

虚谷数据库

C# 标准接口 V3.3.3

开发指南

文档版本 01

发布日期 2024-03-15



版权所有 © 2024 成都虚谷伟业科技有限公司。

声明

未经本公司正式书面许可，任何企业和个人不得擅自摘抄、复制、使用本文档中的部分或全部内容，且不得以任何形式进行传播。否则，本公司将保留追究其法律责任的权利。

用户承诺在使用本文档时遵守所有适用的法律法规，并保证不以任何方式从事非法活动。不得利用本文档内容进行任何侵犯他人权益的行为。

商标声明



为成都虚谷伟业科技有限公司的注册商标。

本文档提及的其他商标或注册商标均非本公司所有。

注意事项

您购买的产品或服务应受本公司商业合同和条款的约束，本文档中描述的部分产品或服务可能不在您的购买或使用范围之内。由于产品版本升级或其他原因，本文档内容将不定期进行更新。

除非合同另有约定，本文档仅作为使用指导，所有内容均不构成任何声明或保证。

成都虚谷伟业科技有限公司

地址：四川省成都市锦江区锦盛路 138 号佳霖科创大厦 5 楼 3-14 号

邮编：610023

网址：www.xugudb.com

前言

概述



本文档主要介绍虚谷数据库 C# 专用接口（XuguClient 程序集）的主要功能和其简单的外围应用，旨在帮助使用虚谷数据库服务的应用开发人员快速开发有关于数据库交互的接口编程。为使用 C# 访问虚谷数据库提供技术支持。

读者对象

- 数据库管理员
- 软件工程师
- 技术支持工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 注意	用于传递设备或环境安全警示信息，若不可避免，可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。
 说明	对正文中重点信息的补充说明。“说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
01	2024-03-15	第一次发布

目录

1	C# 概述	1
1.1	C# 介绍	1
1.2	系统架构	1
1.3	开发流程	1
2	快速入手	2
2.1	编程引用程序集	2
2.2	C# 专用接口的安装和配置	2
2.2.1	Windows 版本	2
2.2.2	Linux 版本	2
3	数据类型	3
4	编程指导	7
4.1	C# 专用接口应用编程的基本步骤	7
4.2	C# 专用接口常见类的方法与属性	8
4.2.1	连接类 XGConnection	8
4.2.2	命令类 XGCommand	10
4.2.3	数据输出类 XGDataReader	13
4.2.4	数据调整类 XGDataAdapter	17
4.2.5	参数类 XGParameters	21
4.2.6	参数集合类 XGParameterCollection	23
4.2.7	事务类 XGTransaction	28
4.2.8	大对象类 XGBlob	29
4.2.9	大对象类 XGClob	30
4.2.10	连接池类 XGConnectionPool	31
4.2.11	数据记录类 XGDataRecord	33
4.2.12	连接串创建类 XGConnectionStringBuilder	36
5	示例说明	38
5.1	不带参数 SQL 的操作	38

5.1.1	多 IP 创建数据库连接	38
5.1.2	创表等初始化 SQL 环境语句	38
5.1.3	不带参数的数据录入	39
5.2	带参数 SQL 的操作	40
5.2.1	带参数 SQL 的数据录入	40
5.2.2	带参数 SQL 数据录入的其它参数形式	41
5.2.3	带参数 SQL 语句执行 UPDATE 操作	42
5.2.4	带参数 SQL 执行 DELETE 操作	42
5.3	一般结果集的获取	42
5.4	结果集更新	43
5.5	存储过程的执行	44
5.5.1	存储过程输入参数的执行	44
5.5.2	存储过程输出参数的执行	45
5.6	存储函数包操作	47
5.6.1	存储函数包的执行	47
5.6.2	存储函数带引用型游标输出结果集	50
5.7	大对象应用	51
6	异常处理	53
6.1	XGConnection Error	53
6.1.1	System.InvalidOperationException	53
6.1.2	System.InvalidOperationException	53
6.1.3	USE DATABASE " + databaseName + " Failure	53
6.1.4	System.ConnectionIsOpenedException	54
6.1.5	System.ActionConnectionException	54
6.1.6	System.ConnectionStrException	54
6.1.7	CLOSE CURSOR Failure	54
6.2	XGCommand Error	55
6.2.1	System.InvalidOperationException	55
6.2.2	System.commandtextException	55
6.2.3	System.CommandPrepareException	55

6.2.4	System.CommandExecuteReaderException	55
6.2.5	System.CommandExecuteException	56
6.3	XGConnectPool Error	59
6.3.1	System.PoolnumOutOfRangeExcepion	59
6.3.2	System.ConnAttributeLessException	59
6.3.3	System.WaitConnTimeOutException	59
6.4	XGDataAdapter Error	60
6.4.1	System.InvalidUpdateException	60
6.5	XGDataReader Error	60
6.5.1	System.WithOutResultException	60
6.5.2	System.InvalidOperationException	60
6.5.3	System.IndexOutOfRangeException	60
6.5.4	this is Get Decimal	61
6.5.5	无法将类型'System.DBNull' 转换为 bool	61
6.5.6	无法将类型'System.DBNull' 转换为 byte	61
6.5.7	无法将类型'System.DBNull' 转换为 DateTime	62
6.5.8	无法将类型'System.DBNull' 转换为 decimal	62
6.5.9	无法将类型'System.DBNull' 转换为 double	62
6.5.10	无法将类型'System.DBNull' 转换为 float	62
6.5.11	无法将类型'System.DBNull' 转换为 short	63
6.6	XGParameters Error	63
6.6.1	ArgumentNullException	63
6.6.2	System.InvalidCastException	63
7	常见问题及解答	65
7.1	关于自动提交的问题	65
7.2	关于存储过程及函数的执行的问题	65
7.3	字符集编码的问题	65
7.4	执行 SQL 语句报错	66
7.4.1	SQL 语句类型与执行函数不匹配	66
7.4.2	SQL 语句的参数绑定错误	66

7.4.3	获取结果集失败	66
7.4.4	数据类型转换错误	66
7.4.5	连接尚未处于活动状态	66
7.5	常见错误定位	67
7.5.1	引用驱动失败	67
7.5.2	数据库连接失败	67

1 C# 概述

1.1 C# 介绍

虚谷数据库 C# 专用接口（Xugu 虚谷数据库 C# 专用接口（XuguClient 程序集）是为 C# 语言在 .net 平台做数据库研发所专业研发的数据库交互封装接口。其功能与 Microsoft 的 SqlClient 以及 Oracle 的 OraClient 类似，为操作虚谷数据库提供相应的类和方法。

1.2 系统架构

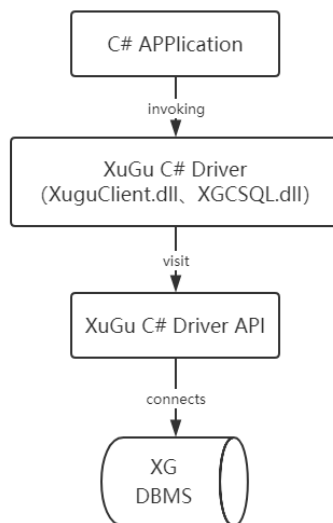
虚谷数据库 C# 专用接口总体结构有两层：

- 上层：用户调用数据的方法，是 XuguClient 程序集提供的类和方法，采用 C# 语言编写，面向对象以类对象和其相应的封装方法实现用户操作数据的目的。
- 下层：其内部与数据库交互的部分：建立连接，接收与发送数据等底层操作均由一个内部的 C++ 动态库实现。

1.3 开发流程

虚谷数据库 C# 专用接口（XuguClient 程序集）使用流程框架如图 1-1 所示。

图 1-1 C# 专用接口使用流程图



2 快速入手

2.1 编程引用程序集

C# 接口程序在代码中使用前，需将 XuguClient 作为程序集被引用至用户的 C# 项目中，文件中使用 using XuguClient 申明其命名空间，以方便用户调用程序集中的各类的实例化以及其方法。

2.2 C# 专用接口的安装和配置

2.2.1 Windows 版本

C# 专用接口是以动态库的形式存在，用户仅仅需要引用 C# 专用接口的动态库文件至本地项目组中，然后使用其中的类函数即可。

操作步骤

1. 引用 XuguClient.dll 文件，点击工程中的“引用”，右键选择“添加引用”，单击“浏览”选择文件的位置，最后点击“确定”。
2. 将 xugusql.dll 文件存放至用户本地项目组的 BIN 目录下，与生成的 exe 可执行文件同级目录。

2.2.2 Linux 版本

操作步骤

1. 添加 dotnet nuget 可用包列表。

```
cp ./XuguClient.2.0.0.nupkg /home/package  
dotnet nuget add source /home/package -n XuguClient
```

2. 添加驱动包。

```
dotnet add package XuguClient
```

3 数据类型

以下是虚谷数据库数据类型与 C# 数据类型的映射，实际参数绑定部分代码请参见章节4.2.6参数集合类 XGParameterCollection 的参数绑定示例。

虚谷数据库数据类型	数据长度 (所占字节)	C# 对应的数据类型	说明
Char(n)	n 字节，最大不超过 64K	char、string	固定串长度为 n 的字符串
Varchar(n)	n 字节，最大不超过 64K	string	最大字符串长度为 n 的可变长度字符串
Binary(n)	n 字节，最大不超过 64K	Array	固定长度为 n 的二进制数据
Tinyint	1 字节	SByte	精度为 3，标度为 0 的有符号精确数字，范围-128– 127
Smallint	2 字节	Int16	精度为 5，标度为 0 的有符号精确数字范围 -32768 – 32767
Integer	4 字节	Int32	精度为 10，标度为 0 的有符号精确数字 C int 的取值
Bigint	8 字节	Int64	精度为 19，标度为 0 的有符号精确数字值 int64 的取值范围
Float	4 字节	Float	浮点数类型
			接下一页

虚谷数据库数据类型	数据长度 (所占字节)	C# 对应的数据类型	说明
Double	8 字节	Double	双精度浮点数字
Bool	1 字节	Bool	布尔类型, 取值 true/false 或者 'T' / 'F'
Numeric(p,s)	20 字节	Decimal	精度为 p, 标度为 s 的有符号精确数字值
Time	4 字节	Datetime	时间数据类型, 时分秒字段
Datetime	8 字节	Datetime	时间戳数据类型, 年月日时分秒字段
Date	4 字节	Datetime	日期数据类型, 年月日字段
Time with time zone	6 字节	string	时间数据类型, 时分秒, 时区字段
Datetime with time zone	10 字节	string	时间戳数据类型, 年月日时分秒, 时区字段
Blob	最大不超过 256MB	Array	二进制大对象类型字段
Clob	最大不超过 256MB	string	字符大对象的存储字段
Interval year	4 字节	string	年间隔, 即两个日期之间的年数字
			接下页

虚谷数据库数据类型	数据长度 (所占字节)	C# 对应的数据类型	说明
Interval month	4 字节	string	月间隔, 即两个日期之间的月数字
Interval day	4 字节	string	日间隔, 即两个日期之间的日数字
Interval hour	4 字节	string	时间间隔, 即为两个日期/时间之间的时数字
Interval minute	TYPE_INTERVAL_MI	string	分间隔, 即为两个日期/时间之间的分数字
Interval second	8 字节	string	秒间隔, 即为两个日期/时间之间的秒数字
Interval day to hour	4 字节	string	日时间间隔, 即为两个日期/时间之间的日时数字
Interval day to minute	4 字节	string	日时分间隔, 即为两个日期/时间之间的日时分数字
Interval day to second	8 字节	string	日时分秒间隔, 即为两个日期/时间之间的日时分秒数字
Interval hour to minute	TYPE_INTERVAL_H2M	string	时分间隔, 即为两个日期/时间之间的时分数字
			接下页

