GBASE

GBase 8a 程序员手册 ADO.NET

www.gbase.cn

GBase 8a 程序员手册 ADO.NET 篇，南大通用数据技术股份有限公司

通讯方式

南大通用数据技术股份有限公司

天津华苑产业区海泰发展六道 6 号海泰绿色产业基地 J 座(300384)

电话：400-013-9696　　邮箱：info@gbase.cn

目 录

前言

手册简介

GBase 8a 程序员手册从程序员进行数据库开发的角度对 GBase 8a 进行详细介绍。

本手册介绍供客户端连接 GBase 8a 服务器用的 GBase ADO.NET 接口驱动程序。本部分内容通过大量示例为用户演示如何使用 GBase ADO.NET 驱动。

公约

下面的文本约定用于本文档：

约　定	说　明
加粗字体	表示文档标题
大写英文（SELECT）	表示 GBase 8a 关键字
等宽字体	表示代码示例
…	表示被省略的内容。

1 GBase ADO.NET 概述

GBase ADO.NET 是一个提供.NET 应用程序与 GBase 数据库之间方便、高效、安全交互的接口程序，使用 100%纯 C#编写，并继承了 Microsoft ADO.NET 类。开发人员可以使用任何一种.NET 开发语言(C#、VB.NET、F#)通过 GBase ADO.NET 操作 GBase 数据库。

GBase ADO.NET 支持以下全部特性:

- 支持针对集群的带负载均衡的连接池功能
- 支持集群高可用功能
- 支持集群负载均衡功能
- 支持 GBase 数据库全部特性，如：存储过程、视图等
- 支持大型数据包，可发送和接收高达 2GB 的 BLOB 数据
- 支持协议压缩，允许对客户端和服务器之间交互的数据流进行压缩
- 支持 Windows 平台下的 TCP/IP 套接字连接
- 支持 Linux 平台下的 TCP/IP 套接字或 Linux 套接字连接
- 无需安装 GBase 数据库的客户端，可通过 GBase ADO.NET 类库实现完整的管理功能

注：此文档为 GBase ADO.NET 开发手册，并不是 Microsoft ADO.NET 的完整指南,关于 Microsoft ADO.NET 的详细介绍请参考 MSDN 上 ADO.NET 相关章节。

2 GBase ADO.NET 版本

1） GBase ADO.NET 8.3.81.51 版本提供如下特性：

支持 GBase 8s 全部特性，如：存储过程、视图等。

2） GBase ADO.NET 8.3.81.52 版本提供如下特性：

支持 GBase 8a 全部特性，如：存储过程、视图等。

3） GBase ADO.NET 8.3.81.53 版本提供如下特性：

- 支持 GBase 8a 集群全部特性，如：存储过程、视图等。
- 针对 GBase 8a 集群提供高可用性特性，包括集群高可用和集群负载均衡。
- 支持 GBase UP，如：存储过程、视图等。

GBase ADO.NET 与 GBase 产品及与.NET Framework 兼容性参见如下表。

版本	支持的产品	支持的.Net Framework 版本	版本兼容性
8.3.81.52	GBase 8a	2.0、3.0、3.5、4.0	↑
8.3.81.53	GBase 8a Cluster	2.0、3.0、3.5、4.0、4.5、4.5.1	↑
8.3.81.53	GBase UP	2.0、3.0、3.5、4.0、4.5、4.5.1	↑

3 安装文件

我们提供的 ADO.NET 接口的 msi 文件格式如下:

GBaseADO.NET-<product version>-<build version>-<os version and archtecture>.msi。

例如: GBaseADO.NET-8.3.81.53-build52.5-Windows-x86.msi。

4 GBase ADO.NET 结构

GBase ADO.NET (全称是 .NET Framework Data Provider For GBase) 提供给.NET 应用程序访问 GBase 数据库、获取数据、管理数据的一套完整的解决方案。

GBase ADO.NET 的四个核心类及若干功能类具有以下功能:

- 建立和管理与 GBase 数据库连接
- 执行 SQL 语句、访问存储过程
- 对结果快速读取
- 对结果集存储及管理
- 使用事务

GBase ADO.NET 的结构如下图所示。

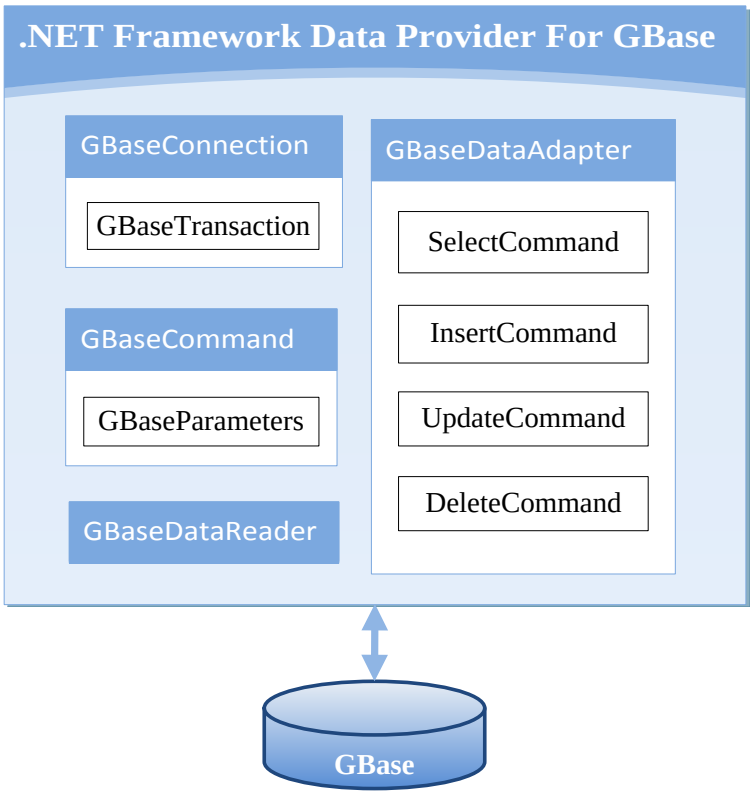


图 4-1 GBase 8a ADO.NET 系统架构图

下面是 GBase ADO.NET 的四个核心类介绍:

- GBaseConnection: 代表一个与 GBase 服务器数据库的连接。这个类不能被继承。
- GBaseCommand: 代表一个要对 GBase 数据库执行操作的 SQL 语句，这个类不能被继承。
- GBaseDataReader: 代表 GBase 数据源中只进且只读的数据流，此类不能被继承。
- GBaseDataAdapter: 是 DataSet 和 GBase 数据库之间的桥接器，用于检索和保存数据。

5 使用 GBase ADO.NET

本章节主要描述.NET 应用程序如何使用 GBase ADO.NET 连接到 GBase 数据源，连接时可以使用哪些参数，以及如何使用接口中提供的功能类操作数据库。

5.1创建数据库连接

.NET 应用程序在使用数据库资源前，首先需要创建数据库连接，与数据库进行必要的交互，获取数据、更新数据，然后关闭数据库连接释放资源。下面的章节将逐步介绍如何使用 GBase ADO.NET 构建连接字符串，创建连接、关闭连接以及获取数据和操作数据的方法。

5.1.1连接字符串

连接字符串是应用通过 GBase ADO.NET 传递到 GBase 数据源时设置的初始化信息。其格式是使用分号分隔的键/值参数对列表，关键字不区分大小写，并将忽略键/值对之间的空格。

下面是一个具体的连接字符串的例子。

```
Server=127.0.0.1;Port=5258;Database=test; User Id=gbase;Password=1111;Pooling=false
```

此连接串连接到 GBase Server 的 test 数据库，Server 的地址是 127.0.0.1，端口号为默认值 5258，验证信息中的用户名是 gbase，密码是 1111，并且不启用连接池。关于更详细的连接串的参数含义，请参考下面介绍的连接参数表。

在使用了 GBase ADO.NET 中继承了 .NET Framework 用于处理连接字符串的“字符串生成器类”后，字符串生成器会将连接串中的关键字转换成类的属性并公开。这将有助于在运行时创建有效的连接字符串，并在创建连接前检查字符串合法性。

在 GBase ADO.NET 中可以使用的连接参数如下表。

表 5-1 连接参数表

关键字	默认值	描　述
User Id, Username, Uid, User name, user, userID		连接 GBaseServer 的用户名。
pwd, Password		连接 GBaseServer 的密码。
Host, Server, Data Source, DataSource, Address, Addr, Network Address	localhost	GBase 服务器的主机名。
Initial Catalog, Database		缺省数据库。
Port	5258	连接端口。
Protocol,Connection Protocol	socket	指明连接到 GBase Server 时使用的通讯协议。对于 socket 连接时可以使用“socket 或 tcp”做为参数值。
compress,Use Compression	false	是否启用在 GBase ADO.NET 与 GBase 数据库之间的传输数据压缩。
Allow Batch, AllowBatch	true	当设置为 true 时，可以在一个 command 执行时指定多条 SQL 语句，语句之间使用“;”分隔。
Old Syntax, OldSyntax, UseOldSyntax, Use Old Syntax	false	设置为 true 时，所有代码中使用‘@’符号做为参数标记，如：GBaseCommand cmd = new GBaseCommand("INSERT INTO Test (id) VALUES (@id)", c)。这个选项不建议使用。
Connect Timeout, Connection Timeout, ConnectionTimeout	15	连接 GBase Server 的超时时间。连接时超过给定的时间后，程序会抛出异常。

关键字	默认值	描　述
Default Command Timeout, command timeout, DefaultCommandTimeout	30	向 GBase Server 发出 SQL 命令后，等待服务器返回的超时时间。
Persist Security Info, PersistSecurityInfo	false	当设置为 false 或 no 时，安全敏感信息，如：密码，在连接打开后，连接对象属性中不可见。可以设置的值有 true 和 false，yes 和 no。
Allow Zero Datetime, AllowZeroDateTime	false	如果设置为 true，将返回针对 date 和 datetime 列的含有非法值的 GBaseDateTime 对象。如果设置为 false，将抛出异常。
Convert Zero Datetime, ConvertZeroDateTime	false	如果设置为 true，则会将对象的非法值转换为合法值，如：0000-00-00 转换成 0100-01-01。
Procedure Cache Size, procedure cache, procedurecache, ProcedureCacheSize	25	设置存储过程缓存个数，接口默认将存储最近使用的 25 个存储过程的 metadata(输入/输出数据类型)。若要禁用缓存，请设置此参数为 0。
Respect Binary Flags, RespectBinaryFlags	true	默认时，执行查询“SELECT CONCAT('hello,world', 1)”后的列值将转换为 byte[]类型。设置为 false 时，则将列值转换为 string 类型。
functions return string, FunctionsReturnString	false	设置为 true 时，所有的服务器函数执行后的结果都转换为 string 类型。

关键字	默认值	描　述
Treat Tiny As Boolean, TreatTinyAsBoolean,	true	设置为 false 时，则 TINYINT(1)将被处理为 INT。
CharSet, Character Set, CharacterSet		表明用于发送给服务器的所有查询中的字符集编码，返回的结果集同样也使用此字符集。
Treat Blobs As UTF8, TreatBlobsAsUTF8	False	二进制 blob 被处理为 UTF8
BlobAsUTF8Include Pattern	null	模式匹配的列被处理为 UTF8
BlobAsUTF8Exclude Pattern	null	模式不匹配的列被处理为 UTF8
Logging	false	开启日志功能，开启日志时需要在应用中的 App.config 中定义如下内容，如果应用中不包含 App.config，则需要手动新建同名文件。
	<configuration> <system.diagnostics> <sources> <source name="gbase" switchName="MySwitch"></source> </sources> <switches> <add name="MySwitch" value="verbose" /> </switches> </system.diagnostics> … </configuration> 注：上述其中 value="verbose" 含义为开启全部日志，	

关键字	默认值	描　述
	可选值可以是字符串，也可为数值，如下列表中定义： a) Off[0] ：不开启； b) Error[1] ：仅限错误信息； c) Warning[2] ：警告消息和错误信息； d) Info[3] ：提示信息、警告消息和错误信息； e) Verbose[4] ：详细消息、提示性消息、警告消息和错误信息。	
Ignore Prepare, IgnorePrepare	true	是否忽略 Prepare 调用。开启此参数时，如果在程序中定义了预处理语句，如：insert into tb values(@a)，则在每次执行相同语句时，如：insert into values(188)，只需将数据 188 发送给服务器。不开启参数时，每次执行都需要将完整的语句发送给服务器。因此当需要执行批量操作时开启此参数可以大量减少网络流量，提高执行效率。
Use Procedure Bodies, procedure bodies, UseProcedureBodies	true	设置为 true 时，当执行存储过程时，参数的添加顺序不需要与存储过程体定义的参数顺序一致。为 false 时，顺序需保持一致。
Allow User Variables, AllowUserVariables	false	是否在执行的 SQL 语句中允许存在用户变量，设置为 true 时，可以在语句中包含用户变量，如：SET @a='test';SELECT @a;。
Auto Enlist, AutoEnlist	true	当使用分布式事务时，该关键字指示 GBase.Data.GBaseClient 是否检测事务上下文并自动在分布式事务范围中登记连接。如果 AutoEnlist=true，连接将自动在事务范围（TransactionScope）中登记。如果 AutoEnlist=false，连接不会自动登

关键字	默认值	描 述
		记 。 登 记 后 使 用 TransactionScope.Complete() 方法可将该分布式事务提交，使用 TransactionScope.Dispose() 方法则回滚。
Interactive Session, interactive	false	设置为 true 时，定义为交互式客户端；为 false 时，定义为非交互式客户端。交互式客户端为与 Server 通讯频繁的一类连接，非交互式客户端为通讯不频繁的连接。
Keep Alive, Keepalive	0	对于 TCP 连接，如果该连接空闲时间（与 Server 没有任何数据交互）超过定义的 Connection Timeout 时间，TCP 协议会自动发出探测数据包（keepalive）检测与服务器是否能够正常通讯。此参数为第一个包发出前的等待时间。默认值 0 为不发送。
MultipleActiveResultSets, Multiple Active ResultSets	false	支持多活动结果集，设置为 true 时，同一个连接中可以支持多个活动结果集的使用；设置为 false 时，同一个连接中只支持一个活动结果集。
集群高可用功能参数 注：Failover，GClusterID，IpList 参数仅用于 8a 集群的高可用功能使用，包括连接池。		
Failover，fail over	false	设置 true 时，开启集群高可用功能
GClusterID, gcluster id		集群 ID，可以为有效的数字字母组合。必须以 a-z 任意字符开头的字符串，gclusterId 组成可以包含 a-z、0-9 所有字符长度为最大为 20。

关键字	默认值	描　述
		当 failover=true 且 IpList 参数不为空时，设置该参数，8a 集群接口的负载均衡开启（负载策略为轮询）。在同一应用程序中，gclusterId 可由程序开发人员自行决定，全工程唯一即可。 注：当设置 gclusteId 时，集群高可用功能自动转换为集群负载均衡。
IpList, ip list		IP 地址列表。列表中的参数以“,”分隔，中间不能含有空格。 例如：现有集群中有三个节点： 1） 192.168.0.2 2） 192.168.0.3 3） 192.168.0.4 当创建数据库连接时，如果当前节点（192.168.0.2）不可用，我们希望把连接请求转移到集群中任意一个可用的节点上时可以使用下面的参数组合： Server=192.168.0.2;User Id=usr;Password=pwd;Failover=true;Ip List=192.168.0.3,192.168.0.4
负载均衡连接池参数		
pooling	true	默认时，则开启连接池功能，GBaseConnection 对象将从连接池中获取。可设置的值为 true 和 false，yes 和 no。
min pool size, min	3	最小空闲连接数。连接池中允许的最小空

关键字	默认值	描　述
idle size, MinimumPoolSize, Minimum Pool Size		闲连接数量。此值为集群最小空闲连接数，如：集群有 3 个节点，每个节点的最小空闲连接数为 1 个。
max pool size,max idle size, MaximumPoolSize, Maximum Pool Size	3	最大空闲连接数。连接池中允许的最大空闲连接数量。此值为集群最大空闲连接数，如：集群有 3 个节点，每个节点的最大空闲连接数为 1 个。
checkout timeout, CheckoutTimeout	0	当连接池中空闲连接用尽后，客户端等待获取新连接的时间，超时后将抛出异常,如设为 0 则无限期等待。单位秒。
initial pool size, InitialPoolSize	3	初始化连接池时缓存连接的数量，取值应在 min Idle size 与 max Idle size 之间。此值为集群初始化缓存连接数，如：集群有 3 个节点，每个节点的初始化缓存连接数为 1 个。
max active size, Maximum Active Size, MaximumActiveSize	2000	最大活动连接数，允许从连接池中借出连接的最大数量。
max idle lifetime, Maximum Idle LifeTime, MaximumIdleLifeTime	28800(8 hrs)	空闲连接存活时间。 该连接在连接池中多长时间没有被借用，则将其关闭，默认与 GBase Server 中允许连接存活的最大超时时间一致。单位秒。
max inuse lifetime, Maximum inuse LifeTime, MaximumInUseLifeTime	0	已用连接存活时间。 连接池借出的连接多长时间后不归还，则认为该连接出现问题，连接池会将其强制关闭。当为 0 时则为无限长时间。单位秒。
test on borrow, TestOnBorrow	false	获取连接前是否检查有效性。是指从空闲连接队列中获取到连接后检查是否能够与

关键字	默认值	描　述
		GBase 集群正常交互。可设置的值有：true 和 false。 **提示：当设置此参数后，会在每次从连接池中获取连接后，进行检查。此操作会有性能上的损失，但可以保证获取到的连接时可用的。**
test on return, TestOnReturn	false	返回连接时是否检查有效性。是指将连接对象返回到空闲队列前检查是否能够与 GBase 集群正常交互。可设置的值有：true 和 false。 **提示：当设置此参数后，当连接返回到连接池中前，进行检查。此操作会有性能上的损失，但可以保证归还到连接池中的连接是可用的。**
test while idle, TestWhileIdle	true	是否检查连接的有效性，包括： 1）检查空闲的连接和借出的连接是否过期，过期条件为 max idle lifetime 和 max inuse lifetime 约束。 2）空闲连接是否能够与 Server 正常交互。 3）动态补充空闲连接数量。 可设置的值有：true 和 false。
invalid idle test period, invalididletestperiod	60	空闲连接检测周期。每隔多长时间检查所有连接池中的空闲连接。单位秒。
supply idle period, supplyidleperiod	60	维护空闲连接周期。每隔多长时间维护连接池中的连接，维护功能包括：补充缺少的空闲连接和删除多余空闲连接。补充条件为可用节点的当前空闲连接数小于最小空闲连接数。删除条件为当前空闲连接数大于最大空闲连接数。维护功能随负载均

关键字	默认值	描　述
		衡连接池开启而开启。单位秒。
LoadBalanceStrategy, load balance strategy, LBS	polling	客户端负载策略，连接池把请求分配给节点的策略。可选值为： 1)　polling ：轮询策略。 2)　mncp ：节点当前连接最小者优先。
connection reset, ConnectionReset	false	设置为 true 时，表明连接从池中获取后是否状态重置。可设置的值为:true 和 false。

5.1.2打开和关闭数据库连接

下面的样例代码使用连接字符串通过 GBaseConnection 类创建连接对象、打开连接、关闭连接。

C# 示例：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Data;
using GBase.Data.GBaseClient;

namespace UsingAdoNet
{
    class Program
    {
        static void Main(string[] args)
        {
            String _ConnStr = "server=192.168.5.41;user
id=root;password=1;database=test;port=5258;pooling=false";
            using (GBaseConnection _Conn = new GBaseConnection())
            {
```

```
                try
                {
                    _Conn.ConnectionString = _ConnStr;
                    _Conn.Open();
                    // do something
                }
                catch (GBaseException ex)
                {
                    Console.WriteLine(ex.StackTrace);
                }
                finally
                {
                    if( _Conn != null )
                        _Conn.Close();
                }
            }
        }
    }
}
```

5.2操作数据库

本章节将详细描述如何通过 GBase ADO.NET 接口访问 GBase 数据库，并读取、更新数据库。

5.2.1读取数据

通过 GBase ADO.NET 接口读取 GBase Server 数据需要下面的步骤：

1) 使用 GBaseConnection 创建数据库连接对象

2) 使用 GBaseCommand 创建命令对象

3) 使用连接对象打开连接

4) 设置命令对象的 CommandText 属性，指明查询语句，并关联连接对象

5) 执行命令对象的 ExecuteReader 方法后返回结果集

- ExecuteReader方法指定 CommandBehavior.SingleResult参数时返回单个结果集。
- ExecuteReader 方法指定 CommandBehavior.Default 参数时返回多个结果集。

6) 关闭数据连接

下面的例子将展示如何循环读取某一列的所有数据，并打印出来。

C# 示例:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Data;
using GBase.Data.GBaseClient;

namespace UsingAdoNet
{
    class Program
    {
        static void Main(string[] args)
        {
            String _ConnStr = "server=192.168.5.41;user
id=root;password=1;database=test;pooling=false";
            using (GBaseConnection _Conn = new
GBaseConnection(_ConnStr))
            {
                try
                {
                    _ String _CmdText = "select * from
`test`.`test`";
```

```
                    GBaseCommand cmd = new GBaseCommand(_CmdText,
_Conn);

                    _Conn.Open();
                    GBaseDataReader reader =
cmd.ExecuteReader(CommandBehavior.SingleResult);
                    while (reader.Read())
                    {
                        Console.WriteLine(reader.GetValue(0));
                    }
                    reader.Close();
                }
                catch (GBaseException ex)
                {
                    Console.WriteLine(ex.StackTrace);
                }
                finally
                {
                    if( _Conn != null )
                        _Conn.Close();
                }
            }
        }
    }
}
```

5.2.2更新数据

通过 GBase ADO.NET 接口修改 GBase Server 数据需要下面的步骤:

1) 使用 GBaseConnection 创建数据库连接对象

2) 使用 GBaseCommand 创建命令对象

3) 使用连接对象打开连接

4) 设置命令对象的 CommandText 属性，指明 SQL 修改语句（Insert 或

Delete 或 Update)，并关联连接对象

5） 调用 GBaseCommand 的 ExecuteNonQuery 方法执行 SQL 修改语句

6） 关闭数据库连接

下面的例子将展示如何打开数据库连接，向 test 表插入一条数据，可以使用同样的步骤 Delete 或者 Update 数据。

C# 示例:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Data;
using GBase.Data.GBaseClient;

namespace UsingAdoNet
{
    class Program
    {
        static void Main(string[] args)
        {
            String _ConnStr = "server=192.168.5.41;user
id=root;password=1;database=test;pooling=false";
            using (GBaseConnection _Conn = new
GBaseConnection(_ConnStr))
            {
                try
                {
                    tring _CmdText = "insert into
`test`(`varchar_column`) values('varchar_value') ";
                    GBaseCommand _Cmd = new GBaseCommand(_CmdText,
_Conn);
```

```
                    _Conn.Open();
                    _Cmd.ExecuteNonQuery();
                }
                catch (GBaseException ex)
                {
                    Console.WriteLine(ex.StackTrace);
                }
                finally
                {
                    if( _Conn != null )
                        _Conn.Close();
                }
            }
        }
    }
}
```

5.3 使用 GBase ADO.NET 访问存储过程

存储过程是一组存储在服务器上的 SQL 语句，GBase ADO.NET 使用 GBaseCommand 对象调用存储过程，并且支持 In、Out、InOut 参数。

5.3.1 通过 GBase ADO.NET 创建存储过程

GBase 数据库中的存储过程可以通过 GBase ADO.NET 直接创建，使用 GBaseCommand 对象的 ExecuteNonQuery 方法。

下面是创建存储过程的步骤：

1) 使用 GBaseConnection 创建数据库连接对象

2) 使用 GBaseCommand 创建命令对象

3) 使用连接对象打开连接

4) 设置命令对象的 CommandText 属性，指明创建存储过程语句，并关联连接对象；

5) 执行命令对象的 ExecuteNonQuery 完成创建；

6) 关闭数据库连接

下面的示例是使用 GBaseCommand 对象创建存储过程的完整过程。

C# 示例：

```
using System.Linq;
using System.Text;
using System.Data;
using GBase.Data;
using GBase.Data.GBaseClient;
using System.Diagnostics;
using System.Collections.Generic;

namespace UsingStoredRoutines
{
    class Program
    {
        static void Main(string[] args)
        {
            GBaseConnection _Conn = new GBaseConnection();
            _Conn.ConnectionString =
"server=192.168.5.41;user=root;database=test;password=1;pooling=fals
e";
            GBaseCommand _Cmd = new GBaseCommand();
            try
            {
                Console.WriteLine("Connecting to GBase...");
                _Conn.Open();
                _Cmd.Connection = _Conn;
                _Cmd.CommandText = "DROP PROCEDURE IF EXISTS
add_emp";
```

```
                    _Cmd.ExecuteNonQuery();
                    _Cmd.CommandText = "DROP TABLE IF EXISTS emp";
                    _Cmd.ExecuteNonQuery();
                    _Cmd.CommandText = "CREATE TABLE emp (empno INT NOT
NULL AUTO_INCREMENT PRIMARY KEY, first_name VARCHAR(20), last_name
VARCHAR(20), birthdate DATE)";
                    _Cmd.ExecuteNonQuery();
                    _Cmd.CommandText = "CREATE PROCEDURE add_emp(" +
                    "IN fname VARCHAR(20), IN lname VARCHAR(20), IN bday
DATETIME, OUT empno INT)" +
                    "BEGIN INSERT INTO emp(first_name, last_name,
birthdate) " +
                    "VALUES(fname, lname, DATE(bday)); SET empno =
LAST_INSERT_ID(); END";
                    _Cmd.ExecuteNonQuery();
                }
                catch (GBaseException ex)
                {
                    Console.WriteLine("Error " + ex.Number + " has
occurred: " + ex.Message);
                }
                finally
                {
                        if (_Conn != null)
                            _Conn.Close();
                        Console.WriteLine("Connection closed.");
                }
            }
        }
    }
```

5.3.2 在GBase ADO.NET 中调用一个存储过程

使用 GBase ADO.NET 调用一个存储过程，要先创建一个 GBaseCommand 对象并且使用属性 CommandText 传递存储过程名，并将属性 CommandType 设为 CommandType.StoredProcedure。为存储过程中的每个输入、输出参数创建一个

GBaseCommand 参数。使用 GBaseCommand.ExecuteNonQuery()方法来调用存储过程。

调用存储过程的步骤:

1) 使用 GBaseConnection 创建数据库连接对象

2) 使用 GBaseCommand 创建命令对象

3) 使用连接对象打开连接

4) 设置命令对象的属性 CommandType 为 CommandType.StoredProcedure,并关联连接对象

5) 设置命令对象的参数与库中要访问的存储过程参数一一对应

6) 执行命令对象的 ExecuteNonQuery 方法完成调用

7) 使用 GBaseCommand.Parameters 的属性 Value 获得输出参数的值

8) 关闭数据库连接

下面是使用 GBase ADO.NET 调用一个存储过程并获取输出参数的完整例子。

C# 示例:

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using GBase.Data;
using GBase.Data.GBaseClient;
using System.Diagnostics;

namespace UsingStoredRoutines
{
    class Program
    {
```

```
            static void Main(string[] args)
            {
                GBaseConnection _Conn = new GBaseConnection();
                _Conn.ConnectionString =
"server=192.168.5.41;user=root;database=test;password=1;pooling=fals
e";
                GBaseCommand _Cmd = new GBaseCommand();
                try
                {
                    Console.WriteLine("Connecting to GBase...");
                    _Conn.Open();
                    _Cmd.Connection = _Conn;

                    _Cmd.CommandText = "add_emp";
                    _Cmd.CommandType = CommandType.StoredProcedure;

                    _Cmd.Parameters.AddWithValue("@lname", "Jones");
                    _Cmd.Parameters["@lname"].Direction =
ParameterDirection.Input;

                    _Cmd.Parameters.AddWithValue("@fname", "Tom");
                    _Cmd.Parameters["@fname"].Direction =
ParameterDirection.Input;

                    _Cmd.Parameters.AddWithValue("@bday",
"1940-06-07");
                    _Cmd.Parameters["@bday"].Direction =
ParameterDirection.Input;

                    _Cmd.Parameters.AddWithValue("@empno",
GBaseDbType.Int32);
                    _Cmd.Parameters["@empno"].Direction =
ParameterDirection.Output;

                    _Cmd.ExecuteNonQuery();

                    Console.WriteLine("Employee number: " +
```

```
_Cmd.Parameters["@empno"].Value);
                    Console.WriteLine("Birthday: " +
_Cmd.Parameters["@bday"].Value);
                }
                catch (GBaseException ex)
                {
                    Console.WriteLine("Error " + ex.Number + " has
occurred: " + ex.Message);
                }
                finally
                {
                    if (_Conn != null)
                        _Conn.Close();
                    Console.WriteLine("None.");
                }
            }
        }
    }
```

5.4GBase ADO.NET 中的事务

GBase ADO.NET 中支持事务，可以使用 GBaseConnection 对象的 BeginTransaction 函数开始一个事务，并默认使用 ReadCommitted 模式初始化。

事务中可以对单个表执行多个操作，或者对多个表执行多个操作，在事务未提交前，事务中的这些命令执行后并不是真正的影响数据库记录。当调用 BeginTransaction 返回对象的 commit 方法时，才会真正的影响记录。

5.4.1在 GBase ADO.NET 中使用事务

在 GBase ADO.NET 使用事务时需下列步骤:

1) 使用 GBaseConnection 创建数据库连接对象

2) 使用 GBaseCommand 创建命令对象

3) 使用连接对象打开连接

4) 使用连接对象的 BeginTransaction 开启事务，返回事务对象

5) 将命令对象与连接对象及事务对象关联

6) 命令对象对表进行一些操作

7) 执行事务对象的提交方法

8) 命令对象执行失败后，执行事务对象的回滚方法

下面的例子展示在 GBase ADO.NET 中如何使用事务。

[C#]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using GBase.Data;
using GBase.Data.GBaseClient;
using System.Diagnostics;

namespace UsingStoredRoutines
{
    class Program
    {
        static void Main(string[] args)
        {
            GBaseConnection _Conn = new GBaseConnection();
            _Conn.ConnectionString =
"server=192.168.5.41;user=root;database=test;password=1;pooling=fals
e";
            GBaseCommand _Cmd = new GBaseCommand();
            GBaseTransaction _Trans = null;
```

```
                try
                {
                    Console.WriteLine("Connecting to GBase...");
                    _Conn.Open();

                    _Trans = _Conn.BeginTransaction();
                    _Cmd.Connection = _Conn;
                    _Cmd.Transaction = _Trans;

                    // 在 GBase 8a 集群下需显示关闭自动提交模式
                    // _Cmd.CommandText = "set autocommit = false";
                    / _Cmd.ExecuteNonQuery();
                    _Cmd.CommandText = "insert into `test`(`f_varchar`)
values('111')";
                    _Cmd.ExecuteNonQuery();
                    _Cmd.CommandText = "insert into `test`(`f_varchar`)
values('222')";
                    _Cmd.ExecuteNonQuery();
                    _Trans.Commit();
                    Console.WriteLine("Transaction has committed.");
                }
                catch (GBaseException ex)
                {
                    _Trans.Rollback();
                    Console.WriteLine("Error " + ex.Number + " has
occurred: " + ex.Message);
                }
                finally
                {
                    if (_Conn != null)
                        _Conn.Close();
                    Console.WriteLine("None.");
                }
            }
        }
    }
```

5.5在 GBase ADO.NET 中的预处理语句

预处理语句是在 GBase ADO.NET 中预先定义的可重复使用的 SQL 语句。在 GBase ADO.NET 中使用预处理语句,能明显改善一个 SQL 语句执行多次时的性能。

5.5.1使用预处理语句

GBase ADO.NET 支持预处理语句，使用预处理语句时需要创建一个 GBaseCommand 命令对象，并且设置相关属性及定义查询语句后调用命令对象的 ExecuteNonQuery()、ExecuteScalar()或 ExecuteReader 方法执行语句。

下面是在 GBase ADO.NET 中定义预处理语句的步骤:

1) 使用 GBaseConnection 创建数据库连接对象

2) 使用 GBaseCommand 创建命令对象

3) 使用连接对象打开连接

4) 设置命令对象的 CommandText 属性，指明预处理语句，并定义参数

5) 执行命令对象的 Prepare 方法

6) 增加命令对象的 Parameters 的参数

7) 设置命令对象 Parameters 的参数值

8) 执行命令对象的 ExecuteNonQuery 方法

下面的样例代码演示如何定义预处理语句和使用预处理语句功能完成数据的批量插入。

C# 示例:

```
using System;
using GBase.Data.GBaseClient;

namespace UsingGBase
```

```
    {
        class Program
        {
            static void Main(string[] args)
            {
                String connectionString =
"server=192.168.5.41;database=test;user
id=root;password=1;pooling=false";
                GBaseConnection conn = new
GBaseConnection(connectionString);
                GBaseCommand cmd = new GBaseCommand();
                try
                {
                    conn.Open();
                    cmd.Connection = conn;

                    cmd.CommandText = "DROP TABLE IF EXISTS gsTable";
                    cmd.ExecuteNonQuery();
                    cmd.CommandText = "CREATE TABLE gsTable (id int,c1
bigint ,c2 varchar(100) )";
                    cmd.ExecuteNonQuery();

                    cmd.CommandText = "INSERT INTO gsTable VALUES(NULL,
@number, @text)";
                    cmd.Prepare();

                    cmd.Parameters.AddWithValue("@number", 1);
                    cmd.Parameters.AddWithValue("@text", "One");

                    for (int i = 1; i <= 1000; i++)
                    {
                        cmd.Parameters["@number"].Value = i;
                        cmd.Parameters["@text"].Value = "A string
value";

                        cmd.ExecuteNonQuery();
                    }
```

```
                }
                catch (GBaseException ex)
                {
                    Console.WriteLine("Error " + ex.Number + " has
occurred: " + ex.Message);
                }
            }
        }
    }
```

5.5.2使用预处理语句的优点

预处理执行比多次直接执行语句要快得多，主要是因为查询只被解析一次。而在直接执行时，查询在每次执行时都要被解析。

下面列举了预处理语句的主要优点：

1） 预处理语句的执行方式减低网络流量。

因为每次对预处理语句的执行只需发送与参数相关的数据，所以该执行方式减低网络流量。

2） 提高了客户端和服务器之间的数据传输效率。

因为预处理语句是在客户端和服务器之间的传输过程中使用二进制协议，提高了客户端和服务器之间的数据传输效率。

5.6使用 GBase ADO.NET 处理日期和时间信息

GBase 数据库和.NET 语言处理日期和时间信息的方式是不同的，GBase 数据库允许的日期有时不能用一个.NET 的数据类型表示，比如“0000-00-00 00:00:00”。如果处理不当，则会产生问题。

在本章节中，我们将说明在使用 GBase ADO.NET 时如何正确处理日期和时间信息。

5.6.1使用无效日期的问题

由于 GBase 数据库允许“0000-00-00”日期值存储到 Date 类型字段，“0000-00-00 00:00:00”和 NULL 时间日期值存储到 DateTime 类型，但这些日期值都是无效的，.NET 框架的日期对象和日期时间对象不支持这些值。对于使用无效日期的开发人员，日期处理上的差异可能产生问题。

因此.NET DataSet 对象不能使用 GBaseDataAdapter 类的 Fill 方法填充，因为无效日期的转换将会引发 System.ArgumentOutOfRangeException 异常。

5.6.2解决无效日期的方法

解决无效日期的方法有如下三种:限制无效日期、处理无效日期和处理 NULL 日期。

5.6.2.1 限制无效日期

日期问题的最好解决办法是限制用户输入无效日期，该方法可以在客户端或服务器端任何一方实现。

1) 客户端实现:

.NET Framework 为应用程序提供了日期控件，当应用程序需要输入日期时使用日期控件完成输入。

2) 服务器端实现:

日期数据进入数据库时都可通过存储过程完成，无效日期的限制在存储过程中完成。

5.6.2.2 处理无效日期

GBaseDateTime 数据类型支持 GBase 数据库支持的相同日期值,缺省情况下

GBaseDataReader.GetValue()方法会为有效日期值返回一个.NET DateTime 对象，并且会为无效日期返回一个错误。可以更改这个缺省方式，让 GBaseDataReader.GetValue() 对于无效日期也返回一个 GBaseDateTime 对象。

要想使 GBase ADO.NET 为无效日期也返回一个 GBaseDateTime 对象，需要在连接字符串中增加下面的键/值对：

Allow Zero Datetime=True

注意：使用 GBaseDateTime 类仍然可能产生问题，下面有一些结论：

- 无效日期的数据绑定仍然可能产生错误（0000-00-00 零日期不会产生错误），如：2012-13-35；
- ToString 方法返回按标准 GBase 数据库格式处理的日期（例如，2005-02-23 08:50:25），这与.NET DateTime 类的 ToString 不同；
- GBaseDateTime 类支持 NULL 日期，但是.NET DateTime 类不支持。如事先不检查 NULL 就把一个 GBaseDateTime 转换为一个 DateTime 时，会产生错误。

5.6.2.3 处理 NULL 日期

NET 的 DateTime 数据类型不支持 NULL 值，在查询中为 DateTime 类型变量赋值时，必须首先检查该值是否为 NULL。使用 GBaseDataReader 得到 DateTime 列数据时，应在将获取到的值赋值给 DateTime 变量前用 IsDBNull 方法检查数据是否值为 NULL，如下面的样例代码中所示。

C# 示例：

```
if (! gsReader.IsDBNull(gsReader.GetOrdinal("gstime")))
    gsTime = gsReader.GetDateTime(gsReader.GetOrdinal("gstime"));
else
    gsTime = DateTime.MinValue;
```

5.7使用 GBase ADO.NET 处理 BLOB 数据

使用 GBase 数据库时经常会在 BLOB 列中存储二进制数据，GBase 数据库支持四种不同的 BLOB 数据类型：TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB。

.NET 应用程序可使用 GBase ADO.NET 访问保存在 BLOB 列中的数据，并能使用客户端代码对这类数据进行操作。

5.7.1将文件写到数据库

本节中的例子会使用一个带有 BLOB 列的表，使用下面的语句生成 file 表。file_id：表主键，file_name：存储的文件名，file_size：存储文件的大小，file：存储文件内容。

```
CREATE TABLE file (
file_id SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
file_name VARCHAR(64) NOT NULL,
file_size MEDIUMINT UNSIGNED NOT NULL,
file MEDIUMBLOB NOT NULL );
```

下面的代码使用 FileStream 对象打开一个文件，把文件内容读到一个字节数组中，并且把它插入到 file 表中。

C# 示例：

```
using System;
using System.IO;
using GBase.Data.GBaseClient;

namespace UsingGBase
{
    class Program
```

```
        {
            static void Main(string[] args)
            {
                GBaseConnection conn = new GBaseConnection();
                GBaseCommand cmd = new GBaseCommand();

                string SQL;
                int FileSize;
                byte[] rawData;
                FileStream fs;
                string strFileName = @"c:\newfile.png";
                conn.ConnectionString =
"server=192.168.5.41;database=test;user
id=root;password=1;pooling=false";

                try
                {
                    fs = new FileStream(@"c:\test.bmp", FileMode.Open,
FileAccess.Read);
                    FileSize = (int)fs.Length;

                    rawData = new byte[FileSize];
                    fs.Read(rawData, 0, FileSize);
                    fs.Close();

                    conn.Open();
                    SQL = "INSERT INTO file VALUES(NULL, @FileName,
@FileSize, @File)";

                    cmd.Connection = conn;
                    cmd.CommandText = SQL;
                    cmd.Parameters.AddWithValue("@FileName",
strFileName);
                    cmd.Parameters.AddWithValue("@FileSize",
FileSize);
                    cmd.Parameters.AddWithValue("@File", rawData);
```

```
                    cmd.ExecuteNonQuery();

                    Console.WriteLine("File Inserted into database
successfully!");

                    conn.Close();
                }
                catch (GBaseException ex)
                {
                    Console.WriteLine("Error " + ex.Number + " has
occurred: " + ex.Message);
                }
            }
        }
    }
```

注意:创建表完成后,需要确定 GBase 数据库系统变量 max_allowed_packet 的值，该变量决定能发送给 GBase 服务器的数据包（比如一行）大小。修改 max_allowed_packet 时需注意 BLOB 数据在网络上传输时需要一些时间,应尽量把该值设为与实际需要相符，并在必要时设置更大值。

要将一个文件写到数据库需要将文件转换为一个字节数组，然后将该字节数组作为一个 INSERT 语句的参数，通过 GBase ADO.NET 将值写入到表中。

5.7.2从数据库中读取 BLOB 数据写入到一个文件

下面的代码从上一节中创建的 file 表中获得一行,并将数据写入到文件中。

C# 示例:

```
using System;
using System.IO;
using GBase.Data.GBaseClient;

namespace UsingGBase
{
```

```
        class Program
        {
            static void Main(string[] args)
            {
                GBaseDataReader gsData;

                GBaseConnection conn = new GBaseConnection();
                GBaseCommand cmd = new GBaseCommand();

                string SQL;
                int FileSize;
                byte[] rawData;
                FileStream fs;

                conn.ConnectionString =
"server=192.168.5.41;uid=root;pwd=1;database=test;pooling=false";

                SQL = "SELECT file_name, file_size, file FROM file";

                try
                {
                    conn.Open();

                    cmd.Connection = conn;
                    cmd.CommandText = SQL;

                    gsData = cmd.ExecuteReader();

                    if (!gsData.HasRows)
                        throw new Exception("There are no BLOBs to
save");


                    gsData.Read();

                    FileSize =
(int)gsData.GetUInt32(gsData.GetOrdinal("file_size"));
                    rawData = new byte[FileSize];
```

```
                    gsData.GetBytes(gsData.GetOrdinal("file"), 0,
rawData, 0, FileSize);

                    fs = new FileStream(@"C:\newfile.png",
FileMode.OpenOrCreate, FileAccess.Write);
                    fs.Write(rawData, 0, FileSize);
                    fs.Close();
                    Console.WriteLine("File successfully written to
disk!");

                    gsData.Close();
                    conn.Close();
                }
                catch (GBaseException ex)
                {
                    Console.WriteLine("Error " + ex.Number + " has
occurred: " + ex.Message);
                }

            }
        }
    }
```

5.8在全局配置文件中添加 DSN

如果您有很多.NET 程序会使用一个相同的连接串访问数据库资源，并且希望此连接串是可以在应用之外进行配置时，可以使用 machine.config 配置文件满足需求。即在配置文件中添加一个 ConnectionString 条目，并指定数据源名字（DSN），应用即可使用数据源名字获取到连接串信息。

下面的步骤将介绍如何在 machine.config 添加一个条目。

- 使用文本编辑器或者 XML 编辑器打开
- 搜索 <connectionStrings> 块

- 使用下面的格式添加一个条目。

<add name="DSNGBaseDatabase"

connectionString="server=127.0.0.1; database=test;pooling=false"

providerName="GBase.Data.GBaseClient">

machine.config 配置文件是.NET Framework 的核心文件，修改不当则会导致不可预料的后果，修改前请先备份。默认位置在：

- .NET Framework 2.0 使用

C:\Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG\machine.config

- .NET Framework 3.0、3.5、4.0、4.5 使用

C:\Windows\Microsoft.NET\Framework\v4.0.30319\CONFIG\machine.config

5.8.1在.NET 应用程序中使用 DSN

在.NET 应用程序中使用 machine.config 文件中的资源时，首先需在工程中引用 .NET Framework 的组件 System.configuration，然后使用 using 语句将 System.configuration 命名空间中的类引入到工程中。

1) 在工程中引用 System.configuration 组件

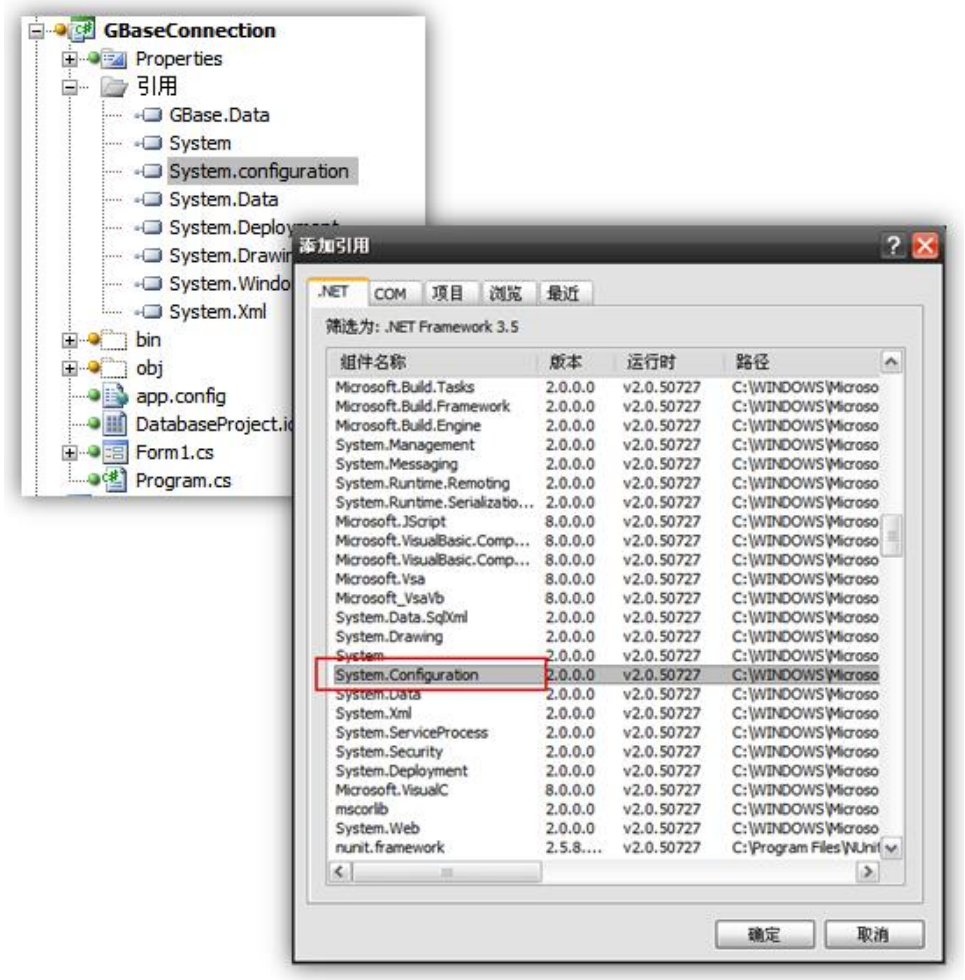


图 5-1 System.configuration 组件

2) 在 .NET 工程中使用数据源名

C# 示例:

```
String _gcConn =
System.Configuration.ConfigurationManager.ConnectionStrings
["DSNGBaseName"].ConnectionString;
```

5.9在GBase ADO.NET 中使用Crystal Reports

Crystal Reports 是.NET 应用程序开发人员用于报表生成的常用工具，在

本节我们将展示如何在 GBase 和 ADO.NET 中使用 Crystal Reports XI。

5.9.1创建数据源

在 Crystal Reports 中创建报表时，可利用两种方法获取 GBase 数据源数据。

1) 第一种方法是在设计报告时使用 GBase ODBC 作为一个 ADO 数据源，可以浏览数据库，并且依靠拖拽来选择表和字段建立报表。

2) 第二种方法是在.NET 程序中创建一个 DataSet 并将其保存为 XML 格式文件，使用该 XML 文件能够作为一个 GBase ADO.NET XML 数据源用来设计报表。当在应用程序中只显示这个报表时可以这样做，但是在设计时很不方便，因为在创建数据集时必须选择所有相关的列。如果漏掉一列，就必须在建立报表前重新创建数据集。

下面的代码演示将一个查询的结果生成一个 DataSet 并把它写入 XML 文件保存到硬盘上。

C# 示例:

```
GBaseConnection conn;
GBaseCommand cmd;
GBaseDataAdapter gsAdapter;

conn = new GBaseConnection();
cmd = new GBaseCommand();
gsAdapter = new GBaseDataAdapter();

conn.ConnectionString =
"server=127.0.0.1;uid=root;pwd=1111;database=test;pooling=false"
;

try
```

```
    {
        cmd.CommandText = "SELECT city.name AS cityName,
city.population AS CityPopulation, " +
        "country.name, country.population, country.continent " +
        "FROM country, city ORDER BY country.continent, country.name";
        cmd.Connection = conn;

        gsAdapter.SelectCommand = cmd;
        gsAdapter.Fill(gsData);

        gsData.WriteXml(@"C:\dataset.xml", XmlWriteMode.WriteSchema);
    }
    catch (GBaseException ex)
    {
        MessageBox.Show(ex.Message, "Report could not be created",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
```

5.9.2创建报表

对大多数应用，标准的报表向导应该能帮助完成一个报表的最初创建。要启动向导,只需要打开 Crystal Reports 并且从 File 菜单中选择 New > Standard Report 选项。

向导首先要求提供一个数据源，如果使用 GBase ODBC 作为数据源，选择该项。如果使用一个保存的 DataSet，就选择 ADO.NET (XML)选项并浏览保存的数据集。剩下的报表创建过程由向导自动完成。

报表创建之后,选择 File 菜单中的 Report Options...菜单项,取消 Report 选项的 Save Data 部分，这样可以避免保存的数据干扰应用程序中的数据加载操作。

5.9.3显示报表

下面的步骤使用.NET WinForm 应用程序显示报表，显示报表前，需使用 5.9.2 的步骤生成报表文件 world_report.rpt。

1) 在工程项目中引用用于显示报表的组件

CrytalDecisions.CrystalReports.Engine

CrystalDecisions.ReportSource

CrystalDecisions.Shared

CrystalDecisions.Windows.Forms

2) 在用户界面上拖放一个 Viewer 控件

3) 在代码中引用水晶报表命名空间

CrystalDecisions.CrystalReports.Engine

4) 使用 GBaseConnection 创建连接对象

5) 使用 GBaseCommand 创建命令对象

6) 使用 GBaseDataAdapter 创建适配器对象

7) 使用 ReportDocument 创建报表对象

8) 使用 DataSet 创建数据集对象

9) 建立命令对象与连接对象的关系

10) 设置命令对象的 CommandText 属性，指明查询语句

11) 从数据源中获取数据后使用适配器对象填充到数据集

12) 使用报表对象加载 world_report.rpt 文件

13) 设置报表对象的数据源为数据集对象

14) 设置 Viewer 空间的报表源为报表对象

15) 显示在界面上

C# 示例:

```
using CrystalDecisions.CrystalReports.Engine;
using System.Data;
using GBase.Data.GBaseClient;

ReportDocument gsReport = new ReportDocument();
DataSet gsData = new DataSet();
GBaseConnection conn;
GBaseCommand cmd;
GBaseDataAdapter gsAdapter;

conn = new GBaseConnection();
cmd = new GBaseCommand();
gsAdapter = new GBaseDataAdapter();

conn.ConnectionString = "server=127.0.0.1;uid=root;" +
    "pwd=12345;database=test;";

try
{
    cmd.CommandText = "SELECT city.name AS cityName,
city.population AS CityPopulation, " + "country.name,
country.population, country.continent " + "FROM country, city ORDER BY
country.continent, country.name";
    cmd.Connection = conn;

    gsAdapter.SelectCommand = cmd;
    gsAdapter.Fill(gsData);

    gsReport.Load(@".\world_report.rpt");
    gsReport.SetDataSource(gsData);
    gsViewer.ReportSource = gsReport;
```

```
}
catch (GBaseException ex)
{
    MessageBox.Show(ex.Message, "Report could not be created",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

5.10 使用 GBase ADO.NET 中的 DataAdapter

在 GBase ADO.NET 中的 DataAdapter 是 GBaseDataAdapter 类，此类可将查询的结果填充到 DataSet 中，还可将 DataSet 中修改的数据保存回数据库。GBaseDataAdapter 可将数据表映射为 DataSet，并在断开连接后使用 DataSet 内容。

DataSet 是数据的内存驻留表示形式，它提供了独立于数据源的一致关系编程模型。DataSet 表示整个数据集，其中包含表、约束和表之间的关系。由于 DataSet 独立于数据源，因此 DataSet 可以包含应用程序本地的数据，也可以包含来自多个数据源的数据。关于 DataSet 更详细的介绍请参考 MSDN 有关章节。

注：如果只需要执行 SQL 语句，就没必要用到 GBaseDataAdapter ，直接用 GBaseCommand 的 Execute 系列方法就可以了。GBaseDataAdapter 的作用是实现 DataSet 和 DB 之间的桥梁：比如将对 DataSet 的修改更新到数据库。

5.10.1 使用 GBaseDataAdapter 填充 DataSet

GBaseDataAdapter 提供了 SelectCommand 属性，当为这些属性的值定义了 GBaseCommand 对象并且设置了 SQL 查询语句后可使用 Fill 方法填充 DataSet。

以下代码展示如何使用 GBaseDataAdapter 填充 DataSet：

C# 示例：

```
String qryString = "SELECT * FROM test";
```

```
GBaseDataAdapter adapter = new GBaseDataAdapter(qryString, conn);
DataSet test = new DataSet();
adapter.Fill(test, "test");
```

5.10.2 使用多个 GBaseDataAdapter 填充 DataSet

一个 DataSet 可以与很多 GBaseDataAdapter 对象一起使用。每个 GBaseDataAdapter 都可用于填充一个或多个 DataTable 对象并将更新解析回数据源。DataRelation 对象可以在本地添加到 DataSet，这样您就可以关联来自不同数据源的数据。一个或多个 GBaseDataAdapter 对象可以处理与每个数据源的通信。

以下代码示例使用同一数据源填充 Customers 表和 Orders 表。已填充的表通过 DataRelation 相关联，这样，客户列表将与相应客户的订单一起显示出来。

- Customers 表定义：

```
CREATE TABLE Customers (
CustomerID varchar(5) NOT NULL PRIMARY KEY,
CompanyName VARCHAR(40) NOT NULL );
```

- Orders 表定义：

```
CREATE TABLE Orders (
OrderID int NOT NULL PRIMARY KEY,
CustomerID VARCHAR(40) NOT NULL );
```

C# 示例

```
GBaseDataAdapter custAdapter = new GBaseDataAdapter(
"SELECT * FROM Customers", _Connection);
GBaseDataAdapter ordAdapter = new GBaseDataAdapter (
"SELECT * FROM Orders", _Connection);
DataSet customerOrders = new DataSet();
```

```
custAdapter.Fill(customerOrders, "Customers");
ordAdapter.Fill(customerOrders, "Orders");

DataRelation relation = customerOrders.Relations.Add("CustOrders",
customerOrders.Tables["Customers"].Columns["CustomerID"],
customerOrders.Tables["Orders"].Columns["CustomerID"]);

foreach (DataRow pRow in customerOrders.Tables["Customers"].Rows)
{
    Console.WriteLine(pRow["CustomerID"]);
    foreach (DataRow cRow in pRow.GetChildRows(relation))
    Console.WriteLine("\t" + cRow["OrderID"]);
}
```

6 GBase ADO.NET 数据类型映射

GBase 数据库与 .NET Framework 使用不同的数据类型。为了在读取和写入数据时保证数据的完整性,GBaseDataReader 将公开用于返回 GBase.Data.Types 对象的 GBase 特定的类型化访问器方法。GBase Server 数据类型和 .NET Framework 数据类型通过 DbType 和 GBaseDbType 类中的枚举值定义,当您指定 GBaseParameter 的数据类型时可以使用这些枚举值。

.NET Framework 类型、DbType 和 GBaseDbType 枚举以及 GBaseDataReader 的访问器方法之间的关系如下表。

.NET Framework 数据类型	GBase 数据库数据类型	DbType 枚举	GBaseDbType 枚举	GBaseDataReader 的 DbType 类型化访问器	GBaseDataReader 的 GBaseDbType 类型化访问器
Int64	bigint	Int64	Int64	GetInt64	
Byte[]	binary	Binary	VarBinary	GetBytes	
Boolean	bit	Boolean(UInt64)	Bit	GetBoolean	
String Char[]	char	AnsiStringFixedLength,	String	GetString GetChars	
DateTime	date	Date	Date	GetDateTime	

.NET Framework 数据类型	GBase 数据库数据类型	DbType 枚举	GBaseDbType 枚举	GBaseDataReader 的 DbType 类型化访问器	GBaseDataReader 的 GBaseDbType 类型化访问器
DateTime	datetime	DateTime	DateTime	GetDateTime	GetGBaseDateTime
Decimal	decimal	Decimal	Decimal	GetDecimal	GetGBaseDecimal
Byte[]	varbinary	Binary	VarBinary	GetBytes	
Double	float	Double	Float	GetFloat	
Int32	int	Int32	Int32	GetInt32	
String Char[]	nchar	String FixedLength	String	GetString GetChars	
Decimal	numeric	Decimal	Decimal	GetDecimal	
String Char[]	nvarchar	String	Varchar	GetString GetChars	
Single	real	Single	Float	GetFloat	

.NET Framework 数据类型	GBase 数据库数据类型	DbType 枚举	GBaseDbType 枚举	GBaseDataReader 的 DbType 类型化访问器	GBaseDataReader 的 GBaseDbType 类型化访问器
Int16	smallint	Int16	Int16	GetInt16	
String Char[]	text	String	Text	GetString GetChars	
TimeSpan	time	Time	Time	GetTimeSpan	
Byte[]	timestamp	Binary	Timestamp	GetBytes	
Byte	tinyint	Byte	Byte	GetByte	
Byte[]	varbinary	Binary	VarBinary	GetBytes	
String Char[]	varchar	AnsiString, String	VarChar	GetString GetChars	
Int64	bigint				
Byte[]	binary				

.NET Framework 数据类型	GBase 数据库数据类型	DbType 枚举	GBaseDbType 枚举	GBaseDataReader 的 DbType 类型化访问器	GBaseDataReader 的 GBaseDbType 类型化访问器
Boolean	bit				
String Char[]	char				
DateTime	date				

7 GBase ADO.NET 高可用特性

GBase ADO.NET 高可用特性是针对 GBase 数据库集群提供客户端高可用及负载均衡相关功能。

本章将介绍在 GBase ADO.NET 中如何开启和使用集群高可用性特性（集群高可用性和集群负载均衡）。

7.1集群高可用性

在通过 GBase ADO.NET 访问 GBase 集群时，如果集群当前节点不可用，希望直接连接到集群中一个可用的节点上时，我们可以使用 GBase ADO.NET 集群高可用性功能（该功能需要 GBase ADO.NET 8.3.81.53 及以上版本）。

GBase ADO.NET 集群高可用性是接口针对 GBase 集群开发的在接口层面的高可用性处理。

高可用性适用于扁平结构的 GBase 集群，在通过 GBase ADO.NET 创建连接时，如果当前节点不可用时，接口会根据相关参数信息把连接数据库请求自动路由到集群其他可用的节点上。

假设存在如下场景：

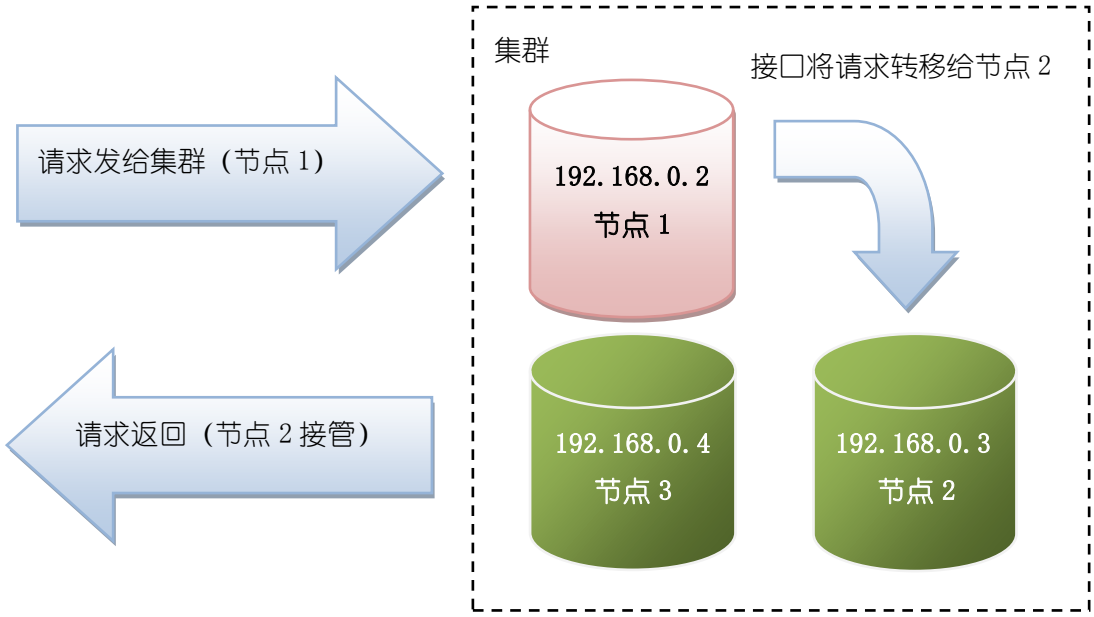
- 现有集群中存在如下三个节点。

节点 1：192.168.0.2 节点 2：192.168.0.3 节点 3：192.168.0.4

- 连接串中设置 server 和 iplist 参数。

```
String _ConnString =
“server=192.168.0.2;failover=true;iplist=192.168.0.3;192.168.0.4”；
```

- 当应用连接到集群中节点 1 时发现故障，此时接口不抛出错误而会将连接请求转移给节点 2，并返回给客户端，下图说明了此种场景。



下面例子介绍如何开启集群高可用功能，并使用 192.168.0.2、192.168.0.3、192.168.0.4 做为高可用节点。

C# 示例：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Data;
using GBase.Data.GBaseClient;

namespace UsingAdoNet
{
    class Program
    {
        static void Main(string[] args)
```

```
            {
                String _ConnStr = "server=192.168.0.2;user
id=root;password=1;database=test;port=5258;pooling=false;failover=tr
ue;iplist=192.168.0.3,192.168.0.4";
                using (GBaseConnection _Conn = new GBaseConnection())
                {
                    try
                    {
                        _Conn.ConnectionString = _ConnStr;
                        _Conn.Open();
                        // do something
                    }
                    catch (GBaseException ex)
                    {
                        Console.WriteLine(ex.StackTrace);
                    }
                    finally
                    {
                        if( _Conn != null )
                            _Conn.Close();
                    }
                }
            }
        }
    }
```

7.2集群负载均衡

如果我们需要将连接请求分摊到集群的每个节点上时，可以使用集群负载均衡功能（该功能需要 GBase ADO.NET 8.3.81.53 及以上版本）。集群负载均衡功能与集群高可用功能都属于客户端负载均衡解决方案。使用此功能时，需同时设置连接串的 Failover、IpList、GClusterID 参数。

集群负载均衡支持多集群的情况，即在一个应用中可以指定多个集群共同分担负载。但在每个连接串中只能指明一个 GClusterID 和此集群的 IpList。

假设存在如下场景：

- 现有集群 G1 中存在三个节点，18 个连接请求。节点 1:192.168.0.2　节点 2：192.168.0.3　节点 3：192.168.0.4。
- 连接串中设置 server、iplist 和 GClusterID 参数。

String _ConnString =

“server=192.168.0.2;failover=true;iplist=192.168.0.3;192.168.0.4;gclusterid=g1”；

- 接口会将 18 个连接请求分摊在节点 1、节点 2 和节点 3 上，分摊后每个节点上会有 6 个请求线程。

注：当某个连接请求访问集群节点 1 时，接口发现节点 1 故障后，在后续的连接请求到来时接口将不再给节点 1 分配请求，而会将请求分配给节点 2 (若节点 2 故障时会分配给节点 3)。

下面例子介绍如何开启集群负载均衡。例子中会创建 18 个连接对象，使用 192.168.0.2、192.168.0.3、192.168.0.4 做为负载均摊节点。

C# 示例：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Data;
using GBase.Data.GBaseClient;

namespace UsingAdoNet
{
    class Program
    {
        static void Main(string[] args)
```

```
            {
                String _ConnStr = "server=192.168.0.2;user
id=root;password=1;database=test;port=5258;pooling=false;failover=tr
ue;iplist=192.168.0.3, 192.168.0.4;gclusterid=g1";
                GBaseConnection[] _Conn = new GBaseConnection[18];
                try
                {
                    for (int i = 0; i < _Conn.Length; i++)
                    {
                        _Conn[i] = new GBaseConnection(_ConnStr);
                        _Conn[i].Open();

                        Thread.Sleep(2000);
                    }
                }
                catch (System.Exception ex)
                {
                    Assert.Fail(ex.Message);
                }
                finally
                {
                    for (int i = 0; i < _Conn.Length; i++)
                    {
                        if (_Conn[i] != null)
                        {
                            if (_Conn[i].State ==
System.Data.ConnectionState.Open )
                                _Conn[i].Close();
                        }
                    }
                }
            }
        }
    }
```

8 GBase ADO.NET 连接池特性

GBase ADO.NET 连接池是针对 GBase 数据库集群提供的客户端连接池功能，且具备负载均衡功能。连接池在 GBase ADO.NET 8.3.81.32 Build 51.0 以上版本提供。

在访问 GBase 集群数据库时，当访问量或并发量很大时，频繁的打开、关闭数据库连接非常耗时，并且会增加数据库的负载压力。GBase ADO.NET 提供的连接池功能可最大限度的减少打开、关闭连接的次数，并可配置不同的策略模式实现负载均衡的功能，将连接请求分摊到集群的各个节点。

本章将介绍在 GBase ADO.NET 中如何开启和使用基于负载均衡的连接池功能。

8.1什么是连接池

连接池是指在客户端连接请求到来时，可以从缓存的若干可用连接中快速获取连接，减少了创建、打开连接的时间。并且在使用后并不关闭，而是重新缓存起来提供给下一个连接请求使用。

8.2什么是负载均衡连接池

基于负载均衡的连接池是在只缓存连接的连接池基础上，根据客户端负载均衡策略模式将客户端连接请求分摊到集群节点，根据服务端策略模式可得到负载最小的缓存连接提供给客户端。

基于负载均衡的连接池有如下特点：

- 缓存所有集群节点的连接
- 提供检查借出连接是否可用
- 提供检测归还连接是否可用

- 根据客户端负载策略以轮询方式获取集群各个节点的缓存连接
- 根据服务端负载策略方式实时或定期获取最小负载节点缓存连接
- 提供定期动态补充连接、清理无效连接、清理过期连接

8.3使用负载均衡连接池

若要使用负载均衡连接池功能，需要在连接串中配置相关的关键字。有关更详细的关键字信息在 5.1.1 章节“表 5-1 连接参数表”中介绍。

假设存在如下场景：

- 现有集群中存在 4 个节点：

192.168.9.173, 192.168.9.174, 192.168.9.175, 192.168.9.176

- 客户端开启负载均衡连接池，设置连接串参数如下：

```
server=192.168.9.173;user id=gbase;password=gbase20110531;
database=test;pooling=true;min idle size=20;max idle size=40;
gclusterid=g1;failover=true; initial pool size=20;
iplist=192.168.9.174,192.168.9.175,192.168.9.176;
test on borrow=true;test on return=true;test while idle=true;
load balance strategy=polling;max inuse lifetime=0;
```

- 客户端使用 GBaseConnection 通过上述连接串进行初始化，并打开连接后。驱动会在集群的每个节点上创建 5 个连接，并缓存起来，并以 polling(轮询)的方式先从 173 节点获取缓存连接。

注：当某个连接请求访问 173 节点时，接口在 173 节点上获取缓存连接后，若发现 173 节点故障，会尝试获取 174 节点的缓存连接，若发现 174 节点故障，会尝试获取 175 节点缓存连接，直到获取到 176 节点的缓存连接。如果所有节点都不可用，则会在默认超时时间后抛出异常。

C# 示例:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using System.Data;
using GBase.Data.GBaseClient;

namespace UsingAdoNet
{
    class Program
    {
        public int _finish = 0;
        public int _thread = 20;
        public int _runTime = 500;

        static void Main(string[] args)
        {
            string enableNewPoolStr =
"server=192.168.9.173;user id=gbase;password=gbase20110531;"+
"pooling=true;Connection lifetime=0;min idle size=20;max idle
size=40;"+
"gclusterid=g1;failover=true;iplist=192.168.9.174,192.168.9.175,
192.168.9.176;"+
"checkout timeout=0;initial pool size=24;max active size=2000;"+

"max idle lifetime=2880000;"+

"test on borrow=true;test on return=true;test while idle=true;"+

"invalid idle test period=30;supply idle period=20;load balance
strategy=polling;max inuse lifetime=0;";
```

```
                Thread[] _WorkThread = new Thread[_thread];
                for (int i = 0; i < _WorkThread.Length; i++)
                {
                    new Thread(WorkThread).Start(enableNewPoolStr);
                    Thread.Sleep(300);
                }
                while (true)
                {
                    if (_finish >= _thread) break;
                    Thread.Sleep(2000);
                }
                // release resources
                GBaseConnection.ClearAllClusterPools();
            }

            /// <summary>
            /// 主工作线程
            /// </summary>
            private void WorkThread(Object _ConnectionStringObj)
            {
                String _ConnectionString =
(String)_ConnectionStringObj;
                int times = _runTime;
                while (times!=0)
                {
                    try
                    {
                        using (GBaseConnection _ConnObject =
                        new GBaseConnection(_ConnectionString))
                        {
                            _ConnObject.Open();
                            using (GBaseCommand _CmdObject =
                            new GBaseCommand())
                            {
                                _CmdObject.CommandText = "SELECT
sleep(1)";
                                _CmdObject.Connection = _ConnObject;
```

```
                                _CmdObject. ExecuteNonQuery();
                            }
                            _ConnObject.Close();
                            _ConnObject.Dispose();
                        }
                    }
                    catch (GBaseException ex)
                    {
                        System.Diagnostics.Debug.WriteLine(ex.Messag
e);
                    }
                    Thread.Sleep(200);
                    times--;
                }
                _finish++;
            }
        }
    }
```

8.4清除连接池资源

在应用中开启连接池时，连接池会缓存连接并提供给请求使用，所以在连接池开启时会将连接资源保存起来，并在设置的缓存连接的生命周期内不会主动关闭。当应用程序关闭时，很可能没有达到生命周期规定的上限，此时在应用程序关闭时不会关闭缓存的连接。通常情况下垃圾回收器（GC）会清除不使用的资源，但有些资源不会清除，比如：连接池缓存的连接。如果这些连接在不使用时没有被清除，则会始终占用 Server 资源，导致 Server 负载过大，造成资源浪费。此时需要开发人员调用清理连接池的方法关闭这些缓存的连接。以下是 GBase ADO.NET 提供的清理连接池的方法：

1) public static void ClearPool(GBaseConnection connection)

功能：清理此连接所在集群连接池的所有缓存连接。

使用场景：当调用驱动的应用程序不在使用此集群连接池资源时调用。

调用方法：GBaseConnection.ClearPool(connection)；

connection 为连接实例。

2) public static void GBaseConnection.ClearAllGcPools()

功能：清理所有集群连接池（多集群）缓存连接。

使用场景：当调用驱动的所有应用程序不在使用集群连接池时调用。

调用方法：GBaseConnection.ClearAllGcPools()；

注：当驱动中仅存在一个集群连接池资源时，调用两种方法的结果一样。

9 GBase ADO.NET VisualStudio 插件

GBase ADO.NET VisualStudio 插件兼容 Microsoft Visual Studio 2008、2010、2012、2013、2015，并可在 SQL SERVER 2008 R2 自带的 SQL Server Business Intelligence Development Studio 中整合。支持 Microsoft Visual Studio 2008 和 2010 的插件会在 GBase ADO.NET 8.3.81.53_Build 52.1 以上版本的安装包中提供.支持 Microsoft Visual Studio 2012、2013、2015 的插件会在 GBase ADO.NET 8.3.81.53_Build 54 以上版本的安装包中提供。

使用 VisualStudio 插件前，如果需要在 VS2008 上使用，需要安装 Microsoft Visual Studio 2008 SDK。如果在 VS2010 上使用，需要安装 Microsoft Visual Studio 2010 SDK SP1。如果在 VS2012 上使用，需要安装 VSIsoShell2010。

9.1什么是 VisualStudio 插件

GBase ADO.NET VisualStudio 插件是在 Microsoft Visual Studio 开发工具中整合，使用 Microsoft VisualStudio SDK 的 DDEX (Data Designer Extensibility) 技术开发的功能集合，安装后会在开发环境中新增 GBase 数据源，方便的构建基于数据驱动的应用程序。插件将提供如下功能：

- 创建基于 GBase 数据源的数据连接；
- 使用插件管理 GBase 数据库（表、视图、存储过程、函数、UDF）；
- GBase SQL 编辑器（创建/保存 GBase SQL Script）

9.2创建基于 GBase 数据源的数据连接

此章节主要介绍如何在 Visual Studio 开发工具的“服务器资源管理器”窗口中创建基于 GBase 数据源的数据连接。

如果“服务器资源管理器”在 Visual Studio 开发工具打开后没有激活，可以选择“视图”菜单中的“服务器资源管理器”，如下图 9-1 所示。在数据连接节点上点击右键后即可添加新连接。

“服务器资源管理器”在英文环境下为“Server Explorer”。

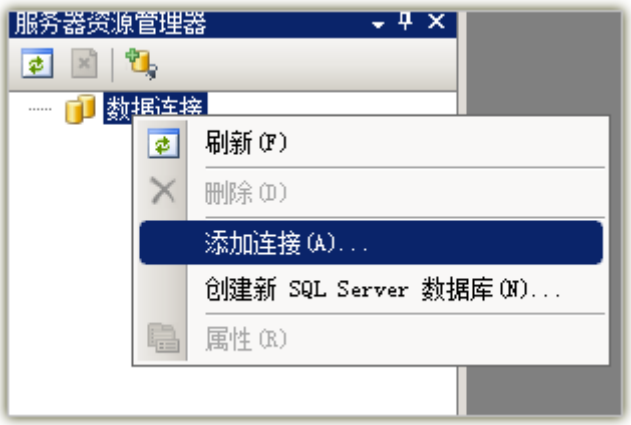


图 9-1 在服务器资源管理器中添加连接

9.2.1打开和关闭数据连接

1） 打开连接

“添加连接”菜单命令执行后，会出现“更改数据源”对话框，如下图 9-2。如果未出现“更改数据源”对话框，则需要在“添加连接”视图中点击“更改”按钮后会出现“更改数据源”对话框，如下图 9-3 所示。

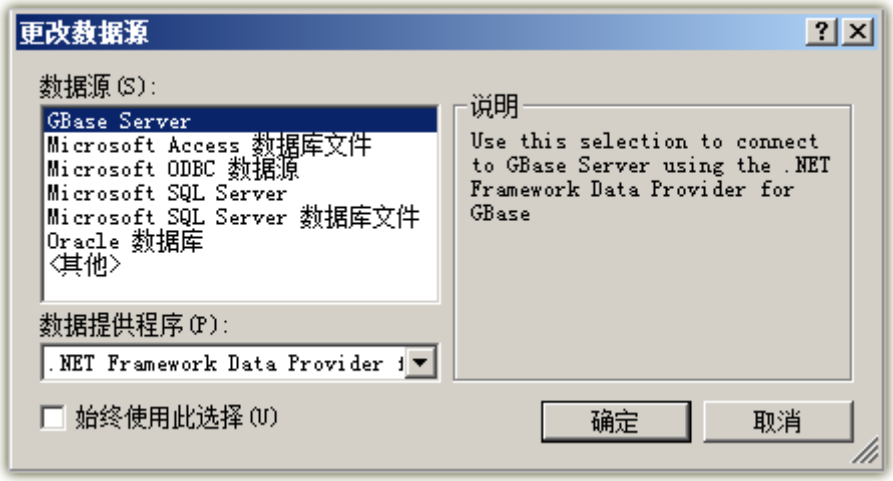


图 9-2 更改数据源对话框

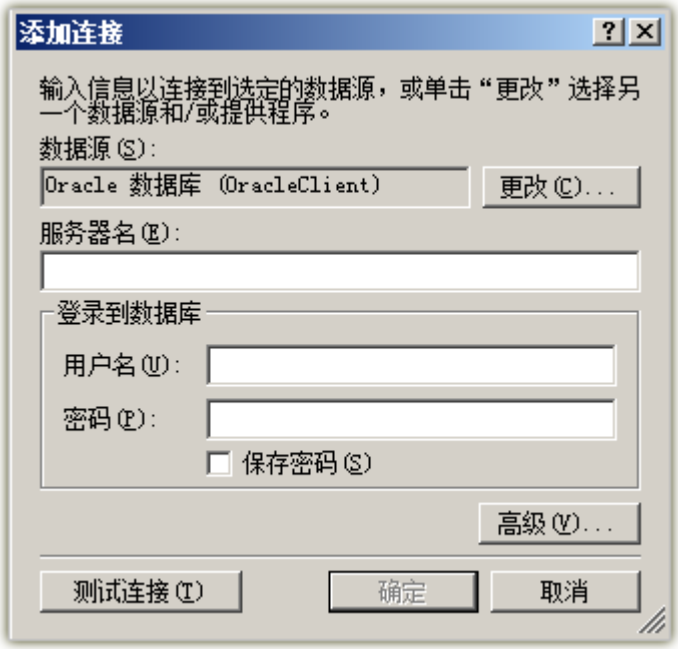


图 9-3 添加连接（示例数据源）对话框

从列表中选择“GBase Server”项目后，则会弹出“添加连接”对话框，如下图 9-4。如果验证信息输入正确并且点击确定按钮后会自动打开数据连接，并且会在“数据连接”节点下新增一个子节点，以 IP(Database)格式命名，如：192.168.11.45(test)。此后的操作都会在此子节点上进行。如：创建表、视图、

存储过程、函数、UDF 等。

图 9-4 添加 GBase 数据源连接对话框

在“添加连接”对话框上输入所需信息的步骤是为新建连接生成连接字符串，如果需要添加额外的连接参数，可以点击高级按钮后在“高级属性”窗口中选择，如下图 9-5 所示。每个参数的含义请参考 5.1.1 连接字符串章节。

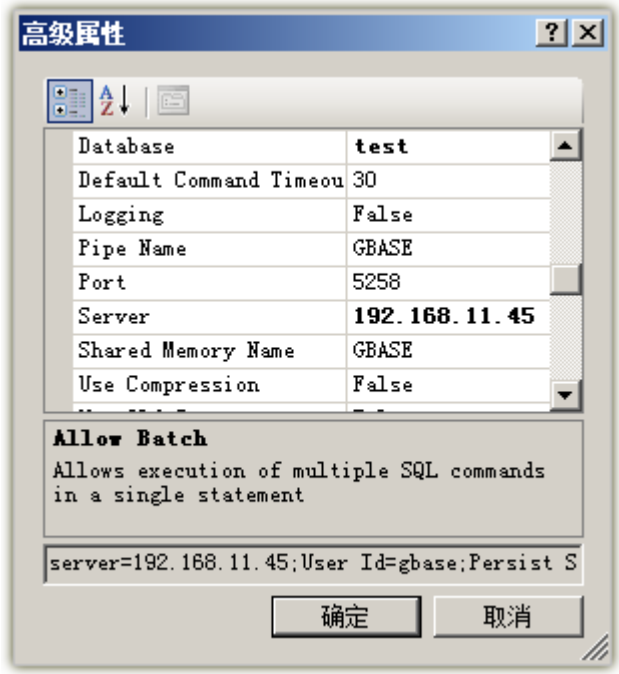


图 9-5 高级属性对话框

2） 关闭连接

在创建的子节点上点击右键后会弹出菜单，选择“关闭连接”命令后，当前连接即被关闭。此时节点前图标会变为 代表此连接已关闭。“关闭连接”命令同样可以在选择节点后，Visual Studio 出现的“数据”菜单中选择。如下图 9-6 所示。

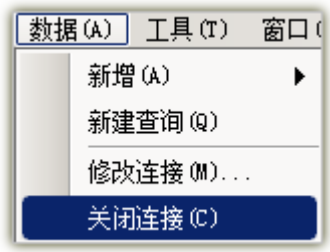


图 9-6 Visual Studio 数据菜单项

9.2.2查看连接信息

在服务器资源管理器中新增了 GBase 数据源节点后，如果要查看当前节点使用的提供程序、版本、类型（厂商）、连接字符串、状态，可在点击创建的子节点后查看“属性”对话框。如下图 9-7 所示。如果“属性”窗口没有显示在当前视图，可以从 Visual Studio 的“视图”菜单中选择“属性窗口”。

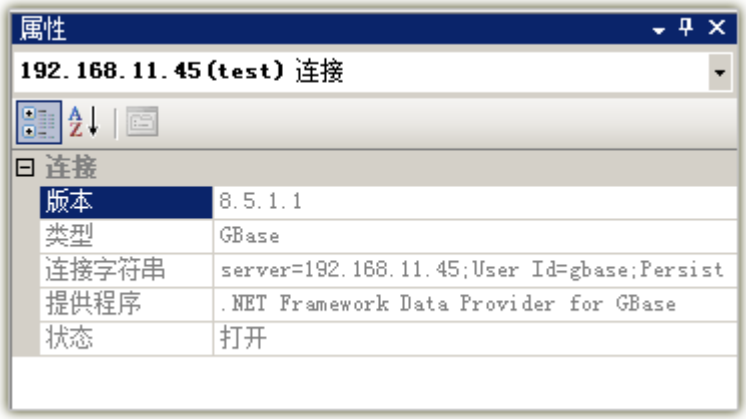


图 9-7 属性窗口

9.2.3保存数据连接

在服务器资源管理器中新增了 GBase 数据源节点后，此节点如果不手动删除，则会自动保存，并且在下次打开 Visual Studio 时会同样显示在此位置。如下图 9-8。

图 9-8 新增基于 GBase 数据源的数据连接

9.3 使用插件管理 GBase 数据库

插件提供了丰富的管理 GBase 数据库的功能，如：创建表\视图、修改表\视图定义、删除表\视图，创建存储过程\函数、修改存储过程\函数定义、删除存储过程\函数，创建 UDF、删除 UDF 等功能，下面的章节将逐一介绍。

9.3.1管理表

包括创建表，修改表定义和删除表功能。这些操作在 GBase 数据源节点展开后的 Tables 节点上进行。

9.3.1.1 创建表

在 Tables 节点上点击右键选择“创建表”命令或者执行 Visual Studio 的“数据”菜单的“新增”子菜单下的“表”命令，则会显示列定义视图（图 9-9）和表定义窗口（图 9-10）。列定义视图分为上下两部分，上面部分输入列名、数据类型等基本信息，下面部分输入列的更详细定义。

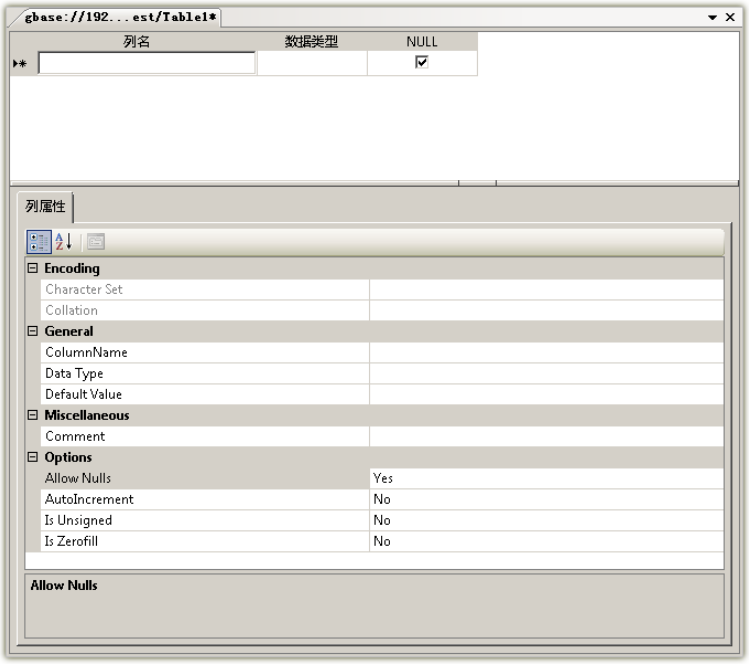

gbase://192...est/Table1*
列名
数据类型
NULL
列属性
Encoding
Character Set
Collation
General
ColumnName
Data Type
Default Value
Miscellaneous
Comment
Options
Allow Nulls
Yes
AutoIncrement
No
Is Unsigned
No
Is Zerofill
No
Allow Nulls

图 9-9 列定义视图

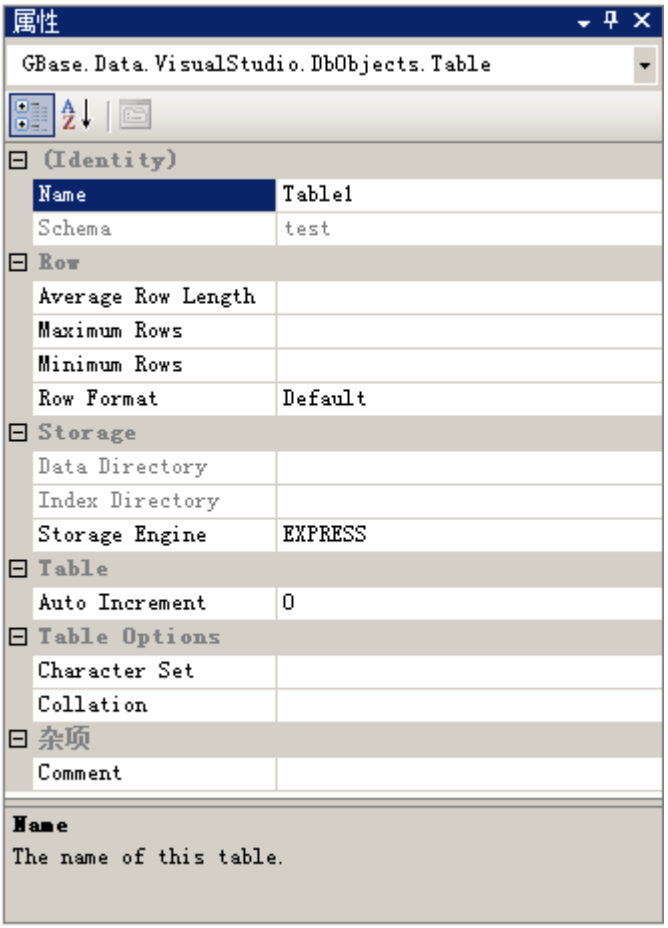


图 9-10 表定义窗口

定义好列信息和表信息后，使用“Ctrl + S”保存定义，或者点击 Visual Studio 的“标准”工具条上的按钮保存。表保存后，会在 Tables 节点下新增一个表节点，以表名命名。

9.3.1.2 修改表定义

右键点击新建表节点后，选择“设计”命令即可编辑表及列。编辑后如果想要保存变更的脚本，可使用 Visual Studio 的“表设计器”菜单的“生成更改脚本”命令，见下图 9-11。编辑完成后，使用“Ctrl + S”保存定义，或者

点击 Visual Studio 的“标准”工具条上的  按钮保存。

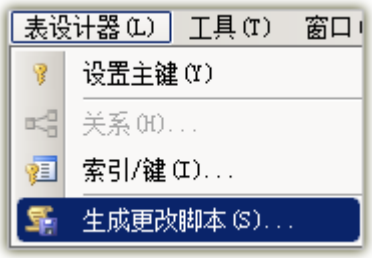


图 9-11 Visual Studio 表设计器菜单

在“保存改变的脚本”对话框中可以看到当前修改生成的脚本，并可以点击确定按钮后保存到 SQL 文件。见下图 9-12。

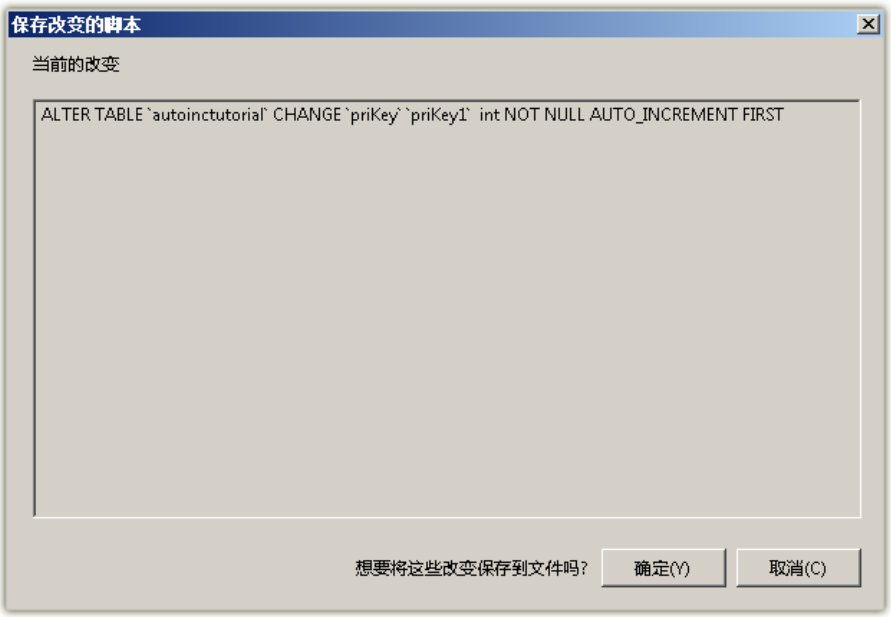


图 9-12 保存改变的脚本对话框

在点击表节点前面“+”号后显示该表定义的列，以列名显示。

9.3.1.3 删除表

在新建表节点上点击右键，执行“删除”命令，可执行删除表操作。如下

图 9-13 所示。

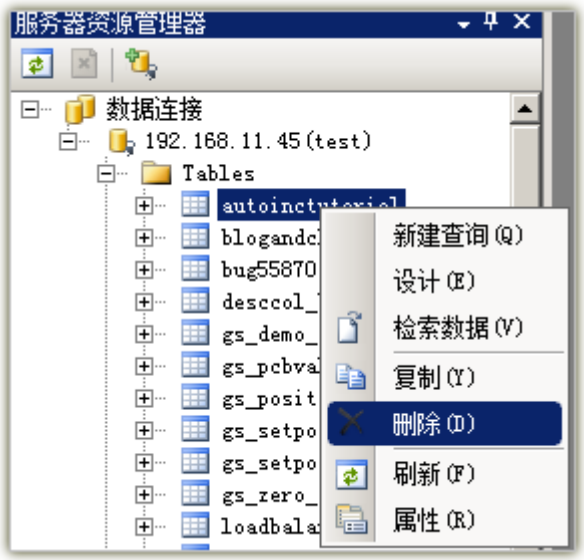


图 9-13 执行删除命令

9.3.2管理视图

包括创建视图，修改视图定义和删除视图功能。这些操作在 GBase 数据源节点展开后的 Views 节点上进行。

9.3.2.1 创建视图

在 Views 节点上点击右键选择“创建视图”命令或者执行 Visual Studio 的“数据”菜单的“新增”子菜单下的“视图”命令，则会显示视图定义模板（图 9-14），输入视图定义的 SQL 语句后，使用“Ctrl + S”保存定义，或者点击 Visual Studio 的“标准”工具条上的按钮保存。视图保存后会在 Views 节点下新增一个视图子节点，以视图名命名。

图 9-14 视图定义模板

9.3.2.2 修改视图定义

右键点击新建视图节点后，选择“修改视图”命令即可编辑该视图 SQL 定义语句。编辑完成后，使用“Ctrl + S”保存定义，或者点击 Visual Studio 的“标准”工具条上的按钮保存。

9.3.2.3 删除视图

在新建视图节点上点击右键，执行“删除”命令，可执行删除视图操作。如下图 9-15 所示。

图 9-15 执行删除命令

9.3.3管理存储过程

包括创建存储过程，修改存储过程定义和删除存储过程功能。这些操作在 GBase 数据源节点展开后的 Stored Procedures 节点上进行。

9.3.3.1 创建存储过程

在 Stored Procedures 节点上点击右键选择“创建存储过程”命令或者执行 Visual Studio 的“数据”菜单的“新增”子菜单下的“存储过程”命令，则会显示存储过程定义模板（图 9-16），输入存储过程定义的 SQL 语句后，使用“Ctrl + S”保存定义，或者点击 Visual Studio 的“标准”工具条上的按钮保存。存储过程保存后会在 Stored Procedures 节点下新增一个存储过程子节点，以存储过程名命名。

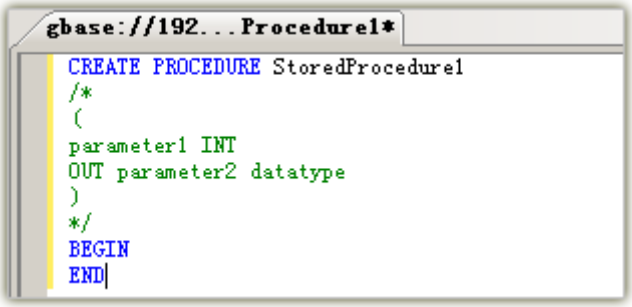


图 9-16 存储过程定义模板

9.3.3.2 修改存储过程定义

右键点击新建存储过程节点后，选择“修改存储过程/函数”命令即可编辑该存储过程 SQL 定义语句。编辑完成后，使用“Ctrl + S”保存定义，或者点击 Visual Studio 的“标准”工具条上的按钮保存。

9.3.3.3 删除存储过程

在新建存储过程节点上点击右键，执行“删除”命令，可执行删除存储过程操作。如下图 9-17 所示。

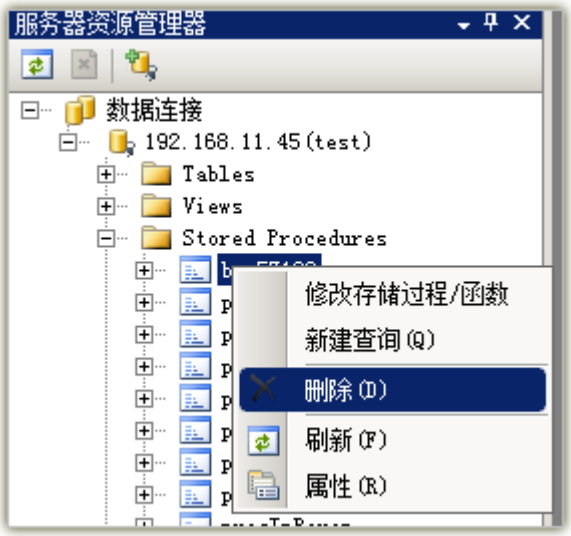


图 9-17 执行删除命令

9.3.4管理函数

包括创建函数，修改函数过程定义和删除函数功能。这些操作在 GBase 数据源节点展开后的 Stored Functions 节点上进行。

9.3.4.1 创建函数

在 Stored Functions 节点上点击右键选择“创建函数”命令或者执行 Visual Studio 的“数据”菜单的“新增”子菜单下的“函数”命令，则会显示函数定义模板（图 9-18），输入函数定义的 SQL 语句后，使用“Ctrl + S”保存定义，或者点击 Visual Studio 的“标准”工具条上的按钮保存。函数保存后会在 Stored Functions 节点下新增一个函数子节点，以函数名命名。

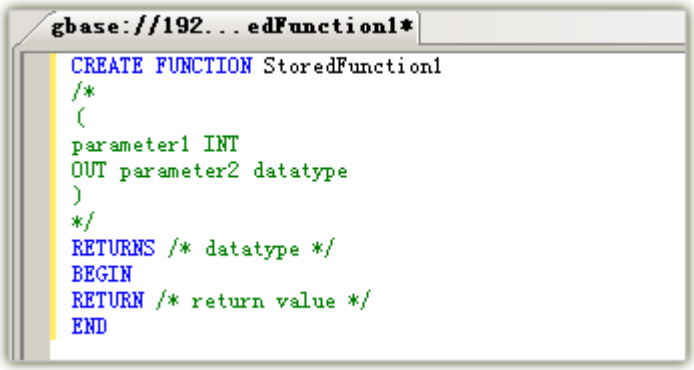


图 9-18 函数定义模板

9.3.4.2 修改函数定义

右键点击新建函数节点后，选择“修改存储过程/函数”命令即可编辑该函数 SQL 定义语句。编辑完成后，使用“Ctrl + S”保存定义，或者点击 Visual Studio 的“标准”工具条上的 按钮保存。

9.3.4.3 删除函数

在新建函数节点上点击右键，执行“删除”命令，可执行删除函数操作。如下图 9-19 所示。

图 9-19 执行删除命令

9.3.5管理用户定义函数(UDF)

包括创建 UDF，删除 UDF 功能。这些操作在 GBase 数据源节点展开后的 UDFs 节点上进行。

9.3.5.1 什么是 UDF

UDF (User-Defined Function)，用户自定义函数，通过添加新函数而对 GBase Server 功能进行扩充，通常使用 C/C++根据 GBase Server 提供的 UDF 编写规则编写成 (Windows 下的 DLL 或 Linux 下的 SO) 二进制文件实现。应用时使用类似“select my_udf()”SQL 语法。有关 UDF 函数 my_udf 如何编写请参考 GBase 8a 及 8a 集群相关手册。

GBase 支持很多内建函数，如：字符串函数、数值函数、日期和时间函数、OLAP 函数等，还可以通过使用 SQL 语句的 Create Function 定义函数。UDF 为用户提供了一种更高效的方式来创建函数。与内建函数类似，有参数也有输出。分为普通函数和聚集函数两种类型，前者能够针对每一行数据进行处理，后者

则用于处理 Group By 这样的情况。

为什么用 UDF？

GBase 本身提供了大量的函数，并且也支持定义函数，为什么我们还需要 UDF 呢？主要有以下几点原因：

1） 比 Function 具有更高的执行效率，并支持聚集函数；

2） 相比修改代码增加函数，更加方便简单

当然 UDF 也是有缺点的，这是因为 UDF 会加载到 GBase 内核服务的内存空间中，不谨慎使用内存很容易导致 GBase 内核服务 Crash 掉。

9.3.5.2 创建 UDF

在 UDFs 节点上点击右键选择“创建 UDF”命令或者执行 Visual Studio 的“数据”菜单的“新增”子菜单下的“UDF”命令，则会显示“创建新 UDF”窗口（图 9-20），输入相关信息后，点击确认保存。

图 9-20 创建新 UDF 窗口

在“创建新 UDF”窗口中名字处输入的自定义函数名需要在 UDFExtend.so 文件中定义。在点击确定后，GBase Server 会检查 UDFExtend.so 中是否定义了名为“FuncName”函数。如果定义正确会在 UDFs 节点下新增子节点，否则会弹出错误提示。

9.3.5.3 删除 UDF

可在 9.4.2 执行 SQL 语句章节中通过 GBase SQL 编辑器使用 Drop 语句执行删除 UDF。

9.3.6查询

包括新建查询（适合执行多表联合查询）和检索数据（单表数据查询）功能。

9.3.6.1 新建查询

在 GBase 数据源根节点下的任意子节点上点击右键选择“创建查询”命令或者执行 Visual Studio 的“数据”菜单的“新建查询”子命令，则会显示图 9-21 所示视图。

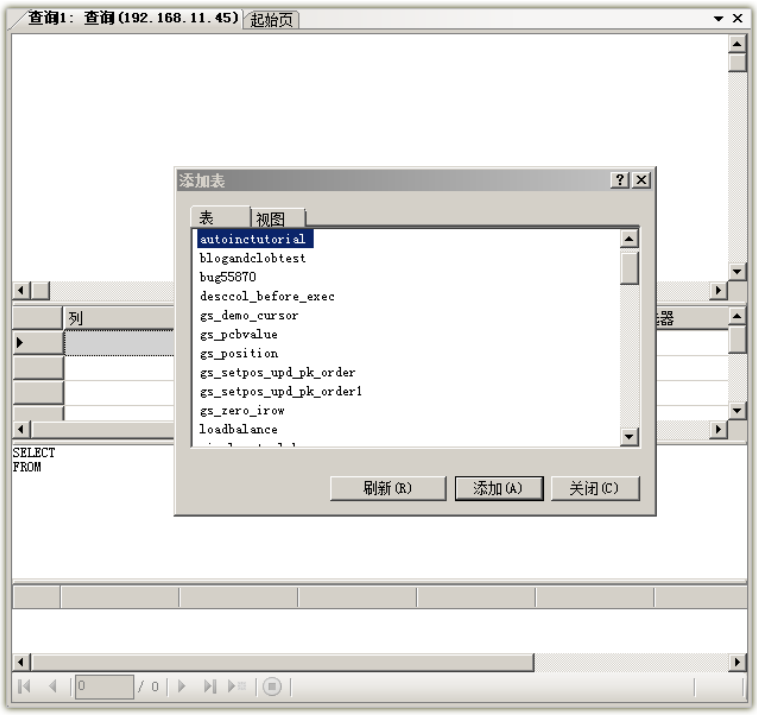


图 9-21 新建查询视图

从上到下依次为“关系图”、“条件”、“SQL”、“结果”，如果需要关闭这些窗格中的某一个，可点击“查询设计器”工具条的对应按钮或者选择 Visual Studio 的“查询分析器”菜单的“窗格”子菜单中对应的菜单命令，如下图 9-22 所示。

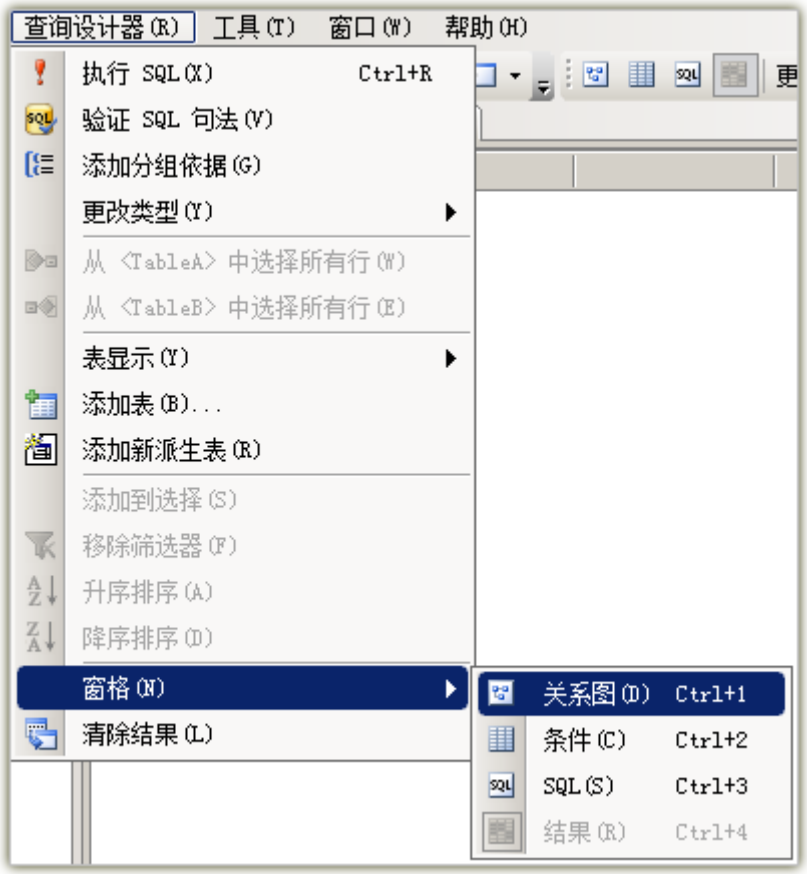


图 9-22 查询设计器工具条

如果“查询设计器”工具条没有显示在当前环境中显示，则可以通过选中 Visual Studio 的“视图”菜单的“工具栏”子菜单的“查询设计器”命令调出工具条。如下图 9-23 所示。

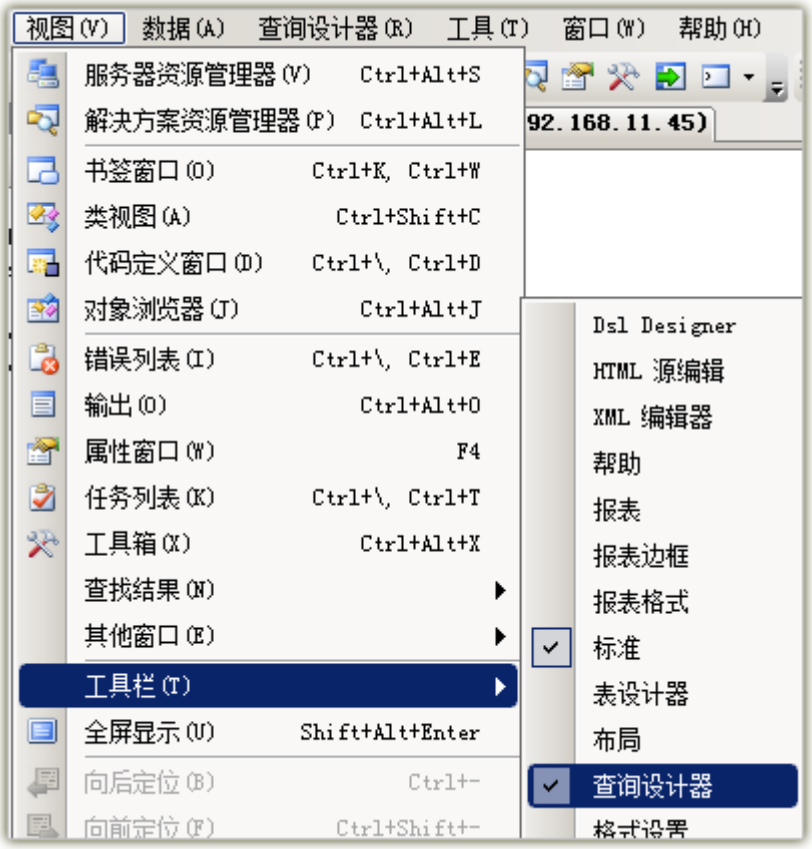


图 9-23 Visual Studio 的视图菜单

从“添加表”窗口可以添加要查询的表和视图，选择“添加”按钮后会将选择的表和视图添加到“关系图”窗口中。完成后点击“查询设计器”的“执行 SQL”按钮 后从“结果”窗口查看结果，如下图 9-24 显示。

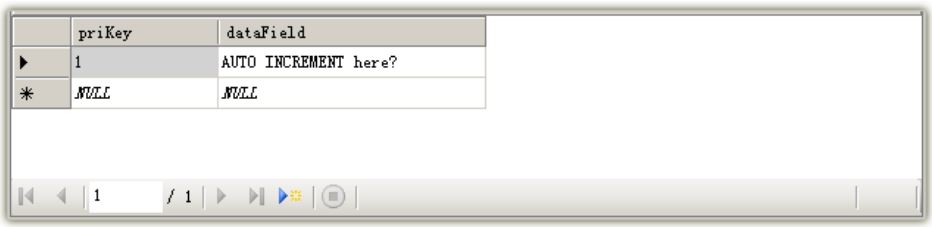


图 9-24 执行 SQL 后的结果窗口

9.3.6.2 检索数据

在 GBase 数据源根节点下的 Tables 和 Views 子节点上点击右键选择“检索数据”命令或者执行 Visual Studio 的“数据”菜单的“检索数据”命令，则会显示图 9-25 所示。

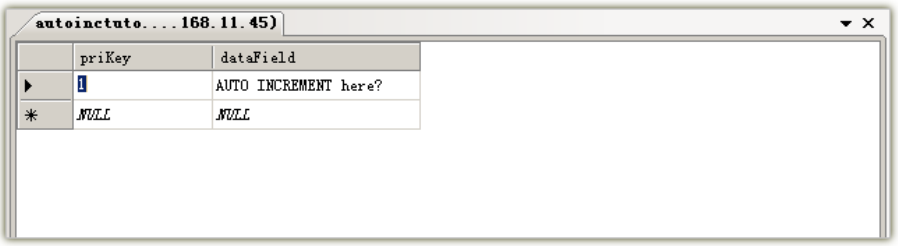


图 9-25 检索数据后的结果窗口

9.4GBase SQL 编辑器

此章节包括如何使用 Visual Studio 新建 GBase 脚本文件，使用 GBase SQL 编辑器编辑脚本文件、执行 SQL 语句和持久化脚本文件。

9.4.1什么是 GBase SQL 编辑器

GBase SQL 编辑器是 VisualStudio 插件中提供编辑、保存 SQL 语法的 SQL 脚本编辑器。可在此脚本文件中使用 GBase 8a SQL 手册中支持的 DML/DDL 语法。

VisualStudio 插件安装后，会自动安装“GBase SQL 编辑器”和 GBase SQL Script 模板。GBase SQL Script 模板安装后，会在 Visual Studio 的新建文件对话框中新增一个名为 GBase 的类别，点选后即可在右边的“Visual Studio 已安装的模板”窗口中看到 GBase SQL Script 模板。如图 9-26 所示。

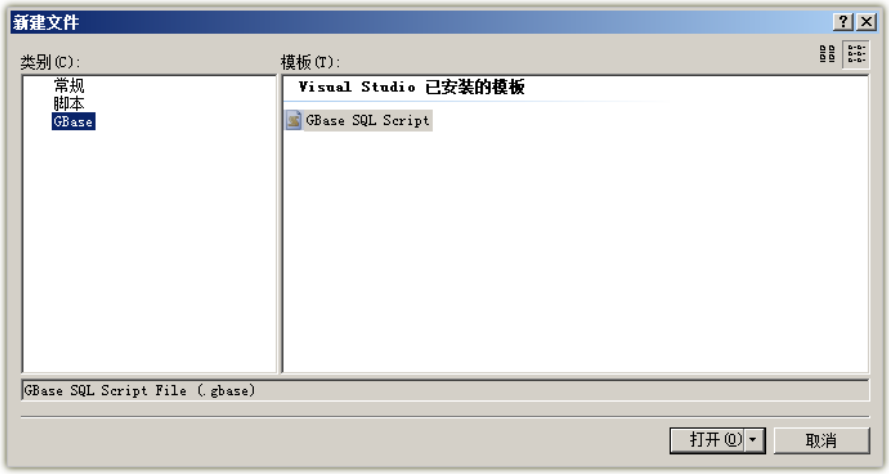


图 9-26 Visual Studio 的新建文件对话框

9.4.2新建GBase脚本文件

可以使用随 VisualStudio 插件自动安装的 GBase SQL Script 模板新建 GBase 脚本文件，新建文件后会出现 GBase SQL 编辑器界面。如下图 9-27 显示。

图 9-27 GBase SQL 编辑器

9.4.3执行SQL语句

在 GBase SQL 编辑器界面的工具条上使用“Connect to GBase …”按钮 建立连接。在“Connect to GBase”对话框中输入必要信息后，激活连接。如果需要更多的连接参数，可点击界面上的“高级”按钮切换窗口样式。如下图 9-28 所示。

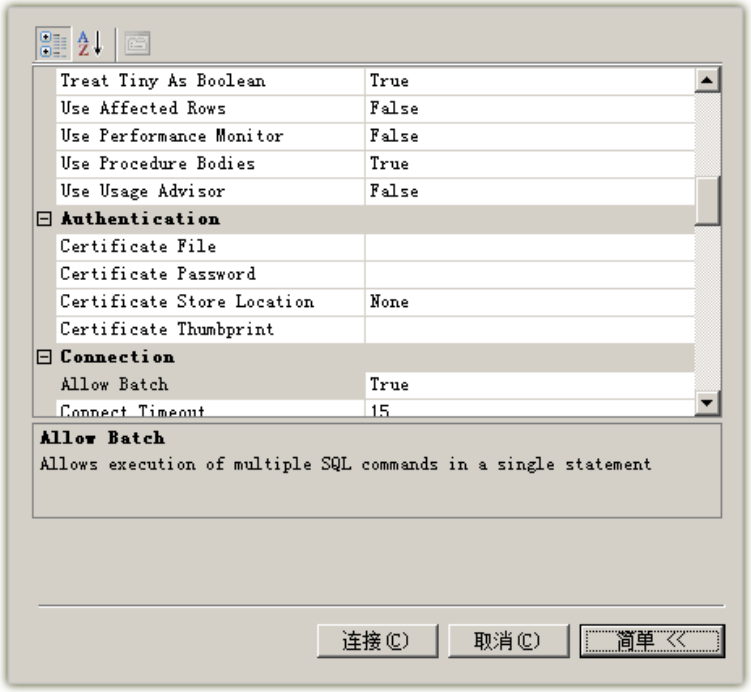


图 9-28 Connect to GBase 对话框

如果连接信息输入正确并且激活后，工具条上的“Disconnect from GBase”按钮和“Run SQL”按钮激活，并且当前连接的主要信息会显示到工具条上。如 9-29 所示。

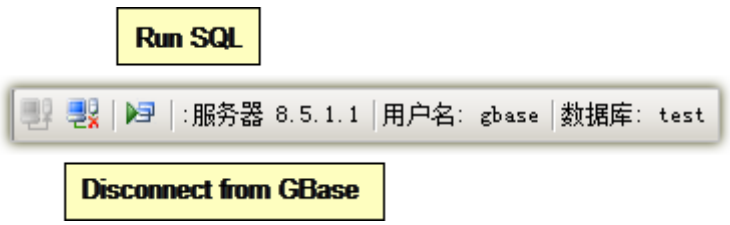


图 9-29 GBase SQL 编辑器工具条

此时可以在 GBase SQL 编辑器中输入 SQL 语句并使用“Run SQL”按钮执行。如下图 9-30 显示，执行后会显示结果窗口。

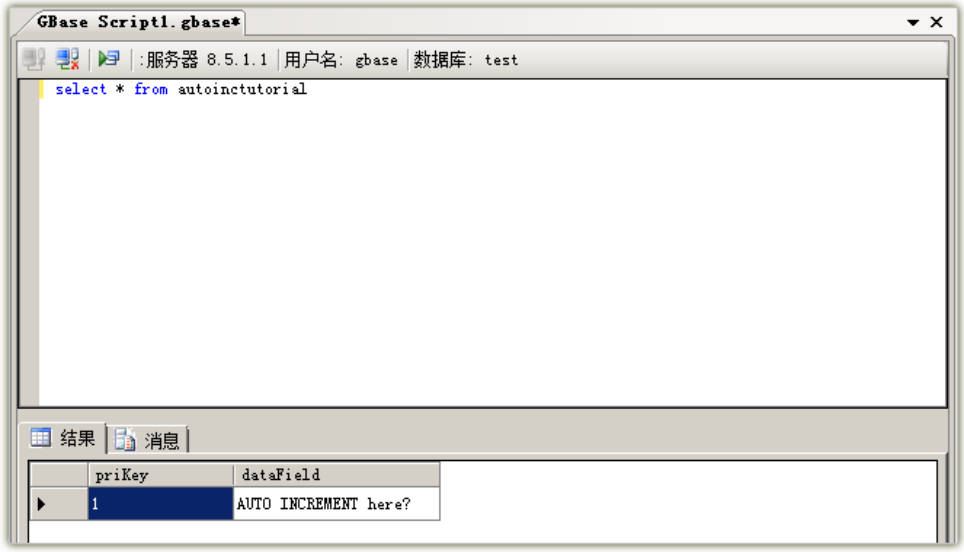


图 9-30 GBase SQL 编辑器

9.4.4持久化脚本文件

当前新建的脚本文件“GBase Script1.gbase”的内容在编辑后会在标题的名字后面显示“*”，此时可使用“Ctrl + S”保存定义，或者点击 Visual Studio 的“标准”工具条上的 按钮保存。

9.5使用 GBase 数据源

此章节介绍 GBase ADO.NET VisualStudio 插件在实际应用中使用的典型场景和在场景中的使用步骤。

9.5.1在微软商业智能(BI)项目中使用 GBase 数据源

微软商业智能（BI）项目是微软公司应对目前的数据挖掘领域的解决方案。在安装 SQL SERVER 2008 R2 版本后会自动安装“SQL Server Business Intelligence Development Studio”,以下简称“SSBIDS”(SSBIDS 就是 Microsoft

Visual Studio 2008 版本编译器)。并内建“Analysis Services 项目”和“Integration Services 项目”等。如果想在 VS2010、VS2012 以及 VS2013 中使用“SSBIDS”,则需要安装 SQL SERVER 2012 以及相应的商业智能安装包。此节将重点介绍在“Analysis Services 项目”中如何使用建立 GBase 数据源。下面是操作步骤。

- 使用“SSBIDS”新建“Analysis Services 项目”

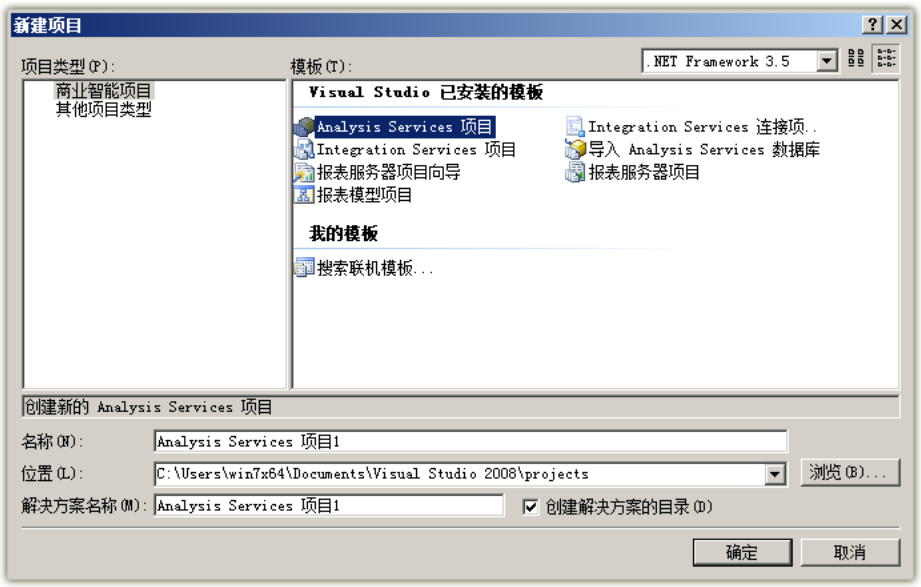


图 9-31 Visual Studio 的新建项目对话框

- 在解决方案管理器的“Analysis Services 项目 1”节点的“数据源”节点位置建立新建 GBase 数据源。

图 9-32 在 Analysis Services 项目 1 中新建数据源

- 在数据源向导中新建数据连接，在提供程序下拉列表中的.NET 提供程序中选择“GBase Data Provider”，在出现的视图界面中输入正确信息后点击确定后完成。并随着向导一步步操作，最后点击完成。在建立 GBase 数据源后，一定要选中“Sql Server Mode”连接串选项。否则会在“新建数据源视图”的最后一步点击确定时弹出错误对话框。

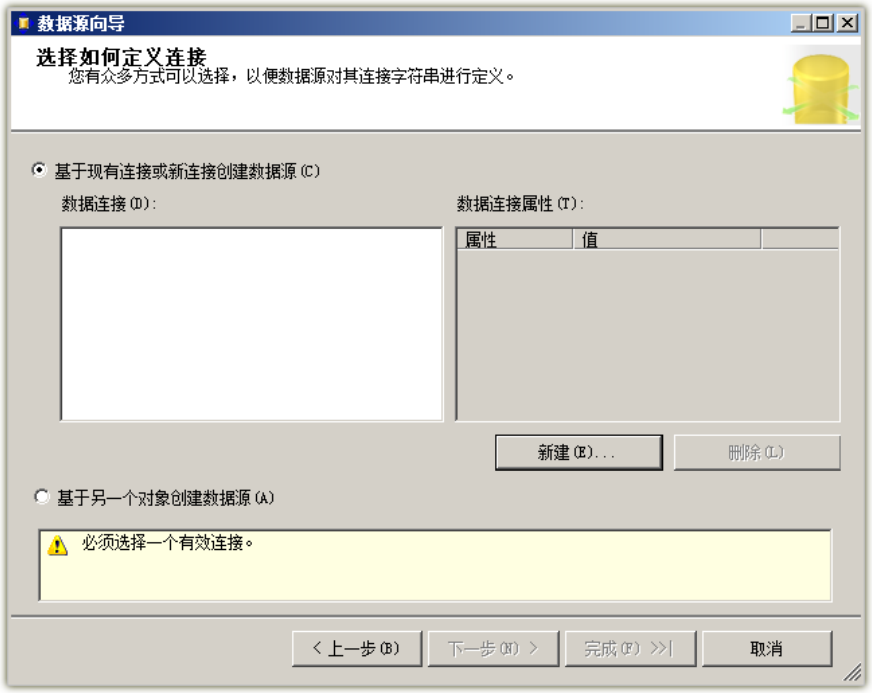


图 9-33 数据源向导窗口

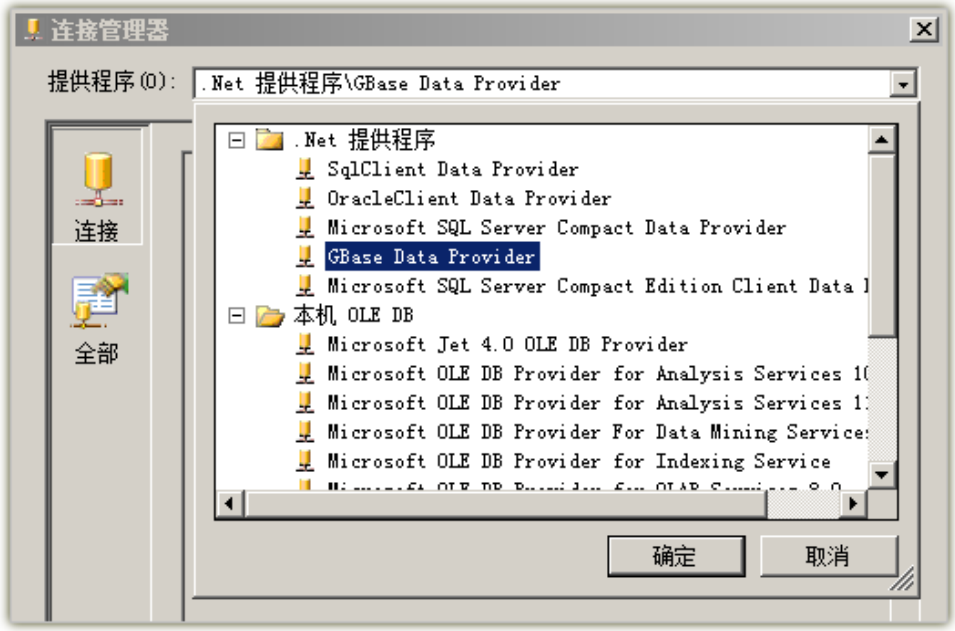


图 9-34 选择 GBase Data Provider 数据源

图 9-35 在 GBase Data Provider 视图输入验证信息

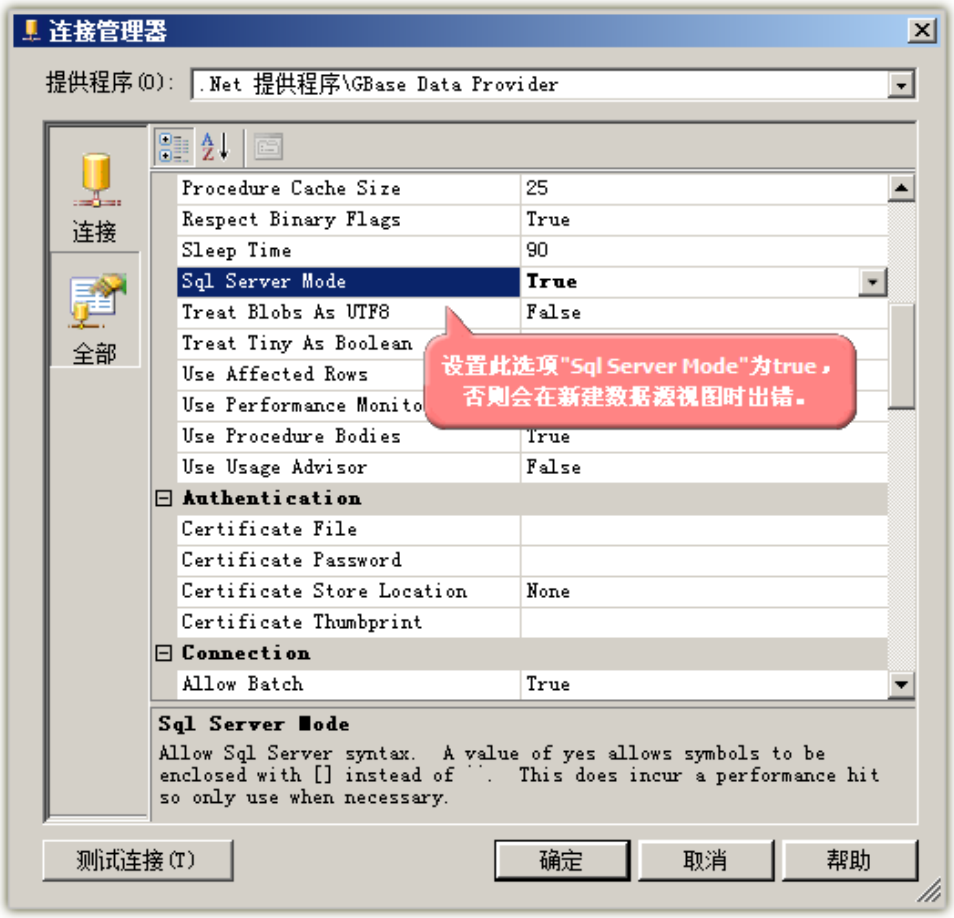


图 9-36 GBase Data Provider 视图全部属性

图 9-37 选择 GBase Data Provider 数据连接

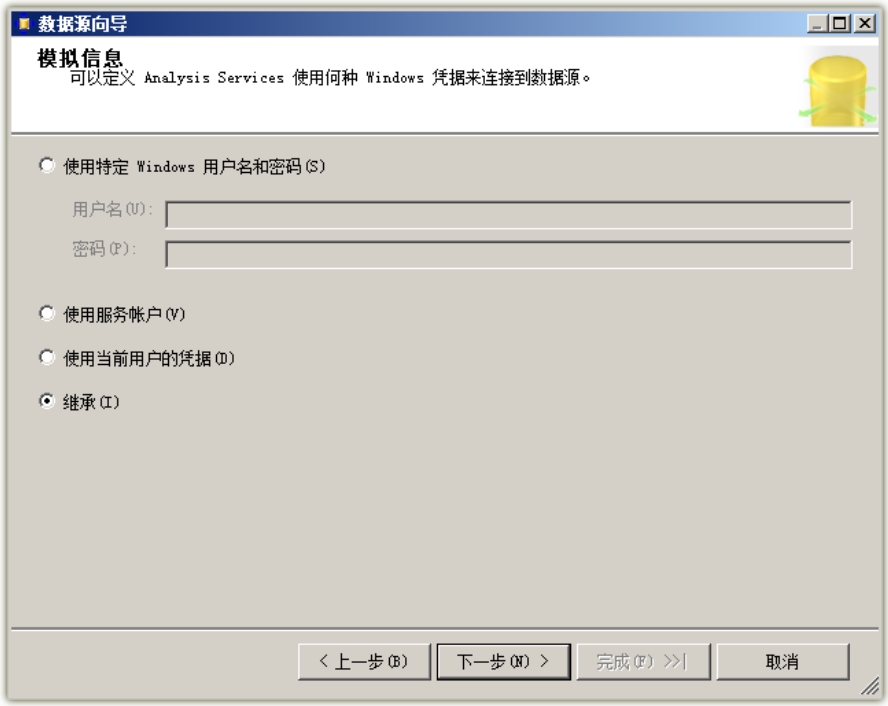


图 9-38 数据源向导窗口 1

图 9-39 数据源向导窗口 2

- 在“Analysis Services 项目 1”的“数据源视图”上新建数据源视图。

图 9-40 Analysis Services 项目 1 中新建数据源视图

- 在“数据源视图向导”窗口中选择新建的 GBase 数据源，新建视图。

图 9-41 数据源视图向导窗口 1

图 9-42 数据源视图向导窗口 2

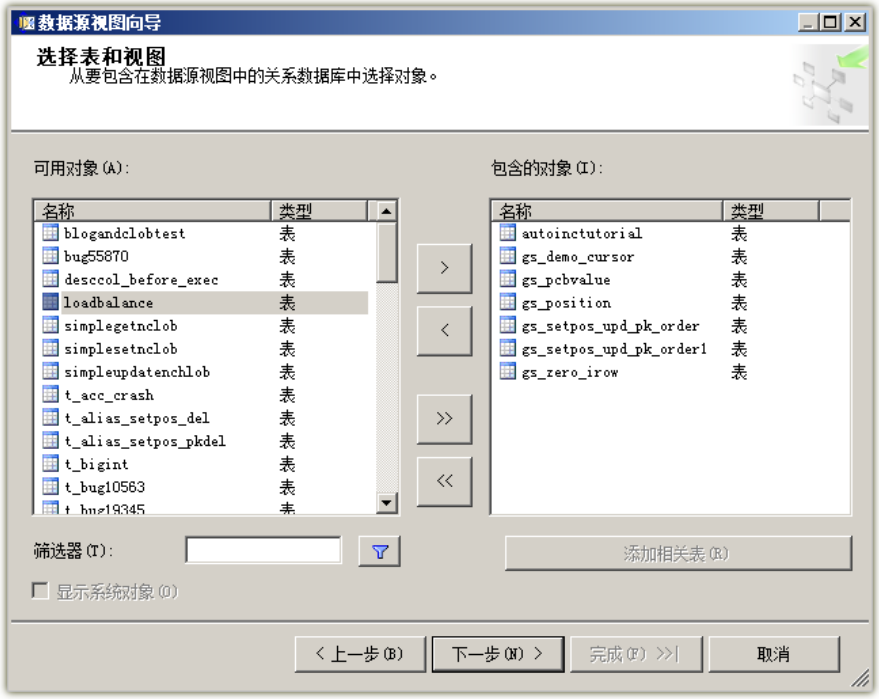


图 9-43 数据源视图向导窗口 3

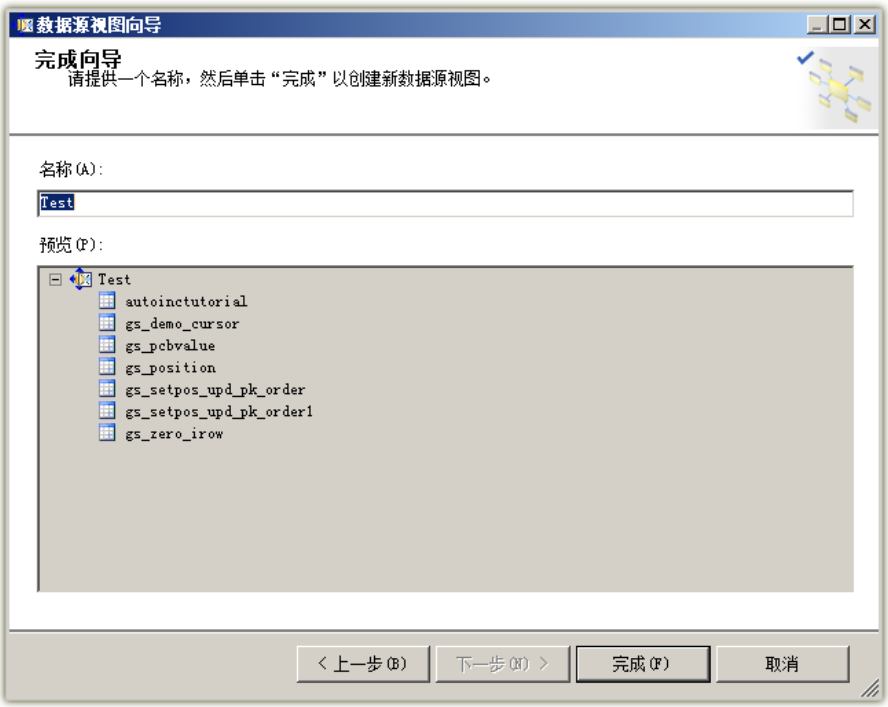


图 9-44 数据源视图向导窗口 4

- 完成新建数据源视图

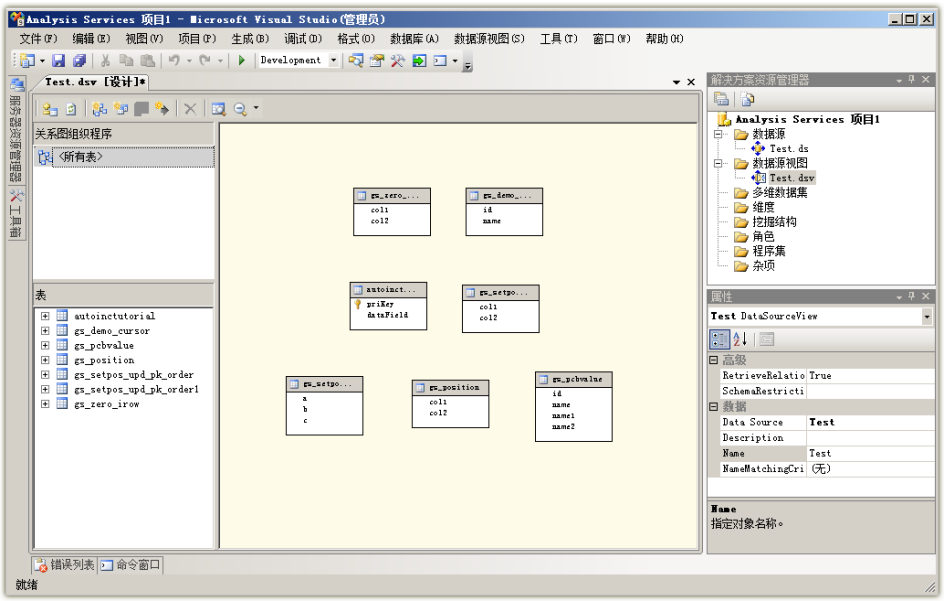


图 9-45 新建后的数据源视图

10 GBase ADO.NET EntityFramework 实体框架支持

GBase ADO.NET 驱动支持 EntityFramework 实体框架。

实体框架，可以理解成微软的一个 ORM 产品，用于支持开发人员通过对概念性应用程序模型编程（而不是直接对关系存储架构编程）来创建数据访问应用程序，目标是降低面向数据的应用程序所需的代码量并减轻维护工作。

本章主要简介 GBase ADO.NET 支持 EntityFramework 以及相应的三种模式的使用，GBase ADO.NET 8.3.81.53_Build 54 以上版本的安装包中提供。

10.1 DatabaseFirst 模式

在项目中添加 EDMX 的实体模型（从数据库生成），步骤如下：

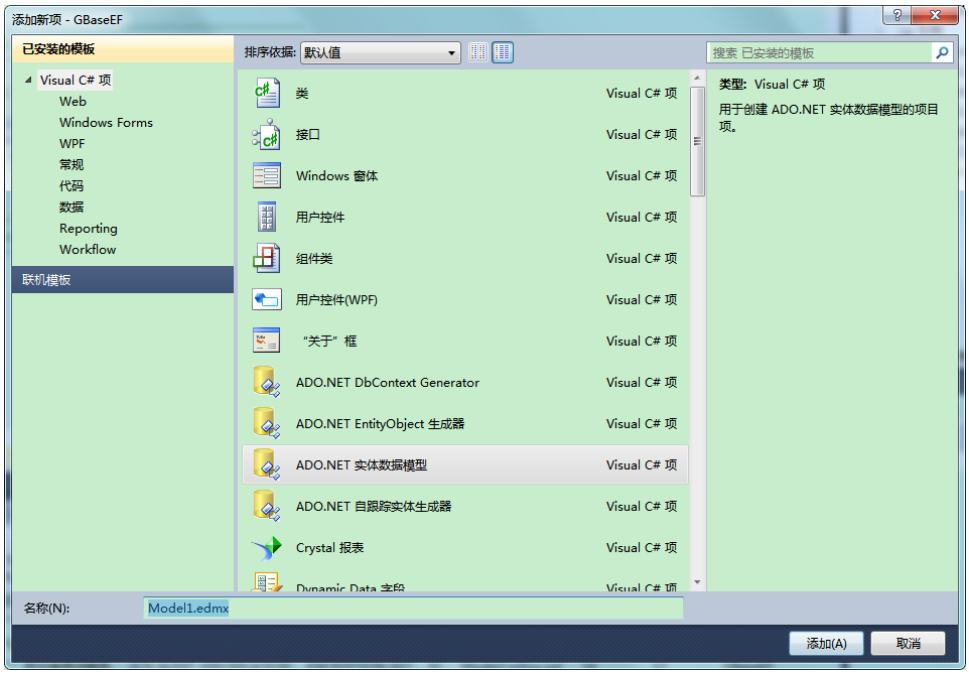


图 10-1 添加 ADO.NET 实体数据模型

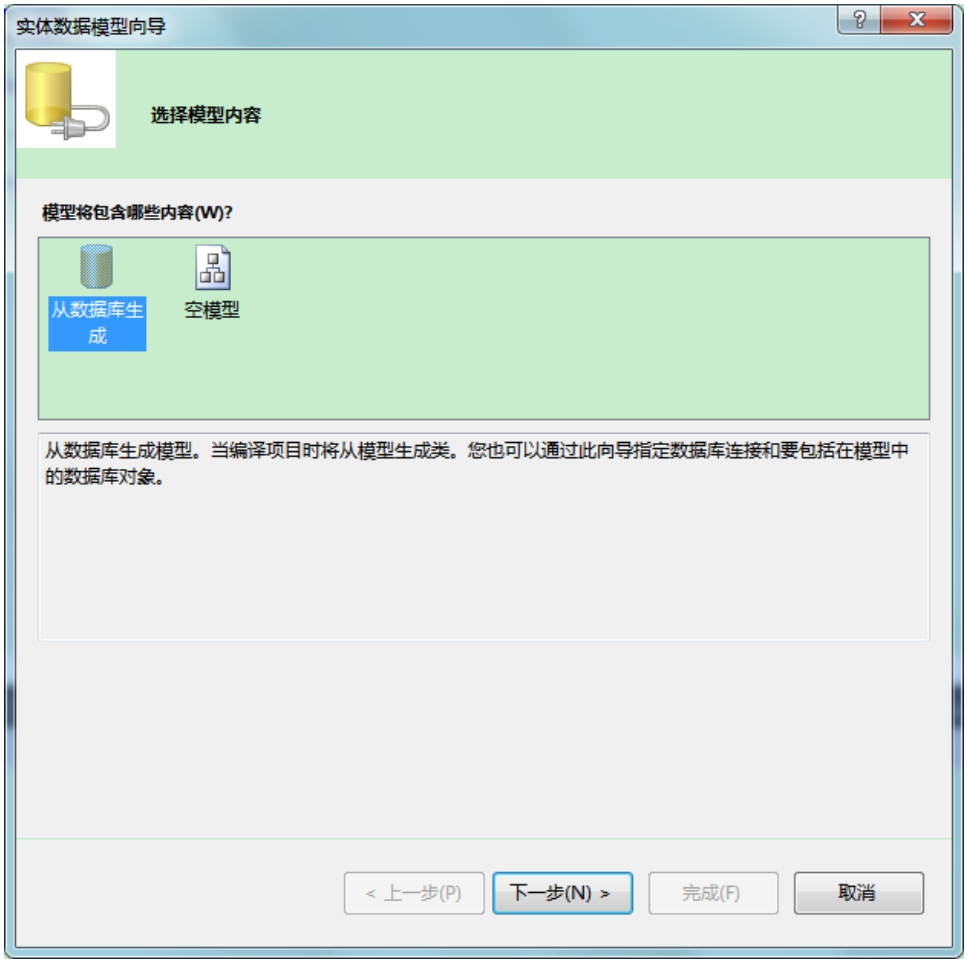

实体数据模型向导
选择模型内容
模型将包含哪些内容(W)?
从数据库生成
空模型
从数据库生成模型。当编译项目时将从模型生成类。您也可以通过此向导指定数据库连接和要包括在模型中的数据库对象。
< 上一步(P)
下一步(N) >
完成(F)
取消

图 10-2 从数据库生成

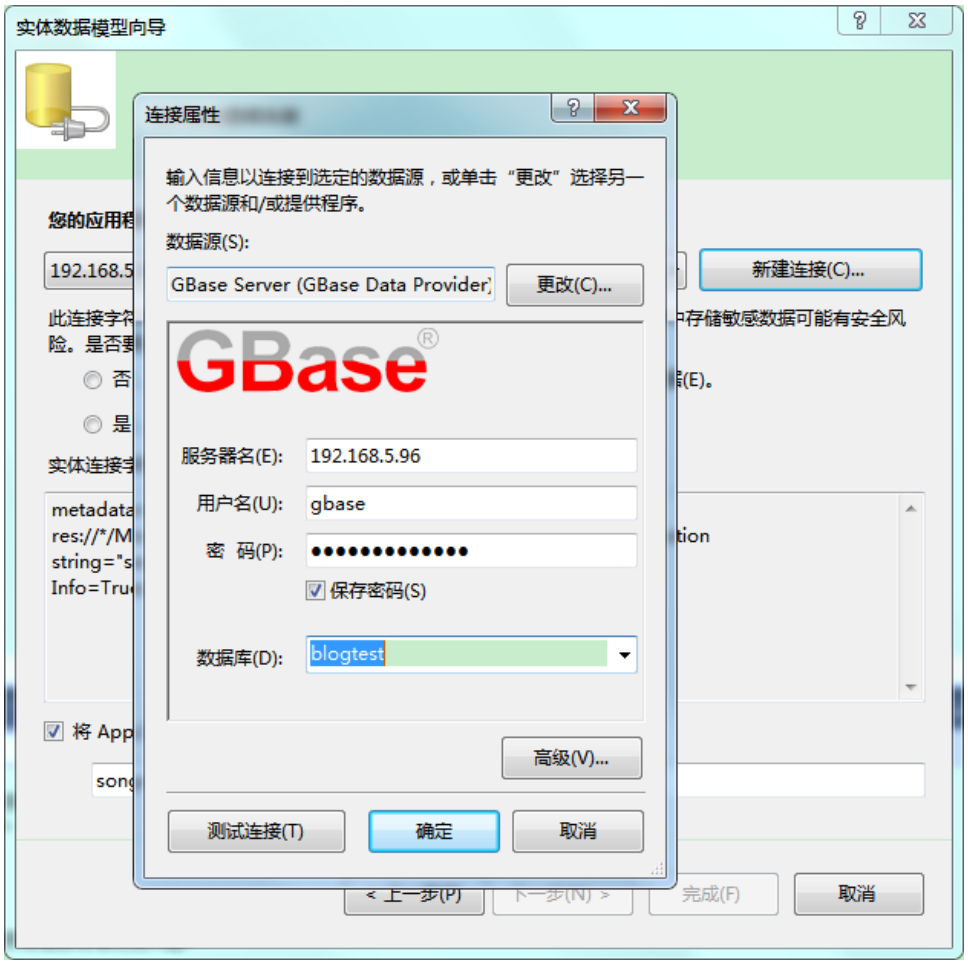


图 10-3 新建数据库连接

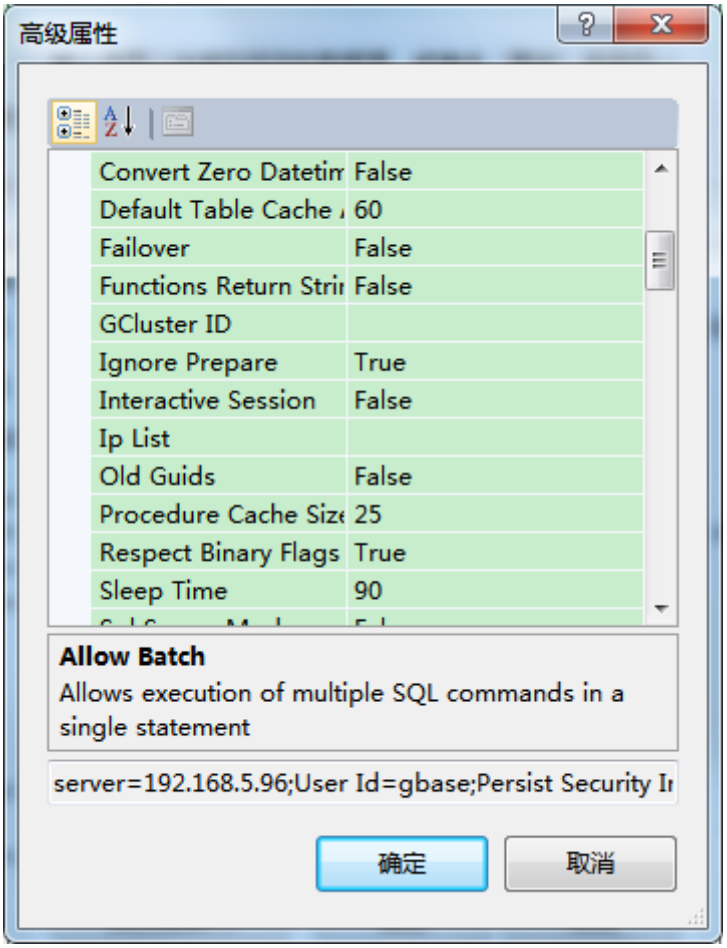


图 10-4 连接高级属性设置

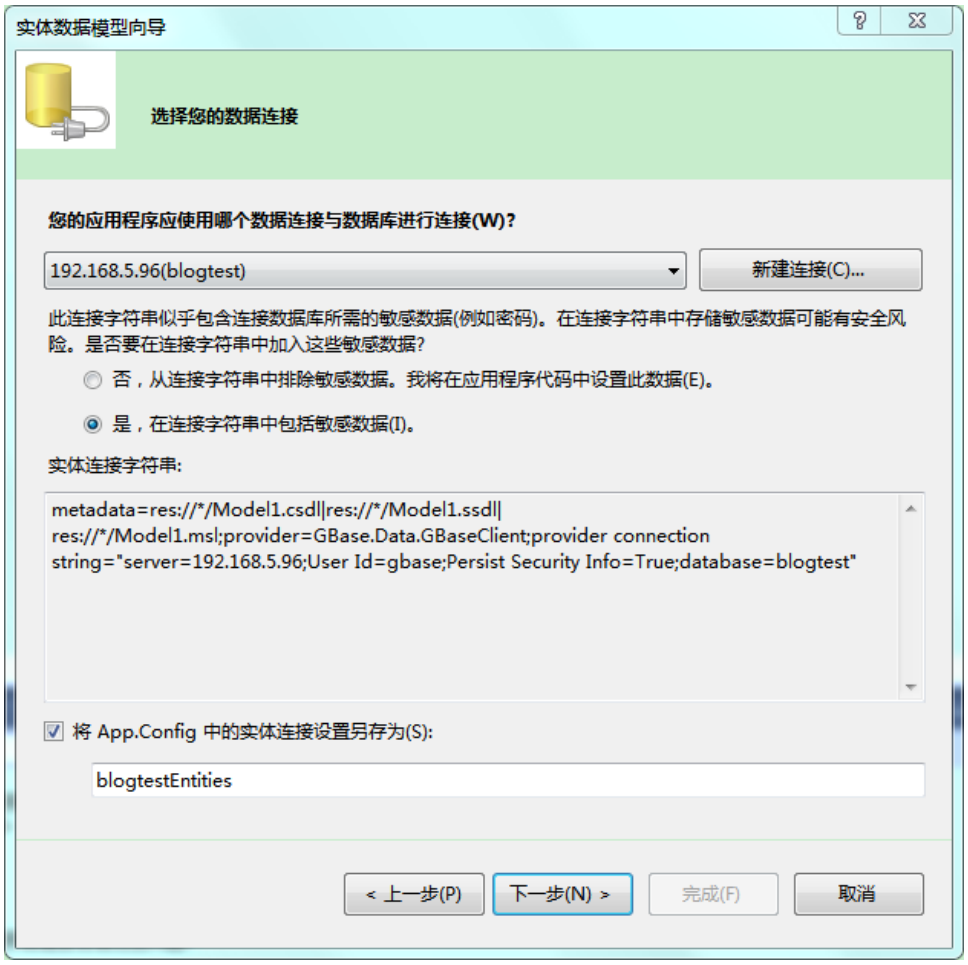


图 10-5 添加 GBase 数据连接

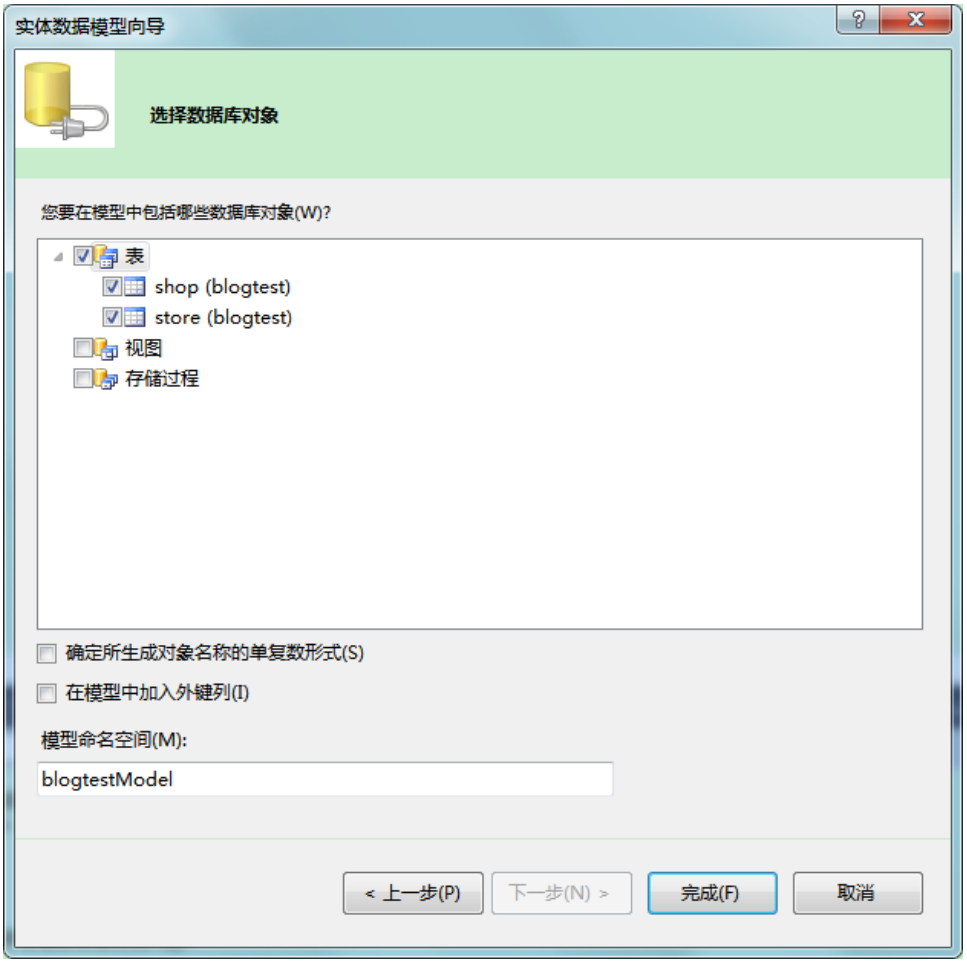


图 10-6 选择数据库对象

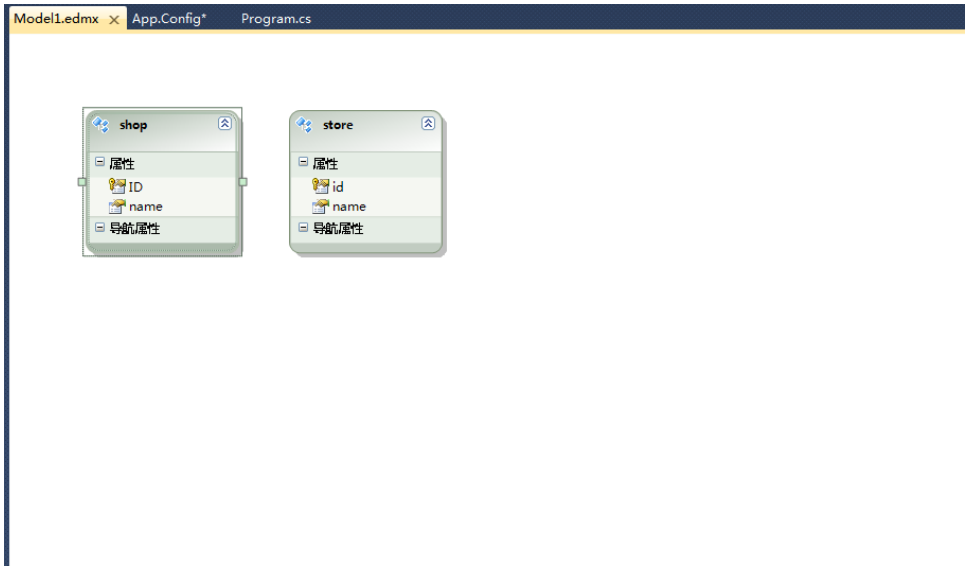


图 10-7 完成创建

10.2 ModelFirst 模式

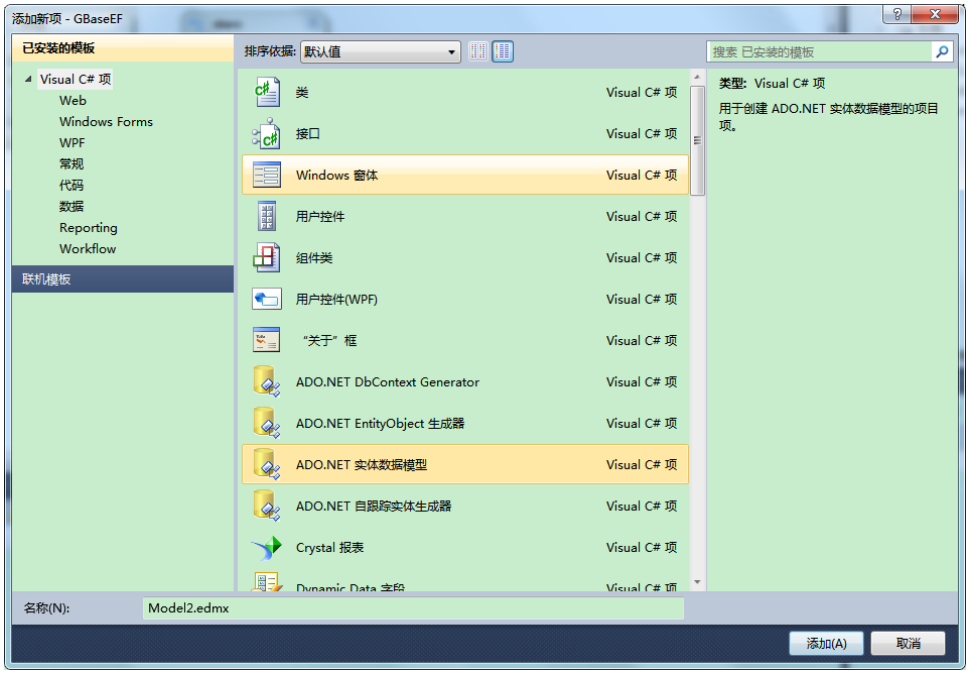


图 9-8 创建 ADO.NET 实体数据类型

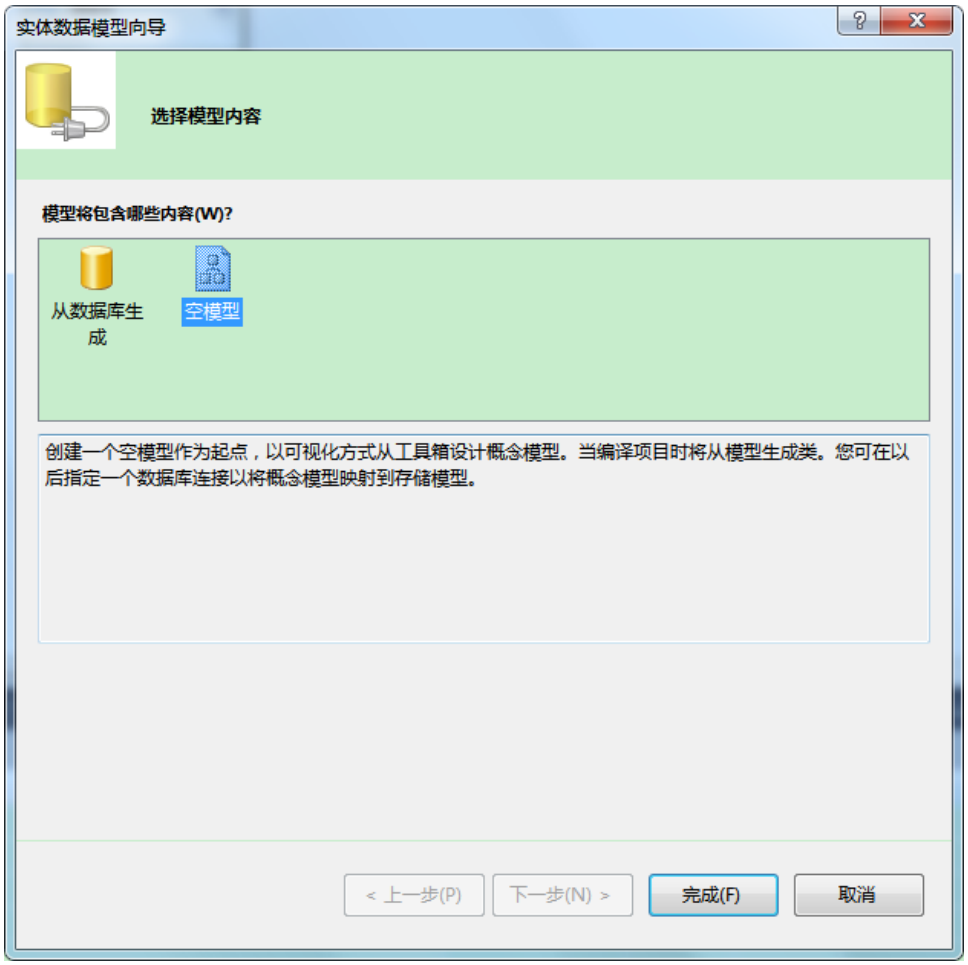


图 10-8 选择空模型

图 10-9 创建空模型

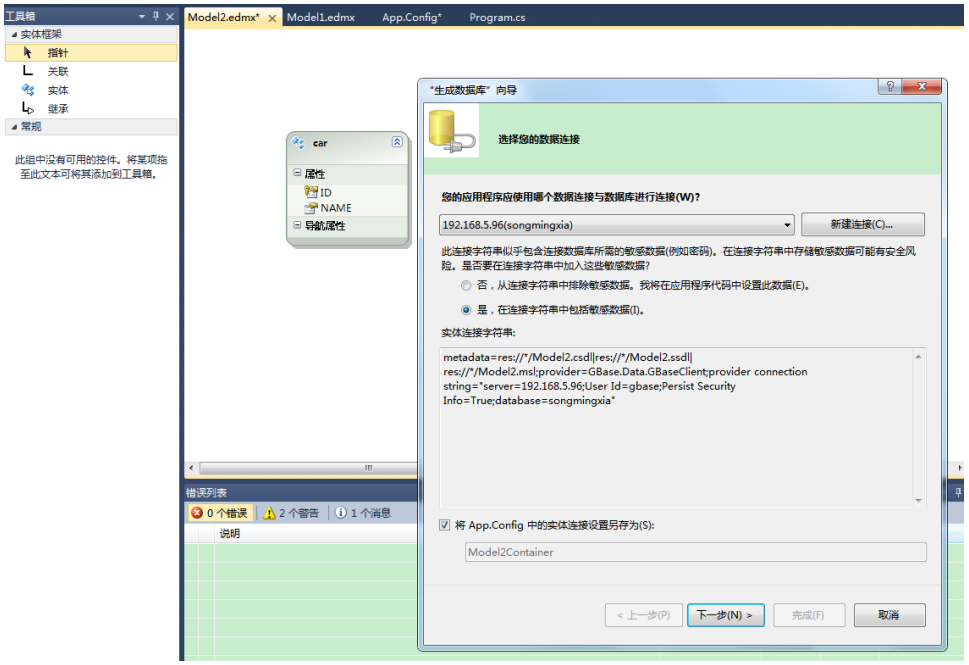


图 10-10 根据模型生成数据库

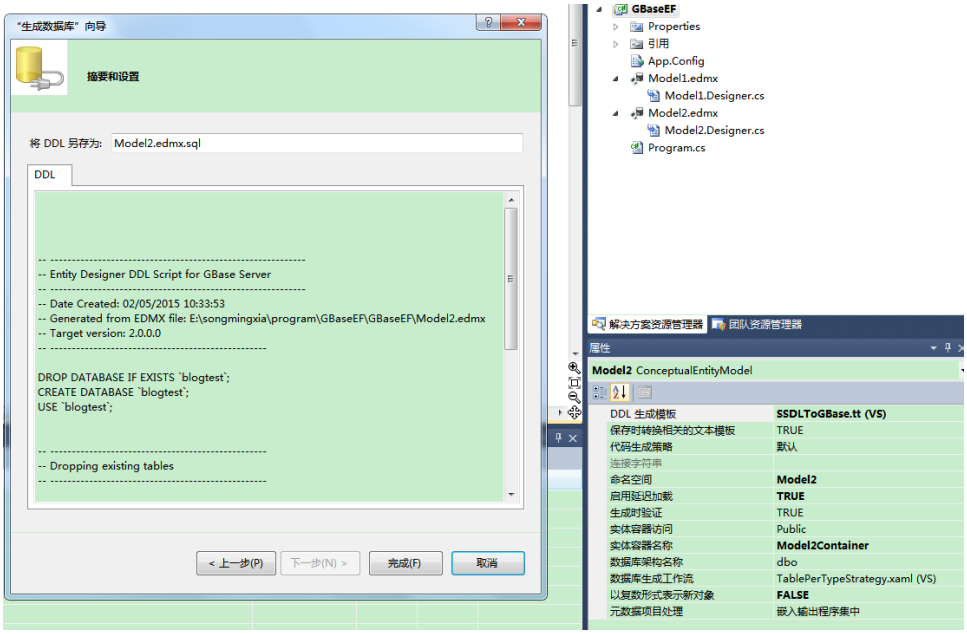


图 10-11 生成数据库脚本

注：获取生成的数据脚本后，可以使用 GBase ADO.NET VisualStudio 插件相关功能执行 SQL。

10.3 CodeFirst 模式

通过编写代码直接操作数据表！需要在 App.config 中配置相应的连接串：