虚谷数据库数据类型	数据长度 (所占字节)	C# 对应的数据类型	说明
Interval hour to second	8 字节	string	时分秒间隔，即为两个日期/时间之间的时分秒数字
Interval minute to second	8 字节	string	分秒间隔，即为两个日期/时间之间的分秒间隔
Interval year to month	4 字节	string	年月间隔，即两个日期之间的年月数字

4 编程指导

4.1 C# 专用接口应用编程的基本步骤

C# 为面向对象编程语言，对数据库编程中须要用到的类有 XGConnection、XGCommand。常用的类有 XGParameters、XGDataReader、XGDataAdapter 等。

常规应用的一般顺序为：

1. 申明连接对象 XGConnection，赋值连接串设定连接信息，建立与数据库服务器的连接。
2. 申明执行命令对象 XGCommand，设定其连接属性为上述连接对象。设定与数据库交互的 sql 语句，有参数进行参数绑定，选择适当的执行方法调用 XGCommand 与数据库交互。有结果集返回则接收结果集，将结果集返回提供给 XGDataReader 对象供用户访问。
3. 释放 XGCommand 对象。
4. 释放 XGConnection 对象。

操作示例如下：

```
Using XuGuClient;

XGConnection conn= new XGConnection();
Conn.ConnectionString= " IP=192.168.2.214;DB=SYSTEM;User=SYSDBA;PWD=
    SYSDBA;Port=5138;AUTO_COMMIT=on;CHAR_SET=GBK " ;
Try{
    Conn.Open();
    XGCommand mycmd= new XGCommand();
    mycmd.Connection=conn;
    mycmd.CommandText= " update t1 set it=5 where pid=1 " ;
    int effect= mycmd.ExecuteNonQuery();
    mycmd.Dispose();
}
Catch(Exception ei)
{
    Console.WriteLine(ei.ToString());
}
Finally{
    conn.Close();
}
```

上述代码是与数据库交互的简单示例：引用虚谷驱动，建立并打开了一个数据库的连接，将 t1 表中的 pid=1 的数据的 it 值置为 5；最终返回值为影响的行数，当表中没有数据符合筛选条件 pid=1 时，影响的行数为 0。

设置正确的会话连接参数和建立数据库连接是操作数据库的基础和前提。

与数据库交互的命令类 XGCommand 类对象：

- 可以设置 sql 语句；
- 可以设定参数；
- 可以设置事务；
- 用户可根据是否需要返回结果或者结果集，选择 ExecuteNonQuery()、ExecuteScalar()、ExecuteReader() 等不同的执行方式。

当收到结果集时可以选择 XGDataReader 对象浏览数据集，也可以选择 XGDataAdapter 对象，将数据填入 DataSet 或者 DataTable。

📖 说明

- 使用 XGDataAdapter 对象结合以上公共控件进行访问时在单表结果集的情况下可以修改结果集的数据，反向同步至数据库端。
- 完成与数据库的交互操作之后，需要关闭相应的类对象释放相关的资源，关闭连接，避免不再使用的资源不能及时释放而增加开销。如不及时释放，有因当前连接数过多而导致新连接不能建立的风险。

4.2 C# 专用接口常见类的方法与属性

4.2.1 连接类 XGConnection

功能

连接类的主要作用是连接数据库，为语句类 XGCommand 提供连接基础，也可以为事务类 XGTransaction 提供事务环境基础。

说明

连接打开方式是使用 Open 函数，连接关闭方式是使用 Close 函数。

在使用连接池时，打开连接的函数略有不同，且关闭连接使用 Close 函数时也不是真的关闭连接，而是将连接释放到连接池。

类属性介绍

属性	说明
ConnectionString	连接串属性，里面是各个连接参数的键值对，包括 IP、DBNAME、USER、PASSWORD、Port、CHARSET 等。其中 IP 是指连接的服务器端的 IP 地址，DBNAME 是连接的数据库名，User 是用户名，Password 是用户对应的口令，Port 表示连接的端口信息，Charset 是指连接的用户端使用的字符集，当客户端的字符集与连接的字符集不匹配时，输出的中文等数据可能会有乱码信息
State	连接状态信息。分为 open, closed, Connecting, Executing, Fetching, Broken 等
DataSource	连接服务器端信息（IP 地址）
Database	连接的数据库信息

类方法介绍

方法	参数	返回值	说明
Open()	-	-	打开数据库连接。如果网络不通，或者服务器端的连接信息有误，或者服务器关闭，那么此函数会抛出异常
Close()	-	-	关闭数据库连接。在启动连接池的情况下，不会关闭连接，而是将连接归还连接池

接下一页

方法	参数	返回值	说明
XGConnection(string conn_str)	conn_str: 连接字符串	-	带连接串的构造函数
Dispose()	-	-	释放连接资源
BeginTransaction()	-	XGTransaction 对象	在当前连接中启动事务环境，这样在后续的操作集合体时，可以根据需要进行提交确认或者集体回滚
CreateCommand()	-	XGCommand 对象	以当前连接为连接属性创建 XGCommand 对象

4.2.2 命令类 XGCommand

功能

此类主要作用是设置 sql 语句与数据库进行交互，一般可以执行的语句按功能可以分为：查询语句，插入语句，修改语句，删除语句，设定某些属性的语句，执行存储过程或函数的语句等。

说明

根据用户的执行目的和对结果的需求不同，提供不同的执行方法供用户进行选择，但均需有 XGConnection 对象作为连接成员，且在执行命令时需要保证连接是打开的。当 sql 中带有参数信息时，XGCommand 类中需设定执行时所需的参数信息，参数一般放在参数集合成员变量 Parameters 内部。有关参数集的部分请参见章节4.2.6参数集合类 XGParameterCollection。

类属性介绍

属性	说明
Connection	连接成员，为 XGConnection 类的对象，需注意执行方法时，保证连接是 open 状态，否则语句执行失败
CommandText	命令执行的 sql 语句。若执行的是存储过程或存储函数，则此处只需填写存储过程或函数的名称，若是包内的过程或函数，则需以“包名. 过程名”这样的格式填写。当填写的内容是存储过程或包时，需要将属性 CommandType 设定为 StoreProcedure
CommandType	命令类型。可设定为一般语句和存储过程两类。默认情况此处类型是一般语句。当执行存储过程时，此处设值为 StoreProcedure
Parameters	参数集成员，在有参数的情况下，提供增添和修改参数的集合，为执行特定的带参数 sql 提供参数支持
Use_Server_Cursor	使用服务器端游标开关。当此属性设置为 true 时，对 select 查询数据的语句有效，此时查询的结果集数据将存放于服务器端，而不是缓冲在本地
Transaction	在显式设定了事务的情况下，可以进行事务操作，但是这个不推荐在命令处使用，而是一般在连接上设定事务，由连接对事务进行统一的提交和回滚操作

类方法介绍

方法	参数	返回值	说明
XGCommand(string cmd_sql, XGConnection conn)	cmd_sql: sql 语句; conn: 连接对象	无	带 sql 与连接信息的构造函数，一步设定连接变量与 sql 语句变量
			接下页

方法	参数	返回值	说明
ExecuteNonQuery()	无	成功: ≥0; 失败: -1	执行非查询型的 sql 语句。当 sql 为 select 语句时直接返回-1; 当语句为 insert、update 或 delete 时, 返回在数据库中所影响的行数。一般的设置型语句也是通过此函数执行。(说明: 此方法无需接收结果集。但如需要达到一定输出的目的, 可以用此方法执行带输出参数的存储过程, 以输出参数达成目的输出)
ExecuteReader()	无	成功: 0; 失败: -1	此函数用于执行能返回结果集的查询语句。当需要服务器端缓存数据时, 请提前设定 Use_Server_Cursor 的值。此函数返回值为 XGDataReader 对象, 是遍历结果集的一个对象

接下页

方法	参数	返回值	说明
ExecuteScalar()	无	成功：0；失败：-1	此函数返回结果集的首行首列，因此一般用于执行统计类查询比较多（类似 select count() 这类的），一般结果集也可以返回首行首列的数据
Dispose()	无	无	资源释放函数，调用此函数会释放申请的资源，对引用的资源的减少引用的统计次数

4.2.3 数据输出类 XGDataReader

功能

这是一个数据集访问类，执行 XGCommand 的 ExecuteReader 方法得到的结果集，可以使用此类对象进行浏览遍历。

说明

该对象使用 read 方法遍历数据集中的每一行，遍历只能单向前滚不能反向遍历。每执行一个 read 方法，当前行的数据发生改变，可调用该方法取得当前行的某一列的信息。

说明

返回数据信息因数据格式不同，需要采用对应的数据类型存放，否则可能发生数据类型转化失败的情况。

类属性介绍

属性	说明
FieldCount	结果集的列数。为用户按列取数提供边界依据
IsClosed	结果集是否已经关闭。已关闭的结果集不可以继续浏览后续数据
RecordsAffected	结果集中所影响的行数。此属性一般在有结果集更新时有用
This[i]	当前行的第 i 列的数据

类方法介绍

方法	参数	返回值	说明
Read()	无	成功: true; 失败: false	读取结果集中的一行数据到当前位置, 并返回成功, 若没有读到数据或者结果集已读取完毕, 再次读数据时返回失败
IsDBNull(i)	i: 结果集当前行的列号	为空: true; 不为空: false	判断当前行第 i 列是否为空
GetValue(i)	i: 结果集当前行的列号	包含数据的 object 对象	取回当前行第 i 列的数据
GetTableName()	无	包含结果集表名的字符串	取回当前结果集的表名
GetString(i)	i: 结果集当前行的列号	成功: 包含数据的字符串; 失败: null	以字符串的形式取回当前行的第 i 列的数据
			接下页

方法	参数	返回值	说明
GetOrdinal(string name)	name: 需要查询的列名	成功: 列序号; 失败: -1	根据列名查询列的序号
GetName(i)	i: 结果集当前行的列号	包含列名的字符串	取得第 i 列的列名
GetFieldType(i)	i: 结果集当前行的列号	System.Type.GetType()	取得第 i 列的数据类型
GetInt64(i)	i: 结果集当前行的列号	Long 类型的数据	以 64 位的 Int 长整型数据取回第 i 列的值
GetInt32(i)	i: 结果集当前行的列号	Int 类型的数据	以 Int 型取数第 i 列的值
GetInt16(i)	i: 结果集当前行的列号	Short 类型数据	以 16 位短整型数据取回第 i 列的值
GetFloat(i)	i: 结果集当前行的列号	Float 类型数据	以浮点型数据取回第 i 列的值
GetDouble(i)	i: 结果集当前行的列号	Double 类型数据	以双精度浮点型数据取回第 i 列的值
GetDecimal(i)	i: 结果集当前行的列号	Decimal 类型数据	以 C# 的 Decimal 型数据取回第 i 列的值
GetDateTime(i)	i: 结果集当前行的列号	DateTime 类型数据	以 datetime 数据类型取回第 i 列数据
GetDate(i)	i: 结果集当前行的列号	DateTime 类型数据	以 datetime 的形式取回日期型数据
			接下页

方法	参数	返回值	说明
GetChars(int i,long dataindex,char[] buffer,int bufferIndex,int length)	i: 结果集当前行的列号; dataindex: 需要截取的字符串的起点位置; buffer: 获取字符串的缓冲区; bufferIndex: buffer 的长度; length: 需要截取的字符串长度	截取字符串的长度	以某个点为起点取回长度 length 的第 i 列的字符串的部分值
GetChar(i)	i: 结果集当前行的列号	包含获取到的数据的字符串	取字符型数据第 i 列的值
GetBytes(int l,long dataindex,byte[] buffer,int bufferIndex,int length)	i: 结果集当前行的列号; dataindex: 需要截取的字符串的起点位置; buffer: 获取字符串的缓冲区; bufferIndex: buffer 的长度; Length: 需要截取的字符串长度	截取字节流长度	以某个点为起点取回长度 length 的第 i 列的字节流的部分值
GetByte(i)	i: 结果集当前行的列号	Byte 类型数据	取字节型数据第 i 列的值
			接下一页

方法	参数	返回值	说明
GetBoolean(i)	i: 结果集当前行的列号	Bool 类型数据	取回 bool 型数据第 i 列的值
Dispose()	-	-	释放非引用型资源。 减少引用型资源的引用次数
Close()	-	-	关闭结果集

4.2.4 数据调整类 XGDataAdapter

功能

此类会根据查询语句类成员变量生成 update、insert、delete 语句等。但其适用范围有限制，只有单表查询生成的结果集才支持此功能，而多表连接查询生成的结果集不支持此功能。

说明

有别于其他数据库，其他的数据库包括 Oracle、SQLServer 均需要表有主键或者唯一值索引等列提供标识列信息。而虚谷的驱动 XuguClient 采用了特殊的机制不需要此要求。即便没有主键列也可以定位修改表列数据。

类属性介绍

属性	说明
SelectCommand	生成当前操作结果集的查询语句。是被 XGDataAdapter 的对象操作的基础数据来源
InsertCommand	对当前结果集补充数据的 sql 语句，补充插入的记录会在 Update 方法调用时生效，但要求当前结果集来自单表
UpdateCommand	对当前结果集更新某些数据的 sql 语句，更新的记录数据会在 Update 方法调用时生效，但要求当前结果集来自单表
接下页	

属性	说明
DeleteCommand	对当前结果集的某些记录进行删除的 sql 语句，删除的记录会在 Update 方法调用时生效，但要求当前结果集来自单表

类方法介绍

方法	参数	返回值	说明
XGDataAdapter(XGCommandselectCommand)	selectCommand: 查询语句	-	以查询语句初始化当前数据调整类
XGDataAdapter(stringselectCommandText,XGConnectionselectConnection)	selectCommandText: 查询 SQL 语句; selectConnection: 连接字符串	-无	以查询的 sql 语句和适当的连接信息初始化当前数据调整类
XGDataAdapter(stringselectCommandText,stringselectConnectionString)	selectCommandText: 查询的 SQL 语句; selectConnectionString: 连接字符串	-	以查询的 sql 语句和适当的连接串初始化当前数据调整类
			接下页

方法	参数	返回值	说明
Fill(DataSet dataset ,string srcTable)	dataset: DataSet 对象; srcTable: dataset 的表单名, 指需要将结果集填入的表单	已在 System.Data.DataSet 中成功添加或刷新的行数	将 select 对应的结果集数据填入 dataset 里面的名为 srcTable 的表单中, 此时会开启数据集变更模式, 如果有更新, 且数据集来自单表而非来自统计或者多个表的联合查询, 那么在调用 update 方法时则可以将更新同步推送至数据库
Fill(DataSet dataset)	dataset: DataSet 对象	已在 System.Data.DataSet 中成功添加或刷新的行数	将当前结果集填入名为 Table1 的表单中
Fill(DataTable dataTable)	dataTable: 需要填入结果集的表单	已在 System.Data.DataSet 中成功添加或刷新的行数	将当前结果集填入名为 dataTable 的表单中
			接下页

方法	参数	返回值	说明
Update(DataSet dataset, string srcTable)	dataset: DataSet 对象; srcTable: dataset 的表单名, 指需要将结果集填入的表单	成功: 0	更新 dataset 内名为 srcTable 表单中被修改过的数据, 并将其同步至数据库中, 但当前记录集不会由 select 语句进行重新刷新, 例如 select 语句中选择的数据行, 此时更新之后部分数据的 pid 不再是 1, 理应刷出当前结果集之外, 但因没有进行刷新操作, 因此还保留在当前结果集中可以供用户继续操作
Update(DataSet dataset)	dataset: DataSet 对象	成功: 0	更新 dataset 中首个表单的记录集有变更的数据并将其同步至数据库中
Update(DataTable dataTable)	dataTable: 需要更新的表单	成功: 0	更新表单中有变更的数据, 将其同步至数据库中
Dispose()	-	-	释放相关的资源

4.2.5 参数类 XGParameters

功能

类继承自 DbParameter，对象一般挂载到 XGParameterCollection 下面，作为 XGCommand 对象的参数集元素而存在。

说明

参数一般有参数名、参数类型、参数的数据类型、参数值等种种属性，其赋值方式也有许多的形式。需要注意参数被使用一次后不会自行失效清空，需要用户自行清空参数链。

类属性介绍

属性	说明
DbType	参数数据类型
Param_no	参数序列号，此参数与 sql 语句相关
ParameterName	参数名
IsNullable	是否可以空
Size	参数大小
Direction	参数的输入输出类型
Value	参数值

类方法介绍

方法	参数	返回值	说明
XGParameters(strin gname,objectvalue)	name: 参数名; value: 参数值	-	带参数名和参数值的 初始化方法
XGParameters(XGP arametersparam)	param: XGPareme- ters 类的参数对象	-	以另一个参数为模板 的初始化方法
			接下页

方法	参数	返回值	说明
XGParameters(stringname,XGDbTypeype)	name: 参数名; type: 参数类型	-	带参数名和参数类型的初始化方法, 在某些特定的情况下可以确定参数的长度
XGParameters(stringname,XGDbTypeype,intsize)	name: 参数名; type: 参数类型; size: 参数长度	-	带参数名、类型、长度的参数初始化方法, 此方法多用于变长值类型的参数, 例如: char、varchar、Binary 等
			接下页

方法	参数	返回值	说明
XGParameters(string parameterName, XGDbType dtType, int size, ParameterDirection parameterDirection, bool isNullable, byte precision, byte scale, string srcColumn, DataWoeVersion srcVersion, object value)	parameterName: 参数名; dtType: 参数数据类型; size: 参数长度; parameterDirection: 参数输入输出类型; isNullable: 参数是否可为空值; precision: 参数精度; scale: 参数范围; srcColumn: 对应的数据库端列名; srcVersion: 对应的数据库端版本号; value: 参数值	-	以较为详细全面的信息初始化参数
Param_isFixed(XGDbType type)	type: 参数数据类型	是: true; 否: false	根据数据类型判断参数是否是固定长度的

4.2.6 参数集合类 XGParameterCollection

功能

此类对象一般作为 XGCommand 对象的成员出现的。为带参数的语句类提供参数支持。

说明

参数集使用完后需要用户自行清空参数链，否则可能发生上一个 sql 执行的参数被填充到当前 sql 的参数执行链的情况。

类属性介绍

属性	说明
Count	参数个数
IsFixedSize	当前参数的类型是否是可变长度的
IsReadOnly	当前参数是否是只读的
IsSynchronized	是否同步的

类方法介绍

方法	参数	返回值	说明
Add(object value)	value: 携带参数值的参数对象	当前参数的位置	添加一个参数, 携带参数值
Add(XGParameter value)	value: XGParameter 的参数对象	XGParameter 的参数对象	添加一个参数, 参数本身为参数类型的实例对象
Add(string parameterName ,object value)	parameterName: 参数名; value: 携带参数值的参数对象	XGParameter 的参数对象	添加一个参数携带参数名以及值
Add(string parameterName ,XGDbType DbType)	parameterName: 参数名; DbType: 参数类型	XGParameter 的参数对象	添加一个参数携带参数名和参数数据类型
Add(string parameterName ,XGDbType DbType ,int size)	parameterName: 参数名; DbType: 参数类型; size: 参数长度	XGParameter 的参数对象	添加一个参数携带参数名、数据类型以及长度
			接下页

方法	参数	返回值	说明
Add(string parameterName ,XGDbType DbType , object value ParameterDirection in_out)	parameterName: 参数名; DbType: 参数类型; value: 参数对象; in_out: 参数输入输出类型	XGParameter的参数对象	添加一个参数, 附带参数名、数据类型、参数值以及输入输出类型
AddRange(Array values)	values: 参数组	-	在参数链中一次添加多个参数
AddWithValue(string parameterName,objectvalue)	parameterName: 参数名; value: 参数对象	XGParameter的参数对象	在参数链中添加一个带参数名, 参数值的参数
Clear()	-	-	清空参数链。XG-Command 执行完成后, 需要执行新的sql 时, 一般需要清空参数链
IndexOf(object value)	value: 参数对象	返回值为 value 的参数的序号	在参数链中寻值为 value 的参数, 并在找到后返回其序号, 否则返回-1
			接下页

方法	参数	返回值	说明
IndexOf(string parameterName)	parameterName: 参数名	返回名为 parameterName 的参数的序号	在参数链中寻找名为 parameterName 的参数，并在找到后返回其序号，否则返回-1
Remove(object value)	value: 参数对象	无	从参数链中移除与 value 指定的相同参数

参数绑定示例

数据库数据类型	C# 绑定参数部分示例代码
Tinyint	<code>t_Cmd.Parameters.Add("COL", XGDbType.TinyInt).Value = 127;</code>
Smallint	<code>t_Cmd.Parameters.Add("COL", XGDbType.SmallInt).Value = 32767;</code>
Integer	<code>t_Cmd.Parameters.Add("COL", XGDbType.Int).Value=2147418111;</code>
Bigint	<code>t_Cmd.Parameters.Add("COL", XGDbType.BigInt).Value=9223090561878;</code>
Float	<code>t_Cmd.Parameters.Add("COL", XGDbType.VarChar).Value="1.1234567";</code>
Double	<code>t_Cmd.Parameters.Add("COL", XGDbType.Double).Value=3.1234567890;</code>
Numeric	<code>t_Cmd.Parameters.Add("COL", XGDbType.Numeric).Value = 3.16 ;</code>
Char	<code>t_Cmd.Parameters.Add("COL", XGDbType.Char).Value = "DO";</code>
接下页	