```
<connectionStrings>
  <add name="BloggingContext"
               connectionString="server=192.168.5.4;User
Id=sysdba;password=1;Initial Catalog=BlogTest;
       Persist Security Info=True;"
       providerName="GBase.Data.GBaseClient"
       />
</connectionStrings>
```

C#代码示例:

```
namespace EF_codefirst
{

    public class Blog
    {
        public int BlogId { get; set; }
        public string Name { get; set; }

        public virtual List<Post> Posts { get; set; }
    }

    public class Post
    {
        public int PostId { get; set; }
        public string Title { get; set; }
        public string Content { get; set; }

        public int BlogId { get; set; }
        public virtual Blog Blog { get; set; }
    }

    public class BloggingContext : DbContext
    {
        public DbSet<Blog> Blogs { get; set; }
        public DbSet<Post> Posts { get; set; }
    }

    class Program
    {

        static void Main(string[] args)
```

```
{
    InsertData();
    QueryData();
}

/// <summary>
/// 插入数据
/// </summary>
public static void InsertData()
{
    try
    {
        using (var db = new BloggingContext())
        {
            //Create and save a new Blog
            Console.Write("Enter a name for a new Blog:");
            var name = Console.ReadLine();

            var blog = new Blog { Name = name };
            db.Blogs.Add(blog);
            db.SaveChanges();
        }
    }
    catch (System.Exception ex)
    {
        throw ex.InnerException;
    }

    QueryData();
}

/// <summary>
```

```
        /// 查询数据
        /// </summary>
        public static void QueryData()
        {
            try
            {
                using (var db = new BloggingContext())
                {
                    //Display all Blogs from the DB
                    var query = from b in db.Blogs
                                orderby b.Name
                                select b;

                    Console.WriteLine("All blogs in the database:");
                    foreach (var item in query)
                    {
                        Console.WriteLine(item.Name);
                    }

                    Console.WriteLine("Press any key to exit...");
                    Console.ReadKey();
                }
            }
            catch (System.Exception ex)
            {
                throw;
            }

        }

    }
```

10.4 使用 EntityFramework 相关问题以及注意事项

由于 GBase 产品特性，在使用 EntityFramework 实体框架时会遇到相应的问题，例如不支持主外键、不支持自增列等，在实际使用时需要规避这些问题。

11 处理连接异常及常用错误代码

当.NET 应用程序连接到服务器时，服务器的状态是不可预知的。因此为应用程序增加一个错误处理是很重要的。当捕获到错误后，应用程序可根据错误代码判断错误等级及采取必要的处理。

例如:当连接出现错误时,GBaseConnection 类将抛出一个 GBaseException 对象，该对象有两个相关的属性用来处理错误:

- Message: 描述当前错误的消息
- Number: GBase 错误号

处理错误时，可根据错误号处理应用程序的响应。两个最常见的连接错误号如下:

- 1042: 不能连接到服务器
- 1045: 无效用户名和/或密码

11.1 处理连接异常

下面的代码显示如何基于实际错误改变应用程序的响应:

C# 示例:

```
GBaseConnection conn;
string gsConnectionString;

gsConnectionString =
"server=192.168.5.33;uid=root;pwd=12345;database=test;pooling=false"
;

try
{
    conn = new GBaseConnection(gsConnectionString);
```

```
        conn.Open();
    }
    catch (GBaseException ex)
    {
        switch (ex.Number)
        {
            case 1042:
                Console.WriteLine("Cannot connect to server.  Contact
administrator");
                break;
            case 1045:
                Console.WriteLine("Invalid username/password, please
try again");
                break;
        }
    }
```

11.2 常用错误代码

错误代码为 GBase Server 返回给应用的错误编号，用于唯一的标识一个错误。错误码在 GBaseErrorCode 枚举中定义。

下表仅提供通过GBase数据库返回给应用的常用错误码及错误描述的参考，具体错误码请参考 GBase 数据库相关手册。

错误码	描　述
1005	不能创建表
1006	数据库不能创建
1007	数据库不能创建，已经存在
1008	数据库不能删除，它不存在
1012	此系统表的记录不能读
1042	不能连接到服务器
1044	指定的用户禁止访问指定的数据库

错误码	描　述
1045	用户名、密码无效
1046	没有数据库被选择
1048	指定的列不能为空
1049	未知的数据库
1050	指定的表已经存在
1051	未知的表
1055	指定的列不在 GROUP BY中
1056	指定的列不能 GROUP
1059	标识符名称太长
1060	列名重复
1061	Key名重复
1065	命令为空
1066	表别名不唯一
1067	指定的列的默认值无效
1068	表有多个主键定义
1071	指定的键太长
1072	表中指定的key列不存在
1073	BLOB列不能作为key
1074	指定类型的列长度太长
1075	只能是一个自增长列，并且必须被定义为主键
1076	服务器准备接受连接
1101	BLOB和Text列不能有默认值
1102	指定的数据库名是错误的
1103	指定的表名无效
1106	指定的存储过程未知
1107	存储过程中的参数数量错误
1108	存储过程中的参数无效
1109	指定的表未知
1111	Group 函数使用错误

错误码	描　述
1113	表必须至少有1列
1115	指定的字符集未知
1130	主机不允许连接
1131	匿名用户不允许连接
1137	指定的表不能被重新打开

12 GBase ADO.NET 常见问题

本章节介绍使用 GBase ADO.NET 操作 GBase 数据库时遇到的常见问题及解决方法。

12.1 使用 GBase ADO.NET 时的版本问题

使用 GBase ADO.NET 接口时，如果.NET 应用程序使用的框架版本为.NET Framework 2.0 则在选择 GBase ADO.NET 时应使用的 .Net Framework 2.0 版本。GBase ADO.NET 目前存在的版本及与.NET Framework 兼容情况，请参考本文档的第二章 GBase ADO.NET 版本中的介绍。

12.2 SQL 语句执行后，长时间没返回的处理

客户端应用在使用接口过程中当 SQL 语句执行后，如果语句过于复杂，直至服务器长时间未将结果返回给客户端时（超时时间默认值 30 秒），会发生超时异常导致出错。若避免此情况的发生可以在连接串中设置 Default Command Timeout 参数为一个合理值，但如果此值太大，也会影响整体性能，请在开发时根据实际应用情况设定。

12.3 SQL 语句兼容问题

在使用 8a 单机版本和 8a 集群版本时，SQL 语句可能会出现不兼容的情况，如：

1. 表自增列的定义：

- 单机版本写法：

  ```
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
  ```

  或者

```
id INT NOT NULL AUTO_INCREMENT , PRIMARY KEY(id)
```

- 集群版本写法:

```
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
```

2. 表主键列的定义:

- 单机版本写法:

```
id INT NOT NULL PRIMARY KEY
```

或者

```
id INT NOT NULL, PRIMARY KEY(id)
```

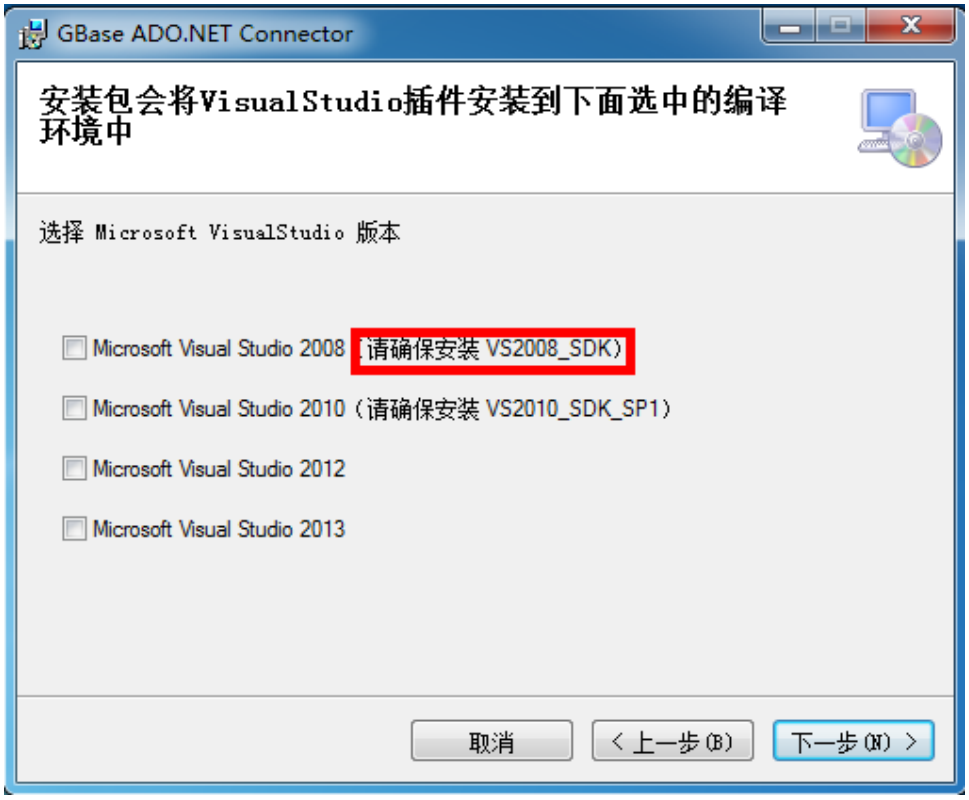
- 集群版本写法:

```
id INT NOT NULL PRIMARY KEY
```

使用 ADO.NET 接口连接至 8a 单机版和 8a 集群版本时,SQL 语法定义请严格参考 SQL 手册。

12.4 VisualStudio 插件正常安装后不能使用

使用 VisualStudio 安装包选择安装 VS 插件时，需确定系统中要确定安装对应插件的 VisualStudio SDK 组件。在安装界面中会有 SDK 确认安装提示，请看下图。



如果在不安装 SDK 的前提下强制使用插件，将会导致莫名其妙的问题。如：在 SSAS 项目中新建数据源的窗口中点击下拉列表并选择 GBase Data Provider 时，会自动关掉窗口。

13 GBase ADO.NET 的客户端类介绍

本章节介绍了 GBase ADO.NET 的客户端类，并且只实现了 Microsoft ADO.NET 的部分类，如果需要了解除本文档外其余的 ADO.NET 的类介绍，请参考 MSDN 上关于 Microsoft ADO.NET 类的相关介绍。

GBase ADO.NET 的客户端类

类	描　　述
GBaseConnection	代表一个与 GBase 服务器数据库连接。这个类不能被继承。
GBaseCommand	代表一个要对 GBase 数据库执行操作的 SQL 语句，这个类不能被继承。
GBaseCommandBuilder	可以创建 GBaseCommandBuilder 对象来自动生成针对单个表操作的 SQL 语句，此类不能被继承。
GBaseDataAdapter	GBaseDataAdapter 是 DataSet 和 GBase 数据库之间的桥接器，用于检索和保存数据。此类不能被继承。
GBaseDataReader	代表 GBase 数据源中只进且只读的数据流，此类不能被继承。
GBaseError	提供由服务器返回的错误代码参考。
GBaseException	返回错误时抛出的异常。这个类不能被继承。
GBaseHelper	帮助类,使用 GBase ADO.NET 对象封装，使得操作 GBase 数据源更加方便。
GBaseInfoMessageEventArgs	提供用于 InfoMessage 事件的数据。这个类不能被继承。
GBaseParameter	代表一个传给 GBaseCommand 的参数。这个类不能被继承。
GBaseParameterCollection	代表一个关于 GBaseCommand 的参数以及和参数对应的 DataSet 中列的集合。

类	描　　述
	这个类不能被继承。
GBaseRowUpdatedEventArgs	提供用于 RowUpdated 事件的数据。这个类不能被继承。
GBaseRowUpdatingEventArgs	提供用于 RowUpdating 事件的数据。这个类不能被继承。
GBaseTransaction	代表一个 GBase 数据库中的事务。这个类不能被继承。
GBaseIpAutoRoute	集群高可用实现类，此类为静态不能被继承。
GBaseLoadBalance	集群负载均衡实现类，此类为不静态不能被继承。
GBaseConnectionSettingCommon	集群高可用和负载均衡新增类，负责解析 Settings 中关于高可用性关键字的类。此类不能继承。

对象事件

委　托	描　述
GBaseInfoMessageEventHandler	代表处理 GBaseConnection 的 InfoMessage 事件的方法。
GBaseRowUpdatedEventHandler	代表处理 GBaseDataAdapter 的 RowUpdated 事件的方法。
GBaseRowUpdatingEventHandler	代表处理 GBaseDataAdapter 的 RowUpdating 事件的方法。
GBaseDbType	指定用于 GBaseParameter 的字段，属性的 GBase 特殊数据类型。

GBase ADO.NET 的数据类型的枚举类

枚　举	描　述
GBaseDbType	指定用于 GBaseParameter 的字段，属性的 GBase 特殊数据类型。

13.1 GBaseCommand 类

代表一个要对 GBase 数据库执行操作的 SQL 语句。这个类不能被继承。对于该类所有成员的列表，参考 GBaseCommand 成员。

- 继承层次

System.Object

|__ System.MarshalByRefObject

|__ System.ComponentModel.Component

|__ System.Data.Common.DbCommand

|__ GBase.Data.GBaseClient.GBaseCommand

- 语法

[Visual Basic]

```
Public NotInheritable Class GBaseCommand _
    Inherits DbCommand _
    Implements ICloneable
```

[C#]

```
public sealed class GBaseCommand : DbCommand, Icloneable
```

- 必要条件

命名空间: GBase.Data.GBaseClient

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 注释

1) GBase ADO.NET 使用“?”符号来定位 SQL 中的参数；

2) GBaseCommand 使用下列方法在 GBase 数据库中执行命令；

项 目	描 述
ExecuteReader	执行后返回一个数据行的方法。
ExecuteNonQuery	执行 INSERT, DELETE 和 UPDATE 语句的方法。
ExecuteScalar	从数据库中返回一个值的方法。

3) 用户可以重置 CommandText 属性，但必须在执行新的或者前面使用过的命令之前关闭 GBaseDataReader。

- 示例

下面的例子创建了一个GBaseCommand 对象和一个 GBaseConnection 对象。GBaseConnection 对象打开并关联 GBaseCommand 的 Connection 属性。然后调用 ExecuteNonQuery，并关闭连接。

```
[Visual Basic]

Public Sub InsertRow(gsConnectionString As String)
    ' If the connection string is null, use a default.
    If gsConnectionString = "" Then
        gsConnectionString = "Database=Test;DataSource=localhost;
"_
        &"User Id=username;
        Password=pass;Pooling=false "
    End If
    Dim gsConnection As New GBaseConnection(gsConnectionString)
    Dim gsInsertQuery As String="INSERT INTO Orders(id, customerId,
"_   & "amount) Values(1001, 23,30.66)"
    Dim gsCommand As New GBaseCommand(gsInsertQuery)
    gsCommand.Connection = gsConnection
    gsConnection.Open()
    gsCommand.ExecuteNonQuery()
    gsCommand.Connection.Close()
End Sub

[C#]
```

```
    public void InsertRow(string gsConnectionString)
    {
        // If the connection string is null, use a default.
        if(gsConnectionString == "")
        {
            gsConnectionString = "Database=Test;Data Source=localhost;
            UserId=username;
            Password=pass";
        }
        GBaseConnection gsConnection = new
GBaseConnection(gsConnectionString);
        string gsInsertQuery = "INSERT INTO Orders (id, customerId,
amount) Values(1001, 23, 30.66)";
        GBaseCommand gsCommand = new GBaseCommand(gsInsertQuery);
        gsCommand.Connection = gsConnection;
        gsConnection.Open();
        gsCommand.ExecuteNonQuery();
        gsCommand.Connection.Close();
    }
```

13.1.1 GBaseCommand 成员

公共构造函数

构造函数	描　述
GBaseCommand	构造函数，初始化 GBaseCommand 类的新实例。

公共属性

属　性	描　述
CommandText	获取或者设置要在数据源中执行的 SQL 语句。
CommandTimeout	获取或设置终止执行命令并生成错误之前的等待时间。

属　性	描　述
CommandType	获取或设置一个值，指明 CommandText 属性 SQL 文本、存储过程名称还是表名。
Connection	获取或设置 GBaseCommand 实例使用的 GBaseConnection 对象。
Container （继承于 Component）	获得包含 Component 的 IContainer。
IsPrepared	返回命令是否准备完成的状态，返回 true 和 false
Parameters	SQL 语句或者存储过程使用的参数集合 GBaseParameterCollection，默认是空集合。
Site （继承于 Component）	获得或者设置 Component 的 ISite。
Transaction	获取或设置事务对象。
UpdatedRowSource	设置或获取当使用 GBaseDataAdapter 的 Update 方法时，如何将结果应用到 DataRow。

公共方法

方　法	描　述
CreateParameter	创建一个 GBaseParameter 对象的实例。
Dispose （继承于 Component）	释放所有 Component 使用的资源。
Equals （继承于 Object）	判断指定的对象是否等于当前的对象。
ExecuteNonQuery	针对一个连接执行一个 SQL 语句并返回影响的行数。
ExecuteReader	执行查询，并返回一个 GBaseDataReader 对象，可读取一行的所有列数据。

方　法	描　述
ExecuteScalar	执行查询，并返回查询结果集的第一行的第一列。忽略其余的行和列。
GetType (继承于 Object)	获取当前实例的类型。
Prepare	执行后将使 GBase 服务器上定义一系列预处理语句生效。
ToString (继承于 Component)	返回类的完全限定名。

事件

事　件	描　述
Dispose (继承于 Component)	调用 Dispose 时触发的事件。

13.1.2　GBaseCommand 构造函数

初始化 GBaseCommand 的新实例。

- 重载列表

1)　初始化 GBaseCommand 的新实例。

GBaseCommand()

2)　使用 SQL 查询语句初始化 GBaseCommand 的新实例。

GBaseCommand(string)

3)　使用 SQL 查询语句和 GBaseConnection 初始化 GBaseCommand 的新实例。

GBaseCommand(string,GBaseConnection)

4)　使用 SQL 查询语句、GBaseConnection 和 GBaseTransaction 实例初始化 GBaseCommand 的新实例。

GBaseCommand(string,GBaseConnection,GBaseTransaction)

- 注释

1)　构造函数执行时，会初始化一些属性的默认值，下面的表显示这些属

性及其默认值。用户可以在程序中设置这些属性的值。

初始化的属性值

属　性	初始值
CommandText	String.Empty
CommandTimeout	30
CommandType	CommandType.Text
Connection	null

- 示例

下面的例子创建了 GBaseCommand 且设置了它的一些属性。

```
[Visual Basic]

Public Sub CreateGBaseCommand()
    Dim sql as String = "SELECT * FROM gstable"
    Dim gsCommand As New GBaseCommand(sql)
    gsCommand.CommandType = CommandType.Text
End Sub


[C#]
public void CreateGBaseCommand()
{
    string sql = "SELECT * FROM gstable";
    GBaseCommand gsCommand = new GBaseCommand(sql);
    gsCommand.CommandType = CommandType.Text;
}
```

13.1.2.1 GBaseCommand 构造函数()

初始化 GBaseCommand 的新实例。

- 语法

[Visual Basic]

```
Public Sub New()
```

[C#]

```
public GBaseCommand()
```

13.1.2.2 GBaseCommand 构造函数(String)

使用 SQL 查询语句初始化 GBaseCommand 的新实例。

- 语法

[Visual Basic]

```
Public Sub New(ByVal cmdText As String)
```

[C#]

```
public GBaseCommand(String cmdText)
```

- 参数

1) cmdText : SQL 查询语句

13.1.2.3 GBaseCommand 构造函数(String, GBaseConnection)

使用 SQL 查询语句和 GBaseConnection 实例初始化 GBaseCommand 的新实例。

- 语法

[Visual Basic]

```
Public Sub New ( _
    cmdText As String, _
    connection As GBaseConnection _
```

```
)
[C#]
public GBaseCommand(
    string cmdText,
    GBaseConnection connection
)
```

- 参数

1) cmdText ： SQL 查询语句;

2) connection ：已经建立连接的 GBaseConnection 对象。

13.1.2.4 GBaseCommand 构造函数 (String, GBaseConnection, GBaseTransaction)

使用 SQL 查询语句、GBaseConnection 和 GBaseTransaction 实例初始化 GBaseCommand 的新实例。

- 语法

```
[Visual Basic]
Public Sub New ( _
    cmdText As String, _
    connection As GBaseConnection, _
    transaction As GBaseTransaction _
)
[C#]
public GBaseCommand(
    string cmdText,
```

```
    GBaseConnection connection,
    GBaseTransaction transaction
)
```

- 参数

1) cmdText ： SQL 查询语句；

2) connection ： 已经建立连接的 GBaseConnection 对象；

3) transaction ： 事务对象。

13.1.3 GBaseCommand 属性

13.1.3.1 CommandText 属性

获取或者设置要在数据源中执行的 SQL 语句，默认是空字符串。

- 语法

```
[Visual Basic]
Public Overrides Property CommandText As String
    Get
    Set
[C#]
public override string CommandText { get; set; }
```

- 实现

IDbCommand.CommandText

- 注释

当 CommandType 属性设置为 StoredProcedure 时， CommandText 属性应该设置为存储过程的名字。如果存储过程名包含特殊字符，用户可能需要使用转

义符语法。在用户调用 Execute 方法的时候，这个命令就执行在 CommandText 中设置的存储过程。

- 示例

下面的例子创建了 GBaseCommand 且设置了它的一些属性。

```
[Visual Basic]
Public Sub CreateGBaseCommand()
    Dim gsCommand As New GBaseCommand()
    gsCommand.CommandText = "SELECT * FROM Gstable ORDER BY"_
    &"id"
    gsCommand.CommandType = CommandType.Text
End Sub


[C#]
public void CreateGBaseCommand()
{
    GBaseCommand gsCommand = new GBaseCommand();
    gsCommand.CommandText = "SELECT * FROM gstable ORDER BY
id";
    gsCommand.CommandType = CommandType.Text;
}
```

13.1.3.2 CommandTimeout 属性

获取或设置终止执行命令并生成错误之前的等待时间，默认值为 0。

- 语法

```
[Visual Basic]

Public Overrides Property CommandTimeout As Integer

    Get

    Set
```

[C#]

```
public override int CommandTimeout { get; set; }
```

- 实现

IDbCommand.CommandTimeout

13.1.3.3 CommandType 属性

获取或设置一个值，指明命令的类型。

- 语法

[Visual Basic]

```
Public Overrides Property CommandType As CommandType
    Get
    Set
```

[C#]

```
public override CommandType CommandType { get; set; }
```

- 属性值

CommandType 可以设置的值有：

  - Text ：SQL 文本命令 (默认)；
  - StoredProcedure ：存储过程的名称；
  - TableDirect ：表的名称。

- 实现

IDbCommand.CommandType

- 注释

当 CommandType 属性设置成 StoredProcedure 时，CommandText 属性应该

设置为存储过程的名字。在用户调用 Execute 方法的时候，这个命令就执行这个存储过程。

- 示例

下面的例子创建了 GBaseCommand 且设置了它的一些属性。

```
[Visual Basic]
Public Sub CreateGBaseCommand()
    Dim gsCommand As New GBaseCommand()
    gsCommand.CommandType = CommandType.Text
End Sub

[C#]
public void CreateGBaseCommand()
{
    GBaseCommand gsCommand = new GBaseCommand();
    gsCommand.CommandType = CommandType.Text;
}
```

13.1.3.4 Connection 属性

获取或设置 GBaseCommand 实例使用的 GBaseConnection 对象。

- 语法

```
[Visual Basic]
Public Property Connection As GBaseConnection
    Get
    Set
[C#]
public GBaseConnection Connection { get; set; }
```

- 注释

当事务在进行中且 Transaction 属性为空时，如果用户设置了 Connection，那么会产生 InvalidOperationException。如果 Transaction 属性不是 null 且事务已经提交或者回滚了，Transaction 将被自动设置为 null。

- 示例

下面的例子创建了 GBaseCommand 且设置了它的一些属性。

```
[Visual Basic]
    Public Sub CreateGBaseCommand()
    Dim gsSelectQuery As String="SELECT * FROM gstable ORDER BY id"
    Dim gsConnectString As String = "Persist SecurityInfo=False; "_
    & "database=test; "_
    & "servser=gsServer"
    Dim gsCommand As New GBaseCommand(gsSelectQuery)
    gsCommand.Connection = New GBaseConnection(gsConnectString)
    gsCommand.CommandType = CommandType.Text
End Sub

[C#]
public void CreateGBaseCommand()
{
    string gsSelectQuery = "SELECT * FROM gstable ORDER BY id";
    string gsConnectString = "Persist
SecurityInfo=False;database=test;
    server = gsServer";
    GBaseCommand gsCommand = new GBaseCommand(gsSelectQuery);
    gsCommand.Connection = new GBaseConnection(gsConnectString);
    gsCommand.CommandType = CommandType.Text;
}
```

13.1.3.5 IsPrepared 属性

获取命令是否准备完成的状态，此属性在调用完 Prepare()方法后则返回 true。

- 语法

[Visual Basic]

```
Public ReadOnly Property IsPrepared As Boolean
    Get
```

[C#]

```
public bool IsPrepared { get; }
```

13.1.3.6 Parameter 属性

SQL 语句或者存储过程使用的参数集合 GBaseParameterCollection，默认是空集合。

- 语法

[Visual Basic]

```
Public ReadOnly Property Parameters As GBaseParameterCollection
    Get
```

[C#]

```
public GBaseParameterCollection Parameters { get; }
```

- 注释

GBase ADO.NET 不支持无名参数。每个加入到集合中的参数必须有一个连接名字。

- 示例

下面的例子创建一个 GBaseCommand 并显示它的参数。要完成这些，要使用 GBaseConnection，一个 SQL SELECT 查询字符串，和一个 GBaseParameter 对象数组。

[Visual Basic]

```
        Public Sub CreateGBaseCommand(gsConnection As GBaseConnection,
_
        gsSelectQuery As String, gsParamArray() As GBaseParameter)
        Dim gsCommand As New GBaseCommand(gsSelectQuery, gsConnection)
        gsCommand.CommandText = "SELECT id, name FROM gstable" _
        &" WHEREage=?age"
        gsCommand.UpdatedRowSource = UpdateRowSource.Both
        gsCommand.Parameters.Add(gsParamArray)
        Dim j As Integer
        For j = 0 To gsCommand.Parameters.Count - 1
            gsCommand.Parameters.Add(gsParamArray(j))
        Next j
        Dim gsMessage As String = ""
        Dim i As Integer
        For i = 0 To gsCommand.Parameters.Count - 1
            gsMessage += gsCommand.Parameters(i).ToString() &
ControlChars.Cr
        Next i
        Console.WriteLine(gsMessage)
    End Sub

    [C#]
    public void CreateGBaseCommand(GBaseConnection gsConnection,
string gsSelectQuery,GBaseParameter[] gsParamArray)
    {
        GBaseCommand gsCommand = new GBaseCommand(gsSelectQuery,
        gsConnection);
        gsCommand.CommandText = "SELECT id, name FROM gstable
        WHERE age=?age";
        gsCommand.Parameters.Add(gsParamArray);
        for (int j=0; j<gsParamArray.Length; j++)
        {
            gsCommand.Parameters.Add(gsParamArray[j]) ;
        }
        string gsMessage = "";
        for (int i = 0; i < gsCommand.Parameters.Count; i++)
```

```
        {
            gsMessage += gsCommand.Parameters[i].ToString() + "\n";
        }
        MessageBox.Show(gsMessage);
    }
```

13.1.3.7 Transaction 属性

获取或设置事务对象。

- 语法

[Visual Basic]

```
Public Property Transaction As GBaseTransaction
    Get
    Set
```

[C#]

```
public GBaseTransaction Transaction { get; set; }
```

- 注释

用户不能设置已经有值的或者是正在执行的 Transaction 属性。如果用户设置了事务属性为一个 GBaseTransaction 对象，而该对象和 GBaseCommand 关联到不同的 GBaseConnection 对象，就会在用户下次尝试执行语句的时候抛出异常。

13.1.3.8 UpdatedRowSource 属性

当使用 DbDataAdapter 的 Update 方法时,获取或设置命令结果如何应用到 DataRow。

- 语法

[Visual Basic]

```
Public Overrides Property UpdatedRowSource As UpdateRowSource
    Get
    Set
```

[C#]

```
public override UpdateRowSource UpdatedRowSource { get; set; }
```

- 实现

IDbCommand.UpdatedRowSource

- 属性

UpdateRowSource 可以设置的值有：

  - None ： 忽略任何返回的参数或行。
  - OutputParameters ： 将输出参数映射到 DataSet 中的已更改的行。
  - FirstReturnedRecord ： 将第一个返回行中的数据映射到 DataSet 中的已更改的行。
  - Both ： 将输出参数和第一个返回行都映射到 DataSet 中的已更改的行。

- 注释

默认的 UpdateRowSource 的值是 Both，除非命令自动生成（在 GBaseCommandBuilder 情况下），此时默认为 None。

13.1.4 GBaseCommand 方法

13.1.4.1 CreateParameter 方法

创建一个 GBaseParameter 对象的实例。

- 语法

[Visual Basic]

```
Public Function CreateParameter As GBaseParameter
```

[C#]

```
public GBaseParameter CreateParameter()
```

- 返回值

创建的 GBaseParameter 对象。

13.1.4.2 ExecuteNonQuery 方法

执行一个 SQL 语句并返回影响的行数。

- 语法

[Visual Basic]

```
Public Overrides Function ExecuteNonQuery As Integer
```

[C#]

```
public override int ExecuteNonQuery()
```

- 返回值

影响的行数。

- 实现

IDbCommand.ExecuteNonQuery()

- 注释

用户可以使用 ExecuteNonQuery 来执行任何类型的数据库操作，这时，任何返回结果集都不可用。任何用于存储过程的参数可以用于绑定数据而且在执行后可以返回完整的结果。对于 UPDATE, INSERT, 和 DELETE 语句，返回值是命令影响的行数。对于其它类型的语句，返回值是-1。

- 示例

下面的例子创建了一个 GBaseCommand 并使用 ExecuteNonQuery 来执行它。例子的参数是要执行的 SQL 语句 (例如 UPDATE, INSERT, or DELETE)和用于连接数据源的字符串。

```
[Visual Basic]
Public Sub CreateGBaseCommand(gsExecuteQuery As String,_
gsConnection_
As GBaseConnection)
    Dim gsCommand As New GBaseCommand (gsExecuteQuery_
    ,gsConnection)
    gsCommand.Connection.Open()
    gsCommand.ExecuteNonQuery()
    gsConnection.Close()
End Sub

[C#]
public void CreateGBaseCommand(string gsExecuteQuery,
GBaseConnection gsConnection)
{
    GBaseCommand gsCommand =New GBaseCommand (gsExecuteQuery,
gsConnection);
    gsCommand.Connection.Open();
    gsCommand.ExecuteNonQuery();
    gsConnection.Close();
}
```

13.1.4.3 ExecuteReader 方法

执行查询，并返回一个 GBaseDataReader 对象，可读取一行的所有列数据。

- 重载列表

1） 使 GBaseConnection 执行 CommandText 属性指定的 SQL 语句，并用返回值构建 GBaseDataReader。

ExecuteReader()

2） 使 GBaseConnection 执行 CommandText 属性指定的 SQL 语句，并使用 CommandBehavior 的一个值构建 GBaseDataReader。

ExecuteReader(CommandBehavior)

13.1.4.3.1 ExecuteReader 方法 ()

使 GBaseConnection 执行 CommandText 属性指定的 SQL 语句，并用返回值构建 GBaseDataReader。

- 语法

[Visual Basic]

Public Function ExecuteReader As GBaseDataReader

[C#]

public GBaseDataReader ExecuteReader();

- 返回值

一个 GBaseDataReader 对象。

- 注释

当 CommandType 属性是 StoredProcedure，CommandText 属性应该设置为存储过程的名字。在用户调用 Execute 方法的时候，命令执行这个存储过程。否

则，在用户调用 Execute 方法执行 CommandText 指定的 SQL 语句，并得到返回值。

- 示例

```
[Visual Basic]
Public Sub CreateGBaseDataReader(gsSelectQuery As String,_
gsConnection_
As GBaseConnection)
    Dim gsCommand As New GBaseCommand(gsSelectQuery, gsConnection)
    gsConnection.Open()
    Dim gsReader As GBaseDataReader
    gsReader = gsCommand.ExecuteReader()
    Try
        While gsReader.Read()
            Console.WriteLine(gsReader.GetString(0))
        End While
        Finally
        gsReader.Close
        gsConnection.Close
    End Try
End Sub

[C#]
public void CreateGBaseDataReader(string gsSelectQuery,
GBaseConnection gsConnection)
{
    GBaseCommand gsCommand = new GBaseCommand(gsSelectQuery,
gsConnection);
    gsConnection.Open();
    MGBaseDataReader gsReader;
    gsReader = gsCommand.ExecuteReader();
    try
    {
        while(gsReader.Read())
        {
```

```
                Console.WriteLine(gsReader.GetString(0));
            }
            finally
            {
                gsReader.Close();
                gsConnection.Close();
            }
        }
    }
```

13.1.4.3.2 ExecuteReader 方法 (CommandBehavior)

使 GBaseConnection 执行 CommandText 属性指定的 SQL 语句，并使用 CommandBehavior 的一个值构建 GBaseDataReader。

- 语法

[Visual Basic]

```
Public Function ExecuteReader ( _
    behavior As CommandBehavior _
) As GBaseDataReader
```

[C#]

```
public GBaseDataReader ExecuteReader(CommandBehavior behavior);
```

- 参数

1) behavior : CommandBehavior 的一个值。

- 返回值

一个 GBaseDataReader 对象。

- 注释

当 CommandType 属性是 StoredProcedure 时，CommandText 属性应该设置为存储过程的名字。在用户调用 Execute 方法的时候，命令执行这个存储过程。

GBaseDataReader 可以支持使用大二进制值的特殊模式，使得读取更有效率。对于更多信息，参考 SequentialAccess 对于 CommandBehavior 的设置。

当使用 GBaseDataReader 的时候，相关的 GBaseConnection 忙于服务 GBaseDataReader。此时，没有其它的操作可以在 GBaseConnection 上执行除非关闭它。这个情况会持续到调用 GBaseDataReader 的 Close 方法。如果 GBaseDataReader 使用设置为 CloseConnection 的 CommandBehavior 创建，关闭 GBaseDataReader 也会自动关闭连接。

注意，当使用 SingleRow 行为调用 ExecuteReader 的时候，用户应该小心使用 SQL 中的 limit 子句，这会让客户端返回所有的行（到达给定的上限）。Read 方法还会返回 false，但是所有的行放入客户端会造成性能上的冲击。如果 limit 子句不需要，就应该避免使用。

13.1.4.4 ExecuteScalar 方法

执行查询，并返回查询结果集的一行的第一列，忽略其余的行和列 。

- 语法

[Visual Basic]

```
Public Overrides Function ExecuteScalar As Object
```

[C#]

```
public override Object ExecuteScalar()
```

- 返回值

执行查询，并返回查询结果集的一行的第一列，忽略其余的行和列。

- 实现

IDbCommand.ExecuteScalar()

- 注释

使用 ExecuteScalar 方法一般从数据库返回一个单一值（例如：求统计行

数)。下面的例子说明一个典型的 ExecuteScalar 应用。

```
[C#]
cmd.CommandText = "select count(*) from region";
Int32 count = (int32) cmd.ExecuteScalar();
```

- 示例

下面的例子创建了一个函数，函数内使用 GBaseCommand 对象并调用 ExecuteScalar 方法,参数为返回聚合结果的 SQL 语句和用于连接数据源的字符串。

```
[Visual Basic]
Public Sub CreateGBaseCommand(gsScalarQuery As String,
gsConnection As GBaseConnection)
    Dim gsCommand As New GBaseCommand(gsScalarQuery, gsConnection)
    gsCommand.Connection.Open()
    gsCommand.ExecuteScalar()
    gsConnection.Close()
End Sub

[C#]
public void CreateGBaseCommand(string gsScalarQuery,
GBaseConnection gsConnection)
{
    GBaseCommand gsCommand = new GBaseCommand(gsScalarQuery,
gsConnection);
    gsCommand.Connection.Open();
    gsCommand.ExecuteScalar();
    gsConnection.Close();
}
```

13.1.4.5 Prepare 方法

执行后将使在 GBase 服务器上定义一系列预处理语句生效。

- 语法

[Visual Basic]

```
Public Overrides Sub Prepare
```

[C#]

```
public override void Prepare()
```

- 实现

IDbCommand.Prepare()

- 示例

下面的例子创建了一个函数，函数内显示了 Prepare 的用法。

[Visual Basic]

```
public sub PrepareExample()
    Dim cmd as New GBaseCommand("INSERT INTO gstable VALUES" _ &"
(?val)", gsConnection)
    cmd.Parameters.Add( "?val", 10 )
    cmd.Prepare()
    cmd.ExecuteNonQuery()
    cmd.Parameters(0).Value = 20
    cmd.ExecuteNonQuery()
end sub
```

[C#]

```
private void PrepareExample()
{
    GBaseCommand cmd = new GBaseCommand("INSERT INTO gstable VALUES
(?val)", gsConnection);
    cmd.Parameters.Add( "?val", 10 );
```

```
        cmd.Prepare();
        cmd.ExecuteNonQuery();
        cmd.Parameters[0].Value = 20;
        cmd.ExecuteNonQuery();
    }
```

13.2 GBaseCommandBuilder 类

自动生成单表命令，当使用 DataSet 对象的方法将变更的内容写回到对应数据库时，需使用 GBaseCommandBuilder 进行协调关联。无法继承此类。

对于该类所有成员的列表，参考 GBaseCommandBuilder 成员。

一、继承层次

```
System.Object
  |__ System.MarshalByRefObject
    |__ System.ComponentModel.Component
      |__ System.Data.Common.DbCommandBuilder
        |__GBase.Data.GBaseClient.GbaseCommandBuilder
```

- 语法

```
[ Visual Basic ]
Public NotInheritable Class GBaseCommandBuilder _
    Inherits DbCommandBuilder
[ C# ]
public sealed class GBaseCommandBuilder : DbCommandBuilder
```

- 必要条件

命名空间: GBase.Data.GBaseClient

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 注释

GBaseDataAdapter 不会自动生成与 DataSet 的变更相一致的 SQL 语句，必须使用 GBaseCommandBuilder 关联。要生成 INSERT、UPDATE 或 DELETE 语句时，需要用到 GBaseDataAdapter 的 SelectCommand 属性自动返回需要的元数据集。SelectCommand 必须返回至少一个主键或者唯一列，如果没有返回，在生成更新语句时就会出现 InvalidOperation 异常，且命令不会生成。

GBaseCommandBuilder 还使用 SelectCommand 引用的 Connection、CommandTimeout 和 Transaction 属性。如果这些属性改变了，用户应该调用 RefreshSchema。否则 InsertCommand、UpdateCommand 和 DeleteCommand 属性会保持它们原先的值。

如果用户调用了 GBaseCommandBuilder 的 Dispose 方法后，GBaseCommandBuilder 会从 GBaseDataAdapter 断开，且不能再用于生成命令。

用户无论何时设置 GBaseDataAdapter 属性，GBaseCommandBuilder 都会把自己注册为 OnRowUpdating 事件的监听器。

- 示例

下面的例子演示了如何创建 GBaseCommandBuilder。

```
[Visual Basic]
Public Shared Function SelectRows(gsConnection As String,_
gsSelectQuery As String, gsTableName As String)_
As DataSet
    Dim gsConn As New GBaseConnection(gsConnection)
    Dim gsDataAdapter As New GBaseDataAdapter()
    gsDataAdapter.SelectCommand = New GBaseCommand(gsSelectQuery,_
gsConn)
    Dim cb As GBaseSqlCommandBuilder = New GbaseCommandBuilder_
    (gsDataAdapter)
    gsConn.Open()
```

```
        Dim ds As DataSet = New DataSet
        gsDataAdapter.Fill(ds, gsTableName)
        ' Code to modify data in DataSet here
        ' Without the GBaseCommandBuilder this line would fail.
        gsDataAdapter.Update(ds, gsTableName)
        gsConn.Close()
    End Function 'SelectRows

    [C#]
    Public Static DataSet SelectRows(string gsConnection, string
gsSelectQuery,
    string gsTableName)
    {
        GBaseConnection gsConn = new GBaseConnection(gsConnection);
        GBaseDataAdapter gsDataAdapter = new GBaseDataAdapter();
        gsDataAdapter.SelectCommand = new GBaseCommand(gsSelectQuery,
gsConn);
        GBaseCommandBuilder cb = new
GBaseCommandBuilder(gsDataAdapter);
        gsConn.Open();
        DataSet ds = new DataSet();
        gsDataAdapter.Fill(ds, gsTableName);
        //code to modify data in DataSet here
        //Without the GBaseCommandBuilder this line would fail
        gsDataAdapter.Update(ds, gsTableName);
        gsConn.Close();
        return ds;
    }
```

13.2.1 GBaseCommandBuilder 成员

公共静态成员（VB 中是 Shared）方法

静态成员方法	描　述
DeriveParameters	从在 GBaseCommand 中指定的存储过

静态成员方法	描　述
	程中检索参数信息并填充指定的 GBaseCommand 对象的 Parameters 集合。

公共构造函数

构造函数	描　述
GBaseCommandBuilder	重　载　函　数　。　初　始　化 GBaseCommandBuilder 类的一个实例。

属性

属　性	描　述
DataAdapter	获取或设置一个用于自动生成 SQL 语句的 GBaseDataAdapter 对象。

公共方法

方　法	描　述
Dispose (继承于 Component)	释放所有 Component 使用的资源。
Equals (继承于 Object)	判断指定的对象是否等于当前的对象。
GetDeleteCommand	获得需要用于执行数据库删除的 GBaseCommand 对象，该对象是自动生成的。
GetInsertCommand	获得需要用于执行数据库插入的 GBaseCommand 对象，该对象是自动生成的。
GetType (继承于 Object)	获取当前实例的类型。
GetUpdateCommand	获得需要用于执行数据库更新 GBaseCommand 对象，该对象是自动生成的。
RefreshSchema	刷新数据库模式（Schema）信息,该信

方　法	描　述
	息用于生成 INSERT, UPDATE, 或 DELET 语句。

13.2.2　GBaseCommandBuilder 构造函数

初始化 GBaseCommandBuilder 类的一个对象。

- 重载列表

1)　初始化 GBaseCommandBuilder 类的一个对象。

GBaseCommandBuilder()

2)　使用 GBaseDataAdapter 对象初始化 GBaseCommandBuilder 类的一个对象。

GBaseCommandBuilder(GBaseDataAdapter)

13.2.2.1.1 GBaseCommandBuilder 构造函数()

初始化 GBaseCommandBuilder 类的一个对象。

- 语法

[Visual Basic]

Public Sub New

[C#]

public GBaseCommandBuilder()

13.2.2.2 GBaseCommandBuilder 构造函数 (GBaseDataAdapter)

使用 GBaseDataAdapter 对象初始化 GBaseCommandBuilder 类的一个对象。

- 语法

[Visual Basic]

```
Public Sub New(ByVal adapter As GBaseDataAdapter)
```

[C#]

```
public GBaseCommandBuilder(GBaseDataAdapter adapter)
```

- 参数

1) adapter : GBaseDataAdapter 对象。

- 注释

GBaseCommandBuilder 把自己作为一个监听器注册到 RowUpdating 事件上，该事件由属性上指定的 GBaseDataAdapter 产生。

13.2.3 GBaseCommandBuilder 属性

13.2.3.1 DataAdapter 属性

获取或设置一个用于自动生成 SQL 语句的 GBaseDataAdapter 对象。

- 语法

[Visual Basic]

```
Public Property DataAdapter As GBaseDataAdapter
    Get
```

Set

[C#]

```
public GBaseDataAdapter DataAdapter { get; set; }
```

- 属性

一个 GBaseDataAdapter 对象。

- 注释

GBaseCommandBuilder 把自己作为一个监听器注册到 RowUpdating 事件上，该事件由属性上指定的 GBaseDataAdapter 产生。

13.2.4 GBaseCommandBuilder 方法

13.2.4.1 DeriveParameters 方法

从在 GBaseCommand 中指定的存储过程中检索参数信息并填充指定的 GBaseCommand 对象的 Parameters 集合。

- 语法

[Visual Basic]

```
Public Shared Sub DeriveParameters ( command As GBaseCommand )
```

[C#]

```
public static void DeriveParameters(GBaseCommand command)
```

- 参数

1) command : GBaseCommand 引用了来自参数信息的存储过程，继承参数被加入到 GBaseCommand 的参数集合中。

- 异常

InvalidOperationException 异常

异常类型	条　　件
InvalidOperationException	命令文本不是有效的存储过程名。

13.2.4.2 GetDeleteCommand 方法

获取需要用于执行数据库删除命令的 GBaseCommand 对象，该对象是自动生成的。

- 语法

[Visual Basic]

```
Public Function GetDeleteCommand As GBaseCommand
```

[C#]

```
public GBaseCommand GetDeleteCommand()
```

- 返回值

用于执行删除操作的 GBaseCommand 对象。

- 注释

用户可以使用 GetDeleteCommand 作为删除命令的基础。例如，用户可以调用 GetDeleteCommand 后修改删除语句内容后重新设置给 GBaseDataAdapter 的 DeleteCommand 属性。

应用程序应该在任何关联到 GBaseCommandBuilder 上的 SQL 语句改变后调用 RefreshSchema。否则，GetDeleteCommand 会依然使用前面语句的信息，这可能是错误的，并且当应用程序调用 Update 或 GetDeleteCommand 的时候，SQL 语句仍是最初产生的。

13.2.4.3 GetInsertCommand 方法

获得需要用于执行数据库插入的 GBaseCommand 对象,该对象是自动生成的。

- 语法

[Visual Basic]

```
Public Function GetInsertCommand As GBaseCommand
```

[C#]

```
public GBaseCommand GetInsertCommand()
```

- 返回值

用于处理插入操作的 GBaseCommand 对象。

- 注释

用户可以使用 GetInsertCommand 作为插入命令的基础。例如，用户可以调用 GetInsertCommand 后修改插入语句内容后重新设置给 GBaseDataAdapter 的 InsertCommand 属性。

应用程序应该在任何关联到 GBaseCommandBuilder 上的 SQL 语句改变后调用 RefreshSchema。否则，GetInsertCommand 会依然使用前面语句的信息，这可能是错误的,并且当应用程序调用 Update 或 GetInsertCommand 的时候,SQL 语句仍是最初产生的。

13.2.4.4 GetUpdateCommand 方法

获得需要用于执行数据库更新 GBaseCommand 对象，该对象是自动生成的。

- 语法

[Visual Basic]

```
Public Function GetUpdateCommand As GBaseCommand
```

[C#]

```
public GBaseCommand GetUpdateCommand()
```

- 返回值

用于处理插入操作的 GBaseCommand 对象。

- 注释

用户可以使用 GetUpdateCommand 作为更新命令的基础。例如，用户可以调用 GetUpdateCommand 后修改插入语句内容后重新设置给 GBaseDataAdapter 的 UpdateCommand 属性。

应用程序应该在任何关联到 GBaseCommandBuilder 上的 SQL 语句改变后调用 RefreshSchema。否则，GetUpdateCommand 会依然使用前面语句的信息，这可能是错误的，并且当应用程序调用 Update 或 GetUpdateCommand 的时候，SQL 语句仍是最初产生的。

13.3 GBaseConnection 类

代表一个打开的 GBase 服务器数据库连接。这个类不能被继承。对于该类所有成员的列表，参考 GBaseConnection 成员。

- 继承层次

```
System.Object
|__ System.MarshalByRefObject
|__ System.ComponentModel.Component
|__ System.ComponentModel.Component
|__ GBase.Data.GBaseClient.GBaseConnection
```

- 语法

[Visual Basic]

```
Public NotInheritable Class GBaseConnection _
    Inherits DbConnection _
    Implements ICloneable
```

[C#]

```
public sealed class GBaseConnection : DbConnection, Icloneable
```

- 必要条件

命名空间: GBase.Data.GBaseClient

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 注释

一个 GBaseConnection 对象代表一个 GBase 数据源的连接。当用户创建一个 GBaseConnection 实例的时候，所有属性被设置为初始值。对于这些值的列表，参考 GBaseConnection 构造函数。

如果 GBaseConnection 完成功能后，用户调用 Close 关闭连接。

- 示例

下面的例子创建了一个 GBaseCommand 和一个 GBaseConnection。GBaseConnection 打开并设置为 GBaseCommand 的 Connection，然后调用了 ExecuteNonQuery，并关闭连接。

```
[Visual Basic]
Public Sub InsertRow(gsConnectionString As String)
    ' If the connection string is null, use a default.
    If gsConnectionString = "" Then
        gsConnectionString ="Database=Test;DataSource=localhost;
"_
        &"UserId=username;Password=pass;pooling=false"
    End If
```

```
        Dim gsConnection As New GBaseConnection(gsConnectionString)
        Dim gsInsertQuery As String = "INSERT INTO Orders (id, customerId,
"_ &"amount)  Values(1001, 23, 30.66)"
        Dim gsCommand As New GBaseCommand(gsInsertQuery)
        gsCommand.Connection = gsConnection
        gsConnection.Open()
        gsCommand.ExecuteNonQuery()
        gsCommand.Connection.Close()
    End Sub

    [C#]
    public void InsertRow(string gsConnectionString)
    {
        // If the connection string is null, use a default.
        if(gsConnectionString == "")
        {
            gsConnectionString = "Database=Test;Data Source=localhost;
            User Id=username;Password=pass;pooling=false";
        }
        GBaseConnection gsConnection = new GbaseConnection
(gsConnectionString);
        string gsInsertQuery = "INSERT INTO Orders (id, customerId,
amount) Values(1001, 23, 30.66)";
        GBaseCommand gsCommand = new GBaseCommand(gsInsertQuery);
        gsCommand.Connection = gsConnection;
        gsConnection.Open();
        gsCommand.ExecuteNonQuery();
        gsCommand.Connection.Close();
    }
```

13.3.1 GBaseConnection 成员

公共构造函数

构造函数	描　述

构造函数	描　述
GBaseConnection	初始化一个新的 GBaseConnection 类实例。

公共属性

属　性	描　述
ConnectionString	获取或设置用于连接 GBase 数据库的字符串。
ConnectionTimeout	获取或设置连接超时时间。
Database	获得当前数据库的名字或在连接打开后使用的数据库名字。
DataSource	获得要连接的 GBase 服务器的名字。
ServerThread	返回这个连接在服务器上的线程 ID。
ServerVersion	一个包含连接的 GBase 数据库版本的字符串。
State	获得当前连接的状态。
UseCompression	获取或设置这个连接与服务器通信时是否使用压缩协议。

公共方法

方　法	描　述
BeginTransaction	重载函数。开始一个数据库事务。
ChangeDatabase	为一个打开的 GBaseConnection 改变当前数据库。
Close	关闭数据库连接。
CreateCommand	创建并返回一个和 GBaseConnection 相关的 GBaseCommand 对象。
Dispose	释放 GBaseConnection 使用的资源。
Equals （继承于 Object）	判断指定的对象是否等于当前的对象。
GetType （继承于 Object）	获取当前实例的类型。
Open	使用 ConnectionString 指定的属性设

方　法	描　述
	置打开一个连接。
Ping	判断是否能够与数据源正常通讯。
ToString (继承于 Component)	返回类的完全限定名。

事件

事　件	描　述
InfoMessage	当 GBase 数据源执行命令或者查询结果返回警告时，该事件发生。
StateChange	当连接状态改变时发生。

13.3.2　GBaseConnection 构造函数

- 重载列表

1)　初始化一个新的 GBaseConnection 类实例。

GBaseConnection()

2)　当给定连接字符串的时候初始化一个新的 GBaseConnection 类实例。

GBaseConnection(string)

- 注释

当创建一个新的 GBaseConnection 实例的时候，其属性设置为下面的初始值，在 ConnectionString 属性中可以重新设置新值。用户可以使用 ConnectionString 属性来改变这些属性值。

初始化时属性初始值

属　性	初始值
ConnectionString	“”
ConnectionTimeout	15

属　性	初始值
Database	“”
DataSource	“”
UseCompression	false

13.3.2.1 GBaseConnection 构造函数 ()

初始化一个新的 GBaseConnection 类实例。

- 语法

[Visual Basic]

```
Public Sub New()
```

[C#]

```
public GBaseConnection()
```

13.3.2.2 GBaseConnection 构造函数 (String)

当给定连接字符串的时候初始化一个新的 GBaseConnection 类实例。

- 语法

[Visual Basic]

```
Public Sub New(ByVal connectionString As String)
```

[C#]

```
public GBaseConnection(string connectionString)
```

- 参数

1)　connectionString ：用于打开 GBase 数据库的连接属性。

13.3.3 GBaseConnection 属性

13.3.3.1 ConnectionString 属性

获取或设置用于连接 GBase 数据库的字符串。

- 语法

[Visual Basic]

```
Public Overrides Property ConnectionString As String
    Get
    Set
```

[C#]

```
public override string ConnectionString { get; set; }
```

- 实现

IDbConnection.ConnectionString

- 注释

1) 用户可以使用 ConnectionString 属性连接数据库。

下面的例子说明了一个典型的连接字符串。

```
" database=gbase;server=GBaseServer;User id=gsUser;Password=gsPass;pooling=false"
```

2) ConnectionString 属性只能在连接打开前设置。

在实际编码过程中，当使用连接串连接到 GBase 数据库后，ConnectionString 将不会包括安全敏感信息，如：Password，除非 ConnectionString 中的 Persist Security Info 参数设置为 true。

3) GBase ADO.NET 会在运行时打开连接前进行语法分析。

如果语法上包含错误，就会出现运行时异常。其它错误只有当尝试打开连接时才会出现，如：用户名或密码错误等。

4） 连接字符串的基础格式要使用特定格式。

连接字符串的基础格式包含使用分号分割开的键/值对，等于号(=)连接每个键和值。值中要包含前面或者后面的空格，可以使用单引号引用，并且空格在值中不会忽略。单引号可以用于连接字符串而不用使用分隔符（例如，Data Source= gs'Server)。若要在值中要包含一个等号（=），连接串则可以这样写"key=’ hello=world’ "，关键词是 "key" 值是"’ hello=world’ "。关键词对大小写不敏感。

5） 有关 ConnectionString 中关键词的详细介绍，请参考文档的“5.1.1 连接字符串”中的内容。

- 示例

下面的例子演示如何创建一个 GBaseConnection 并设置它的一些属性。

```
[Visual Basic]
Public Sub CreateConnection()
    Dim gsConnection As New GBaseConnection()
    gsConnection.ConnectionString = "Persist
SecurityInfo=False;database=gsDB;server=gsHost;Connect
Timeout=30;user id=gsUser; pwd=gsPass;pooling=false"
    gsConnection.Open()
End Sub 'CreateConnection

[C#]
public void CreateConnection()
{
    GBaseConnection gsConnection = new GBaseConnection();
    gsConnection.ConnectionString = "Persist
SecurityInfo=False;database=gsDB;server=gsHost;Connect
Timeout=30;user id=gsUser; pwd=gsPass";
```

```
    gsConnection.Open();
}
```

13.3.3.2 ConnectionTimeout 属性

获取或设置连接超时时间，值为“0”时没有限制。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property ConnectionTimeout As Integer
    Get
```

[C#]

```
public override int ConnectionTimeout { get; }
```

- 实现

IDbConnection.ConnectionTimeout

- 异常

ArgumentException 异常

异常类型	条　件
ArgumentException	ConnectionTimeout 设置的值非法

- 示例

下面的例子创建一个 GBaseConnection 并在连接字符串中设置它的一些属性。

[Visual Basic]

```
Public Sub CreateSqlConnection()
    Dim gsConnection As New GBaseConnection()
    gsConnection.ConnectionString = "Persist Security"_
```

```
        &"Info=False;Username=user;Password=pass;database=test1; "_
        &"server=localhost;Connect Timeout=30;pooling=false "
        gsConnection.Open()
    End Sub

    [C#]
    public void CreateSqlConnection()
    {
        GBaseConnection gsConnection = new GBaseConnection();
        gsConnection.ConnectionString = "Persist
SecurityInfo=False;Username=user;
Password=pass;database=test1;server=localhost;Connect
Timeout=30;pooling=false";
        gsConnection.Open();
    }
```

13.3.3.3 Database 属性

获得当前数据库的名字或在连接打开后使用的数据库名字。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property Database As String
    Get
```

[C#]

```
public override string Database { get; }
```

- 实现

IDbConnection.Database

- 注释

Database 属性不能动态更新，如果用户想改变数据库可以使用

ChangeDatabase 方法改变当前数据库，此时这个属性会变为新的数据库值。

- 示例

下面的例子创建了一个 GBaseConnection 并且显示了它的一些只读属性。

```
[Visual Basic]
Public Sub CreateGBaseConnection()
    Dim gsConnString As String = _
    "Persist SecurityInfo=False;
database=test;server=localhost;"_
    &"user id=joeuser;pwd=pass;pooling=false"
    Dim gsConnection As New GBaseConnection( gsConnString )
    gsConnection.Open()
    MessageBox.Show( "Server Version: " +
gsConnection.ServerVersion _
                        + ControlChars.NewLine + "Database: " +_
    gsConnection.Database )
    gsConnection.ChangeDatabase( "test2" )
    MessageBox.Show( "ServerVersion: " +
gsConnection.ServerVersion _
                        + ControlChars.NewLine + "Database: " +_
    gsConnection.Database )
    gsConnection.Close()
End Sub


[C#]
public void CreateGBaseConnection()
{
    string gsConnString = "Persist Security
Info=False;database=test;server=localhost;user
id=joeuser;pwd=pass;pooling=false";
    GBaseConnection gsConnection = new
GBaseConnection( gsConnString );
    gsConnection.Open();
    MessageBox.Show( "Server Version: " +
gsConnection.ServerVersion  + "\nDatabase: " + gsConnection.Database );
```

```
        gsConnection.ChangeDatabase( "test2" );
        MessageBox.Show( "ServerVersion: " +
gsConnection.ServerVersion  + "\nDatabase: " + gsConnection.Database );
        gsConnection.Close();
    }
```

13.3.3.4 DataSource 属性

获得要连接的 GBase 服务器的名字。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property DataSource As String
    Get
```

[C#]

```
public override string DataSource { get; }
```

13.3.3.5 ServerThread 属性

返回这个连接在服务器上的线程 ID。

- 语法

[Visual Basic]

```
Public ReadOnly Property ServerThread As Integer
    Get
```

[C#]

```
public int ServerThread { get; }
```

13.3.3.6 ServerVersion 属性

一个包含连接的 GBase 数据库版本的字符串。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property ServerVersion As String
    Get
```

[C#]

```
public override string ServerVersion { get; }
```

- 异常

InvalidOperationException 异常

异常类型	条　件
InvalidOperationException	连接关闭。

- 示例

下面的例子将创建一个 GBaseConnection,打开并显示它的一些属性后关闭连接。

[Visual Basic]

```
Public Sub CreateGBaseConnection(gsConnString As String)
    Dim gsConnection As New GBaseConnection(gsConnString)
    gsConnection.Open()
    MessageBox.Show("ServerVersion: " + gsConnection.ServerVersion
+ "\nState: " + gsConnection.State.ToString())
    gsConnection.Close()
End Sub
```

[C#]

```
public void CreateGBaseConnection(string gsConnString)
```

```
    {
        GBaseConnection gsConnection = new
GBaseConnection(gsConnString);
        gsConnection.Open();
        MessageBox.Show("ServerVersion: " + gsConnection.ServerVersion
+ "\nState: " + gsConnection.State.ToString());
        gsConnection.Close();
    }
```

13.3.3.7 State 属性

获得当前连接的状态。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property State As ConnectionState
    Get
```

[C#]

```
public override ConnectionState State { get; }
```

- 实现

IDbConnection.State

- 注释

允许的值有:

1) 从 Closed 到 Open, 使用连接对象的 Open 方法;

2) 从 Open 到 Closed, 使用连接对象的 Close 方法 或 Dispose 方法。

- 示例

下面的例子创建一个 GBaseConnection,打开它显示它的一些属性然后关闭连接。

```
[Visual Basic]
Public Sub CreateGBaseConnection(gsConnString As String)
    Dim gsConnection As New GBaseConnection(gsConnString)
    gsConnection.Open()
    MessageBox.Show("ServerVersion: " + gsConnection.ServerVersion
+ "\nState: " + gsConnection.State.ToString())
    gsConnection.Close()
End Sub

[C#]
public void CreateGBaseConnection(string gsConnString)
{
    GBaseConnection gsConnection = new
GBaseConnection(gsConnString);
    gsConnection.Open();
    MessageBox.Show("ServerVersion: " + gsConnection.ServerVersion
+ "\nState: " + gsConnection.State.ToString());
    gsConnection.Close();
}
```

13.3.3.8 UseCompression 属性

获取或设置这个连接与服务器通信时是否使用压缩协议。

- 语法

```
[Visual Basic]
Public ReadOnly Property UseCompression As Boolean
    Get
[C#]
public bool UseCompression { get; }
```

13.3.4 GBaseConnection 方法

13.3.4.1 BeginTransaction 方法

- 重载列表

1) 开始一个数据库事务。

BeginTransaction()

2) 使用一个指定的隔离级别开始一个数据库事务。

BeginTransaction(IsolationLevel)

13.3.4.1.1 BeginTransaction 方法 ()

开始一个数据库事务。

- 语法

[Visual Basic]

Public Function BeginTransaction As GBaseTransaction

[C#]

public GBaseTransaction BeginTransaction()

- 返回值

代表新事务的对象。

- 注释

这个命令等价于在 GBase 数据库中执行 BEGIN TRANSACTION 命令，用户必须使用 Commit 方法提交事务，使用 Rollback 方法回滚事务。