数据库数据类型	C# 绑定参数部分示例代码
Varchar	t_Cmd.Parameters.Add("COL",XGDbType.VarChar).Value="XuGu";
Clob	t_Cmd.Parameters.Add("COL",XGDbType.LongVarChar).Value=t_Clob;
Blob	t_Cmd.Parameters.Add("COL", XGDbType.LongVarBinary).Value = t_Blob;
Boolean	t_Cmd.Parameters.Add("COL", XGDbType.Bool).Value = true ;
Date	t_Cmd.Parameters.Add("COL",XGDbType.Date).Value="2018-5-10";
DateTime	t_Cmd.Parameters.Add("COL",XGDbType.DateTime).Value="2018-5-109:42:45";
DateTimeWithTimeZ one	t_Cmd.Parameters.Add("COL", XGDbType.VarChar).Value = "2018-5-14 10:34:45 +8:00" ;
Time	t_Cmd.Parameters.Add("COL", XGDbType.Time).Value = "9:46:32";
TimeWithTimeZone	t_Cmd.Parameters.Add("COL",XGDbType.VarChar).Value="10:34:45+8:00";
IntervalYear	t_Cmd.Parameters.Add("COL", XGDbType.VarChar).Value = "22" ;
IntervalMonth	t_Cmd.Parameters.Add("COL", XGDbType.VarChar).Value = "8";
IntervalHour	t_Cmd.Parameters.Add("COL", XGDbType.VarChar).Value = "9";
IntervalMinute	t_Cmd.Parameters.Add("COL", XGDbType.VarChar).Value = "10";
IntervalSecond	t_Cmd.Parameters.Add("COL", XGDbType.VarChar).Value = "11";
接下页	

数据库数据类型	C# 绑定参数部分示例代码
IntervalYearToMonth	<code>t_Cmd.Parameters.Add("COL",XDbType.IntervalY2M).Value="22-8";</code>
IntervalDayToHour	<code>t_Cmd.Parameters.Add("COL",XDbType.VarChar).Value="1123";</code>
IntervalDayToMinute	<code>t_Cmd.Parameters.Add("COL",XDbType.VarChar).Value="1123:23";</code>
IntervalDayToSecond	<code>t_Cmd.Parameters.Add("COL",XDbType.IntervalD2s).Value="802523:23:23";</code>
IntervalHourToMinute	<code>t_Cmd.Parameters.Add("COL",XDbType.VarChar).Value="23:23";</code>
IntervalHourToSecond	<code>t_Cmd.Parameters.Add("COL",XDbType.VarChar).Value="23:23:23";</code>
IntervalMinuteToSecond	<code>t_Cmd.Parameters.Add("COL",XDbType.VarChar).Value="23:23";</code>
Binary	<code>t_Cmd.Parameters.Add("COL",XDbType.Binary).Value="t_Binary";</code>
VarBinary	<code>t_Cmd.Parameters.Add("COL",XDbType.VarBinary).Value="t_VarBinary";</code>

4.2.7 事务类 XGTransaction

功能

用于创建事务和操作事务。

说明

事务依托于连接，里面包含一个或者多个 XGCommand 的执行命令。多命令 sql 的执行在最后进行统一的提交或者回滚，从而保证事务特性。

类属性介绍

属性	说明
Connection	事务所属的连接
IsolationLevel	事务隔离级别

类方法介绍

方法	参数	返回值	说明
Commit()	-	-	提交事务
Rollback()	-	-	回滚事务
Dispose()	-	-	回滚结束事务

4.2.8 大对象类 XGBlob

功能

操作大对象的类，主要功能是插入 BLOB 文件和导出 BLOB 文件。

说明

主要针对图片等二进制文件在虚谷数据库的导入导出操作。

类属性介绍

无

类方法介绍

方法	参数	返回值	说明
BeginChunkWrite()	-	-	创建一个大对象句柄，写入数据的前置方法
EndChunkWrite()	-	-	标识大对象写入数据完毕
			接下页

方法	参数	返回值	说明
Write(byte[] b_data , int index ,int length)	b_data: 需要录入的大对象数据; index: 大对象的偏移位置, 即需要在大对象的什么偏移位置进行写入; length: 录入的大对象长度	写入的数据长度	往大对象中写入数据, b_data 表示需要录入的大对象数据, index 表示需要在大对象的什么偏移位置进行写入, length 表示本次录入的大对象数据长度, 可分段写入注意偏移量即可
Close()	-	-	关闭大对象并释放在驱动内部的大对象资源
Dispose()	-	-	释放大对象资源, 包括对象句柄

4.2.9 大对象类 XGClob

功能

操作大对象的类, 主要功能是插入 CLOB 文件和导出 CLOB 文件。

说明

主要针对 txt 等文本文件在虚谷数据库的导入导出操作。

类属性介绍

无

类方法介绍

方法	参数	返回值	说明
BeginChunkWrite()	-	-	创建一个大对象句柄，开始写入数据的前置方法
EndChunkWrite()	-	-	标识大对象写入数据完毕
Write(byte[] b_data , int index ,int length)	b_data: 需要录入的大对象数据; index: 大对象的偏移位置，即需要在大对象的什么偏移位置进行写入; length: 录入的大对象长度	写入的数据长度	往大对象中写入数据，b_data 表示需要录入的大对象数据，index 表示需要在大对象的什么偏移位置进行写入，length 表示本次录入的大对象数据长度，可分段写入注意偏移量即可
Close()	-	-	关闭大对象并释放在驱动内部的大对象资源
Dispose()	-	-	释放大对象资源，包括对象句柄

4.2.10 连接池类 XGConnectionPool

功能

连接池类的主要功能是满足大量的短操作的数据库访问行为，包括建立连接、操作数据、关闭连接。

说明

当用户使用连接池时，用户关闭连接，实际是将连接归还连接池。当用户新建连接，实际是从连接池分配连接，极大地提高了创建和关闭连接的效率。

类属性介绍

无

类方法介绍

方法	参数	返回值	说明
SetConnAttribute_IP (stringip_str)	ip_str: 设置连接池 IP 地址	-	设定连接池内的连接 服务端 IP 地址
SetConnAttribute_P ort(intport)	port: 设置连接池端 口号	-	设定连接池内的连接 监听端口
SetConnAttribute_D ataBase(stringdbna me)	dbname: 设置连接 池的连接数据库名	-	设定连接池内的连接 数据库名
SetConnAttribute_U ser(stringuser)	user: 设置连接池的 连接用户名	-	设定连接池内的连接 用户的用户名
SetConnAttribute_P assword(stringpass word)	password: 设置连接 池的连接用户的口令	-	设定连接池内的连接 用户的口令
SetConnAttribute_ch arset(stringcharset)	charset: 设置连接池 的连接字符集	-	设定连接池内的连接 字符集
SetConnAttribute_Mi nPoolconns(intmins)	mins: 设置连接池的 最小连接数	-	设定连接池的最小连 接数
SetConnAttribute_M axPoolconns(intmax s)	maxs: 设置连接池 的最大连接数	-	设定连接池的最大连 接数, 当正在使用的 用户活动连接数等于 此数时, 后续用户使 用连接时需等候

接下一页

方法	参数	返回值	说明
SetConnAttribute_TimeOut(intntime)	ntime: 设置等候超时的时间	-	设定等候超时
GetConnection()	-	XGConnection 对象	从连接池中分配一个连接, 效果等同于单独申明一个连接加 Open 操作
Return_conntoPool()	-	-	归还使用后的连接回连接池
Dispose()	-	-	释放连接池的资源, 释放连接池内的未处于使用状态的连接, 将正在使用的连接移除出连接池变为一般连接进行管理

4.2.11 数据记录类 XGDataRecord

功能

用于提取结果集当前行的数据。

说明

以数据访问类 XGDataReader 为基础的转到当前行的对象的类。

类属性介绍

属性	说明
FieldCount	结果集行的列数

类方法介绍

方法	参数	返回值	说明
IsDBNull(int i)	i: 指定的列序号	为空: true; 不为空: false	指定列序号 i 的值是否为空
GetValue(int i)	i: 指定的列序号	包含指定列序号 i 的 object 对象	取得指定列序号 i 的值
GetValues(object[] values)	values: 获取当前行的所有列值的缓冲区	获取的列数量	依次取得当前行所有列的列值放在一个数组里面
GetBoolean(int i)	i: 指定的列序号	True 或者 false	以 bool 类型取得当前行的第 i 列值
GetByte(int i)	i: 指定的列序号	Byte 类型数据	取字节型数据第 i 列的值
GetBytes(int i, long dataIndex, byte[] buffer, int bufferIndex, int length)	i: 指定的列序号; dataindex: 需要截取的字节流的起点位置; buffer: 获取字节流的缓冲区; bufferIndex: buffer 的长度; Length: 需要截取的字节流长度	截取字节流的长度	以某个点为起点取回长度 length 的第 i 列的字节流的部分值, 此方法可支持分段取数
GetChar(int i)	i: 指定的列序号	包含获取到的数据的字符串	取字符型数据第 i 列的值
接下页			

方法	参数	返回值	说明
GetChars(int i,long dataIndex,char[] buffer,int bufferIndex,int length)	i: 指定的列序号; dataindex: 需要截取的字符串的起点位置; buffer: 获取字符串的缓冲区; bufferIndex: buffer 的长度; Length: 需要截取的字符串长度	截取字符串的长度	以某个点为起点取回长度 length 的第 i 列的字符串的部分值
GetDataTypeName(int i)	i: 指定的列序号	包含数据类型的字符串	取得第 i 列的数据类型名称
GetDateTime(int i)	i: 指定的列序号	DateTime 类型数据	以 datetime 数据类型取回第 i 列数据
GetDecimal(int i)	i: 指定的列序号	Decimal 类型数据	以 C# 的 Decimal 型数据取回第 i 列的值
GetDouble(int i)	i: 指定的列序号	Double 类型数据	以双精度浮点型数据取回第 i 列的值
GetFieldType(int i)	i: 指定的列序号	System.Type.GetType()	取得第 i 列的数据类型
GetFloat(int i)	i: 指定的列序号	Float 类型数据	以浮点型数据取回第 i 列的值
GetInt64(int i)	i: 指定的列序号	Long 类型数据	以 64 位的 Int 长整型数据取回第 i 列的值
			接下页

方法	参数	返回值	说明
GetInt32(int i)	i: 指定的列序号	Int 类型数据	以 Int 型取数第 i 列的值
GetInt16(int i)	i: 指定的列序号	Short 类型数据	以 16 位短整型数据取回第 i 列的值
GetName(int i)	i: 指定的列序号	包含列名的字符串	取得第 i 列的列名
GetOrdinal(string name)	name: 需要查询的列名	列序号	根据列名查询列的序号
GetString(int i)	i: 指定的列序号	包含当前行的第 i 列的数据的字符串	以字符串的形式取回当前行的第 i 列的数据

4.2.12 连接串创建类 XGConnectionStringBuilder

功能

组装连接字符串或通过键值提取连接字符串的相应数据。

类属性介绍

属性	说明
ConnectionString	组装后的连接串信息
Count	连接属性键值对的个数
IsFixedSize	连接串创建器是否有固定的大小
IsReadOnly	连接串创建器是否为只读
Keys	属性键信息
Values	属性键的值信息

类方法介绍

方法	参数	返回值	说明
Add(string keyword ,object value)	keyword: 键值; value: 与键值对应的数值	-	添加键值对
Clear()	-	-	清除所有键值信息
ContainsKey(string keyword))	keyword: 键值	包含: true; 不包含: false	判断是否包含指定的键值信息
EquivalentTo(DbConnectionString-BUILDER connectionStringBuilder)	connectionStringBuilder: 连接串构造类对象	相等: true; 不相等: false	判断是否与指定的连接串构造类对象完全相等
Remove(string keyword)	keyword: 键值	True	从键值信息链里寻找并移除指定键值对信息
ShouldSerialize(string keyword)	keyword: 键值	包含: true; 不包含: false	判断实例中是否存在指定的键信息
TryGetValue(string keyword ,out object value)	keyword: 键值; value: 与键值对应的数值	包含: true; 不包含: false	判断实例中是否存在指定的键值信息