如果用户没有指定一个隔离级别，会使用默认的隔离级别。要使用

BeginTransaction 指定一个隔离级别,可以使用带有 IsolationLevel 的重载函数。

- 异常

InvalidOperationException 异常

异常类型	条　件
InvalidOperationException	不支持并发事务。

- 示例

下面的例子中演示了如何使用 BeginTransaction 的 Commit 和 Rollback 方法。

```
[Visual Basic]
Public Sub RunTransaction(gsConnString As String)
    Dim gsConnection As New GBaseConnection(gsConnString)
    gsConnection.Open()
    Dim gsCommand As GBaseCommand = gsConnection.CreateCommand()
    Dim gsTrans As GBaseTransaction
    ' Start a local transaction
    gsTrans = gsConnection.BeginTransaction()
    ' Must assign both transaction object and connection
    ' to Command object for a pending local transaction
    gsCommand.Connection = gsConnection
    gsCommand.Transaction = gsTrans
    Try
        gsCommand.CommandText = "Insert into Test (id, desc) VALUES"
_ &" (100, 'Description')"
        gsCommand.ExecuteNonQuery()
        gsCommand.CommandText = "Insert into Test (id, desc) VALUES"
_ &" (101, 'Description')"
        gsCommand.ExecuteNonQuery()
        gsTrans.Commit()
        Console.WriteLine("Both records are written to database.")
        Catch e As Exception
        Try
```

```
                gsTrans.Rollback()
                Catch ex As GBaseException
                If Not gsTrans.Connection Is Nothing Then
                    Console.WriteLine("An exception of type " + _
    ex.GetType().ToString() + " was encountered while attempting to roll
back the transaction.")
                End If
            End Try
            Console.WriteLine("An exception of type " +
e.GetType().ToString() + "was encountered while inserting the data.")
            Console.WriteLine("Neither record was written to
database.")
            Finally
                gsConnection.Close()
        End Try
    End Sub
    [C#]
    public void RunTransaction(string gsConnString)
    {
        GBaseConnection gsConnection = new
GBaseConnection(gsConnString);
        gsConnection.Open();
        GBaseCommand gsCommand = gsConnection.CreateCommand();
        GBaseTransaction gsTrans;
        // Start a local transaction
        gsTrans = gsConnection.BeginTransaction();
        // Must assign both transaction object and connection
        // to Command object for a pending local transaction
        gsCommand.Connection = gsConnection;
        gsCommand.Transaction = gsTrans;
        try
        {
            gsCommand.CommandText = "insert into Test (id, desc) VALUES
            (100, 'Description')";
            gsCommand.ExecuteNonQuery();
            gsCommand.CommandText = "insert into Test (id, desc) VALUES
```

```
            (101, 'Description')";
            gsCommand.ExecuteNonQuery();
            gsTrans.Commit();
            Console.WriteLine("Both records are written to database.");
        }
        catch(Exception e)
        {
            try
            {
                gsTrans.Rollback();
            }
            catch (SqlException ex)
            {
                if (gsTrans.Connection != null)
                {
                    Console.WriteLine("An exception of type " +
ex.GetType() +" was encountered while attempting to roll back the
transaction.");
                }
            }
            Console.WriteLine("An exception of type " + e.GetType() +
                        " was encountered while inserting the data.");
            Console.WriteLine("Neither record was written to
database.");
        }
        finally
        {
            gsConnection.Close();
        }
    }
```

13.3.4.1.2 BeginTransaction 方法 (IsolationLevel)

使用一个指定的隔离级别开始一个数据库事务。

- 语法

```
[Visual Basic]
Public Function BeginTransaction ( _
    iso As IsolationLevel _
) As GBaseTransaction
[C#]
public GBaseTransaction BeginTransaction(IsolationLevel iso)
```

- 参数

1) iso ： 指定事务使用的隔离级别。

- 返回值

代表新事务的对象。

- 异常

InvalidOperationException 异常

异常类型	条　件
InvalidOperationException	不支持并发异常。

13.3.4.2 ChangeDatabase 方法

为一个打开的 GBaseConnection 改变当前数据库。

- 语法

```
[Visual Basic]
Public Overrides Sub ChangeDatabase ( _
    databaseName As String _
)
```

[C#]

```
public override void ChangeDatabase(
    string databaseName
)
```

- 参数

1) databaseName ： 要使用的数据库名字。

- 实现

IDbConnection.ChangeDatabase(String)

- 注释

在 database 参数中提供的数值必须是一个有效的数据库名字。database 参数不能包含 null 值、一个空字符串、或者一个有空白字符的字符串。

当用户对 GBase 使用连接池的时候，用户关闭连接，它会返回连接池。下一次连接会从池中返回，重置连接请求要在用户执行任何操作之前完成。

- 异常

异常

异常类型	条　件
ArgumentException	数据库名无效。
InvalidOperationException	连接没有打开。
GBaseException	不能更改数据库。

- 示例

下面的列子创建了一个 GBaseConnection 并显示了它的一些只读属性。

[Visual Basic]

```
Public Sub CreateGBaseConnection()
    Dim gsConnString As String =
            "Persist Security
```

```
Info=False;database=test;server=localhost; "_
        &"user id=joeuser;pwd=pass;pooling=false"

        Dim gsConnection As New GBaseConnection(gsConnString)
        gsConnection.Open()
        MessageBox.Show( "Server Version: " +
gsConnection.ServerVersion _
                            + ControlChars.NewLine + "Database: " + _
gsConnection.Database )
        gsConnection.ChangeDatabase( "test2" )
        MessageBox.Show( "ServerVersion: " +
gsConnection.ServerVersion _
                            + ControlChars.NewLine + "Database: " + _
gsConnection.Database )
        gsConnection.Close()
    End Sub

    [C#]
    public void CreateGBaseConnection()
    {
        string gsConnString = "Persist Security
Info=false;database=test;server=localhost; " _
    & "user id=joeuser;pwd=pass;pooling=false";
        GBaseConnection gsConnection = new
GBaseConnection( gsConnString );
        gsConnection.Open();
        MessageBox.Show("Server Version:"+gsConnection.ServerVersion
+ "\nDatabase: " + gsConnection.Database );
        gsConnection.ChangeDatabase( "test2" );
        MessageBox.Show( "ServerVersion: "  +
gsConnection.ServerVersion + "\nDatabase: " + gsConnection.Database );
        gsConnection.Close();
    }
```

13.3.4.3 Close 方法

关闭数据库的连接。

- 语法

[Visual Basic]

```
Public Overrides Sub Close
```

[C#]

```
public override void Close()
```

- 实现

```
IDbConnection.Close()
```

- 注释

1) Close 方法会回滚任何未完成的事务

2) 如果在应用程序中开启连接池，则 Close 方法调用后会释放连接到池中

- 示例

下面的例子创建了一个 GBaseConnection 后打开，显示它的一些属性然后关闭连接。

[Visual Basic]

```
Public Sub CreateGBaseConnection(gsConnString As String)
    Dim gsConnection As New GBaseConnection(gsConnString)
    gsConnection.Open()
    MessageBox.Show("ServerVersion: " + gsConnection.ServerVersion
+ ControlChars.NewLine + "State: " + gsConnection.State.ToString())
    gsConnection.Close()
End Sub
```

[C#]

```
    public void CreateGBaseConnection(string gsConnString)
    {
        GBaseConnection gsConnection = new
GBaseConnection(gsConnString);
        gsConnection.Open();
        MessageBox.Show("ServerVersion: " + gsConnection.ServerVersion
+ "\nState: " + gsConnection.State.ToString());
        gsConnection.Close();
    }
```

13.3.4.4 CreateCommand 方法

创建并返回一个与 GBaseConnection 关联的 GBaseCommand 对象。

- 语法

[Visual Basic]

```
Public Function CreateCommand As GBaseCommand
```

[C#]

```
public GBaseCommand CreateCommand()
```

- 返回值

一个 GBaseCommand 对象。

13.3.4.5 Open 方法

使用 ConnectionString 指定的参数打开一个数据库连接。

- 语法

[Visual Basic]

```
Public Overrides Sub Open
```

[C#]

public void Open()

- 实现

IDbConnection.Open()

- 注释

如果有可用的则 GBaseConnection 从连接池中获取连接，否则它建立新的到 GBase 的连接实例。

- 异常

异常

异常类型	条　件
InvalidOperationException	不能打开没有指定数据源或者服务器的连接。
GBaseException	打开连接时出现连接等级（connection-level）错误。

- 示例

下面的例子创建了一个 GBaseConnection 对象并打开，显示它的一些属性后关闭连接。

```
[Visual Basic]
Public Sub CreateGBaseConnection(gsConnString As String)
    Dim gsConnection As New GBaseConnection(gsConnString)
    gsConnection.Open()
    MessageBox.Show("ServerVersion: " + gsConnection.ServerVersion
+ ControlChars.NewLine + "State: " + gsConnection.State.ToString())
    gsConnection.Close()
End Sub

[C#]
public void CreateGBaseConnection(string gsConnString)
{
```

```
        GBaseConnection gsConnection = new
GBaseConnection(gsConnString);
        gsConnection.Open();
        MessageBox.Show("ServerVersion: " + gsConnection.ServerVersion
+ "\nState: " + gsConnection.State.ToString());
        gsConnection.Close();
    }
```

13.3.4.6 Ping 方法

判断是否能够与服务器正常通讯，此方法需要正常连接后才能使用。

- 语法

[Visual Basic]

```
Public Function Ping As Boolean
```

[C#]

```
public bool Ping()
```

13.3.5 GBaseConnection 事件

13.3.5.1 InfoMessage 事件

当 GBase 数据源执行命令或者查询结果返回警告时，该事件发生。

- 语法

[Visual Basic]

```
Public Event InfoMessage As GBaseInfoMessageEventHandler
```

[C#]

```
public event GBaseInfoMessageEventHandler InfoMessage
```

13.4 GBaseDataAdapter 类

GBaseDataAdapter 是 DataSet 和 GBase 数据库之间的桥接器，是 DataSet 和 GBase 数据库之间的桥接器，用于检索和保存数据。此类不能被继承。

对于该类所有成员的列表，参考 GBaseDataAdapter 成员。

- 继承层次

System.Object

|__ System.MarshalByRefObject

|__ System.ComponentModel.Component

|__ System.Data.Common.DataAdapter

|__ System.Data.Common.DbDataAdapter

|__ GBase.Data.GBaseClient.GBaseDataAdapter

- 语法

[Visual Basic]

```
Public NotInheritable Class GBaseDataAdapter _
    Inherits DbDataAdapter _
    Implements IDbDataAdapter, IDataAdapter, ICloneable
```

[C#]

```
public sealed class GBaseDataAdapter: DbDataAdapter,
        IDbDataAdapter, IDataAdapter, ICloneable
```

- 必要条件

命名空间: GBase.Data.GBaseClient

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 注释

GBaseDataAdapter 是 DataSet 和 GBase Server 之间的桥接器，用于检索和保存数据。GBaseDataAdapter 通过对数据源使用适当的 SQL 语句映射 Fill（将数据源中的数据填充到 DataSet）和 Update（将 DataSet 的数据变更写回到数据源中）来提供这一桥接。更新是逐行进行的。对于每个已插入、修改和删除的行，Update 方法会确定其执行类型（Insert、Update 或 Delete）。

根据更改类型，执行 Insert、 Update 或 Delete 命令模板将已修改的行传递给数据源。 当 GBaseDataAdapter 填充 DataSet 时，它为返回的数据创建表和列（如果这些表和列尚不存在）。但是，除非 MissingSchemaAction 属性设置为 AddWithKey，否则这个隐式创建的架构中不包括主键信息。也可以使用 FillSchema，让 GBaseDataAdapter 创建 DataSet 的架构，并在用数据填充它之前就将主键信息包括进去。

- 示例

在下面的例子中，创建了一个 GBaseCommand 和一个 GBaseConnection。GBaseConnection 为打开状态并设 Connection 用于 GBaseCommand。例子然后调用了 ExecuteNonQuery 并关闭连接。要完成成这些，向 ExecuteNonQuery 传送了连接字符串和查询字符串，是 INSERT 语句。

```
[Visual Basic]
Public Function SelectRows(dataSet As DataSet, connection As String,
query_ As String) As DataSet
        Dim conn As New GBaseConnection(connection)
        Dim adapter As New GBaseDataAdapter()
        adapter.SelectCommand = new GBaseCommand(query, conn)
        adapter.Fill(dataset)
        Return dataset
End Function
[C#]
```

```
    public DataSet SelectRows(DataSet dataset,string connection,string
query)
    {
        GBaseConnection conn = new GBaseConnection(connection);
        GBaseDataAdapter adapter = new GBaseDataAdapter();
        adapter.SelectCommand = new GBaseCommand(query, conn);
        adapter.Fill(dataset);
        return dataset;
    }
```

13.4.1 GBaseDataAdapter 成员

公共构造函数

构造函数	描 述
GBaseDataAdapter	构造函数，初始化一个新的 GBaseDataAdapter 对象。

公共属性

属 性	描 述
AcceptChangesDuringFill (继承于 DataAdapter)	获取或设置一个值，该值指示在任何 Fill 操作过程中，在将 AcceptChanges 添加到 DataTable 之后是否在 DataRow 上调用它。
Container (继承于 Component)	获得包含 Component 的 IContainer。
ContinueUpdateOnError (继承于 DataAdapter)	获取或设置一个值指明是否当行更新出现错误时生成一个异常。
DeleteCommand	获取或设置一个 SQL 语句或存储过程用于从 DataSet 删除记录。
InsertCommand	获取或设置一个 SQL 语句或存储过程用于插入记录到 DataSet。
MissingMappingAction	获取或设置当传入的数据没有匹配的

属　性	描　述
(继承于 DataAdapter)	表或列时需要执行的操作。
MissingSchemaAction (继承于 DataAdapter)	设置或获取当现有 DataSet 架构与传入数据不匹配时需要执行的操作。
SelectCommand	获取或设置一个 SQL 语句或存储过程用于从数据源中选择数据。
Site (继承于 Component)	获取或设置 Component 的 ISite。
TableMappings (继承于 DataAdapter)	获得用于提供一个源表和一个 DataTable 之间主服务器映射的集合。
UpdateCommand	获取或设置一个 SQL 语句或存储过程用于更新数据源中数据。

公共方法

方　法	描　述
Dispose (继承于 Component)	释放 Component 使用的资源。
Equals (继承于 Object)	判断指定的对象是否等于当前的对象。
Fill (继承于 DbDataAdapter)	重载函数。使用 SelectCommand 的结果集填充 DataSet。
FillSchema (继承于 DbDataAdapter)	重载函数。将名为“Table”的 DataTable 添加到指定的 DataSet 中，并根据指定的 SchemaType 配置架构以匹配数据源中的架构。
GetFillParameters (继承于 DbDataAdapter)	获得当用户执行 SELECT 语句时使用的参数集。
GetType (继承于 Object)	获取当前实例的类型。
ToString (继承于 Component)	返回类的完全限定名。

方　法	描　述
Update （继承于 DbDataAdapter）	重载函数。分别调用 INSERT，UPDATE 或 DELETE 语句来对 DataSet 中的行进行插入更新和删除。

事件

事　件	描　述
Disposed （继承于 Component）	当组件通过调用 Dispose() 方法销毁时发生。
FillError （继承于 DbDataAdapter）	在 fill 操作过程中发生错误时返回。
RowUpdated	在针对一个数据源执行命令之后的更新过程中发生。
RowUpdating	在针对一个数据源执行命令之前的更新过程中发生。

13.4.2　GBaseDataAdapter 构造函数

初始化 GBaseDataAdapter 类的一个新实例。

- 重载列表

1）　初始化 GBaseDataAdapter 类的一个新实例。

GBaseDataAdapter()

2）　使用 GBaseCommand 作为 SelectCommand 属性初始化 GBaseDataAdapter 类的一个新实例。

GBaseDataAdapter(GBaseCommand)

3）　使用 SelectCommand 和一个 GBaseConnection 对象初始化 GBaseDataAdapter 类的一个新实例。

GBaseDataAdapter(String,GBaseConnection)

4) 使用 SelectCommand 和一个连接字符串初始化 GBaseDataAdapter 类的一个新实例。

GBaseDataAdapter(String,String)

- 注释

当 GBaseDataAdapter 的实例被创建，下面的属性设置为初始值。用户可以通过独立的调用这些属性来改变它们的数值。

初始化时设置的属性初始值

属　性	初始值
MissingMappingAction	MissingMappingAction.Passthrough
MissingSchemaAction	MissingSchemaAction.Add

- 示例

下面的例子创建了一个 GBaseDataAdapter 对象并设置它的一些属性。

```
[Visual Basic]
Public Sub CreateSqlDataAdapter()
    Dim conn As GBaseConnection = New GBaseConnection("Data" _
     & "Source=localhost; database=test;pooling=false")
    Dim da As GBaseDataAdapter = New GBaseDataAdapter
    da.MissingSchemaAction = MissingSchemaAction.AddWithKey
    da.SelectCommand = New GBaseCommand("SELECT id, name" _
                            & "FROM gstable", conn)
    da.InsertCommand = New GBaseCommand("INSERT INTO gstable" _   &
" (id, name) VALUES (?id, ?name)", conn)
    da.UpdateCommand = New GBaseCommand("UPDATE gstable" _
                    & "SET id=?id, name=?name WHERE id=?oldId", conn)
    da.DeleteCommand = New GBaseCommand("DELETE FROM gstable" _ &
"WHERE id=?id", conn)
    da.InsertCommand.Parameters.Add("?id", GBaseDbType.VarChar,
5,_
  "id")
    da.InsertCommand.Parameters.Add("?name",
```

```
GBaseDbType.VarChar,_
    40, "name")
        da.UpdateCommand.Parameters.Add("?id", GBaseDbType.VarChar, _
    5, "id")
        da.UpdateCommand.Parameters.Add("?name",GBaseDbType.VarChar,
40, "name")
        da.UpdateCommand.Parameters.Add("?oldId",
GBaseDbType.VarChar,_
         5, "id").SourceVersion = DataRowVersion.Original
        da.DeleteCommand.Parameters.Add("?id", GBaseDbType.VarChar,_
    5, "id").SourceVersion = DataRowVersion.Original
    End Sub

    [C#]
    Public Static void CreateSqlDataAdapter()
    {
        GBaseConnection conn = new GBaseConnection("Data
Source=localhost;database=test;pooling=false");
        GBaseDataAdapter da = new GBaseDataAdapter();
        da.MissingSchemaAction = MissingSchemaAction.AddWithKey;
        da.SelectCommand = new GBaseCommand("SELECT id, name FROM
gstable", conn);
        da.InsertCommand = new GBaseCommand("INSERT INTO gstable (id,
name) " +"VALUES (?id, ?name)", conn);
        da.UpdateCommand = new GBaseCommand("UPDATE gstable SET id=?id,
name=?name " +"WHERE id=?oldId", conn);
        da.DeleteCommand = new GBaseCommand("DELETE FROM gstable WHERE
id=?id", conn);
        da.InsertCommand.Parameters.Add("?id", GBaseDbType.VarChar, 5,
"id");
        da.InsertCommand.Parameters.Add("?name", GBaseDbType.VarChar,
40, "name");
        da.UpdateCommand.Parameters.Add("?id", GBaseDbType.VarChar, 5,
"id");
        da.UpdateCommand.Parameters.Add("?name", GBaseDbType.VarChar,
    40, "name");
```

```
        da.UpdateCommand.Parameters.Add("?oldId", GBaseDbType.VarChar,
5, "id").SourceVersion = DataRowVersion.Original;
        da.DeleteCommand.Parameters.Add("?id", GBaseDbType.VarChar, 5,
"id").SourceVersion = DataRowVersion.Original;
    }
```

13.4.2.1 GBaseDataAdapter 构造函数 ()

初始化 GBaseDataAdapter 类的一个新实例。

- 语法

[Visual Basic]

```
Public Sub New
```

[C#]

```
public GBaseDataAdapter()
```

13.4.2.2 GBaseDataAdapter 构造函数 (GBaseCommand)

使用 GBaseCommand 作为 SelectCommand 属性初始化 GBaseDataAdapter 类的一个新实例。

- 语法

[Visual Basic]

```
Public Sub New ( _
    selectCommand As GBaseCommand _
)
```

[C#]

```
public GBaseDataAdapter(    GBaseCommand selectCommand )
```

- 参数

1) selectCommand :GBaseCommand 是一个 SELECT 语句或是存储过程并作为 SelectCommand 属性。

13.4.2.3 GBaseDataAdapter 构造函数 (String, GBaseConnection)

使用 SelectCommand 和一个 GBaseConnection 对象初始化 GBaseDataAdapter 类的一个新实例。

- 语法

[Visual Basic]

```
Overloads Public Sub New( _
ByVal selectCommandText As String, _
ByVal conn As GBaseConnection _
)
```

[C#]

```
public GBaseDataAdapter(string selectCommandText,GBaseConnection conn)
```

- 参数

1) selectCommandText: 一个可以是 SQL SELECT 语句或者存储过程的字符串，用于 GBaseDataAdapter 的 SelectCommand 属性。

2) conn: 代表连接的 GBaseConnection。

13.4.2.4 GBaseDataAdapter 构造函数 (String, String)

使用 SelectCommand 和一个连接字符串初始化 GBaseDataAdapter 类的一个新实例。

- 语法

[Visual Basic]

```
Overloads Public Sub New(_
ByVal selectCommandText As String, _
ByVal selectConnString As String _
)
```

[C#]

```
public GBaseDataAdapter(string selectCommandText,string selectConnString);
```

- 参数

1) selectCommandText :一个可以是 SELECT 语句或者存储过程的字符串，用于 GBaseDataAdapter 的 SelectCommand 属性;

2) selectConnString : 连接字符串。

13.4.3 GBaseDataAdapter 属性

13.4.3.1 DeleteCommand 属性

获取或设置一个 SQL 语句或存储过程用于从数据集删除记录。

- 语法

[Visual Basic]

```
Public Property DeleteCommand As GBaseCommand
    Get
    Set
[C#]
public GBaseCommand DeleteCommand { get; set; }
```

- 注释

在 Update 时，如果这个属性没有设置且主键信息没有在 DataSet 中，如果用户设置了 SelectCommand 属性并使用 GBaseCommandBuilder，那么 DeleteCommand 可以自动生成。而后任何用户没有设置的命令会由 GBaseCommandBuilder 生成，这个生成逻辑需要在 DataSet 中有主键信息。

- 示例

下面的例子中演示如何创建 GBaseDataAdapter 对象并设置 SelectCommand 和 DeleteCommand 属性。

```
[Visual Basic]
Public Shared Function CreateCustomerAdapter(conn As
GBaseConnection) _ As GBaseDataAdapter
    Dim da As GBaseDataAdapter = New GBaseDataAdapter()
    Dim cmd As GBaseCommand
    Dim parm As GBaseParameter
    ' Create the SelectCommand.
    cmd = New GBaseCommand("SELECT * FROM gstable WHERE id=?"_
&"id AND name=?name", conn)
    cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15)
    cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15)
    da.SelectCommand = cmd
    ' Create the DeleteCommand.
    cmd = New GBaseCommand("DELETE FROM gstable WHERE id=?id"_
, conn)
    parm = cmd.Parameters.Add("?id", GBaseDbType.VarChar, 5, "id")
    parm.SourceVersion = DataRowVersion.Original
```

```
        da.DeleteCommand = cmd
        Return da
    End Function

    [C#]
    Public Static GBaseDataAdapter
CreateCustomerAdapter(GbaseConnection  conn)
    {
        GBaseDataAdapter da = new GBaseDataAdapter();
        GBaseCommand cmd;
        GBaseParameter parm;
        // Create the SelectCommand.
        cmd = new GBaseCommand("SELECT * FROM gstable WHERE id=?id
                                AND name=?name", conn);
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15);
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15);
        da.SelectCommand = cmd;
        // Create the DeleteCommand.
        cmd = new GBaseCommand("DELETE FROM gstable WHERE id=?id",
conn);
        parm = cmd.Parameters.Add("?id", GBaseDbType.VarChar, 5, "id");
        parm.SourceVersion = DataRowVersion.Original;
        da.DeleteCommand = cmd;
        return da;
    }
```

13.4.3.2 InsertCommand 属性

获取或设置一个 SQL 语句或存储过程，用于将记录插入到数据集。

- 语法

[Visual Basic]

```
Public Property InsertCommand As GBaseCommand
```

Get

Set

[C#]

```
public GBaseCommand InsertCommand {get; set;}
```

- 注释

在 Update 时，如果这个属性没有设置且主键信息没有在 DataSet 中，这时用户设置了 SelectCommand 属性并使用 GBaseCommandBuilder，那么 InsertCommand 可以自动生成。而后任何用户没有设置的命令会由 GBaseCommandBuilder 生成，这个生成逻辑需要在 DataSet 中有主键信息。

- 示例

下面的例子中演示如何创建 GBaseDataAdapter 对象并设置 SelectCommand 和 InsertCommand 属性。

[Visual Basic]

```
Public Shared Function CreateCustomerAdapter(conn As
GBaseConnection) _ As GBaseDataAdapter
        Dim da As GBaseDataAdapter = New GBaseDataAdapter()
        Dim cmd As GBaseCommand
        Dim parm As GBaseParameter
        ' Create the SelectCommand.
        cmd = New GBaseCommand("SELECT * FROM gstable WHERE id=?"_
        &"id AND name=?name", conn)
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15)
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15)
        da.SelectCommand = cmd
        ' Create the InsertCommand.
        cmd = New GBaseCommand("INSERT INTO gstable (id,name) "_
        &"VALUES(?id, ?name)", conn)
        cmd.Parameters.Add( "?id", GBaseDbType.VarChar, 15, "id" )
        cmd.Parameters.Add( "?name", GBaseDbType.VarChar, 15, "name" )
        da.InsertCommand = cmd
```

```
        Return da
    End Function

    [C#]
    Public Static GBaseDataAdapter
CreateCustomerAdapter(GBaseConnection conn)
    {
        BaseDataAdapter da = new GBaseDataAdapter();
        GBaseCommand cmd;
        GBaseParameter parm;
        // Create the SelectCommand.
        cmd = new GBaseCommand("SELECT * FROM gstable WHERE id=?id AND
name=?name", conn);
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15);
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15);
        da.SelectCommand = cmd;
        // Create the InsertCommand.
        cmd = new GBaseCommand("INSERT INTO gstable (id,name) VALUES
(?id,?name)", conn);
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15, "id" );
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15, "name" );
        da.InsertCommand = cmd;
        return da;
    }
```

13.4.3.3 SelectCommand 属性

获取或设置一个 SQL 语句或存储过程，用于从数据源中选择数据。

- 语法

```
[Visual Basic]

Public Property SelectCommand As GBaseCommand

    Get
```

Set

[C#]

```
public GBaseCommand SelectCommand {get; set;}
```

- 属性值

是一个 GBaseCommand，从数据源中选择数据。

- 注释

如果 SelectCommand 没有返回任何行，在使用 Fill 方法时就不会有表加入到 DataSet，也不会出现异常。

- 示例

下面的例子中演示如何创建 GBaseDataAdapter 对象并设置 SelectCommand 和 InsertCommand 属性。

[Visual Basic]

```
Public Shared Function CreateCustomerAdapter(conn As
GBaseConnection)_ As GBaseDataAdapter
        Dim da As GBaseDataAdapter = New GBaseDataAdapter()
        Dim cmd As GBaseCommand
        Dim parm As GBaseParameter
        ' Create the SelectCommand.
        cmd = New GBaseCommand("SELECT * FROM gstable WHERE id=?"_
        &"id AND name=?name", conn)
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15)
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15)
        da.SelectCommand = cmd
        ' Create the InsertCommand.
        cmd = New GBaseCommand("INSERT INTO gstable (id,name) "_
        &"VALUES (?id, ?name)", conn)
        cmd.Parameters.Add( "?id", GBaseDbType.VarChar, 15, "id" )
        cmd.Parameters.Add( "?name", GBaseDbType.VarChar, 15, "name" )
        da.InsertCommand = cmd
        Return da
```

```
    End Function

    [C#]
    Public Static GBaseDataAdapter
CreateCustomerAdapter(GBaseConnection conn)
    {
        GBaseDataAdapter da = new GBaseDataAdapter();
        GBaseCommand cmd;
        GBaseParameter parm;
        // Create the SelectCommand.
        cmd = new GBaseCommand("SELECT * FROM gstable WHERE id=?id
         AND name=?name", conn);
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15);
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15);
        da.SelectCommand = cmd;
        // Create the InsertCommand.
        cmd = new GBaseCommand("INSERT INTO gstable (id,name) VALUES
        (?id,?name)", conn);
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15, "id" );
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15, "name" );
        da.InsertCommand = cmd;
        return da;
    }
```

13.4.3.4 UpdateCommand 属性

获取或设置一个 SQL 语句或者存储过程，用于更新数据源中的记录。

- 语法

```
[Visual Basic]

Public Property UpdateCommand As GBaseCommand

    Get

    Set
```

[C#]

```
public GBaseCommand UpdateCommand {get; set;}
```

- 注释

在 Update 时，如果这个属性没有设置且主键信息没有在 DataSet 中，如果用户设置了 SelectCommand 属性并使用 GBaseCommandBuilder，那么 UpdateCommand 可以自动生成。而后任何用户没有设置的命令会由 GBaseCommandBuilder 生成，这个生成逻辑需要在 DataSet 中有主键信息。

- 示例

下面的例子中演示如何创建 GBaseDataAdapter 对象并设置 SelectCommand 和 InsertCommand 属性。

[Visual Basic]

```
Public Shared Function CreateCustomerAdapter(conn As
GBaseConnection)_ As GBaseDataAdapter
        Dim da As GBaseDataAdapter = New GBaseDataAdapter()
        Dim cmd As GBaseCommand
        Dim parm As GBaseParameter
        ' Create the SelectCommand.
        cmd = New GBaseCommand("SELECT * FROM gstable WHERE id=?"_
        &"id AND name=?name", conn)
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15)
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15)
        da.SelectCommand = cmd
        ' Create the UpdateCommand.
        cmd = New GBaseCommand("UPDATE gstable SET id=?id, name=?"_
        &"name WHERE id=?oldId", conn)
        cmd.Parameters.Add( "?id", GBaseDbType.VarChar, 15, "id" )
        cmd.Parameters.Add( "?name", GBaseDbType.VarChar, 15, "name" )
        parm = cmd.Parameters.Add("?oldId", GBaseDbType.VarChar, 15,
"id")
        parm.SourceVersion = DataRowVersion.Original
        da.UpdateCommand = cmd
        Return da
```

```
    End Function

    [C#]
    public static GBaseDataAdapter
CreateCustomerAdapter(GbaseConnection  conn)
    {
        GBaseDataAdapter da = new GBaseDataAdapter();
        GBaseCommand cmd;
        GBaseParameter parm;
        // Create the SelectCommand.
        cmd = new GBaseCommand("SELECT * FROM gstable WHERE id=?id AND
name=?name", conn);
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15);
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15);
        da.SelectCommand = cmd;
        // Create the UpdateCommand.
        cmd = new GBaseCommand("UPDATE gstable SET id=?id, name=?name
WHERE id=?oldId", conn);
        cmd.Parameters.Add("?id", GBaseDbType.VarChar, 15, "id" );
        cmd.Parameters.Add("?name", GBaseDbType.VarChar, 15, "name" );
        parm = cmd.Parameters.Add( "?oldId", GBaseDbType.VarChar, 15,
"id" );
        parm.SourceVersion = DataRowVersion.Original;
        da.UpdateCommand = cmd;
        return da;
    }
```

13.4.4 GBaseDataAdapter 方法

13.4.4.1 Fill 方法

重载函数。在 DataSet 中添加或刷新行。该方法详细介绍请参考 MSDN 上 DbDataAdapter.Fill 方法。

- 重载列表

1) 在 DataSet 中添加或刷新行。

public override int Fill(DataSet dataSet)

2) 在 DataTable 中添加或刷新行，限于 SelectCommand 返回单一表结果集的情况。

public int Fill(DataTable dataTable)

3) 在 DataSet 中添加或刷新 srcTable 指定表名的记录。

public int Fill(DataSet dataSet, string srcTable)

4) 在 DataTable 中添加或刷新行，从 startRecord 指定的开始， 数量为 maxRecords 指定的最大数目。

public int Fill(int startRecord, int maxRecords, params DataTable[] dataTables)

5) 在 DataSet 中添加或刷新 srcTable 指定表名的记录，从 startRecord 指定的开始，数量为 maxRecords 指定的最大数目。

public int Fill(DataSet dataSet, int startRecord, int maxRecords, string srcTable)

13.4.4.2 FillSchema 方法

重载函数。填充 DataSet 架构，并根据指定的 SchemaType 配置架构。该方法详细介绍请参考 MSDN 上 DbDataAdapter.FillSchema 方法。

- 重载列表

1) 填充 dataSet 架构，并根据指定的 SchemaType 配置架构。

public override DataTable[] FillSchema(DataSet dataSet,SchemaType schemaType)

2) 根据指定的 SchemaType 配置指定 DataTable 的架构。

public DataTable FillSchema(DataTable dataTable,SchemaType schemaType)

3) 根据 srcTable 指定的表名填充 dataSet 架构，并根据指定的 SchemaType 配置架构。

public DataTable[] FillSchema(DataSet dataSet,SchemaType schemaType,string srcTable)

13.4.4.3 Update 方法

重载函数。将 DataSet 或者 DataRow 中的变更写回数据源。该方法详细介绍请参考 MSDN 上 DbDataAdapter.Update 方法。

- 重载列表

1) 为指定 DataSet 中每个已插入、已更新或已删除的行调用相应的 INSERT、UPDATE 或 DELETE 语句。

public override int Update(DataSet dataSet)

2) 为指定的 DataRow 对象数组中每个已插入、已更新或已删除的行调用相应的 INSERT、UPDATE 或 DELETE 语句。

public int Update(DataRow[] dataRows)

3) 为指定 DataTable 中每个已插入、已更新或已删除的行调用相应的 INSERT、UPDATE 或 DELETE 语句。

public int Update(DataTable dataTable)

4) 为具有指定 DataTable 名称的 DataSet 中每个已插入、已更新或已删除的行调用相应的 INSERT、UPDATE 或 DELETE 语句。

```
public int Update(DataSet dataSet, string srcTable)
```

13.4.5 GBaseDataAdapter 事件

13.4.5.1 RowUpdated 事件

在针对一个数据源执行命令之后的更新过程中发生。如果试图更新，该事件就会发生。

- 语法

[Visual Basic]

```
Public Event RowUpdated As GBaseRowUpdatedEventHandler
```

[C#]

```
public event GBaseRowUpdatedEventHandler RowUpdated
```

- 事件数据

事件句柄接受一个包含关于事件数据的 GBaseRowUpdatedEventArgs 参数。下面的 GBaseRowUpdatedEventArgs 属性提供了特定事件的信息。

属性提供的特定事件的信息

属　性	描　述
Command	事件发生时，获取或设置的 GBaseCommand 对象。
Errors	当执行 Command 的时候，获取任何.NET 框架数据源产生的错误。
RecordsAffected	得到 SQL 语句在执行插入、更新、删除时影响的行数。
Row	当前更新的 DataRow。
StatementType	获取执行的 SQL 语句类型。

属　性	描　述
Status	获取 Command 的 UpdateStatus 属性。
TableMapping	获取 DataTableMapping。

13.4.5.2 RowUpdating 事件

在针对一个数据源执行命令之前的更新过程中发生。如果试图更新，该事件就会发生。

- 语法

[Visual Basic]

```
Public Event RowUpdating As GBaseRowUpdatingEventHandler
```

[C#]

```
public event GBaseRowUpdatingEventHandler RowUpdating;
```

- 事件数据

事件句柄接受一个包含关于事件的数据的 GBaseRowUpdatingEventArgs 参数。下面的 GBaseRowUpdatingEventArgs 属性提供了特定事件的信息。

属性提供的特定事件的信息

属　性	描　述
Command	事件发生时，获取或设置的 GBaseCommand 对象。
Errors	当执行 Command 的时候，获取任何.NET 框架数据源产生的错误。
Row	当前更新的 DataRow。
StatementType	获取执行的 SQL 语句类型。
Status	获取 Command 的 UpdateStatus 属性。
TableMapping	获取 DataTableMapping。

13.5 GBaseDataReader 类

代表 GBase 数据源中只进且只读的数据流，此类不能被继承。对于该类所有成员的列表，参考 GBaseDataReader 成员。

- 继承层次

```
System.Object
  |__ System.MarshalByRefObject
    |__ System.Data.Common.DbDataReader
      |__ GBase.Data.GBaseClient.GBaseDataReader
```

- 语法

```
[ Visual Basic ]
Public NotInheritable Class GBaseDataReader _
    Inherits DbDataReader _
    Implements IDataReader, IDisposable, IDataRecord
[ C# ]
public sealed class GBaseDataReader : DbDataReader,
    IDataReader, IDisposable, IDataRecord
```

- 必要条件

命名空间: GBase.Data.GBaseClient

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 注释

要创建一个 GBaseDataReader，用户必须调用 GBaseCommand 对象的 ExecuteReader 方法。

IsClosed 和 RecordsAffected 是唯一能够在 GBaseDataReader 关闭后调用的属性。尽管 RecordsAffected 属性可以在 GBaseDataReader 存在的任何时间访问，也总是在调用了 Close 之后才调用 RecordsAffected 来保证获得准确的返回值。

由于 GBaseDataReader 提供只进的只读数据访问方式，如果用户试图修改使用诸如 GetValue 返回的数值，就会出现警告。

- 示例

下面的例子演示创建了 GBaseDataReader 对象后，使用 GBaseDataReader 对象读取数据并将结果打印到控制台。

```
[Visual Basic]
Public Sub ReadGsData(gsConnString As String)
    Dim gsSelectQuery As String = "SELECT OrderID, CustomerID FROM"_
&"Orders"
    Dim gsConnection As New GBaseConnection(gsConnString)
    Dim gsCommand As New GBaseCommand(gsSelectQuery, gsConnection)
    gsConnection.Open()
    Dim gsReader As GBaseDataReader
    gsReader = gsCommand.ExecuteReader()
    ' Always call Read before accessing data.
    While gsReader.Read()
        Console.WriteLine((gsReader.GetInt32(0) & ", " &_
        gsReader.GetString(1)))
    End While
    ' always call Close when done reading.
    gsReader.Close()
    ' Close the connection when done with it.
    gsConnection.Close()
End Sub 'ReadGsData
```

```
[C#]
public void ReadGsData(string gsConnString)
{
    string gsSelectQuery = "SELECT OrderID, CustomerID FROM Orders";
    GBaseConnection gsConnection = new
GBaseConnection(gsConnString);
    GBaseCommand gsCommand = new
GBaseCommand(gsSelectQuery,gsConnection);
    gsConnection.Open();
    GBaseDataReader gsReader;
    gsReader = gsCommand.ExecuteReader();
    //Always call Read before accessing data.
    while (gsReader.Read())
    {
        Console.WriteLine(gsReader.GetInt32(0) + ", " +
gsReader.GetString(1));
    }
    // always call Close when done reading.
    gsReader.Close();
    // Close the connection when done with it.
    gsConnection.Close();
}
```

13.5.1 GBaseDataReader 成员

公共属性

属　性	描　述
YieldCount	获取当前行的列数。
HasRows	获取一个数值指明 GBaseDataReader 是否包含一行或多行。
IsClosed	获得一个值指明 GBaseDataReader 是否关闭。
Item	重载函数。根据给定的列索引，返回指

属　性	描　述
	定列的原始列值。
RecordsAffected	得到由 SQL 语句更改的，插入的，或者删除的行数。

公共方法

方　法	描　述
Close	关闭 GBaseDataReader 对象。
Equals　(继承于 Object)	判断指定的对象是否等于当前的对象。
GetBoolean	获取指定列的 Boolean 值。
GetByte	获取指定列的字节值。
GetBytes	获取指定列的字节数组值。
GetChar	获取指定列的单个字符值。
GetChars	获取指定列的字符数组值。
GetDataTypeName	获取指定列的数据类型的字符串。
GetDateTime	获取指定列的 DateTime 类型值。
GetDecimal	获取指定列的 Decimal 值。
GetDouble	获取指定列的双精度浮点数值。
GetFieldType	获取指定列的数据类型对象。
GetFloat	获取指定列的单精度浮点数值类型。
GetInt16	获取指定列的 16 位有符号整数值。
GetInt32	获取指定列的 32 位有符号整数值。
GetInt64	获取指定列的 64 位有符号整数值。
GetGBaseDateTime	获取指定列的 GBaseDateTime 值。
GetName	获得指定列的名字。
GetOrdinal	通过给定的列名，获得列序号。
GetSchemaTable	返回描述 GBaseDataReader 列元数据的 DataTable。
GetString	得到指定列的 String 值。
GetTimeSpan	得到指定列的 TimeSpan 值。
GetType　(继承于 Object)	获取当前实例的类型。

方　法	描　述
GetUInt16	获取指定列的 16 位无符号整数值。
GetUInt32	获取指定列的 32 位无符号整数值。
GetUInt64	获取指定列的 64 位无符号整数值。
GetValue	获取指定列的原类型数值，返回 Object。
GetValues	获得当前行中所有属性，返回属性的数量。
IsDBNull	获得一个值，判断指定列的值是否为空。
NextResult	当获取多条 SQL 语句对应的结果集时，此方法获取下一个结果集。
Read	使数据读取器读取下一条记录。
ToString （继承于 Object）	返回类的完全限定名。

13.5.2 GBaseDataReader 属性

13.5.2.1 FieldCount 属性

获取当前行中的列数。

- 语法

```
[ Visual Basic ]
Public Overrides ReadOnly Property FieldCount As Integer
    Get
[ C# ]
public override int FieldCount { get; }
```

- 实现

IDataRecord.FieldCount

13.5.2.2 HasRows 属性

获取一个数值指明 GBaseDataReader 是否包含一行或多行。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property HasRows As Boolean
    Get
```

[C#]

```
public override bool HasRows { get; }
```

13.5.2.3 IsClosed 属性

获得一个值指明 GBaseDataReader 是否关闭。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property IsClosed As Boolean
Get
```

[C#]

```
public override bool IsClosed { get; }
```

- 实现

IDataReader.IsClosed

13.5.2.4 Item 属性

重载属性。根据给定的列索引，返回指定列的原始列值。

- 重载列表

1) 根据给定的序号，获取指定列值。

Item(Int32)

2) 根据给定的字符串，获取指定列值。

Item(String)

13.5.2.4.1 Item 属性 (Int32)

根据给定的序号，获取指定列值。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Default Property Item ( _
    i As Integer _
) As Object
    Get
```

[C#]

```
public override Object this[    int i] { get; }
```

- 实现

IDataRecord.Item(Int32)

13.5.2.4.2 Item 属性 (String)

根据给定的字符串，获取指定列值。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Default Property Item ( _
    name As String _
) As Object
    Get
```

[C#]

```
public override Object this[    string name] { get; }
```

- 实现

IDataRecord.Item(String)

13.5.2.5 RecordsAffected 属性

得到 SQL 语句更改的，插入的，或者删除的行数。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property RecordsAffected As Integer
    Get
```

[C#]

```
public override int RecordsAffected { get; }
```

- 实现

IDataReader.RecordsAffected

13.5.3 GBaseDataReader 方法

13.5.3.1 Close 方法

关闭 GBaseDataReader 对象。

- 语法

[Visual Basic]

Public Overrides Sub Close

[C#]

public override void Close()

- 实现

IDataReader.Close()

13.5.3.2 GetBoolean 方法

获取指定列的 Boolean 值。

- 重载列表

1) 根据给定的序号，获取指定列的 Boolean 值。

GetBoolean(Int32)

2) 根据给定的列名，获取指定列的 Boolean 值。

GetBoolean(String)

13.5.3.2.1 GetBoolean 方法 (Int32)

根据给定的序号，获取指定列的 Boolean 值。

- 语法

[Visual Basic]

```
Public Overrides Function GetBoolean ( _
    i As Integer _
) As Boolean
```

[C#]

```
public override bool GetBoolean(int i)
```

- 参数

1) i : 从 0 开始的列序数

- 实现

IDataRecord. GetBoolean (Int32)

13.5.3.2.2 GetBoolean 方法 (String)

根据给定的列名，获取指定列的 Boolean 值。

- 语法

[Visual Basic]

```
Public Function GetBoolean ( _
    name As String _
) As Boolean
```

[C#]

```
public bool GetBoolean(
    string name
)
```

- 参数

1) name ：列名称

13.5.3.3 GetByte 方法

获取指定列的字节类型值。

- 重载列表

1) 根据给定的序号，获取指定列的字节类型值。

GetByte(Int32)

2) 根据给定的列名，获取指定列的字节类型值。

GetByte(String)

13.5.3.3.1 GetByte 方法 (Int32)

根据给定的序号，获取指定列的字节类型值。

- 语法

```
[Visual Basic]
Public Overrides Function GetByte ( _
    i As Integer _
) As Byte
[C#]
public override byte GetByte(
```

```
    int i
)
```

- 参数

1) i : 从 0 开始的列序数

- 实现

IDataRecord.GetByte(Int32)

13.5.3.3.2 GetByte 方法 (String)

根据给定的列名，获取指定列的字节类型值。

- 语法