5 示例说明

5.1 不带参数 SQL 的操作

5.1.1 多 IP 创建数据库连接

```
XGConnection t_Conn = new XGConnection();  
t_Conn.ConnectionString = "IPS=127.0.0.1,192.168.2.93; DB=SYSTEM;  
    USER=SYSDBA; PWD=SYSDBA; PORT=5138; AUTO_COMMIT=ON; CHAR_SET=GBK"  
    ;  
t_Conn.Open();  
XGCommand cmd2 = t_Conn.CreateCommand();  
cmd2.CommandText = "select table_name as db_id,db_id as table_name  
    from sys_tables;";
```

5.1.2 创表等初始化 SQL 环境语句

```
XGConnection conn = new XGConnection();  
conn.ConnectionString = conn_xg ;  
Try  
{  
    conn.Open();  
    XGCommand cmd = new XGCommand();  
    cmd.Connection = conn;  
    cmd.CommandText = "select count(*) from user_tables where  
        table_name='TA'";  
    if (Convert.ToInt32(cmd.ExecuteScalar()) == 1)  
    {  
        cmd.CommandText = "drop table TA cascade";  
        cmd.ExecuteNonQuery();  
    }  
    cmd.CommandText = "select count(*) from user_tables where  
        table_name='TP'";  
    if (Convert.ToInt32(cmd.ExecuteScalar()) == 1)  
    {  
        cmd.CommandText = "drop table TP cascade";  
        cmd.ExecuteNonQuery();  
    }  
    cmd.CommandText = "select count(*) from user_sequences where  
        seq_name='SEQ_TP'";  
    if (Convert.ToInt32(cmd.ExecuteScalar()) == 1)  
    {  
        cmd.CommandText = "drop sequence SEQ_TP cascade";  
        cmd.ExecuteNonQuery();  
    }  
    cmd.CommandText = "select count(*) from user_sequences where  
        seq_name='SEQ_TA'";  
    if (Convert.ToInt32(cmd.ExecuteScalar()) == 1)  
    {  
        cmd.CommandText = "drop sequence SEQ_TA cascade";
```

```
        cmd.ExecuteNonQuery();
    }
    string sql_str1 = "create table ta(id number,pid integer,
        p_float float,p_double double,pkey bigint,p_sint smallint,
        p_tint tinyint,name char(100),descri varchar(100),modify_time
        datetime default sysdate,p_numr numeric(4,2),p_clob clob,
        p_bool boolean,p_date date default sysdate,p_time time
        default sysdate,p_blob blob)";
    string sql_str2 = "create table tp(id number,pid integer,pname
        char(100),descri varchar(100),modify_time datetime default
        sysdate)";
    string sql_str3 = "create sequence seq_ta minvalue 1 maxvalue
        999999999 start with 1 increment by 1 cache 20";
    string sql_str4 = "create sequence seq_tp minvalue 1 maxvalue
        999999999 start with 1 increment by 1 cache 20";
    string sql_str5 = "create or replace view v_ap as select ta.id
        as id1,tp.id id2,ta.name,tp.pname,ta.modify_time from tp
        left join ta on tp.pid=ta.pid";
    string sql_str6 = "create or replace trigger trig_identity_ta
        before insert on ta for each row begin if inserting and :new.
        id is null then :new.id := seq_ta.nextval; end if; end
        trig_identity_ta;";
    string sql_str7 = "create or replace trigger trig_identity_tp
        before insert on tp for each row begin if inserting and :new.
        id is null then :new.id := seq_tp.nextval; end if; end
        trig_identity_tp;";

    cmd.CommandText = sql_str1;
    cmd.ExecuteNonQuery();
    cmd.CommandText = sql_str2;
    cmd.ExecuteNonQuery();
    cmd.CommandText = sql_str3;
    cmd.ExecuteNonQuery();
    cmd.CommandText = sql_str4;
    cmd.ExecuteNonQuery();
    cmd.CommandText = sql_str5;
    cmd.ExecuteNonQuery();
    cmd.CommandText = sql_str6;
    cmd.ExecuteNonQuery();
    cmd.CommandText = sql_str7;
    cmd.ExecuteNonQuery();
}
catch (System.Exception ex)
{
    Console.WriteLine(ex.ToString());
    conn.Close();
    Console.WriteLine("测试关闭连接后连接当前状态" + conn.State.
        ToString());
}
}
```

5.1.3 不带参数的数据录入

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg ;
Try
```

```
{
    string insert_sql_ta = "insert into ta(pid,name) values(1,'张三')";
    string insert_sql_tp = "insert into tp(pid,pname,modify_time)
        values(1,'财务部',to_date('2016-04-21 12:12:00','yyyy-mm-dd
            hh:mi:ss'))";

    XGConnection conn = new XGConnection();

    conn.ConnectionString = conn_xg;

    conn.Open();
    XGCommand cmd = new XGCommand();
    cmd.Connection = conn;
    cmd.CommandText = insert_sql_ta;
    cmd.ExecuteNonQuery();

    cmd.CommandText = insert_sql_tp;
    cmd.ExecuteNonQuery();
}
catch (System.Exception ex)
{
    Console.WriteLine(ex.ToString());
    conn.Close();
    Console.WriteLine("测试关闭连接后连接当前状态" + conn.State.
        ToString());
}
```

5.2 带参数 SQL 的操作

5.2.1 带参数 SQL 的数据录入

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg ;
Try
{
    conn.Open();
    XGCommand cmd = new XGCommand();
    cmd.Connection = conn;
    string sql = "insert into tp(pid,pname) values(?,?)";
    cmd.CommandText = sql;
    XGTransaction trans = conn.BeginTransaction();//为连接创建显示
        事务，此时该连接事务处于非自动提交状态
    cmd.Transaction = trans;//指定申明命令事务环境

    //参数赋值
    cmd.Parameters.Add("pid", XGDbType.Int).Value = 2;
    cmd.Parameters.Add("pname", XGDbType.VarChar).Value = "开发部";
    cmd.ExecuteNonQuery();
    //重复执行参数赋值前，需清除前次参数
    cmd.Parameters.Clear();
    cmd.Parameters.Add("pid", XGDbType.Int ).Value = 3;
    cmd.Parameters.Add("pname", XGDbType.VarChar).Value = "测试部";
```



```

cmd.ExecuteNonQuery();

trans.Commit();//提交命令事务
return 1;
}
catch (System.Exception ex)
{
    return 0;
}

```

5.2.2 带参数 SQL 数据录入的其它参数形式

```

XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg ;
Try
{
    conn.Open();
    XGCommand cmd = new XGCommand();
    cmd.Connection = conn;
    string sql = "insert into tp(pid,pname) values(?,?)";
    cmd.CommandText = sql;
    //参数赋值
    XGParameters pidParam = new XGParameters("pid", XGDbType.Int);
    pidParam.Direction = ParameterDirection.Input;
    pidParam.Value = 2;
    cmd.Parameters.Add(pidParam);
    string clstring = "李四";
    XGParameters nammeParam = new XGParameters("name", XGDbType.
        VarChar);
    nammeParam.Direction = ParameterDirection.Input;
    nammeParam.Value = clstring;
    cmd.Parameters.Add(nammeParam);

    cmd.ExecuteNonQuery();

    cmd.Parameters.Clear();
    cmd.Parameters.Add("pid", XGDbType.Int).Value = 2;
    cmd.Parameters.Add("name", XGDbType.VarChar).Value = "王五";
    cmd.ExecuteNonQuery();
    //重复执行参数赋值前,需清除前次参数
    cmd.Parameters.Clear();
    cmd.Parameters.Add("pid", XGDbType.Int).Value = 3;
    cmd.Parameters.Add("name", XGDbType.VarChar).Value = "赵七";
    cmd.ExecuteNonQuery();
    cmd.Parameters.Clear();
    cmd.Parameters.Add("pid", XGDbType.Int).Value = 3;
    cmd.Parameters.Add("name", XGDbType.VarChar).Value = "钱八";
    cmd.ExecuteNonQuery();
    return 1;
}
catch (System.Exception ex)
{
    Console.WriteLine(ex.ToString());
    conn.Close();
    return 0;
}

```

```
}
```

5.2.3 带参数 SQL 语句执行 UPDATE 操作

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg ;
Try
{
    conn.Open();
    XGCommand cmd = new XGCommand();
    cmd.Connection = conn;
    string sql_str = "update tp set descri=?,modify_time=?";
    cmd.CommandText = sql_str;

    cmd.Parameters.Add("descri", XGDbType.VarChar).Value = "测试数
        据变更";
    string strTime = "2017-12-12 12:11:11";
    DateTime dtTime = DateTime.Parse(strTime);
    cmd.Parameters.Add("modify_time", XGDbType.DateTime).Value =
        dtTime;
    int ret = cmd.ExecuteNonQuery();
    return ret;
}
catch (System.Exception ex)
{
    Console.WriteLine(ex.ToString());
    conn.Close();
    return 0;
}
```

5.2.4 带参数 SQL 执行 DELETE 操作

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg ;
Try
{
    conn.Open();
    string sql_str = "delete from ta where pid=3";
    XGCommand cmd = new XGCommand(sql_str, conn);
    cmd.Connection = conn;

    Int32 ret = cmd.ExecuteNonQuery(); //return effect num
    return ret;
}
catch (System.Exception ex)
{
    return 0;
}
```

5.3 一般结果集的获取

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg;

try
{
    conn.Open();
    XGCommand cmd = new XGCommand();
    cmd.Connection = conn;
    string sql_sql = "select * from ta";
    XGDataAdapter da = new XGDataAdapter(sql_sql, conn);
    DataSet ds = new DataSet();
    da.Fill(ds, "ta");
    for (int i = 0; i < ds.Tables[0].Rows.Count;i++ )
    {
        string val = "人员表数据： ";
        for (int j = 0; j < ds.Tables[0].Columns.Count;j++ )
        {
            val += ds.Tables[0].Rows[i][j].ToString()+"|";
        }
    }
    string sql_str2 = "select * from tp";
    XGCommand cmd2 = conn.CreateCommand();// 或者 new DBCommand(
        sql_str, conn);
    cmd.CommandText = sql_str2;
    XGDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        string val = "人员数据： ";
        string id = reader.GetInt32(0).ToString();
        string pid = reader.GetString(1);
        string pname = reader.GetString(2);
        string desc = reader.IsDBNull(3) ? null : reader.GetString
            (3);
        string date = reader.GetDateTime(4).ToString();
        val += id + "|" + pid + "|" + pname + "|" + desc + "|" +
            date + "|";
        Console.WriteLine(val);
    }
}
catch (System.Exception ex)
{
    Console.WriteLine(ex.ToString());
    conn.Close();
    return 0;
}
```

5.4 结果集更新

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg;

try
```

```
{
    conn.Open();
    XGCommand cmd = new XGCommand();
    cmd.Connection = conn;
    string sql_sql = "select * from tp";
    XGDataAdapter adapter = new XGDataAdapter();
    adapter.SelectCommand = new XGCommand(queryString, conn);
    string tableName = "tp";
    DataSet dataSet = new DataSet();
    adapter.Fill(dataSet, tableName);
    foreach (DataRow dr in dataSet.Tables["tp"].Rows)
    {
        if (dr["PNAME"].ToString().Trim().Equals("测试部"))
        {
            dr.Delete(); // 删除DataSet 中的行
            break;
        }
    }
    dataSet.Tables[tableName].Rows[0][1] = 38; // 更新DataSet中第一行
    第2列的值
    dataSet.Tables[tableName].Rows[1][3] = "开发项目";
    string[] dd = new String[5] { "124", "24", "dsf", "dsgrt", "
    2016-12-12 12:11:11" };
    dataSet.Tables[tableName].Rows.Add(dd); // 增加一行 考虑参数的形
    式
    // 增加一行
    adapter.Update(dataSet, tableName);

    return 0;
}
catch (System.Exception ex)
{
    Console.WriteLine(ex.ToString());
    conn.Close();
    return 0;
}
```