```
[Visual Basic]
Public Function GetByte ( _
    name As String _
) As Byte
[C#]
public byte GetByte(
    string name
)
```

- 参数

1) name : 列名称

13.5.3.4 GetBytes 方法

获取指定列的字节数组值。

- 语法

```
[Visual Basic]
Public Overrides Function GetBytes ( _
    i As Integer, _
    fieldOffset As Long, _
    buffer As Byte(), _
    bufferoffset As Integer, _
    length As Integer _
) As Long
[C#]
public override long GetBytes(
    int i,
    long fieldOffset,
    byte[] buffer,
    int bufferoffset,
    int length
)
```

- 参数

1) i : 从 0 开始的列序数;

2) fieldOffset : 字段值索引, 从该处开始读操作;

3) buffer : 放置数据流的目标缓存区;

4) bufferoffset: 目标缓存区存放数据的开始索引;

5) length : 存放数据的最大长度。

- 返回值

读取字节的实际数量。

- 实现

IDataRecord.GetBytes(Int32, Int64, Byte[], Int32, Int32)

13.5.3.5 GetChar 方法

获取指定列的单个字符值。

- 重载列表

1) 根据给定的序号，获取指定列的单个字符值。

GetChar(Int32)

2) 根据给定的列名，获取指定列的单个字符值。

GetChar(String)

13.5.3.5.1 GetChar 方法 (Int32)

根据给定的序号，获取指定列的单个字符值。

- 语法

```
[Visual Basic]
Public Overrides Function GetChar ( _
    i As Integer _
) As Byte
[C#]
public override byte GetChar (
```

```
    int i
)
```

- 实现

IDataRecord.GetChar(Int32)

- 参数

1) i : 从 0 开始的列序数

13.5.3.5.2 GetChar 方法 (String)

根据给定的列名，获取指定列的单个字符值。

- 语法

```
[Visual Basic]
Public Function GetChar ( _
    name As String _
) As Byte
[C#]
public byte GetChar(
    string name
)
```

- 参数

1) name : 列名称

13.5.3.6 GetChars 方法

从指定列偏移开始，读取字符流到指定缓冲偏移开始的数组中。

- 语法

```
[Visual Basic]
Public Overrides Function GetChars ( _
    i As Integer, _
    fieldoffset As Long, _
    buffer As Char(), _
    bufferoffset As Integer, _
    length As Integer _
) As Long
[C#]
public override long GetChars(
    int i,
    long fieldoffset,
    char[] buffer,
    int bufferoffset,
    int length
)
```

- 参数

1) i : 从 0 开始的列序数;
2) fieldOffset : 字段值索引, 从该处开始读操作;
3) buffer : 放置数据流的目标缓存区;
4) bufferoffset: 目标缓存区存放数据的开始索引;
5) length : 存放数据的最大长度。

- 实现

IDataRecord.GetChars(Int32, Int64, Char[], Int32, Int32)

13.5.3.7 GetDataTypeName 方法

获取指定列的数据类型的字符串。

- 语法

```
[Visual Basic]
Public Overrides Function GetDataTypeName ( _
    i As Integer _
) As String
[C#]
public override string GetDataTypeName(
    int i
)
```

- 参数

1) i : 从 0 开始的列序数

- 返回值

返回数据类型的字符串。

- 实现

IDataRecord.GetDataTypeName(Int32)

13.5.3.8 GetDateTime 方法

获取指定列的 DateTime 类型值。

- 重载列表

1) 根据给定的序号，获取指定列的 DateTime 类型值。

GetDateTime(Int32)

2) 根据给定的列名，获取指定列的 DateTime 类型值。

GetDateTime(String)

- 注释

注意：GBase 数据库允许 Date 列包含值 0000-00-00 和 DateTime 列包含 0000-00-00 00:00:00。然而.NET DateTime 数据类型不能包含或者表现这些数值。要从可能包含 0 值的列中读取一个日期或日期时间值时，可以使用 GetGBaseDateTime 方法，并且在连接字符串中指定 Allow Zero Datetime 参数。要获取这个参数的更多信息，请参考 5.1.1 连接字符串章节中的连接参数表。

13.5.3.8.1 GetDateTime 方法 (Int32)

根据给定的序号，获取指定列的 DateTime 类型值。

- 语法

```
[Visual Basic]
Public Overrides Function GetDateTime ( _
    i As Integer _
) As DateTime
[C#]
public override DateTime GetDateTime(
    int i
)
```

- 参数

1） i ： 从 0 开始的列序数

13.5.3.8.2 GetDateTime 方法 (String)

根据给定的列名，获取指定列的 DateTime 类型值。

- 语法

[Visual Basic]

```
Public Overrides Function GetDateTime ( _
    i As Integer _
) As DateTime
```

[C#]

```
public override DateTime GetDateTime(
    int i
)
```

- 参数

1） name ： 列名称

13.5.3.9 GetDecimal 方法

获取指定列的 Decimal 类型值。

- 重载列表

1） 根据给定的序号，获取指定列的 Decimal 类型值。

GetDecimal(Int32)

2） 根据给定的列名称，获取指定列的 Decimal 类型值。

GetDecimal(String)

13.5.3.9.1 GetDecimal 方法 (Int32)

根据给定的序号，获取指定列的 Decimal 类型值。

- 语法

[Visual Basic]

```
Public Overrides Function GetDecimal ( _
    i As Integer _
) As Decimal
```

[C#]

```
public override decimal GetDecimal(
    int i
)
```

- 参数

1) i ：从 0 开始的列序数

- 实现

IDataRecord.GetDecimal(Int32)

13.5.3.9.2 GetDecimal 方法 (String)

根据给定的列名称，获取指定列的 Decimal 类型值。

- 语法

[Visual Basic]

```
Public Function GetDecimal ( _
    column As String _
```

```
) As Decimal
```

[C#]

```
public decimal GetDecimal(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.10 GetDouble 方法

获取指定列的双精度浮点数值。

- 重载列表

1) 根据给定的序号，获取指定列的双精度浮点数值。

GetDouble(Int32)

2) 根据给定的列名称，获取指定列的双精度浮点数值。

GetDouble(String)

13.5.3.10.1 GetDouble 方法 (Int32)

根据给定的序号，获取指定列的双精度浮点数值。

- 语法

[Visual Basic]

```
Public Overrides Function GetDouble ( _
    i As Integer _
) As Double
```

[C#]

```
public override double GetDouble(
    int i
)
```

- 参数

1) i : 从 0 开始的列序数

- 实现

IDataRecord.GetDouble(Int32)

13.5.3.10.2 GetDouble 方法 (String)

根据给定的列名称，获取指定列的双精度浮点数值。

- 语法

[Visual Basic]

```
Public Function GetDouble ( _
    column As String _
) As Double
```

[C#]

```
public double GetDouble(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.11 GetFieldType 方法

获取指定列的数据类型对象。

- 重载列表

1) 根据给定的序号，获取指定列的数据类型对象。

GetFieldType(Int32)

2) 根据给定的列名称，获取指定列的数据类型对象。

GetFieldType(String)

13.5.3.11.1 GetFieldType 方法 (Int32)

根据给定的序号，获取指定列的数据类型对象。

- 语法

```
[Visual Basic]
Public Overrides Function GetFieldType ( _
    i As Integer _
) As Type
[C#]
public override Type GetFieldType(
    int i
)
```

- 参数

1) i ：从 0 开始的列序数

- 实现

IDataRecord.GetFieldType(Int32)

13.5.3.11.2 GetFieldType 方法 (String)

根据给定的列名称，获取指定列的数据类型对象。

- 语法

[Visual Basic]

```
Public Function GetFieldType ( _
    column As String _
) As Type
```

[C#]

```
public Type GetFieldType(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.12 GetFloat 方法

获取指定列的单精度浮点数值类型。

- 重载列表

1) 根据给定的序号，获取指定列的单精度浮点数值类型。

GetFloat(Int32)

2) 根据给定的列名称，获取指定列的单精度浮点数值类型。

GetFloat(String)

13.5.3.12.1 GetFloat 方法 (Int32)

根据给定的序号，获取指定列的单精度浮点数值类型。

- 语法

[Visual Basic]

```
Public Overrides Function GetFloat ( _
    i As Integer _
) As Single
```

[C#]

```
public override float GetFloat(
    int i
)
```

- 参数

1) i ：从 0 开始的列序数

- 实现

IDataRecord.GetFloat(Int32)

13.5.3.12.2 GetFloat 方法 (String)

根据给定的列名称，获取指定列的单精度浮点数值类型。

- 语法

[Visual Basic]

```
Public Function GetFloat ( _
    column As String _
```

```
) As Single
```

[C#]

```
public float GetFloat(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.13 GetInt16 方法

获取指定列的 16 位有符号整数值。

- 重载列表

1) 根据给定的序号，获取指定列的 16 位有符号整数值。

GetInt16(Int32)

2) 根据给定的列名称，获取指定列的 16 位有符号整数值。

GetInt16(String)

13.5.3.13.1 GetInt16 方法 (Int32)

根据给定的序号，获取指定列的 16 位有符号整数值。

- 语法

[Visual Basic]

```
Public Overrides Function GetInt16 ( _
    i As Integer _
) As Short
```

[C#]

```
public override short GetInt16(
    int i
)
```

- 参数

1) i ：从 0 开始的列序数

- 实现

IDataRecord.GetInt16(Int32)

13.5.3.13.2 GetInt16 方法 (String)

根据给定的列名称，获取指定列的 16 位有符号整数值。

- 语法

[Visual Basic]

```
Public Function GetInt16 ( _
    column As String _
) As Short
```

[C#]

```
public short GetInt16(
    string column
)
```

- 参数

1) column ：列名称

13.5.3.14 GetInt32 方法

获取指定列的 32 位有符号整数值。

- 重载列表

1) 根据给定的序号，获取指定列的 32 位有符号整数值。

GetInt32(Int32)

2) 根据给定的列名称，获取指定列的 32 位有符号整数值。

GetInt32(String)

13.5.3.14.1 GetInt32 方法 (Int32)

根据给定的序号，获取指定列的 32 位有符号整数值。

- 语法

```
[Visual Basic]
Public Overrides Function GetInt32 ( _
    i As Integer _
) As Integer
[C#]
public override Interger GetInt16(
    int i
)
```

- 参数

1) i : 从 0 开始的列序数

- 实现

IDataRecord.GetInt32(Int32)

13.5.3.14.2 GetInt32 方法 (String)

根据给定的列名称，获取指定列的 32 位有符号整数值。

- 语法

[Visual Basic]

```
Public Function GetInt32 ( _
    column As String _
) As Integer
```

[C#]

```
public Integer GetInt32(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.15 GetInt64 方法

按照 64 位有符号整数获取指定列的值。

- 重载列表

1) 根据给定的序号，获取指定列的 64 位有符号整数值。

GetInt64(Int32)

2) 根据给定的列名称，获取指定列的 64 位有符号整数值。

GetInt64(String)

13.5.3.15.1 GetInt64 方法 (Int32)

根据给定的序号，获取指定列的 64 位有符号整数值。

- 语法

[Visual Basic]

```
Public Overrides Function GetInt64 ( _
    i As Integer _
) As Long
```

[C#]

```
public override long GetInt64(
    int i
)
```

- 参数

1) i ： 从 0 开始的列序数

- 实现

IDataRecord.GetInt64(Int32)

13.5.3.15.2 GetInt64 方法 (String)

根据给定的列名称，获取指定列的 64 位有符号整数值。

- 语法

[Visual Basic]

```
Public Function GetInt64 ( _
    column As String _
```

```
) As Long
[C#]
public long GetInt64(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.16 GetGBaseDateTime 方法

获取指定列的 GBaseDateTime 值。

- 重载列表

1) 根据给定的序号，获取指定列的 GBaseDateTime 值。

GetGBaseDateTime (Int32)

2) 根据给定的列名称，获取指定列的 GBaseDateTime 值。

GetGBaseDateTime(String)

13.5.3.16.1 GetGBaseDateTime 方法 (Int32)

根据给定的序号，获取指定列的 GBaseDateTime 值。

- 语法

```
[Visual Basic]
Public Function GetGBaseDateTime ( _
    column As Integer _
) As GBaseDateTime
```

```
[C#]
public override long GetGBaseDateTime(
    int column
)
```

- 参数

1) column : 从 0 开始的列序数

13.5.3.16.2 GetGBaseDateTime 方法 (String)

根据给定的列名称，获取指定列的 GBaseDateTime 值。

- 语法

```
[Visual Basic]
Public Function GetGBaseDateTime ( _
    column As String _
) As GBaseDateTime
[C#]
public GBaseDateTime GetGBaseDateTime(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.17 GetName 方法

获得指定列的名字。

- 语法

[Visual Basic]

```
Public Overrides Function GetName ( _
    i As Integer _
) As String
```

[C#]

```
public override string GetName(
    int i
)
```

- 参数

1) i ：从 0 开始的列序数

- 实现

IDataRecord.GetName(Int32)

13.5.3.18 GetOrdinal 方法

通过给定的列名，获得列序数。

- 语法

[Visual Basic]

```
Public Overrides Function GetOrdinal ( _
    name As String _
) As Integer
```

[C#]

```
public override int GetOrdinal(
    string name
)
```

- 参数

1) name ： 列名称

- 实现

IDataRecord.GetOrdinal(String)

13.5.3.19 GetSchemaTable 方法

返回描述 GBaseDataReader 列元数据的 DataTable。

- 语法

```
[Visual Basic]
Public Overrides Function GetSchemaTable As DataTable
[C#]
public override DataTable GetSchemaTable()
```

- 实现

IDataReader.GetSchemaTable()

13.5.3.20 GetString 方法

得到指定列的 String 值。

- 重载列表

1) 根据给定的序号，得到指定列的 String 值。

GetString(Int32)

2) 根据给定的列名称，得到指定列的 String 值。

GetString(String)

13.5.3.20.1 GetString 方法 (Int32)

根据给定的序号，得到指定列的 String 值。

- 语法

```
[Visual Basic]
Public Overrides Function GetString ( _
    i As Integer _
) As String
[C#]
public override String GetString(
    int i
)
```

- 参数

1) i ： 从 0 开始的列序数

- 实现

IDataRecord.GetString(Int32)

13.5.3.20.2 GetString 方法 (String)

根据给定的列名称，得到指定列的 String 值。

- 语法

```
[Visual Basic]
Public Function GetString ( _
    column As String _
) As Long
[C#]
public long GetString(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.21 GetTimeSpan 方法

得到指定列的 TimeSpan 值。

- 重载列表

1) 根据给定的序号，得到指定列的 TimeSpan 值。

GetTimeSpan(Int32)

2) 根据给定的列名称，得到指定列的 TimeSpan 值。

GetTimeSpan(String)

13.5.3.21.1 GetTimeSpan 方法 (Int32)

根据给定的序号，得到指定列的 TimeSpan 值。

- 语法

[Visual Basic]

```
Public Function GetTimeSpan ( _
    column As Integer _
) As TimeSpan
```

[C#]

```
public TimeSpan GetTimeSpan(
    int column
)
```

- 参数

1) column : 从 0 开始的列序数

13.5.3.21.2 GetTimeSpan 方法 (String)

根据给定的列名称，得到指定列的 TimeSpan 值。

- 语法

[Visual Basic]

```
Public Function GetTimeSpan ( _
    column As String _
) As TimeSpan
```

[C#]

```
public TimeSpan GetTimeSpan(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.22 GetUInt16 方法

获取指定列的 16 位无符号整数值。

- 重载列表

1) 根据给定的序号，获取指定列的 16 位无符号整数值。

GetUInt16(Int32)

2) 根据给定的列名称，获取指定列的 16 位无符号整数值。

GetUInt16(String)

13.5.3.22.1 GetUInt16 方法 (Int32)

根据给定的序号，获取指定列的 16 位无符号整数值。

- 语法

```
[Visual Basic]
Public Function GetUInt16 ( _
    column As Integer _
) As UShort
[C#]
public ushort GetUInt16(
    int column
)
```

- 参数

1) column : 从 0 开始的列序数

13.5.3.22.2 GetUInt16 方法 (String)

根据给定的列名称，获取指定列的 16 位无符号整数值。

- 语法

[Visual Basic]

```
Public Function GetUInt16 ( _
    column As String _
) As UShort
```

[C#]

```
public ushort GetUInt16(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.23 GetUInt32 方法

获取指定列的 32 位无符号整数值。

- 重载列表

1) 根据给定的序号，获取指定列的 32 位无符号整数值。

GetUInt32(Int32)

2) 根据给定的列名称，获取指定列的 32 位无符号整数值。

GetUInt32(String)

13.5.3.23.1 GetUInt32 方法 (Int32)

根据给定的序号，获取指定列的 32 位无符号整数值。

- 语法

[Visual Basic]

```
Public Function GetUInt32 ( _
    column As Integer _
) As UInteger
```

[C#]

```
public uint GetUInt32(
    int column
)
```

- 参数

1) column ： 从 0 开始的列序数

13.5.3.23.2 GetUInt32 方法 (String)

根据给定的列名称，获取指定列的 32 位无符号整数值。

- 语法

[Visual Basic]

```
Public Function GetUInt32 ( _
    column As String _
) As UInteger
```

[C#]

```
public uint GetUInt32(
    string column
)
```

- 参数

1) column ：列名称

13.5.3.24 GetUInt64 方法

获取指定列的 64 位无符号整数值。

- 重载列表

1) 根据给定的序号，获取指定列的 64 位无符号整数值。

GetUInt64(Int32)

2) 根据给定的列名称，获取指定列的 64 位无符号整数值。

GetUInt64(String)

13.5.3.24.1 GetUInt64 方法 (Int32)

根据给定的序号，获取指定列的 64 位无符号整数值。

- 语法

```
[Visual Basic]
Public Function GetUInt64 ( _
    column As Integer _
) As ULong
[C#]
public ulong GetUInt64(
```

```
    int column
)
```

- 参数

1) column : 从 0 开始的列序数

13.5.3.24.2 GetUInt64 方法 (String)

根据给定的列名称，获取指定列的 64 位无符号整数值。

- 语法

```
[Visual Basic]
Public Function GetUInt64 ( _
    column As String _
) As ULong
[C#]
public ulong GetUInt64(
    string column
)
```

- 参数

1) column : 列名称

13.5.3.25 GetValue 方法

获取指定列的原类型数值，返回 Object。

- 语法

```
[Visual Basic]
```

```
Public Overrides Function GetValue ( _
    i As Integer _
) As Object
```

[C#]

```
public override Object GetValue(
    int i
)
```

- 参数

1) i ： 从 0 开始的列序数

- 实现

IDataRecord.GetValue(Int32)

13.5.3.26 GetValues 方法

获得当前行中所有属性，返回属性的数量。

- 语法

[Visual Basic]

```
Public Overrides Function GetValues ( _
    values As Object() _
) As Integer
```

[C#]

```
public override int GetValues(
    Object[] values
)
```

- 参数

1) values ：获取的所有属性

- 实现

IDataRecord.GetValues(Object[])

13.5.3.27 IsDBNull 方法

获得一个值，判断指定列的值是否为空。

- 语法

```
[Visual Basic]
Public Overrides Function IsDBNull ( _
    i As Integer _
) As Boolean
[C#]
public override bool IsDBNull(
    int i
)
```

- 参数

1) i ：从 0 开始的列索引

- 实现

IDataRecord.IsDBNull(Int32)

13.5.3.28 NextResult 方法

当获取多条 SQL 语句对应的结果集时，此方法获取下一个结果集。

- 语法

[Visual Basic]

```
Public Overrides Function NextResult As Boolean
```

[C#]

```
public override bool NextResult()
```

- 实现

```
IDataReader.NextResult()
```

13.5.3.29 Read 方法

使数据读取器读取下一条记录。

- 语法

[Visual Basic]

```
Public Overrides Function Read As Boolean
```

[C#]

```
public override bool Read()
```

- 实现

```
IDataReader.Read()
```

13.6 GBaseError 类

提供由服务器返回的错误代码参考。对于该类所有成员的列表，参考 GBaseError 成员。

- 继承层次

System.Object

|__ GBase.Data.GBaseClient.GBaseError

- 语法

[Visual Basic]

Public Class GBaseError

[C#]

public class GBaseError

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 必要条件

命名空间: GBase.Data.GBaseClient

13.6.1 GBaseError 成员

公共构造函数

构造函数	描　述
GBaseError	初始化一个新的 GBaseError 类实例。

公共属性

属　性	描　述
Code	错误代码
Level	错误等级

属　性	描　述
Message	错误信息

公共方法

方　法	描　述
Equals　(继承于 Object)	判断指定的对象是否等于当前的对象。
GetType　(继承于 Object)	获取当前实例的类型。
ToString　(继承于 Object)	返回类的完全限定名。

13.6.2　GBaseError 构造函数

- 语法

[Visual Basic]

```
Public Sub New ( _
    level As String, _
    code As Integer, _
    message As String _
)
```

[C#]

```
public GBaseError(string level,int code,string message)
```

- 参数

1) level : 错误等级

2) code : 错误代码

3) message : 错误消息

13.6.3 GBaseError 属性

13.6.3.1 Level 属性

- 语法

[Visual Basic]

```
Public ReadOnly Property Level As String
    Get
```

[C#]

```
public string Level { get; }
```

13.6.3.2 Code 属性

- 语法

[Visual Basic]

```
Public ReadOnly Property Code As String
    Get
```

[C#]

```
public string Code { get; }
```

13.6.3.3 Message 属性

- 语法

[Visual Basic]

```
Public ReadOnly Property Message As String
```

Get

[C#]

public string Message { get; }

13.7 GBaseException 类

返回错误时抛出的异常，这个类不能被继承。对于该类所有成员的列表，参考 GBaseException 成员。

- 继承层次

System.Object

|__ System.Exception

|__ System.SystemException

|__ System.Runtime.InteropServices.ExternalException

|__ System.Data.Common.DbException

|__GBase.Data.GBaseClient.GBaseException

- 语法

[Visual Basic]

<SerializableAttribute> _

Public NotInheritable Class GBaseException _

Inherits DbException

[C#]

[SerializableAttribute]

public sealed class GBaseException : DbException

- 必要条件

命名空间: GBase.Data.GBaseClient

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 注释

每当接口程序遇到服务器生成的错误时，都将创建此类。(客户端错误作为标准的.NET Framework 异常引发)。

任何打开的连接在遇到异常的时候都不会自动关闭。如果客户端应用程序认为异常是致命的，应该在捕获到异常信息后关闭任何 GBaseDataReader 对象或者 GBaseConnection 对象。

- 示例

下面的例子演示当连接 GBase 数据库时因为缺少用户名和密码而生成一个 GBaseException，然后显示异常信息。

```
[Visual Basic]
Public Sub ShowException()
    Dim gsSelectQuery As String = "SELECT column1 FROM table1"
    Dim gsConnection As New GBaseConnection ("Data"_
&" Source=localhost;Database=Sample;pooling=false")
    Dim gsCommand As New GBaseCommand(gsSelectQuery,_ gsConnection)
    Try
        gsCommand.Connection.Open()
    Catch e As GBaseException
        MessageBox.Show( e.Message )
    End Try
End Sub

[C#]
public void ShowException()
{
    string gsSelectQuery = "SELECT column1 FROM table1";
```

```
        GBaseConnection gsConnection =new
        GBaseConnection("DataSource=localhost;Database=Sample;poolin
g=false");
        GBaseCommand gsCommand = new
GBaseCommand(gsSelectQuery,gsConnection);
        try
        {
            gsCommand.Connection.Open();
        }
        catch (GBaseException e)
        {
            MessageBox.Show( e.Message );
        }
    }
```

13.7.1 GBaseException 成员

公共属性

属　性	描　述
HelpLink （继承于 Exception）	获取或设置一个关于这个异常的帮助文件的链接。
InnerException （继承于 Exception）	获得引起当前异常的异常实例。
Message （继承于 Exception）	获得描述当前异常的信息。
Number	获得一个描述当前错误的号码。
Source （继承于 Exception）	获取或设置应用程序的名称或者引起错误的对象名称。
StackTrace （继承于 Exception）	在异常抛出时，获得调用堆栈上的字符串表示形式。
TargetSite （继承于 Exception）	获得抛出异常的方法。

公共方法

方　法	描　述
Equals （继承于 Object）	判断指定的对象是否等于当前的对象。
GetBaseException （继承于 Exception）	当在子类中覆盖的时候，返回根异常。
GetType （继承于 Object）	获取当前实例的类型。
ToString （继承于 Exception）	创建并返回一个代表当前异常的字符串。

13.7.2 GBaseException 属性

13.7.2.1 Number 属性

获得一个描述当前错误的号码。

- 语法

```
[Visual Basic]
Public ReadOnly Property Number As Integer
    Get
[C#]
public int Number { get; }
```

13.8 GBaseHelper 类

帮助类，使用 GBase ADO.NET 对象封装，使得操作 GBase 数据源更加方便。对于该类所有成员的列表，参考 GBaseHelper 成员。

- 继承层次

```
System.Object
|__ GBase.Data.GBaseClient.GBaseHelper
```

- 语法

[Visual Basic]

```
Public NotInheritable Class GBaseHelper
```

[C#]

```
public sealed class GBaseHelper
```

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 必要条件

命名空间： GBase.Data.GBaseClient

13.8.1 GBaseHelper 成员

公共静态成员（Visual Basic 中是 Shared）方法

静态成员方法	描 述
ExecuteDataRow	执行一句 SQL 命令并返回结果中的首行。方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。返回 DataRow 对象。
ExecuteDataset	重载函数。执行一句 SQL 命令并返回带有数据的 DataSet。
ExecuteNonQuery	重载函数。对 GBase 数据库执行一句 SQL 命令，返回受影响的行数。
ExecuteReader	重载函数。对 GBase 数据库执行一句 SQL 命令，返回 GBaseReader 对象。
ExecuteScalar	重载函数。对 GBase 数据库执行一句命

静态成员方法	描　述
	令,返回结果集中第一行第一列object值对象。
UpdateDataSet	使用 DataSet 中的数据更新指定的表。

公共方法

方　法	描　述
Equals　(继承于 Object)	判断指定的对象是否等于当前的对象。
GetType　(继承于 Object)	获取当前实例的类型。
ToString　(继承于 Object)	返回类的完全限定名。

13.8.2　GBaseHelper 方法

13.8.2.1 ExecuteDataRow 方法

执行一句 SQL 命令并返回结果中的首行。方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。返回 DataRow。

- 语法

```
[Visual Basic]
Public Shared Function ExecuteDataRow ( _
    connectionString As String, _
    commandText As String, _
    ParamArray parms As GBaseParameter() _
) As DataRow
[C#]
public static DataRow ExecuteDataRow(
```

```
    string connectionString,
    string commandText,
    params GBaseParameter[] parms
)
```

- 参数

1) connectionString ：用于连接的设置；

2) commandText ：要执行的命令；

3) parms ：命令的参数。

- 返回值

包含结果集首行的 DataRow。

13.8.2.2 ExecuteDataset 方法

执行一句 SQL 命令并返回带有数据的 DataSet。

- 重载列表

1) 执行一句 SQL 命令并返回带有数据的 DataSet。使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。

ExecuteDataset(GBaseConnection,String)

2) 执行一句 SQL 命令并返回带有数据的 DataSet。使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。通过 GBaseParameter[]参数传递命令对象使用的参数。

ExecuteDataset(GBaseConnection,String,GBaseParameter[])

3) 执行一句 SQL 命令并返回带有数据的 DataSet。方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。

ExecuteDataset(String,String)

4) 执行一句 SQL 命令并返回带有数据的 DataSet。方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。通过 GBaseParameter[]参数传递命令对象使用的参数。

ExecuteDataset(String,String,GBaseParameter[])

13.8.2.2.1 ExecuteDataset 方法 (GBaseConnection, String)

执行一句 SQL 命令并返回带有数据的 DataSet。使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。

- 语法

```
[Visual Basic]
Public Shared Function ExecuteDataset ( _
    connection As GBaseConnection, _
    commandText As String _
) As DataSet
[C#]
public static DataSet ExecuteDataset(
    GBaseConnection connection,
    string commandText
)
```

- 参数

1) connection : GBaseConnection 连接对象；

2) commandText : 要执行的 SQL 语句。

- 返回值

包含结果集的 DataSet。

13.8.2.2.2 ExecuteDataset 方法 (GBaseConnection, String, GBaseParameter[])

执行一句 SQL 命令并返回带有数据的 DataSet。使用给定的 GBaseConnection 对象创建,GBaseConnection 对象的状态在执行之后保持不变。通过 GBaseParameter[]参数传递命令对象使用的参数。

- 语法

```
[Visual Basic]
Public Shared Function ExecuteDataset ( _
    connectionString As String, _
    commandText As String, _
    ParamArray commandParameters As GBaseParameter() _
) As DataSet
[C#]
public static DataSet ExecuteDataset(
    string connectionString,
    string commandText,
    params GBaseParameter[] commandParameters
)
```

- 参数

1) connection : GBaseConnection 连接对象;

2) commandText : 要执行的 SQL 语句;

3) commandParameters : 用于 GBaseCommand 对象的参数数组。

- 返回值

包含结果集的 DataSet。

13.8.2.2.3 ExecuteDataset 方法 (String, String)

执行一句 SQL 命令并返回带有数据的 DataSet。方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。

- 语法

[Visual Basic]

```
Public Shared Function ExecuteDataset ( _
    connectionString As String, _
    commandText As String _
) As DataSet
```

[C#]

```
public static DataSet ExecuteDataset(
    string connectionString,
    string commandText
)
```

- 参数

1) connectionString : 用于创建 GBaseConnection 对象的连接字符串;

2) commandText : 要执行的 SQL 语句。

- 返回值

包含结果集的 DataSet。

13.8.2.2.4 ExecuteDataset 方法 (String, String, GBaseParameter[])

执行一句 SQL 命令并返回带有数据的 DataSet。方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。通过 GBaseParameter[]参数传递命令对象使用的参数。

- 语法

[Visual Basic]

```
Public Shared Function ExecuteDataset ( _
    connection As GBaseConnection, _
    commandText As String, _
    ParamArray commandParameters As GBaseParameter() _
) As DataSet
```

[C#]

```
public static DataSet ExecuteDataset(
    GBaseConnection connection,
    string commandText,
    params GBaseParameter[] commandParameters
)
```

- 参数

1) connectionString ：用于创建 GBaseConnection 对象的连接字符串；

2) commandText ：要执行的 SQL 语句；

3) commandParameters ：用于 GBaseCommand 对象的参数数组。

- 返回值

包含结果集的 DataSet。

13.8.2.3 ExecuteNonQuery 方法

对 GBase 数据库执行一句 SQL 命令，返回受影响的行数。

- 重载列表

1） 对 GBase 数据库执行一句 SQL 命令。使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。

ExecuteNonQuery(GBaseConnection,String,GBaseParameter[])

2） 对 GBase 数据库执行一个 SQL 命令。使用给定的 ConnectionString 创建一个 GBaseConnection。方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。通过 GBaseParameter[]参数传递命令对象使用的参数。

ExecuteNonQuery(String,String,GBaseParameter[])

13.8.2.3.1 ExecuteNonQuery 方法 (GBaseConnection, String, GBaseParameter[])

对 GBase 数据库执行一句 SQL 命令。使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。

- 语法

```
[Visual Basic]
Public Shared Function ExecuteNonQuery ( _
    connection As GBaseConnection, _
    commandText As String, _
    ParamArray commandParameters As GBaseParameter() _
```

```
) As Integer
```

[C#]

```
public static int ExecuteNonQuery(
    GBaseConnection connection,
    string commandText,
    params GBaseParameter[] commandParameters
)
```

- 参数

1) connection ：GBaseConnection 连接对象；

2) commandText ：要执行的 SQL 语句；

3) commandParameters ：用于 GBaseCommand 对象的参数数组。

- 返回值

受影响的行数

13.8.2.3.2 ExecuteNonQuery 方法 (String, String, GBaseParameter[])

对 GBase 数据库执行一个 SQL 命令。方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。通过 GBaseParameter[]参数传递命令对象使用的参数。

- 语法

[Visual Basic]

```
Public Shared Function ExecuteNonQuery ( _
    connectionString As String, _
```

```
        commandText As String, _
        ParamArray parms As GBaseParameter() _
    ) As Integer
    [C#]
    public static int ExecuteNonQuery(
        string connectionString,
        string commandText,
        params GBaseParameter[] parms
    )
```

- 参数

1) connectionString ：用于创建 GBaseConnection 对象的连接字符串；

2) commandText ：要执行的 SQL 命令；

3) parms ：用于 GBaseCommand 对象的参数数组。

- 返回值

受影响的行数

13.8.2.4 ExecuteReader 方法

对 GBase 数据库执行一句 SQL 命令，返回 GBaseReader 对象。

- 重载列表

1) 对 GBase 数据库执行一句 SQL 命令，方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。

ExecuteReader(String,String)

2） 对 GBase 数据库执行一句 SQL 命令，方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。通过 GBaseParameter[]参数传递命令对象使用的参数。

ExecuteReader(String,String,GBaseParameter[])

3） 对 GBase 数据库执行一句 SQL 命令，使用指定的 GBaseConnection 对象创建，GBaseConnection 在使用完成后状态保持不变。

ExcuteReader(GBaseConnection,String)

4） 对 GBase 数据库执行一句 SQL 命令，使用指定的 GBaseConnection 对象创建，GBaseConnection 在使用完成后状态保持不变。通过 GBaseParameter[]参数传递命令对象使用的参数。

ExcuteReader(GBaseConnection,String,GBaseParameter[])

13.8.2.4.1 ExecuteReader 方法 (String, String)

对 GBase 数据库执行一句 SQL 命令，方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。

- 语法

[Visual Basic]

```
Public Shared Function ExecuteReader ( _
    connectionString As String, _
    commandText As String _
) As GBaseDataReader
```

[C#]

```
public static GBaseDataReader ExecuteReader(
    string connectionString,
```

```
    string commandText
)
```

- 参数

1) connectionString ：用于创建 GBaseConnection 对象的连接字符串

2) commandText ：要执行的 SQL 语句

- 返回值

准备读取命令结果的 GBaseDataReader 对象。

13.8.2.4.2 ExecuteReader 方法 (String, String, GBaseParameter[])

对 GBase 数据库执行一句 SQL 命令，方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。通过 GBaseParameter[]参数传递命令对象使用的参数。

- 语法

```
[Visual Basic]
Public Shared Function ExecuteReader ( _
    connectionString As String, _
    commandText As String, _
    ParamArray commandParameters As GBaseParameter() _
) As GBaseDataReader
[C#]
public static GBaseDataReader ExecuteReader(
    string connectionString,
    string commandText,
```

```
    params GBaseParameter[] commandParameters
)
```

- 参数

1) connectionString ：用于创建 GBaseConnection 对象的连接字符串；

2) commandText ：要执行的 SQL 语句；

3) commandParameters ：用于 GBaseCommand 对象的参数数组。

- 返回值

准备读取命令结果的 GBaseDataReader 对象。

13.8.2.4.3 ExecuteReader 方法（GBaseConnection, String）

对 GBase 数据库执行一句 SQL 命令，使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。

- 语法

```
[Visual Basic]
Public Shared Function ExecuteReader ( _
    connection As GBaseConnection, _
    commandText As String _
) As GBaseDataReader
[C#]
public static GBaseDataReader ExecuteReader(
    GBaseConnection connection,
    string commandText
```

)

- 参数

1) connection : GBaseConnection 连接对象;

2) commandText : 要执行的 SQL 语句。

- 返回值

准备读取命令结果的 GBaseDataReader 对象。

13.8.2.4.4 ExecuteReader 方法 (GBaseConnection,String,GBaseParameter[])

对 GBase 数据库执行一句 SQL 命令，使用给定的 GBaseConnection 对象创建,GBaseConnection 对象的状态在执行之后保持不变。通过 GBaseParameter[] 参数传递命令对象使用的参数。

- 语法

```
[Visual Basic]
Public Shared Function ExecuteReader ( _
    connection As GBaseConnection, _
    commandText As String, _
    ParamArray commandParameters As GBaseParameter() _
) As GBaseDataReader
[C#]
public static GBaseDataReader ExecuteReader(
    GBaseConnection connection,
    string commandText,
```

params GBaseParameter[] commandParameters

)

- 参数

1) connection : GBaseConnection 连接对象;

2) commandText : 要执行的 SQL 语句;

3) commandParameters : 用于 GBaseCommand 对象的参数数组。

- 返回值

准备读取命令结果的 GBaseDataReader 对象。

13.8.2.5 GBaseHelper.ExecuteScalar 方法

对 GBase 数据库执行一句 SQL 命令。

- 重载列表

1) 对 GBase 数据库执行一句 SQL 命令，使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。

ExecuteScalar(GBaseConnection,String)

2) 对 GBase 数据库执行一句 SQL 命令，使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。通过 GBaseParameter[]参数传递命令对象使用的参数。

ExecuteScalar(GBaseConnection,String,GBaseParameter[])

3) 对 GBase 数据库执行一句 SQL 命令，方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。

ExecuteScalar(String,String)

4) 对 GBase 数据库执行一句 SQL 命令，方法执行期间，一个新的

GBaseConnection 对象会被创建、打开和关闭。通过 GBaseParameter[]参数传递命令对象使用的参数。

ExecuteScalar(String,String,GBaseParameter[])

13.8.2.5.1 ExecuteScalar 方法 (GBaseConnection, String)

对 GBase 数据库执行一句 SQL 命令，使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。

- 语法

[Visual Basic]

```
Public Shared Function ExecuteScalar ( _
    connection As GBaseConnection, _
    commandText As String _
) As Object
```

[C#]

```
public static Object ExecuteScalar(
    GBaseConnection connection,
    string commandText
)
```

- 参数

1) connection : GBaseConnection 连接对象

2) commandText : 要执行的 SQL 语句

- 返回值

返回结果集中的第一行的第一列的值，如果值为空则返回 null。

13.8.2.5.2 ExecuteScalar 方法 (GBaseConnection, String, GBaseParameter[])

对 GBase 数据库执行一句 SQL 命令，使用给定的 GBaseConnection 对象创建，GBaseConnection 对象的状态在执行之后保持不变。通过 GBaseParameter[] 参数传递命令对象使用的参数。

- 语法

[Visual Basic]

```
Public Shared Function ExecuteScalar ( _
    connection As GBaseConnection, _
    commandText As String, _
    ParamArray commandParameters As GBaseParameter() _
) As Object
```

[C#]

```
public static Object ExecuteScalar(
    GBaseConnection connection,
    string commandText,
    params GBaseParameter[] commandParameters
)
```

- 参数

1) connection : GBaseConnection 连接对象；

2) commandText : 要执行的 SQL 语句；

3) commandParameters : 用于 GBaseCommand 对象的参数数组。

- 返回值

结果集中的第一行的第一列的值，或者值为空则返回 null。

13.8.2.5.3 ExecuteScalar 方法 (String, String)

对 GBase 数据库执行一句 SQL 命令，方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。

- 语法

[Visual Basic]

```
Public Shared Function ExecuteScalar ( _
    connectionString As String, _
    commandText As String _
) As Object
```

[C#]

```
public static Object ExecuteScalar(
    string connectionString,
    string commandText
)
```

- 参数

1) connectionString ：用于创建 GBaseConnection 对象的连接字符串；

2) commandText ：要执行的 SQL 语句。

- 返回值

结果集中的第一行的第一列的值，如果值为空则返回 null。

13.8.2.5.4 ExecuteScalar 方法 (String, String, GBaseParameter[])

对 GBase 数据库执行一句 SQL 命令，方法执行期间，一个新的 GBaseConnection 对象会被创建、打开和关闭。通过 GBaseParameter[]参数传递命令对象使用的参数。

- 语法

[Visual Basic]

```
Public Shared Function ExecuteScalar ( _
    connectionString As String, _
    commandText As String, _
    ParamArray commandParameters As GBaseParameter() _
) As Object
```

[C#]

```
public static Object ExecuteScalar(
    string connectionString,
    string commandText,
    params GBaseParameter[] commandParameters
)
```

- 参数

1) connectionString ：用于创建 GBaseConnection 对象的连接字符串；

2) commandText ：要执行的 SQL 语句；

3) commandParameters ：用于 GBaseCommand 对象的参数数组。

- 返回值

结果集中第一行的第一列，如果结果集为空则返回 null 引用。

13.8.2.6 GBaseHelper.UpdateDataSet 方法

使用 DataSet 中的数据更新指定的表。

- 语法

```
[Visual Basic]
Public Shared Sub UpdateDataSet ( _
    connectionString As String, _
    commandText As String, _
    ds As DataSet, _
    tablename As String _
)
[C#]
public static void UpdateDataSet(
    string connectionString,
    string commandText,
    DataSet ds,
    string tablename
)
```

- 参数

1) connectionString ：用于创建 GBaseConnection 对象的连接字符串；

2) commandText ：要执行的 SQL 语句；

3) ds ：用于更新的包含新数据的 DataSet；

4) tablename ：数据库中要更新的表名。

13.9 GBaseInfoMessageEventArgs 类

提供用于 InfoMessage 事件的数据，InfoMessage 事件是当 GBase 返回一个警告或信息性消息时发生，对于该类所有成员的列表，参考 GBaseInfoMessageEventArgs 成员。

- 继承层次

System.Object

|__ System.EventArgs

|__ GBase.Data.GBaseClient.GBaseInfoMessageEventArgs

- 语法

[Visual Basic]

```
Public Class GBaseInfoMessageEventArgs _
    Inherits EventArgs
```

[C#]

```
public class GBaseInfoMessageEventArgs : EventArgs
```

- 线程安全性

这个类型的公共静态成员(在 Visual Basic 中为 Shared)对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 必要条件

命名空间：GBase.Data.GBaseClient

13.9.1 GBaseInfoMessageEventArgs 成员

公共构造函数

构造函数	描 述
GBaseInfoMessageEventArgs	初始化一个新的 GBaseInfoMessageEventArgs 类实例。

公共字段

字 段	描 述
errors	获取从服务器发送来的警告的集合。

公共方法

方 法	描 述
Equals (继承于 Object)	判断指定的对象是否等于当前的对象。
GetType (继承于 Object)	获取当前实例的类型。
ToString (继承于 Object)	返回类的完全限定名。

13.9.2 GBaseInfoMessageEventArgs 构造函数

初始化一个新的 GBaseInfoMessageEventArgs 类实例。

- 语法

[Visual Basic]

```
Public Sub New
```

[C#]

```
public GBaseInfoMessageEventArgs()
```

13.9.3 GBaseInfoMessageEventArgs 字段

13.9.3.1 GBaseInfoMessageEventArgs.errors 字段

获取从服务器发送来的警告的集合。

- 语法

[Visual Basic]

```
Public errors As GBaseError()
```

[C#]

```
public GBaseError[] errors
```

13.10 GBaseParameter 类

代表一个传给 GBaseCommand 的参数。这个类不能被继承。对于该类所有成员的列表，参考 GBaseParameter 成员。

- 继承层次

```
System.Object
  |__ System.MarshalByRefObject
    |__ System.Data.Common.DbParameter
      |__ GBase.Data.GBaseClient.GBaseParameter
```

- 语法

[Visual Basic]

```
Public NotInheritable Class GBaseParameter _
    Inherits DbParameter _
```

Implements IDbDataParameter, IDataParameter, ICloneable

[C#]

public sealed class GBaseParameter : DbParameter,

IDbDataParameter, IDataParameter, ICloneable

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 必要条件

命名空间：GBase.Data.GBaseClient

13.10.1 GBaseParameter 成员

公共构造函数

构造函数	描　述
GBaseParameter	重载函数。初始化 GBaseParameter 类的一个新实例。

公共属性

属　性	描　述
DbType	获取或设置参数的 DbType 类型。
Direction	获取或设置一个指明参数是只能输入、只能输出、双向，还是一个存储过程的返回值。
IsNullable	获取或设置一个值指明参数是否可以是 null 值。
GBaseDbType	获取或设置参数的 GBaseDbType 类型。
ParameterName	获取或设置 GBaseParameter 的名字。

属　性	描　述
Precision	获取或设置用于显示 Value 属性的最大位数。
Scale	获取或设置获得的 Value 的小数位位数。
Size	获取或设置列中数据按字节计算的最大大小。
SourceColumn	获取或设置源列的名称，该源列映射到 DataSet 并用于加载或返回 Value 时使用。
SourceVersion	获取或设置用于装载 Value 时的 DataRowVersion。
Value	获取或设置参数的值。

公共方法

方　法	描　述
Equals　(继承于 Object)	判断指定的对象是否等于当前的对象。
ResetDbType	重置与此 GBaseParameter 关联的类型。
GetType　(继承于 Object)	获取当前实例的类型。
ToString	获得一个包含 ParameterName 的字符串。

13.10.2　GBaseParameter 构造函数

初始化 GBaseParameter 类的一个新实例。

- 重载列表

1)　初始化 GBaseParameter 类的一个新实例。

GBaseParameter()

2） 使用参数名和新的 GBaseParameter 的值初始化 GBaseParameter 类的一个新实例。

GBaseParameter(String,Object)

3） 使用参数名和参数 GBaseDbType 类型初始化 GBaseParameter 类的一个新实例。

GBaseParameter(String,GBaseDbType)

4） 使用参数名、参数 GBaseDbType 类型、和参数长度初始化 GBaseParameter 类的一个新实例。

GBaseParameter(String,GBaseDbType,Int32)

5） 使用参数名、参数的 GBaseDbType 类型、参数长度、源列名，初始化 GBaseParameter 类的一个新实例。

GBaseParameter(String,GBaseDbType,Int32,String)

6） 使用参数名、参数 GBaseDbType 类型、参数长度、ParameterDirection、是否可为 null、参数精度、源列名、一个可用的 DataRowVersion、参数值，初始化 GBaseParameter 类的一个新实例。

GBaseParameter(String,GBaseDbType,Int32,ParameterDirection,Boolean,Byte,Byte,String,DataRowVersion,Object)

13.10.2.1 GBaseParameter 构造函数 ()

初始化 GBaseParameter 类的一个新实例。

- 语法

[Visual Basic]

Public Sub New

[C#]

public GBaseParameter()

13.10.2.2 GBaseParameter 构造函数 (String, Object)

使用参数名和新的 GBaseParameter 的值初始化 GBaseParameter 类的一个新实例。

- 语法

[Visual Basic]