5.5 存储过程的执行

5.5.1 存储过程输入参数的执行

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg;
try
{
    conn.Open();
    XGCommand cmd = new XGCommand();
    cmd.Connection = conn;
    cmd.CommandText = "select count(*) from user_tables where
    table_name='T_PROC10'"; // T_pack_func1
    if (Convert.ToInt32(cmd.ExecuteScalar()) == 1)
    {
        cmd.CommandText = "drop table T_PROC10 cascade";
    }
}
```

```
        cmd.ExecuteNonQuery();
    }
    cmd.CommandText = "create table T_PROC10(pid int ,pname varchar
        )";
    cmd.ExecuteNonQuery();
    string sql = "create or replace PROCEDURE P1(pid INT,pname
        VARCHAR )";
    sql += " AS BEGIN ";
    sql += " insert into T_PROC10 values(pid,pname)";
    sql += " end; ";
    cmd.CommandText = sql;
    cmd.ExecuteNonQuery();
    cmd.CommandText = "P1";
    cmd.CommandType = CommandType.StoredProcedure;
    //参数赋值
    cmd.Parameters.Add("pid", XGDbType.Int).Value = 2;
    cmd.Parameters.Add("pname", XGDbType.VarChar).Value = "开发部";
    cmd.ExecuteNonQuery();
    //重复执行参数赋值前,需清除前次参数
    cmd.Parameters.Clear();
    cmd.Parameters.Add("pid", XGDbType.Int).Value = 3;
    cmd.Parameters.Add("pname", XGDbType.VarChar).Value = "测试部";
    cmd.ExecuteNonQuery();
    return 1;
}
catch (System.Exception ex)
{
    Console.WriteLine(ex.ToString());
    conn.Close();
    return 0;
}
```

5.5.2 存储过程输出参数的执行

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg;

try
{
    conn.Open();
    XGCommand cmd = new XGCommand();
    cmd.Connection = conn;
    cmd.CommandText = "select count(*) from user_tables where
        table_name='TEST_CSH_PROC'";
    if (Convert.ToInt32(cmd.ExecuteScalar()) == 1)
    {
        cmd.CommandText = "drop table TEST_CSH_PROC cascade";
        cmd.ExecuteScalar();
    }
    cmd.CommandText = "CREATE TABLE TEST_CSH_PROC(ID INT,SS VARCHAR
        (20),KK DATE,SP NUMERIC(12,5),PP FLOAT,TT FLOAT)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "insert into TEST_CSH_PROC values(1,'testss',
        TO_DATE('2016-1-1 23:45:09','YYYY-MM-DD HH24:MI:SS')
        ,321.34,123.123,321.123)";
```

```

cmd.ExecuteNonQuery();
cmd.CommandText = "insert into TEST_CSH_PROC values(1,'testss',
    TO_DATE('2016-1-1 23:45:09','YYYY-MM-DD HH24:MI:SS')
    ,111.23,231.12,322)";
cmd.ExecuteNonQuery();
cmd.CommandText = "insert into TEST_CSH_PROC values(1,'testss',
    TO_DATE('2016-1-1 23:45:09','YYYY-MM-DD HH24:MI:SS')
    ,234.123,986.234,9322)";
cmd.ExecuteNonQuery();
string sql_str = "CREATE OR REPLACE PROCEDURE P1(IN_ID IN OUT
    INT, OSS OUT VARCHAR, OKK OUT DATE, OSP OUT NUMERIC, OPP OUT
    FLOAT, OTT OUT FLOAT)";
sql_str += "AS TEMP_ID INT; BEGIN";
sql_str += " select count(*), ss, kk, sum(sp), sum(pp), sum(tt) into
    temp_id, oss, okk, osp, opp, ott from TEST_CSH_PROC where id=
    IN_ID group by ss, kk;";
sql_str += " IN_ID:=TEMP_ID; end;";
cmd.CommandText = sql_str;
cmd.ExecuteNonQuery();
cmd.CommandType = CommandType.StoredProcedure;
cmd.CommandText = "P1";
XGParameters par_id = new XGParameters("IN_ID", XGDbType.Int);
par_id.Direction = ParameterDirection.InputOutput;
par_id.Value = 1;
cmd.Parameters.Add(par_id);
XGParameters par_oss = new XGParameters("OSS", XGDbType.VarChar
    , 500);
par_oss.Direction = ParameterDirection.Output;
cmd.Parameters.Add(par_oss);
XGParameters par_okk = new XGParameters("OKK", XGDbType.
    DateTime);
par_okk.Direction = ParameterDirection.Output;
cmd.Parameters.Add(par_okk);
XGParameters par_osp = new XGParameters("OSP", XGDbType.Numeric
    );
par_osp.Direction = ParameterDirection.Output;
cmd.Parameters.Add(par_osp);
XGParameters par_opp = new XGParameters("OPP", XGDbType.Double)
    ;
par_opp.Direction = ParameterDirection.Output;
cmd.Parameters.Add(par_opp);
XGParameters par_ott = new XGParameters("OTT", XGDbType.Double)
    ;
par_ott.Direction = ParameterDirection.Output;
cmd.Parameters.Add(par_ott);
cmd.ExecuteNonQuery();

Console.WriteLine("in_id=" + par_id.Value.ToString());
Console.WriteLine("oss=" + par_oss.Value.ToString());
Console.WriteLine("okk=" + par_okk.Value.ToString());
Console.WriteLine("osp=" + par_osp.Value.ToString());
Console.WriteLine("opp=" + par_opp.Value.ToString());
Console.WriteLine("ott=" + par_ott.Value.ToString());
cmd.Dispose();
conn.Dispose();

```

```
        return 0;
    }
    catch (System.Exception ex)
    {
        Console.WriteLine(ex.ToString());
        conn.Close();
        return 0;
    }
}
```

5.6 存储函数包操作

5.6.1 存储函数包的执行

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg;

try
{
    conn.Open();
    XGCommand cmd = new XGCommand();
    cmd.Connection = conn;
    cmd.CommandText = "select count(*) from user_tables where
        table_name='T_PACK_FUNC1'";//T_pack_func1
    if (Convert.ToInt32(cmd.ExecuteScalar()) == 1)
    {
        cmd.CommandText = "drop table T_PACK_FUNC1 cascade";
        cmd.ExecuteScalar();
    }

    cmd.CommandText = "create table T_pack_func1(c1 int ,c2 double,
        c3 datetime,c4 numeric(32,8),c5 varchar)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = " insert into T_pack_func1 values(1,null
        , '2017-07-10 15:01:35',33493.23423,'this is the func1 values
        1')";
    cmd.ExecuteNonQuery();
    cmd.CommandText = " insert into T_pack_func1 values
        ( 2,23423.23,null,98763.2333,'here is the func1 the No2 value
        and so we need ')" ;
    cmd.ExecuteNonQuery();
    cmd.CommandText = "insert into T_pack_func1 values
        ( 3,972.332,'2017-07-09 19:45:22',null,'so we get the third
        3 value')";
    cmd.ExecuteNonQuery();

    cmd.CommandText = "insert into T_pack_func1 values
        ( 4, 243.2342,'2017-07-04 20:35:18',3454.32,null)";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "insert into T_pack_func1 values
        ( 5,843.23,'2008-09-01 12:25:38',205.23,')";
    cmd.ExecuteNonQuery();
}
```

```

cmd.CommandText = "select count(*) from user_tables where
    table_name='T_PACK_PAN1'";//T_pack_pan1
if (Convert.ToInt32(cmd.ExecuteScalar()) == 1)
{
    cmd.CommandText = "drop table T_PACK_PAN1 cascade";
    cmd.ExecuteScalar();
}
cmd.CommandText = "create table T_pack_pan1(c1 bigint,c2 char
    (50),c3 date,c4 int, c5 float)";
cmd.ExecuteNonQuery();
cmd.CommandText = "insert into T_pack_pan1 values(123,'
    sdishosho hereess ','2017-07-01',null,324.56)";
cmd.ExecuteNonQuery();
cmd.CommandText = " insert into T_pack_pan1 values(456,'some
    get out the sdishosho hereess ',null,234,45.324) ";
cmd.ExecuteNonQuery();
cmd.CommandText = " insert into T_pack_pan1 values(789,null
    ,'2017-07-03',34,23445.23) ";
cmd.ExecuteNonQuery();
cmd.CommandText = "insert into T_pack_pan1 values(678,'the 4
    fourth in ','2017-07-04',44,null)";
cmd.ExecuteNonQuery();

cmd.CommandText = "select count(*) from user_PACKAGES where
    PACK_name='PACK_NAME1'";//PACK_NAME1
if (Convert.ToInt32(cmd.ExecuteScalar()) == 1)
{
    cmd.CommandText = "alter package pack_name1 recompile";
    cmd.ExecuteScalar();
}
Else
{
    string sql_pack_head = "create or replace package
        pack_name1 is ";
    sql_pack_head += " function pa_func1(aa in int,";
    sql_pack_head += " bb in out double,";
    sql_pack_head += " cc out datetime,";
    sql_pack_head += " dd out numeric,";
    sql_pack_head += " ee out varchar)";
    sql_pack_head += " return datetime;";
    sql_pack_head += " procedure proc_pan1";
    sql_pack_head += "(aa in bigint,bb in out char(50), cc out
        date, dd out int, ee out float);";
    sql_pack_head += " end; ";
    cmd.CommandText = sql_pack_head;
    cmd.ExecuteNonQuery();
    string sql_pack_body = "create or replace package body
        pack_name1 is";
    sql_pack_body += " function pa_func1(";
    sql_pack_body += " aa in int, bb in out double, cc out
        datetime, dd out numeric, ee out varchar )";
    sql_pack_body += " return datetime as TMP_DT DATETIME;
        begin";
    sql_pack_body += " select c2 ,c3,c4,c5 into bb,cc,dd,ee
        from T_pack_func1 where c1=aa;";
}

```



```

        sql_pack_body += " TMP_DT:=cc;"; //sql_pack_body += "";
        sql_pack_body += " return TMP_DT;end;";
        sql_pack_body += " procedure proc_pan1(";
        sql_pack_body += " aa in bigint, bb in out char(50)";
        sql_pack_body += " , cc out date, dd out int, ee out float)";
        sql_pack_body += " as begin ";
        sql_pack_body += " select c2 ,c3,c4,c5 into bb,cc,dd,ee";
        sql_pack_body += " from T_pack_pan1 where c1=aa;";
        sql_pack_body += " end; ";
        sql_pack_body += " end; ";
        cmd.CommandText = sql_pack_body;
        cmd.ExecuteNonQuery();
    }
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.CommandText = "PACK_NAME1.PA_FUNC1";
    XGParameters aa = new XGParameters("AA", XGDbType.Int);
    aa.Direction = ParameterDirection.Input;
    aa.Value = 2; // 2 3 4 5
    cmd.Parameters.Add(aa);