```
Public Sub New ( _
    parameterName As String, _
    value As Object _
)
```

[C#]

```
public GBaseParameter(
    string parameterName,
    Object value
)
```

- 参数

1) parameterName: 参数名;

2) value: 参数值。

13.10.2.3 GBaseParameter 构造函数 (String, GBaseDbType)

使用参数名和参数的 GBaseDbType 类型初始化 GBaseParameter 类的一个新实例。

- 语法

```
[Visual Basic]
Public Sub New ( _
    parameterName As String, _
    dbType As GBaseDbType _
)
[C#]
public GBaseParameter(
    string parameterName,
    GBaseDbType dbType
)
```

- 参数

1) parameterName: 参数名;

2) dbType : 参数的 GBaseDbType 类型。

13.10.2.4 GBaseParameter 构造函数 (String, GBaseDbType, Int32)

使用参数名、参数 GBaseDbType 类型、和参数的长度初始化 GBaseParameter

类的一个新实例。

- 语法

```
[Visual Basic]
Public Sub New ( _
    parameterName As String, _
    dbType As GBaseDbType, _
    size As Integer _
)
[C#]
public GBaseParameter(
    string parameterName,
    GBaseDbType dbType,
    int size
)
```

- 参数

1) parameterName : 参数名;

2) dbType : 参数数据类型;

3) size : 参数的长度。

13.10.2.5 GBaseParameter 构造函数 (String, GBaseDbType, Int32, String)

使用参数名、参数的 GBaseDbType 类型、参数的长度、源列名，初始化 GBaseParameter 类的一个新实例。

- 语法

[Visual Basic]

```
Public Sub New ( _
    parameterName As String, _
    dbType As GBaseDbType, _
    size As Integer, _
    sourceColumn As String _
)
```

[C#]

```
public GBaseParameter(
    string parameterName,
    GBaseDbType dbType,
    int size,
    string sourceColumn
)
```

- 参数

1) parameterName : 参数名;

2) dbType : 参数值;

3) size : 参数的长度;

4) sourceColumn : 源列名。

13.10.2.6 GBaseParameter 构造函数 (String, GBaseDbType, Int32, ParameterDirection, Boolean, Byte, Byte, String, DataRowVersion, Object)

使用参数名、参数类型、参数大小、ParameterDirection、参数精度、参数精度、源列、一个可用的 DataRowVersion、参数值来初始化 GBaseParameter 类的一个新实例 。

- 语法

[Visual Basic]

```
Public Sub New ( _
    parameterName As String, _
    dbType As GBaseDbType, _
    size As Integer, _
    direction As ParameterDirection, _
    isNullable As Boolean, _
    precision As Byte, _
    scale As Byte, _
    sourceColumn As String, _
    sourceVersion As DataRowVersion, _
    value As Object _
)
```

[C#]

```
public GBaseParameter(
    string parameterName,
    GBaseDbType dbType,
    int size,
    ParameterDirection direction,
    bool isNullable,
    byte precision,
    byte scale,
    string sourceColumn,
    DataRowVersion sourceVersion,
    Object value
)
```

- 参数

1) parameterName : 参数名;
2) dbType : 参数的 GBaseDbType 类型;
3) size : 参数的长度;
4) direction : ParameterDirection 值之一;
5) isNullable : 值是否可以为 null;
6) precision : 小数点左右两侧的总位数;
7) scale: 小数位总个数;
8) sourceColumn : 源列的名称;
9) sourceVersion : DataRowVersion 值之一;
10) value : 参数值。

- 异常

ArgumentException 异常

异常类型	条　　件
ArgumentException	dbType 参数中提供的值为无效数据类型时引发。

13.10.3 GBaseParameter 属性

13.10.3.1 DbType 属性

获取或设置参数的 DbType 类型。

- 语法

```
[Visual Basic]
Public Overrides Property DbType As DbType
    Get
    Set
[C#]
public override DbType DbType { get; set; }
```

- 实现

IDataParameter.DbType

13.10.3.2 Direction 属性

获取或设置一个指明参数是只能输入、只能输出、双向，还是一个存储过程的返回值。

- 语法

[Visual Basic]

```
Public Overrides Property Direction As ParameterDirection
    Get
    Set
```

[C#]

```
public override ParameterDirection Direction { get; set; }
```

- 实现

IDataParameter.Direction

13.10.3.3 IsNullable 属性

获取或设置一个值指明参数是否可以是 null 值。

- 语法

[Visual Basic]

```
Public Overrides Property IsNullable As Boolean
    Get
    Set
```

[C#]

```
public override bool IsNullable { get; set; }
```

- 实现

IDataParameter.IsNullable

13.10.3.4 GBaseDbType 属性

获取或设置参数的 GBaseDbType 类型。

- 语法

[Visual Basic]

```
Public Property GBaseDbType As GBaseDbType
    Get
    Set
```

[C#]

```
public GBaseDbType GBaseDbType { get; set; }
```

13.10.3.5 ParameterName 属性

获取或设置 GBaseParameter 的名字。

- 语法

[Visual Basic]

```
Public Overrides Property ParameterName As String
    Get
    Set
```

[C#]

```
public override string ParameterName { get; set; }
```

- 实现

IDataParameter.ParameterName

13.10.3.6 Precision 属性

获取或设置用于显示 Value 属性的最大位数。

- 语法

[Visual Basic]

```
Public Property Precision As Byte
    Get
    Set
```

[C#]

```
public byte Precision { get; set; }
```

- 实现

IDbDataParameter.Precision

13.10.3.7 Scale 属性

获取或设置获得的 Value 的小数位位数。

- 语法

[Visual Basic]

```
Public Property Scale As Byte
    Get
    Set
```

[C#]

```
public byte Scale { get; set; }
```

- 实现

IDbDataParameter.Scale

13.10.3.8 Size 属性

获取或设置列中数据按字节计算的最大大小。

- 语法

[Visual Basic]

```
Public Overrides Property Size As Integer
    Get
    Set
```

[C#]

```
public override int Size { get; set; }
```

- 实现

IDbDataParameter.Size

13.10.3.9 SourceColumn 属性

获取或设置源列的名称，该源列映射到 DataSet 并用于加载或返回 Value 时使用。

- 语法

[Visual Basic]

```
Public Overrides Property SourceColumn As String
    Get
    Set
```

[C#]

```
public override string SourceColumn { get; set; }
```

- 实现

IDataParameter.SourceColumn

13.10.3.10 SourceVersion 属性

获取或设置用于装载 Value 时的 DataRowVersion。

- 语法

[Visual Basic]

```
Public Overrides Property SourceVersion As DataRowVersion
    Get
    Set
```

[C#]

```
public override DataRowVersion SourceVersion { get; set; }
```

- 实现

IDataParameter.SourceVersion

13.10.3.11 Value 属性

获取或设置参数的值。

- 语法

[Visual Basic]

```
Public Overrides Property Value As Object
    Get
    Set
```

[C#]

```
public override Object Value { get; set; }
```

- 实现

IDataParameter.Value

13.10.4 GBaseParameter 方法

13.10.4.1 ResetDbType 方法

重置与此 SqlParameter 关联的类型。

- 语法

[Visual Basic]

```
Public Overrides Sub ResetDbType
```

[C#]

```
public override void ResetDbType()
```

13.10.4.2 ToString 方法

获得一个包含 ParameterName 的字符串。

- 语法

[Visual Basic]

```
Public Overrides Function ToString As String
```

[C#]

```
public override string ToString()
```

13.11 GBaseParameterCollection 类

代表一个关于 GBaseCommand 以及它们各自映射到 DataSet 中列上的参数的集合。这个类不能被继承。对于该类所有成员的列表，参考 GBaseParameterCollection 成员。

- 继承层次

System.Object

|__ System.MarshalByRefObject

|__ System.Data.Common.DbParameterCollection

|__ GBase.Data.GBaseClient.GBaseParameterCollection

- 语法

[Visual Basic]

```
Public NotInheritable Class GBaseParameterCollection _
        Inherits DbParameterCollection
```

[C#]

```
public sealed class GBaseParameterCollection :
DbParameterCollection
```

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 注释

集合中参数的个数必须和命令文本中参数占位符的个数相等，否则会产生异常。

- 必要条件

命名空间： GBase.Data.GBaseClient

13.11.1 GBaseParameterCollection 成员

公共属性

属 性	描 述
Count	获取集合中 GBaseParameter 对象的个数。
IsFixedSize	获取一个值，该值指示 GBaseParameterCollection 是否具有固定大小。
IsReadOnly	获取一个值，该值指示 GBaseParameterCollection 是否是只读的。
IsSynchronized	获取一个值，该值指示 GBaseParameterCollection 是否是同步的。
Item	重载函数。使用指定的属性获得 GBaseParameter。在 C#中，这个属性是 GBaseParameterCollection 的索引。
SyncRoot	获取可用于同步 GBaseParameterCollection 访问的对象。

公共方法

方 法	描 述
Add	把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

方 法	描 述
Clear	从集合中移除所有对象。
Contains	获得一个值，指明是否有 GBaseParameter 对象存在于集合中。
CopyTo	从 GBaseParameterCollection 中复制 GBaseParameter 对象到指定的数组中。
Equals （继承于 Object）	判断指定的对象是否等于当前的对象。
GetType （继承于 Object）	获取当前实例的类型。
IndexOf	重载函数。获取集合中 GBaseParameter 的位置。
Insert	在指定的位置插入一个 GBaseParameter。
Remove	从集合中移除一个 GBaseParameter。
RemoveAt	重载函数。从集合中移除 GBaseParameter。
ToString （继承于 Object）	返回类的完全限定名。

13.11.2 GBaseParameterCollection 属性

13.11.2.1 Count 属性

获取集合中 GBaseParameter 对象的个数。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property Count As Integer
    Get
```

[C#]

```
public override int Count { get; }
```

- 实现

ICollection.Count

13.11.2.2 IsFixedSize 属性

获取一个值，该值指示 GBaseParameterCollection 是否具有固定大小\。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property IsFixedSize As Boolean
Get
```

[C#]

```
public override bool IsFixedSize { get; }
```

- 实现

IList.IsFixedSize

13.11.2.3 IsReadOnly 属性

获取一个值，该值指示 GBaseParameterCollection 是否是只读的。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property IsReadOnly As Boolean
    Get
```

[C#]

```
public override bool IsReadOnly { get; }
```

- 实现

IList.IsReadOnly

13.11.2.4 IsSynchronized 属性

获取一个值，该值指示 GBaseParameterCollection 是否是同步的。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property IsSynchronized As Boolean
    Get
```

[C#]

```
public override bool IsSynchronized { get; }
```

- 实现

ICollection.IsSynchronized

13.11.2.5 Item 属性

重载函数。使用指定的属性获得 GBaseParameter。在 C#中，这个属性是 GBaseParameterCollection 的索引。

- 重载列表

1) 按照指定的索引位置获得 GBaseParameter。

Item(Int32)

2) 按照指定的名称获得 GBaseParameter。

Item(String)

13.11.2.5.1 Item 属性 (Int32)

按照指定的索引位置获得 GBaseParameter。

- 语法

```
[Visual Basic]
Public Default Property Item ( _
    index As Integer _
) As GBaseParameter
    Get
    Set
[C#]
public GBaseParameter this[
    int index
] { get; set; }
```

13.11.2.5.2 Item 属性 (String)

按照指定的名称获得 GBaseParameter。

- 语法

```
[Visual Basic]
Public Default Property Item ( _
    name As String _
) As GBaseParameter
    Get
```

```
    Set
[C#]
public GBaseParameter this[
    string name
] { get; set; }
```

13.11.2.6 SyncRoot 属性

获取可用于同步 GBaseParameterCollection 访问的对象。

- 语法

```
[Visual Basic]
Public Overrides ReadOnly Property SyncRoot As Object
    Get
[C#]
public override Object SyncRoot { get; }
```

- 实现

```
ICollection.SyncRoot
```

13.11.3 GBaseParameterCollection 方法

13.11.3.1 Add 方法

把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

- 重载列表

1) 把指定的 GBaseParameter 对象加入到 GBaseParameterCollection

中。

Add(GBaseParameter)

2) 把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

Add(Object)

3) 给定参数名和数据类型，把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

Add(String, GBaseDbType)

4) 给定参数名、数据类型和列长度，把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

Add(String, GBaseDbType, Int32)

5) 给定参数名、数据类型、列长度和源列名，把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

Add(String, GBaseDbType, Int32, String)

6) 给定参数名和值，把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

Add(String, Object)

13.11.3.1.1 Add 方法 (GBaseParameter)

把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

- 语法

[Visual Basic]

```
Public Function Add ( _
    value As GBaseParameter _
```

```
) As GBaseParameter
```

[C#]

```
public GBaseParameter Add(
    GBaseParameter value
)
```

- 参数

1) value ： 要加入到集合的 GBaseParameter 对象。

- 返回值

新 GBaseParameter 对象的索引。

13.11.3.1.2 Add 方法 (Object)

把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

- 语法

[Visual Basic]

```
Public Overrides Function Add ( _
    value As Object _
) As Integer
```

[C#]

```
public override int Add(
    Object value
)
```

- 参数

1) value ： 要加入到集合的 GBaseParameter 对象。

- 返回值

新 GBaseParameter 对象的索引。

13.11.3.1.3 Add 方法 (String, GBaseDbType)

给定参数名和数据类型，把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

- 语法

```
[Visual Basic]
Public Function Add ( _
    parameterName As String, _
    dbType As GBaseDbType _
) As GBaseParameter
[C#]
public GBaseParameter Add(
    string parameterName,
    GBaseDbType dbType
)
```

- 参数

1) parameterName : 参数的名称;

2) dbType : GBaseDbType 的值之一。

- 返回值

新 GBaseParameter 对象的索引。

13.11.3.1.4 Add 方法 (String, GBaseDbType, Int32)

给定参数名、数据类型和列长度，把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

- 语法

[Visual Basic]

```
Public Function Add ( _
    parameterName As String, _
    dbType As GBaseDbType, _
    size As Integer _
) As GBaseParameter
```

[C#]

```
public GBaseParameter Add(
    string parameterName,
    GBaseDbType dbType,
    int size
)
```

- 参数

1) parameterName ： 参数的名称；

2) dbType ： GBaseDbType 的值之一；

3) size ： 列的长度。

- 返回值

新 GBaseParameter 对象的索引。

13.11.3.1.5 Add 方法 (String, GBaseDbType, Int32, String)

给定参数名、数据类型、列长度和源列名，把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

- 语法

[Visual Basic]

```
Public Function Add ( _
    parameterName As String, _
    dbType As GBaseDbType, _
    size As Integer, _
    sourceColumn As String _
) As GBaseParameter
```

[C#]

```
public GBaseParameter Add(
    string parameterName,
    GBaseDbType dbType,
    int size,
    string sourceColumn
)
```

- 参数

1) parameterName : 参数的名称;

2) dbType : GBaseDbType 的值之一;

3) size : 列的长度;

4) sourceColumn : 源列的名称;

- 返回值

新 GBaseParameter 对象的索引。

13.11.3.1.6 Add 方法 (String, Object)

给定参数名和值，把指定的 GBaseParameter 对象加入到 GBaseParameterCollection 中。

- 语法

[Visual Basic]

```
<ObsoleteAttribute("Add(String parameterName, Object value) 已经放弃.请使用 AddWithValue(String parameterName, Object value)"代替)> _
Public Function Add ( _
    parameterName As String, _
    value As Object _
) As GBaseParameter
```

[C#]

```
[ObsoleteAttribute("Add(String parameterName, Object value) 已经放弃.请使用 AddWithValue(String parameterName, Object value)"代替)]
public GBaseParameter Add(
    string parameterName,
    Object value
)
```

- 参数

1) parameterName : 参数的名称;

2) value : 加入到集合的 GBaseParameter 的值。

- 返回值

新 GBaseParameter 对象的索引。

13.11.3.2 Clear 方法

从集合中移除所有对象。

- 语法

[Visual Basic]

```
Public Overrides Sub Clear
```

[C#]

```
public override void Clear()
```

- 实现

```
IList.Clear()
```

13.11.3.3 Contains 方法

获得一个值，指明是否有 GBaseParameter 对象存在于集合中。

- 重载列表

1) 获得一个值，指明是否有 GBaseParameter 存在于集合中。

```
Contains(Object)
```

2) 获得一个值，指明是否有指定参数的 GBaseParameter 存在于集合中。

```
Contains(String)
```

13.11.3.3.1 Contains 方法 (Object)

获得一个值，指明是否有 GBaseParameter 存在于集合中。

- 语法

[Visual Basic]

```
Public Overrides Function Contains ( _
    value As Object _
) As Boolean
```

[C#]

```
public override bool Contains(
    Object value
)
```

- 参数

1) value ： 要找的 GBaseParameter 对象的值 。

- 返回值

如果集合包含 GBaseParameter 对象为真否则为假。

- 实现

IList.Contains(Object)

13.11.3.3.2 Contains 方法 (String)

获得一个值，指明是否有指定参数的 GBaseParameter 存在于集合中。

- 语法

[Visual Basic]

```
Public Overrides Function Contains ( _
    parameterName As String _
) As Boolean
```

[C#]

```
public override bool Contains(
    string parameterName
)
```

- 参数

1) name ：要找得 GBaseParameter 对象的名字 。

- 返回值

如果集合包含 GBaseParameter 对象为真否则为假。

- 实现

IDataParameterCollection.Contains(String)

13.11.3.4 CopyTo 方法

从 GBaseParameterCollection 中复制 GBaseParameter 对象到指定的数组中。

- 语法

[Visual Basic]

```
Public Overrides Sub CopyTo ( _
    array As Array, _
    index As Integer _
)
```

[C#]

```
public override void CopyTo(
    Array array,
```

```
    int index
)
```

- 实现

ICollection.CopyTo(Array, Int32)

13.11.3.5 IndexOf 方法

获取集合中 GBaseParameter 的位置。

- 重载列表

1) 获取集合中 GBaseParameter 的位置。

IndexOf(Object)

2) 使用指定的参数名，获取集合中 GBaseParameter 的位置。

IndexOf(String)

13.11.3.5.1 IndexOf 方法 (Object)

获取集合中 GBaseParameter 的位置。

- 语法

[Visual Basic]

```
Public Overrides Function IndexOf ( _
    value As Object _
) As Integer
```

[C#]

```
public override int IndexOf(
    Object value
```

)

- 参数

1) value ： 要定位的 GBaseParameter 对象 。

- 返回值

GBaseParameter 对象在集合中的位置，从 0 开始。

- 实现

IList.IndexOf(Object)

13.11.3.5.2 IndexOf 方法 (String)

使用指定的参数名，获取集合中 GBaseParameter 的位置。

- 语法

```
[Visual Basic]
Public Overrides Function IndexOf ( _
    parameterName As String _
) As Integer
[C#]
public override int IndexOf(
    string parameterName
)
```

- 参数

1) parameterName ： 要返回的 GBaseParameter 对象的名字 。

- 返回值

GBaseParameter 对象在集合中的位置，从 0 开始。

- 实现

IDataParameterCollection.IndexOf(String)

13.11.3.6 Insert 方法

在指定的位置插入一个 GBaseParameter。

- 语法

[Visual Basic]

```
Public Overrides Sub Insert ( _
    index As Integer, _
    value As Object _
)
```

[C#]

```
public override void Insert(
    int index,
    Object value
)
```

- 参数

1) index : 在集合中要插入的位置;

2) value : 要插入的 GBaseParameter。

- 实现

IList.Insert(Int32, Object)

13.11.3.7 Remove 方法

从集合中移除一个 GBaseParameter。

- 语法

[Visual Basic]

```
Public Overrides Sub Remove ( _
    value As Object _
)
```

[C#]

```
public override void Remove(
    Object value
)
```

- 参数

1) value : 要移除的对象。

- 实现

IList.Remove(Object)

13.11.3.8 RemoveAt 方法

从集合中移除指定的 GBaseParameter。

- 重载列表

1) 从集合中移除指定索引的 GBaseParameter 。

RemoveAt(Int32)

2) 从集合中移除指定参数名的 GBaseParameter 。

RemoveAt(String)

13.11.3.8.1 RemoveAt 方法 (Int32)

从集合中移除指定索引的 GBaseParameter。

- 语法

[Visual Basic]

```
Public Overrides Sub RemoveAt ( _
    index As Integer _
)
```

[C#]

```
public override void RemoveAt(
    int index
)
```

- 参数

1) index : 从 0 开始的参数索引 。

- 实现

IList.RemoveAt(Int32)

13.11.3.8.2 RemoveAt 方法 (String)

从集合中移除指定参数名的 GBaseParameter。

- 语法

[Visual Basic]

```
Public Overrides Sub RemoveAt ( _
```

```
    parameterName As String _
)
[C#]
public override void RemoveAt(
    string parameterName
)
```

- 参数

1) name ：要返回的 GBaseParameter 对象的名字 。

- 实现

IDataParameterCollection.RemoveAt(String)

13.12 GBaseRowUpdatedEventArgs 类

提供用于 RowUpdated 事件的数据。这个类不能被继承。对于该类所有成员的列表，参考 GBaseRowUpdatedEventArgs 成员。

- 继承层次

```
System.Object
  |__ System.EventArgs
    |__ System.Data.Common.RowUpdatedEventArgs
      |__ GBase.Data.GBaseClient.GBaseRowUpdatedEventArgs
```

- 语法

```
[Visual Basic]
Public NotInheritable Class GBaseRowUpdatedEventArgs _
                    Inherits RowUpdatedEventArgs
```

[C#]

public sealed class GBaseRowUpdatedEventArgs : RowUpdatedEventArgs

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 必要条件

命名空间：GBase.Data.GBaseClient

13.12.1 GBaseRowUpdatedEventArgs 成员

公共构造函数

构造函数	描 述
GBaseRowUpdatedEventArgs	初始化一个新的 GBaseRowUpdatedEventArgs 类实例。

公共属性

属 性	描 述
Command	重载函数。当调用 Update 的时候获取或设置 GBaseCommand。
Errors （继承于 RowUpdatedEventArgs）	当 Command 执行时获取.NET Framework 下接口生成的任何错误。
RecordsAffected （继承于 RowUpdatedEventArgs）	获取执行 SQL 语句更新、插入或者删除的行数。
Row （继承于 RowUpdatedEventArgs）	获得通过 Update 发送的 DataRow。
StatementType （继承于 RowUpdatedEventArgs）	获得执行的 SQL 语句类型。

属　性	描　述
Status (继承于 RowUpdatedEventArgs)	获得 Command 的 UpdateStatus 属性。
TableMapping (继承于 RowUpdatedEventArgs)	获得通过 Update 发送的 DataTableMapping。

公共方法

方　法	描　述
Equals (继承于 Object)	判断指定的对象是否等于当前的对象。
GetType (继承于 Object)	获取当前实例的类型。
ToString (继承于 Object)	返回类的完全限定名。

13.12.2 GBaseRowUpdatedEventArgs 构造函数

初始化一个新的 GBaseRowUpdatedEventArgs 类实例。

- 语法

```
[Visual Basic]
Public Sub New ( _
    row As DataRow, _
    command As IDbCommand, _
    statementType As StatementType, _
    tableMapping As DataTableMapping _
)
[C#]
public GBaseRowUpdatedEventArgs(
    DataRow row,
```

```
    IDbCommand command,
    StatementType statementType,
    DataTableMapping tableMapping
)
```

- 参数

1) row ： 通过 Update 发送的 DataRow ；

2) command ： 当调用 Update 时执行的 IDbCommand ；

3) statementType ： 指定查询类型的 StatementType 值之一 ；

4) tableMapping ： 通过 Update 发送的 DataTableMapping。

13.12.3 GBaseRowUpdatedEventArgs 属性

13.12.3.1 Command 属性

当调用 Update 的时候获取或设置 GBaseCommand。

- 语法

```
[Visual Basic]
Public ReadOnly Property Command As GBaseCommand
    Get
[C#]
public GBaseCommand Command { get; }
```

13.13 GBaseRowUpdatingEventArgs 类

提供用于 RowUpdating 事件的数据。这个类不能被继承。对于该类所有成

员的列表，参考 GBaseRowUpdatingEventArgs 成员。

- 继承层次

System.Object

|__ System.EventArgs

|__ System.Data.Common.RowUpdatingEventArgs

|__ GBase.Data.GBaseClient.GBaseRowUpdatingEventArgs

- 语法

[Visual Basic]

Public NotInheritable Class GBaseRowUpdatingEventArgs _

Inherits RowUpdatingEventArgs

[C#]

public sealed class GBaseRowUpdatingEventArgs : RowUpdatingEventArgs

- 线程安全性

这个类型的公共静态成员（在 Visual Basic 中为 Shared）对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 必要条件

命名空间：GBase.Data.GBaseClient

13.13.1 GBaseRowUpdatingEventArgs 成员

公共构造函数

构造函数	描 述
GBaseRowUpdatingEventArgs	初始化一个新的 GBaseRowUpdatedEventArgs 类实例。

公共属性

属 性	描 述
Command	重载函数。当调用 Update 的时候获取或设置 GBaseCommand。
Errors (继承于 RowUpdatedEventArgs)	当执行 Command 执行时获取.NET Framework 下接口生成的任何错误。
Row (继承于 RowUpdatedEventArgs)	获得通过 Update 发送的 DataRow。
StatementType (继承于 RowUpdatedEventArgs)	获得执行的 SQL 语句类型。
Status (继承于 RowUpdatedEventArgs)	获得 Command 的 UpdateStatus 属性。
TableMapping (继承于 RowUpdatedEventArgs)	获得通过 Update 发送的 DataTableMapping。

公共方法

方 法	描 述
Equals (继承于 Object)	判断指定的对象是否等于当前的对象。
GetType (继承于 Object)	获取当前实例的类型。
ToString (继承于 Object)	返回类的完全限定名。

13.13.2 GBaseRowUpdatingEventArgs 构造函数

初始化一个新的 GBaseRowUpdatedEventArgs 类实例。

- 语法

[Visual Basic]

```
Public Sub New ( _
    row As DataRow, _
    command As IDbCommand, _
    statementType As StatementType, _
    tableMapping As DataTableMapping _
)
```

```
[C#]
public GBaseRowUpdatingEventArgs(
    DataRow row,
    IDbCommand command,
    StatementType statementType,
    DataTableMapping tableMapping
)
```

- 参数

1) row ： 通过 Update 发送的 DataRow ；

2) command ： 当调用 Update 时执行的 IDbCommand ；

3) statementType ： 指定查询类型的 StatementType 值之一 ；

4) tableMapping ： 通过 Update 发送的 DataTableMapping。

13.13.3 GBaseRowUpdatingEventArgs 属性

13.13.3.1 Command 属性

当调用 Update 的时候获取或设置 GBaseCommand。

- 语法

[Visual Basic]

```
Public Property Command As GBaseCommand
    Get
    Set
```

[C#]

```
public GBaseCommand Command { get; set; }
```

13.14 GBaseTransaction 类

代表一个 GBase 数据库中的事务。这个类不能被继承。对于该类型的所有成员，参考 GBaseTransaction 成员。

- 继承层次

```
System.Object
  |__ System.MarshalByRefObject
    |__ System.Data.Common.DbTransaction
      |__ GBase.Data.GBaseClient.GBaseTransaction
```

- 语法

[Visual Basic]

```
Public NotInheritable Class GBaseTransaction _
Inherits DbTransaction
```

[C#]

```
public sealed class GBaseTransaction : DbTransaction
```

- 必要条件

命名空间：GBase.Data.GBaseClient

- 线程安全性

这个类型的公共静态成员(在 Visual Basic 中为 Shared)对于多线程操作是保证线程安全的，对于实例不保证线程安全性。

- 注释

程序通过在 GBaseConnection 对象上调用 BeginTransaction 来创建一个 GBaseTransaction 对象。所有后面关于事务的操作（例如：提交或者回滚事务）都在 GBaseTransaction 对象上进行。

- 示例

下面的例子创建了一个 GBaseConnection 和一个 GBaseTransaction 对象，还示范了如何使用 BeginTransaction、Commit 和 Rollback 方法。

```
[Visual Basic]
Public Sub RunTransaction(gsConnString As String)
    Dim gsConnection As New GBaseConnection(gsConnString)
    gsConnection.Open()
    Dim gsCommand As GBaseCommand = gsConnection.CreateCommand()
    Dim gsTrans As GBaseTransaction
    ' Start a local transaction
    gsTrans = gsConnection.BeginTransaction()
    ' Must assign both transaction object and connection
    ' to Command object for a pending local transaction
    gsCommand.Connection = gsConnection
    gsCommand.Transaction = gsTrans
    Try
        gsCommand.CommandText = "Insert into Region ("_
        &"RegionID,"_
        &"RegionDescription)VALUES (100, "_
        &"Description')"
        gsCommand.ExecuteNonQuery()
        gsCommand.CommandText = "Insert into Region ("_
        &"RegionID, "_
```

```
            &"RegionDescription) VALUES (101, "_
            &"'Description')"
            gsCommand.ExecuteNonQuery()
            gsTrans.Commit()
            Console.WriteLine("Both records are written to database.")
        Catch e As Exception
            Try
                gsTrans.Rollback()
                Catch ex As GBaseException
                If Not gsTrans.Connection Is Nothing Then
                    Console.WriteLine("An exception of type " &_
ex.GetType().ToString() &_
                    " was encountered while attempting to roll back the
transaction.")
                End If
            End Try
            Console.WriteLine("An exception of type "& _
e.GetType().ToString()&_
            "was encountered while inserting the data.")
            Console.WriteLine("Neither record was written to
database.")
        Finally
            gsConnection.Close()
        End Try
    End Sub 'RunTransaction

    [C#]
    public void RunTransaction(string gsConnString)
    {
        GBaseConnection gsConnection = new
GBaseConnection(gsConnString);
        gsConnection.Open();
        GBaseCommand gsCommand = gsConnection.CreateCommand();
        GBaseTransaction gsTrans;
        // Start a local transaction
        gsTrans = gsConnection.BeginTransaction();
```

```
        // Must assign both transaction object and connection
        // to Command object for a pending local transaction
        gsCommand.Connection = gsConnection;
        gsCommand.Transaction = gsTrans;
        try
        {
            gsCommand.CommandText = "Insert into Region (RegionID,
RegionDescription) VALUES (100, 'Description')";
            gsCommand.ExecuteNonQuery();
            gsCommand.CommandText = "Insert into Region (RegionID,
RegionDescription) VALUES (101, 'Description')";
            gsCommand.ExecuteNonQuery();
            gsTrans.Commit();
            Console.WriteLine("Both records are written to database.");
        }
        catch(Exception e)
        {
            try
            {
                gsTrans.Rollback();
            }
            catch (GBaseException ex)
            {
                if (gsTrans.Connection != null)
                {
                    Console.WriteLine("An exception of type " +
ex.GetType() +" was encountered while attempting to roll back the
transaction.");
                }
            }
            Console.WriteLine("An exception of type " + e.GetType() +
                                " was encountered while inserting
the data.");
            Console.WriteLine("Neither record was written to
database.");
        }
        finally
```

```
        {
            gsConnection.Close();
        }
    }
```

13.14.1 GBaseTransaction 成员

公共属性

属　性	描　述
Connection	获取和事务相关的 GBaseConnection 对象，当事务无效的时候是一个空引用（在 Visual Basic 中为空）。
IsolationLevel	为这个事务指定隔离级别。

公共方法

方　法	描　述
Commit	提交数据库事务。
Equals （继承于 Object）	判断指定的对象是否等于当前的对象。
GetType （继承于 Object）	获取当前实例的类型。
Rollback	从一个未完成的状态回滚事务。
ToString （继承于 Object）	返回类的完全限定名。

13.14.2 GBaseTransaction 属性

13.14.2.1 Connection 属性

获得关于这个对象的 GBaseConnection 对象，或者如果当事务不再有效时是空引用（在 Visual Basic 中为空）。

- 语法

[Visual Basic]

```
Public ReadOnly Property Connection As GBaseConnection
    Get
```

[C#]

```
public GBaseConnection Connection { get; }
```

- 注释

一个独立的应用程序可以有多个数据库连接，每个连接有 0 或者多个事务。此属性使您能够获取与该 GBaseTransaction 实例相关联的连接对象。

13.14.2.2 IsolationLevel 属性

指定这个事务的隔离级别，默认是 ReadCommitted。

- 语法

[Visual Basic]

```
Public Overrides ReadOnly Property IsolationLevel As IsolationLevel
    Get
```

[C#]

```
public override IsolationLevel IsolationLevel { get; }
```

- 属性值

允许的值有:

- Unspecified ：正在使用与指定隔离级别不同的隔离级别，但是无法确定该级别。
- Chaos ：无法覆盖隔离级别更高的事务中挂起的更改。
- ReadUncommitted ：可以进行脏读，意思是说，不发布共享锁，

也不接受独占锁。

- ReadCommitted ：在正在读取数据时保持共享锁，以避免脏读，但是在事务结束之前可以更改数据，从而可能导致不可重复的读取或幻像数据。
- RepeatableRead ：在查询中使用的所有数据上放置锁，以防止其他用户更新这些数据。防止不可重复的读取，但是仍可以有幻像行。
- Serializable ：在 DataSet 上放置范围锁，以防止在事务完成之前由其他用户更新行或向数据集中插入行。
- Snapshot ：通过在一个应用程序正在修改数据时存储另一个应用程序可以读取的相同数据版本来减少阻止。表示您无法从一个事务中看到在其他事务中进行的更改，即便重新查询也是如此。

- 注释

不支持并行事务。隔离级别应用于整个事务。

13.14.3 GBaseTransaction 方法

13.14.3.1 Commit 方法

提交数据库事务 。

- 语法

[Visual Basic]

```
Public Overrides Sub Commit
```

[C#]

```
public override void Commit()
```

- 实现

IDbTransaction.Commit()

- 注释

Commit 方法等价于 GBase 数据库的 COMMIT 命令。

13.14.3.2 Rollback 方法

从一个未完成的状态回滚事务。

- 语法

[Visual Basic]

Public Overrides Sub Rollback

[C#]

public override void Rollback()

- 实现

IDbTransaction.Rollback()

- 注释

回滚方法等价于 GBase 数据库的 ROLLBACK 命令。事务只能从未结束的状态回滚（在调用了 BeginTransaction 之后和 Commit 之前）。

13.15 GBaseDbType 枚举

一个特定的数据类型，用于 GBaseParameter 。

- 语法

[Visual Basic]

Public Enumeration GBaseDbType

[C#]

public enum GBaseDbType

- 成员

GBaseDbType 的成员

成员名称	描　述
Decimal	Decimal 一个固定精度和大小的数值，在-10^38 -1 和 10^38 -1 之间。
Byte	有符号的范围从-128 到 127。无符号的范围从 0 到 255。
Int16	一个 16 位的有符号数。范围是-32768 到 32767。无符号范围是 0 到 65536。
Int24	指定一个 24 位（3 个字节）有符号或无符号值。
Int32	一个 32 位有符号整数。
Int64	一个 64 位有符号整数。
Float	一个小的（单精度）浮点型数值。允许值从 -3.402823466E+38 到 -1.175494351E-38，0，1.175494351E-38 到 3.402823466E+38。
Double	一个正常大小的(双精度)浮点型数值。允许从-1.7976931348623157E+308 到 -2.2250738585072014E-308，0，和 2.2250738585072014E-308 到 1.7976931348623157E+308。
Timestamp	一个时间戳。范围从’1970-01-01 00:00:00’到 2037 年的某时。
Date	支持范围从’1000-01-01’到’9999-12-31’的日期。
Time	时间范围是’-838:59:59’到’838:59:59’。

成员名称	描 述
Datetime	支持的范围是 '1000-01-01 00:00:00.000000' 到 '9999-12-31 23:59:59.999999'。
Year	2 位或者 4 位格式（默认是 4 位）。允许的值从 1901 到 2155，2 位格式从 1970-2069。
Newdate	废除的，使用 Datetime 或 Date 类型。
VarString	一个可变长度的String包括0到65535个字符。
Bit	Bit 字段数据类型
NewDecimal	新的 Decimal 类型
Enum	一个枚举值。一个只能有一个值的字符对象，从值列表'value1','value2', ..., NULL 中选择，或者特殊的 “”错误值。一个 ENUM 可以有最多 65535 中不同的值。
Set	一个集合。可以有 0 个或者多个字符对象，每个必须从值列表'value1', 'value2', ...选择一个，一个 SET 最多可有 64 个成员。
TinyBlob	最多可 255 (2^8 - 1) 个字符的 BLOB 或 TEXT 列。
MediumBlob	最多可 16777215 (2^24 - 1) 个字符的 BLOB 或 TEXT 列。
LongBlob	最多可 4294967295 或 4G (2^32 - 1) 个字符的 BLOB 或 TEXT 列。
Blob	BLOB 或 TEXT 列最多可 65535 (2^16 - 1) 个字符。
VarChar	一个可变长度的字符串，包含 0 到 255 个字节。
String	一个固定长度的字符串。

成员名称	描　述
Geometry	Geometric(GIS)数据类型。
UByte	无符号 8 位数。
UInt16	无符号 16 位数。
UInt24	无符号 24 位数。
UInt32	无符号 32 位数。
UInt64	无符号 64 位数。
Binary	固定长度的二进制字符串。
VarBinary	可变长度的二进制字符串。
TinyText	一个有最大长度 255 (2^8 - 1) 个字符串的 TEXT 列。
MediumText	一个有最大长度 16777215 (2^24 - 1) 个字符串的 TEXT 列。
LongText	一个有最大长度 4294967295 或 4G (2^32 - 1) 个字符串的 TEXT 列。
Text	一个有最大长度 65535 (2^16 - 1) 个字符串的 TEXT 列。
Guid	一个 Guid 列

- 必要条件

命名空间: GBase.Data.GBaseClient

13.16 GBaseInfoMessageEventHandler 委托

代表一个可以处理一个 GBaseConnection 的 InfoMessage 事件的方法。

- 语法

```
[Visual Basic]
Public Delegate Sub GBaseInfoMessageEventHandler ( _
    sender As Object, _
    args As GBaseInfoMessageEventArgs _
```

```
)
[C#]
public delegate void GBaseInfoMessageEventHandler(
    Object sender,
    GBaseInfoMessageEventArgs args
)
```

- 参数

1) sender ：事件源；

2) args ：包含事件数据的 GBaseInfoMessageEventArgs 对象。

13.17 GBaseRowUpdatedEventHandler 委托

代表处理一个 GBaseDataAdapter 的 RowUpdated 事件的方法。

- 语法

```
[Visual Basic]
Public Delegate Sub GBaseRowUpdatedEventHandler ( _
    sender As Object, _
    e As GBaseRowUpdatedEventArgs _
)
[C#]
public delegate void GBaseRowUpdatedEventHandler(
    Object sender,
    GBaseRowUpdatedEventArgs e
)
```

- 参数

1) sender ：事件源；

2) e ：包含事件数据的 GBaseRowUpdatedEventArgs。

- 必要条件

命名空间：GBase.Data.GBaseClient

13.18 GBaseRowUpdatingEventHandler 委托

代表处理一个 GBaseDataAdapter 的 RowUpdating 事件的方法。

- 语法

```
[Visual Basic]
Public Delegate Sub GBaseRowUpdatingEventHandler ( _
    sender As Object, _
    e As GBaseRowUpdatingEventArgs _
)
[C#]
public delegate void GBaseRowUpdatingEventHandler(
    Object sender,
    GBaseRowUpdatingEventArgs e
)
```

- 参数

1) sender ：事件源。

2) e ：包含事件数据的 GBaseRowUpdatingEventArgs。

- 必要条件

命名空间： GBase.Data.GBaseClient

14 程序示例

14.1 执行加载数据命令获取任务 ID 和跳过行数

通过GBaseCommand执行“Load Data Infile”数据加载命令后，可以通过RecordsTaskID获取加载任务的ID，通过RecordsSkipped获取加载任务跳过行数。

示例如下：

```
using (GBaseConnection con = new GBaseConnection(connStr))
{
    con.Open();
    GBaseCommand cmd = new GBaseCommand("", con);
    //load data infile
    cmd.CommandText = @"load data infile 'ftp://test:123456@192.168.7.182/1.txt' into table test fields terminated by ',';";
    cmd.CommandTimeout = 400;
    int recordsRefected = cmd.ExecuteNonQuery();
    long recordsSkipped = cmd.RecordsSkipped;//跳过行数
    long recordsTaskId = cmd.RecordsTaskID;//任务 ID
```

```
}
```

14.2 Blob Uri 大数据字段的存取

支持 GBase UP 的 Blob Uri 字段的存取，使用时注意以下几点要求：

- 必须使用 Prepare
- 必须使用 Parameters
- 大文件必须使用 FileStream 作为 Parameters 的参数
- Select 时必须将 Blob Uri 字段放在所有字段最后
- Select 时 DataReader 必须使用 CommandBehavior.SequentialAccess 流式读取模式
- Select 时先执行 GetBlobLength()获取 Blob Uri 字段值的长度，再循环执行 GetBlob 获取

使用代码示例如下：

```
string connstr =
"server=192.168.8.29;database=song;uid=gbase;pwd=gbase20110531;pooling=false;
Ignore Prepare=false;";//连接参数中必须指定Ignore Prepare=false


FileStream fsin;
string filepath = @"d:\1.bmp";
string tablename = "test1";
fsin = new FileStream(filepath,FileMode.Open,FileAccess.Read);
long fsinlength = fsin.Length;
string sql = "drop table if exists " + tablename;
using (GBaseConnection conn = new GBaseConnection(connstr))
{
    conn.Open();
```

```
using (GBaseCommand cmd = new GBaseCommand(sql,conn))
{
    cmd.ExecuteNonQuery();
    cmd.CommandText = "create table " + tablename + " (a int, b blob uri, c
varchar(30))";
    cmd.ExecuteNonQuery();

    cmd.CommandText = "insert into " + tablename + " values(@a,@b,@c)";
    cmd.CommandTimeout = 20000;
    cmd.Parameters.AddWithValue("@a", 1);
    cmd.Parameters.AddWithValue("@b", fsin); //将FileStream作为参数传入
    cmd.Parameters.AddWithValue("@c", "song");
    cmd.Prepare();//必须先Prepare
    cmd.ExecuteNonQuery();
}
fsin.Close();

FileStream fsout;
string outpath = @"d:\outfile.bmp";
if (File.Exists(outpath))
{
    File.Delete(outpath);
}
fsout = new FileStream(outpath, FileMode.OpenOrCreate, FileAccess.Write);

sql = "select a,c,b from " + tablename; //select时将blob uri字段放在最后
using (GBaseCommand cmd = new GBaseCommand(sql,conn))
{
    cmd.Prepare();//开启prepare
    cmd.CommandTimeout = 20000;//timeout设置大，防止在读取较大数据
时超时退出
```

```
            using (GBaseDataReader reader =
cmd.ExecuteReader(CommandBehavior.SequentialAccess)) //必须设置为流式读取
            {
                while (reader.Read())
                {
                    Assert.AreEqual(1, reader.GetValue(0));
                    Assert.AreEqual("song", reader.GetValue(1));
                    long len = 0;
                    len = reader.GetBlobLength(2);//读取blob uri时，先获取列值的
长度 GetBlobLength
                    if (len == 0)
                    {
                        Console.WriteLine("blob uri is null");
                    }
                    Assert.AreEqual(len, fsinlength);//判断获取的长度等于存入时
的长度

                    byte[] result = null;
                    long alreadyRead = 0;//已经读取的长度
                    while (alreadyRead < len)//循环获取值
                    {
                        result = reader.GetBlob(2, alreadyRead); //读满，一个数据
包最大16M，内存够用 GetBlob
                        fsout.Write(result, 0, result.Length);
                        alreadyRead += result.Length;
                    }

                    fsout.Close();
                }
            }
        }
}
```

GBASE

南大通用数据技术股份有限公司
General Data Technology Co., Ltd.

微博二维码　微信二维码

技术支持热线：400-013-9696