    XGParameters bb = new XGParameters("BB", XGDbType.Double);
    bb.Direction = ParameterDirection.InputOutput;
    bb.Value = 1.0;
    cmd.Parameters.Add(bb);

    XGParameters cc = new XGParameters("CC", XGDbType.DateTime);
    cc.Direction = ParameterDirection.Output;
    cmd.Parameters.Add(cc);
    XGParameters dd = new XGParameters("DD", XGDbType.Numeric);
    dd.Direction = ParameterDirection.Output;
    cmd.Parameters.Add(dd);

    XGParameters ee = new XGParameters("ee", XGDbType.VarChar, 200)
        ;
    ee.Direction = ParameterDirection.Output;
    cmd.Parameters.Add(ee);

    XGParameters ff = new XGParameters("ff", XGDbType.DateTime);
    ff.Direction = ParameterDirection.ReturnValue;
    cmd.Parameters.Add(ff);
    cmd.ExecuteNonQuery();

    Console.WriteLine("aa=" + aa.Value.ToString());
    Console.WriteLine("bb=" + bb.Value.ToString());
    Console.WriteLine("cc=" + cc.Value.ToString());
    Console.WriteLine("dd=" + dd.Value.ToString());
    Console.WriteLine("ee=" + ee.Value.ToString());
    Console.WriteLine("ff=" + ff.Value.ToString());
    return 0;
}
catch (System.Exception ex)
{
    Console.WriteLine(ex.ToString());
    conn.Close();
    return 0;
}

```

```
}

```

5.6.2 存储函数带引用型游标输出结果集

SQL 准备:

```
create table ta(id number,pid integer, name char(100),descri
  varchar(100),modify_time datetime default sysdate, p_date date
  default sysdate,p_time time default sysdate ) ;
create table tp(id number,pid integer,pname char(100),descri
  varchar(100),modify_time datetime default sysdate);
insert into tp values(1, 38 , '财务部', '测试数据变更', '
  2017-12-12 12:11:11' );
insert into tp values(2, 1 , '财务部', '开发项目', '
  2017-12-12 12:11:11');
insert into tp values( 3, 2 , '开发部', '测试数据变更', '
  2017-12-12 12:11:11');
insert into tp values(124,24 , 'dsf', 'dsgrt', '2016-12-12 12:11:11
  ');

insert into ta values( 1, 1 , '张三', NULL, '
  2017-08-01 09:34:49.458' , '2017-08-01', '09:34:49.458') ;
insert into ta values( 2, 2 , '李四', NULL, '
  2017-08-01 09:36:51.113' , '2017-08-01', '09:36:51.113') ;
insert into ta values( 3, 2 , '王五', NULL, '
  2017-08-01 09:36:53.427' , '2017-08-01', '09:36:53.427') ;

create or replace function myfunc_outrefur(refcur_out out
  sys_refcursor)
return sys_refcursor is refcur_ret sys_refcursor;
begin
open refcur_ret for select * from ta;
open refcur_out for select * from tp;
return refcur_ret;
end ;
/
```

代码示例:

```
XGConnection conn = new XGConnection();
conn.ConnectionString = conn_xg;
try
{
  conn.Open();
  XGCommand cmd = new XGCommand();
  cmd.Connection = conn;
  cmd.CommandType = CommandType.StoredProcedure;
  cmd.CommandText = "myfunc_outrefur";
  cmd.Parameters.Clear();

  XGParameters pa1= new XGParameters("refcur1",XGDbType.RefCursor
  );
  pa1.Direction= ParameterDirection.Output;
  cmd.Parameters.Add(pa1);
```

```
XGParameters pa2 = new XGParameters("refcur2", XGDbType.  
    RefCursor);  
pa2.Direction = ParameterDirection.ReturnValue;  
cmd.Parameters.Add(pa2);  
ret = cmd.ExecuteNonQuery();  
//output to reader  
XGDataReader reader_out1 = ((XGRefCursor)pa1.Value).  
    GetDataReader();  
Console.WriteLine("reader_out1.getname(2) = " + reader_out1.  
    GetName(2));  
XGDataReader reader_out2 = ((XGRefCursor)pa2.Value).  
    GetDataReader();  
Console.WriteLine("reader_out2.getname(2) = " + reader_out2.  
    GetName(2));  
reader_out1.Close();  
reader_out1.Dispose();  
reader_out2.Close();  
reader_out2.Dispose();  
//  
cmd.Parameters.Clear();  
cmd.Dispose();  
conn.Close();  
conn.Dispose();  
}  
catch (System.Exception ex)  
{  
    Console.WriteLine(ex.ToString());  
    conn.Close();  
    return 0;  
}
```

5.7 大对象应用

代码示例：

```
public static string filepath = "D:\\csharp_data\\data254.mp4";  
//blob clob  
public static int test_blob()  
{  
    XGConnection conn = new XGConnection();  
    conn.ConnectionString = conn_xg;  
    Try  
    {  
        conn.Open();  
        XGCommand cmd = new XGCommand();  
        cmd.Connection = conn;  
        cmd.CommandText = "SELECT COUNT(*) FROM USER_TABLES WHERE  
            TABLE_NAME='TEST_BLOB'";  
        if (Convert.ToInt32(cmd.ExecuteScalar()) == 0)  
        {  
            cmd.CommandText = "CREATE TABLE TEST_BLOB(ID INT,SS  
                BLOB)";  
            cmd.ExecuteNonQuery();  
        }  
    }  
}
```

```
else
{
    cmd.CommandText = "DROP TABLE TEST_BLOB";
    cmd.ExecuteNonQuery();
    cmd.CommandText = "CREATE TABLE TEST_BLOB(ID INT,SS
        BLOB) ";
    cmd.ExecuteNonQuery();
}
cmd.CommandText = "INSERT INTO TEST_BLOB VALUES(1,NULL)";
cmd.ExecuteNonQuery();
cmd.CommandText = "UPDATE TEST_BLOB SET SS=? WHERE ID=1";
XGBlob xb = new XGBlob();
FileAccess acces = FileAccess.Read;
FileMode filemode = FileMode.Open;
long big_count = 0;
using (FileStream fs = new FileStream(filepath, filemode,
    acces))
{
    byte[] b = new byte[64768];
    Int64 amountRead = 0;
    amountRead = fs.Read(b, 0, b.Length);
    xb.BeginChunkWrite();
    Int64 have_r = amountRead;
    while (amountRead > 0)
    {
        big_count += amountRead; // count
        have_r = have_r + b.Length;
        xb.write(b, 0, (Int32)amountRead);
        amountRead = fs.Read(b, 0, b.Length);
    }
    xb.EndChunkWrite();
}
cmd.Parameters.Add("?", XGDbType.LongVarBinary, xb,
    ParameterDirection.Input);
cmd.ExecuteNonQuery();
xb.Close();
}
catch (Exception ei)
{
    Console.WriteLine(ei.ToString());
}
finally
{
    conn.Close();
    Console.WriteLine("测试 关闭连接后 连接当前状态" + conn.
        State.ToString());
}
return 0;
}
```

6 异常处理

6.1 XGConnection Error

6.1.1 System.InvalidOperationException

错误描述

System.InvalidOperationException: 连接尚未处于活动状态。

错误原因

- 没有可用的活动连接。
- 连接对象未调用.Open() 方法将连接状态置为活动状态。

分析与建议

- 在初始化连接串后，应执行.Open() 操作打开连接，然后再使用。
- 建议排查代码保证预先调用了.Open() 方法。

6.1.2 System.InvalidOperationException

错误描述

System.InvalidOperationException: 连接已经是本地事务处理或分布式事务处理的一部分。

错误原因

当前连接已经执行过一次.BeginTransaction() 加入了某个事务环境。

分析与建议

排查代码，如果当前连接已经执行过一次.BeginTransaction() 加入了事务，那么不应重复加入事务环境，如需开启新事务，那么提交或者回滚当前事务即可。

6.1.3 USE DATABASE ” + databaseName + ” Failure

错误描述

USE DATABASE ” + databaseName + ” Failure。

错误原因

切换数据库失败，连接不可用或者用户没有相应权限。

分析与建议

查看连接是否处于活动状态，检查当前连接的用户有无切换到目标数据库的权限。

6.1.4 System.ConnectionIsOpenedException

错误描述

System.ConnectionIsOpenedException: 连接已经打开。

错误原因

打开已经打开的连接报错。

分析与建议

当前连接已经调用过一个.Open() 操作，目前连接已经处于活动状态了，不可以重复打开。

6.1.5 System.ActionConnectionException

错误描述

System.ActionConnectionException: 来自连接池的连接不允许调用 Open 方法。

错误原因

打开连接池的连接另有方法。

分析与建议

在调用 GetConnection() 时，从连接池来的连接对象已经被打开了，不需要 Open 进行重复打开。

6.1.6 System.ConnectionStrException

错误描述

System.ConnectionStrException: 连接串不允许为空。

错误原因

连接对象初始化未对连接串属性赋值。

分析与建议

需要在执行.Open() 方法之前对连接串属性进行赋值, 其中包括 IP、端口、数据库、用户名、密码、字符集等等。

6.1.7 CLOSE CURSOR Failure

错误描述

CLOSE CURSOR” + ser_cursorname + ” Failure。

错误原因

关闭游标失败。

分析与建议

需要排查连接是否已经处于不活动状态，确认游标未曾被关闭。

6.2 XGCommand Error

6.2.1 System.InvalidOperationException

错误描述

System.InvalidOperationException: 指定的连接无效或者尚未打开。

错误原因

没有可用的连接。

分析与建议

Command 对象在执行 SQL 操作前应与活动的连接进行绑定，没有绑定的会报错，需排查。

6.2.2 System.commandtextException

错误描述

System.commandtextException:CommandText 不能为空。

错误原因

命令不能为空。

分析与建议

执行 ExecuteNonQuery() 之前应对 CommandText 赋值，请进行排查。

6.2.3 System.CommandPrepareException

错误描述

System.CommandPrepareException:sqltext err:

错误原因

prepare 出错。

分析与建议

prepare 语句执行错误，请参照内部返回的错误码进行排查。

6.2.4 System.CommandExecuteReaderException

错误描述

System.CommandExecuteReaderException:here is a reader already Opened.

错误原因

已经有了一个打开的 reader。

分析与建议

当前 CMD 对象已经打开一个 reader 对象了，应关闭后再打开新的 reader。

6.2.5 System.CommandExecuteException

6.2.5.1 sqlexecute err:%s

错误描述

System.CommandExecuteException:sqlexecute err:%s。

错误原因

SQL 语句执行出错。

分析与建议

SQL 语句执行错误，请参照内部返回的错误码进行排查。

6.2.5.2 only select cmd can use server_cursor

错误描述

System.CommandExecuteException:only select cmd can use server_cursor。

错误原因

只有查询语句才能生成 reader。

分析与建议

CMD 对象调用.ExecuteReader 方法时没有使用 select 查询语句，请检查 SQL 语句。

6.2.5.3 ParameterDirection Error

错误描述

System.CommandExecuteException:ParameterDirection Error。

错误原因

参数输入输出类型绑定出错。

分析与建议

游标型参数只能为输出型，或者输入输出型，请验证。

6.2.5.4 Parameter Un-Expected datatype Error

错误描述

System.CommandExecuteException:Parameter Un-Expected datatype Error。

错误原因

输入的参数类型尚未支持。

分析与建议

参数的数据类型应在许可的范围之内，不应超出，用户可根据需要进行转化后再进行绑定，通用型最高的字符串类型。

6.2.5.5 BindParamsByPos RefCursor not output or return error

错误描述

System.CommandExecuteException:BindParamsByPos RefCursor not output or return error。

错误原因

输出型游标的参数输入输出类型异常。

分析与建议

游标对象一般仅支持输出型参数，请检测其输入输出类型。

6.2.5.6 BindParamsByPos error

错误描述

System.CommandExecuteException:BindParamsByPos error。

错误原因

输入的参数类型尚未支持。

分析与建议

按序号绑定的参数类型不在支持的目标范围，请转换后再绑定。

6.2.5.7 BindParamsByName RefCursor not output nor return error

错误描述

System.CommandExecuteException:BindParamsByName RefCursor not output nor return error。

错误原因

输出型游标的参数输入输出类型异常。

分析与建议

游标对象一般仅支持输出型参数，请检测其输入输出类型。

6.2.5.8 BindParamsByName bindtype error

错误描述

System.CommandExecuteException:BindParamsByName bindtype error。

错误原因

输入的参数类型尚未支持。

分析与建议

按参数名绑定的参数类型不在支持的目标范围，请转换后再绑定。

6.2.5.9 BindParamsArrayByPos inputoutput error

错误描述

System.CommandExecuteException:BindParamsArrayByPos inputoutput error: ” + ”

BindParamsArray do not support output。

错误原因

批量参数不支持输入型以外的类型。

分析与建议

批量插入方式仅支持输入型参数不支持输出型参数，请根据需求选择参数输入输出类型。

6.2.5.10 BindParamsArrayByPos bindtype error

错误描述

System.CommandExecuteException:BindParamsArrayByPos bindtype error。

错误原因

批量插入不支持大对象类型。

分析与建议

批量数据插入不支持含 Clob、Blob 等大对象字段的表结构，如有需要请转换类型后插入，或采用单条插入方式。

6.3 XGConnectPool Error

6.3.1 System.PoolNumOutOfRangeException

错误描述

- System.PoolNumOutOfRangeException: 连接池最小连接数非法。
- System.PoolNumOutOfRangeException: 连接池最大连接数非法。

错误原因

- 最小连接数属性赋值不合理。
- 最大连接数属性赋值不合理。

分析与建议

- 最小连接数应处于 1-998 之间，不应超出此范围。
- 最大连接数应处于 1-999 之间，不应超出此范围。

6.3.2 System.ConnAttributeLessException

错误描述

System.ConnAttributeLessException: 连接参数缺少。

错误原因

建立连接的参数不足。

分析与建议

建立连接池的属性 IP、数据库名、用户名、密码都应正确提供，请验证。

6.3.3 System.WaitConnTimeOutException

错误描述

System.WaitConnTimeOutException: 连接分配等候超时。

错误原因

连接池的连接数量不足或使用后未归还，导致新连接分配失败。

分析与建议

提供足够使用的连接数，检查分配出去的连接已经正常归还。

6.4 XGDataAdapter Error

6.4.1 System.InvalidUpdateException

错误描述

System.InvalidUpdateException: 多表连接查询或基于统计的查询不允许数据更改。

错误原因

多表连接查询不允许使用 update 反馈至数据库端。

分析与建议

单表查询时可以 update 到数据库的表数据内，多表连接查询的结果 update 至服务器端不支持，建议使用单独的 SQL 语句对各个表进行 update。

6.5 XGDataReader Error

6.5.1 System.WithOutResultException

错误描述

System.WithOutResultException: 没有适当的结果集返回。

错误原因

结果集数量为 0。

分析与建议

查看生产 reader 的 SQL 是否正常生成结果集。

6.5.2 System.InvalidOperationException

错误描述

System.InvalidOperationException: 对象的当前状态使该操作无效。

错误原因

对象已经关闭或处于非打开状态。

分析与建议

结果集未处于正常的打开状态，结果集相关属性不能正常的反馈，请排查 reader 的状态。

6.5.3 System.IndexOutOfRangeException

错误描述

- System.IndexOutOfRangeException: 指定的列序号无效。

- System.IndexOutOfRangeException: 指定的数据偏移量无效。

错误原因

- 列序号输入错误。
- 数据偏移量超界。

分析与建议

- 列序号应根据生成的结果集的列数进行填充，其值从 0 开始，不应超过结果集的最大列数。
- Bytes 类型的数据有其长度，虽然可以分段获取，但是不应超过其最大长度。

6.5.4 this is Get Decimal

错误描述

this is Get Decimal: 数据超过 System.Decimal 最大表示范围请使用 GetString 取数。

错误原因

值转化错误。

分析与建议

当数值长度的精度超过其数据类型表示范围后，应采用字符串获取值，避免范围超界。

6.5.5 无法将类型'System.DBNull'转换为 bool

错误描述

无法将类型'System.DBNull'转换为 bool。

错误原因

值转化错误。

分析与建议

Bool 类型值只能存放 true 或者 false 不能存放空值，应该在取值前先判断数据是否为空。

6.5.6 无法将类型'System.DBNull'转换为 byte

错误描述

无法将类型'System.DBNull'转换为 byte。

错误原因

值转化错误。

分析与建议

byte 类型值不能存放空值，应该在取值前先判断数据是否为空。

6.5.7 无法将类型'System.DBNull' 转换为 DateTime

错误描述

无法将类型' System.DBNull' 转换为 DateTime。

错误原因

值转化错误。

分析与建议

DateTime 类型值不能存放空值，应该在取值前先判断数据是否为空。

6.5.8 无法将类型'System.DBNull' 转换为 decimal

错误描述

无法将类型' System.DBNull' 转换为 decimal。

错误原因

值转化错误。

分析与建议

decimal 类型值不能存放空值，应该在取值前先判断数据是否为空。

6.5.9 无法将类型'System.DBNull' 转换为 double

错误描述

无法将类型' System.DBNull' 转换为 double。

错误原因

值转化错误。

分析与建议

double 类型值不能存放空值，应该在取值前先判断数据是否为空。

6.5.10 无法将类型'System.DBNull' 转换为 float

错误描述

无法将类型' System.DBNull' 转换为 float。

错误原因

值转化错误。

分析与建议

float 类型值不能存放空值，应该在取值前先判断数据是否为空。

6.5.11 无法将类型'System.DBNull' 转换为 short

错误描述

无法将类型' System.DBNull' 转换为 short。

错误原因

值转化错误。

分析与建议

short 类型值不能存放空值，应该在取值前先判断数据是否为空。

6.6 XGParameters Error

6.6.1 ArgumentNullException

6.6.1.1 the input parameter is null

错误描述

ArgumentNullException:the input parameter is null。

错误原因

输入参数为 null。

分析与建议

应填入初始化后的 parameter 对象。

6.6.1.2 the input parameter is null, the param NO

错误描述

ArgumentNullException:the input parameter is null,the param NO。

错误原因

参数数组中某个参数为 null。

分析与建议

排查一次性绑定的参数数组，查看其中某个元素是否为空。

6.6.2 System.InvalidCastException

错误描述

System.InvalidCastException:the supplied obj cannot be cast to a XGParameters object。

错误原因

参数类型转化失败。

分析与建议

传入的参数不能转化为 XGParameters 对象，建议克隆可用对象后编辑使用。

7 常见问题及解答

7.1 关于自动提交的问题

C# 专用接口的连接字符串中的自动提交（auto commit），其默认配置为 true，即命令发往服务器后立刻提交。

当自动提交参数设置为 false 时，命令发往服务器后，需要用户显式地发送“commit”命令到服务器，之前的操作才会被提交，否则被回滚。但有的时候，没有发送 commit 命令到服务器，但还是提交了 SQL 语句，这是为什么呢？

解答

有一些 SQL 语句，比如 Create Table .Truncate Table 等 DDL 语句涉及数据库资源的分配回收的问题，这类语句自动带有 commit 的功能，此语句之前发送的 SQL 语句会在本条语句执行之前被提交。所以，这类语句会隐式的提交之前的事务，如果涉及用户主动回滚则需要注意这类 SQL 语句。

7.2 关于存储过程及函数的执行的问题

一般情况下执行 SQL 命令，直接发送 SQL 语句即可，在别的数据库执行存储过程有种调用方式是直接发送存储过程名和参数（有参数的情况），而虚谷数据库的存储过程调用不是直接输入存储过程名而是需要键入 execute+ 存储过程名。发送 SQL 命令多一个 execute 关键字。

7.3 字符集编码的问题

某些时候，数据库字符集和客户端字符集会出现不相同的情况，此时涉及转码。用户可能发现插入数据库的是一个值，而查询返回是另一个值（可能为乱码）。

解答

有一种办法是更改两端字符集的一端，使二者字符集一致，就不涉及转码问题。如果做不到，必须要转码，则要保证转码的一致性，即入库的时候转码了，那么出库的时候也需要转码，入库的时候没有转码，出库的时候也不要转码。

7.4 执行 SQL 语句报错

7.4.1 SQL 语句类型与执行函数不匹配

错误定位

错误返回 System.CommandExecuteException:sqlexecute err:-1。

解决方法

不同的 SQL 类型（SELECT、UPDATE、DELETE、INSERT）对应不同的执行函数，对应规则请参见章节4.2.2命令类 XGCommand 的类方法介绍。

7.4.2 SQL 语句的参数绑定错误

错误定位

报错 System.CommandExecuteException:ParameterDirection Error。

解决方法

游标参数只能设置为输入型参数或者输出型参数，请检查是否设置正确。

7.4.3 获取结果集失败

错误定位

报错 System.CommandExecuteException:only select cmd can use server_cursor。

解决方法

只有查询语句才能生成 reader，请检查 SQL 语句是否为 SELECT 类型。

7.4.4 数据类型转换错误

错误定位

报错无法将类型“System.DBNull”转换为***。

解决方法

目标类型值不能存放空值，应该在取值前先判断数据是否为空，或者设置该值可以为空（Nullable 属性为 true）。

7.4.5 连接尚未处于活动状态

错误定位

报错 System.InvalidOperationException: 连接尚未处于活动状态。

解决方法

对象操作前应有一个活动连接与其绑定，建议将活动连接与对象显式绑定。

7.5 常见错误定位

7.5.1 引用驱动失败

错误定位

检查自己的工程项目 BIN 目录下是否存在最新的 XGCSQL.dll 文件和 XuguClient.dll 文件。

解决方法

若不存在 XuguClient.dll 文件，请在工程中添加引用；若不存在 XGCSQL.dll 文件，请手动复制到该目录下。

7.5.2 数据库连接失败

7.5.2.1 网络不通

错误定位

通过本机 PING 虚谷数据库服务器的 IP 地址。

解决方法

修改本机或者虚谷数据库 IP 地址，确保两个 IP 地址在同一网段或者中间有 VPN 连接。

7.5.2.2 连接参数错误

错误定位

检查连接字符串是否与虚谷数据库相应参数（IP 地址、端口号、数据库名、用户名、用户密码等）匹配。

解决方法

修改代码中的连接字符串，确保其中的 IP 地址、端口、数据库名、用户名和用户密码正确。

7.5.2.3 用户密码连续错误三次

错误定位

判断是否存在连续三次输入用户名密码错误的情况。

解决方法

等待 IP 冻结时间（3 分钟）结束，然后输入正确的用户名密码。



成都虚谷伟业科技有限公司

联系电话：400-8886236

官方网站：www.xugudb.com