GBASE

GBase Python

接口开发手册

www.gbase.cn

GBase Python 接口开发手册，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

1 GBase Python 概述 2
2 GBase Python 版本 2
3 GBase Python 架构 2
4 使用 GBase Python 4
4.1 创建数据库连接 4
4.1.1 连接参数 4
4.1.2 打开和关闭数据库连接 5
4.2 使用游标操作数据库 5
4.2.1 执行 SQL 语句 6
4.2.2 获取结果集 7
4.2.3 获取多个结果集数据 10
4.2.4 获取下一个结果集 11
4.2.5 执行存储过程 12
4.2.6 执行事务 14
4.2.7 插入二进制数据 15
4.2.8 批量插入数据 16
5 GBase Python 常见问题 17
5.1 SQL 语句执行后，长时间没返回的处理 17
6 GBase Python 客户端类 17
6.1 连接类 18
6.1.1 属性 18
6.1.2 方法 19
6.2 游标类 24
6.2.1 属性 24
6.2.2 方法 25
6.3 异常类 32

1 GBase Python 概述

GBase Python 接口是 Python 语言连接并使用 GBase 数据库的接口驱动程序。GBase Python 接口基于 Python Database API Specification 2.0 标准编写。接口兼容标准的同时并支持如下特性:

- 完全支持 GBase 8a 及 8a 集群的特性
- 完全支持 SQL 标准语法
- 支持二进制流插入、更新
- 支持批量插入优化
- 支持多 SQL 语句执行和获取多结果集
- 支持 TCP/IP 协议
- 支持 Python 的 datetime 和 GBase 时间类型的映射。

2 GBase Python 版本

表格 2-1 版本支持

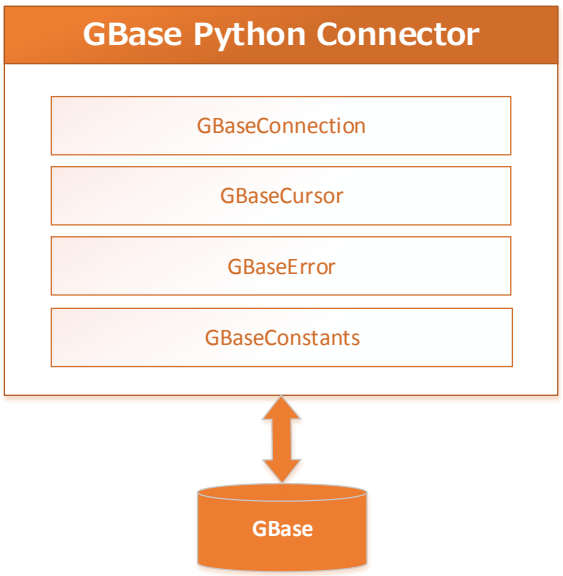
接口版本	支持的产品	支持的 Python 语言	版本兼容性
1.0	GBase 8a 单机 GBase 8a 集群	2.7, 2.6	↑

3 GBase Python 架构

GBase Python 接口提供了统一的客户端访问数据库、获取数据、管理数据的方式，使用如下介绍的核心类完成所有数据库操作。

- 与服务器建立连接，完成握手、初始化
- 执行 SQL 语句、访问存储过程、使用事务
- 对结果集的快速读取
- 快速获取下一个结果集
- 输出定位准确的日志内容

下图展示了 GBase Python 的整体结构。



图表 3-1 GBase Python 结构

- GBaseConnection ：GBase 数据库的连接。
- GBaseCursor ：执行 GBase 数据库操作的游标类，可以执行 SQL 语句、存储过程、获取结果集。
- GBaseError ：异常处理类，定义接口抛出的异常。
- GBaseConstants ：常量类，定义客户端标记、字符集等。

4 使用 GBase Python

本章节主要介绍如何使用 GBase Python 接口创建数据库连接和操作数据库。使用接口类前需要使用 python 语法的 from...import... 命令在 python 文件头部引用接口类。

```
from GBaseConnector import connect,GBaseError
```

4.1 创建数据库连接

创建数据库连接，并通过指定连接参数对连接对象进行初始化。

4.1.1 连接参数

创建连接前需要指定连接参数，参数列表将指明服务器地址、端口号、数据库等相关参数。连接参数定义时参考如下格式：

```
config = {'host':'172.16.3.10','port':5258,'database':'test',
          'user':'gbase','passwd':'gbase20110531'}
```

下面的表格针对每个连接参数给出说明和定义以及约束。

表格 4-1 GBase Python 参数表

参数名	参数类型	参数含义	默认值
基本参数			
host	string	主机地址。GBase 服务器地址。	127.0.0.1
user	string	用户名。登陆数据库的用户名。	
password(passwd)	string	密码。登陆数据库的密码。	
database(db)	string	数据库名称。连接后默认的数据库。	
port	int	端口号。连接数据库时使用的	5258

		端口号。	
charset	string	连接数据库使用的字符集。取值 utf8/gbk	utf8
use_unicode	bool	是否使用 unicode 字符。	True
connect_timeout	int	连接超时时间。创建连接时的超时时间。	30
autocommit	bool	是否使用自动提交模式。	True

4.1.2 打开和关闭数据库连接

操作数据库前需要建立与数据库的连接，并在不使用此连接后需要关闭连接并释放资源，如下所示。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
                'passwd':'111111',
                'port':5258}
    try:
        conn = connect()
        conn.connect(**config)
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

4.2 使用游标操作数据库

使用游标操作数据库资源，包括：执行 SQL 语句、调用存储过程、执行事务、获取结果集操作。

4.2.1 执行 SQL 语句

连接创建后，可以使用连接的 cursor 方法生成游标，然后可以使用游标执行 SQL 语句和获取结果集。

4.2.1.1 执行单条语句

使用游标执行单条 SQL 语句。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
                'passwd':'111111',
                'port':5258}
    try:
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("CREATE DATABASE test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

4.2.1.2 执行多条语句

使用游标执行多条 SQL 语句。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
```

```
            'db':'test'}
    try:
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("CREATE TABLE IF NOT EXISTS test(id INT, name VARCHAR(20))")
        cur.execute("INSERT INTO test VALUES(%s,%s)", (1,'hello'))

        iters = cur.execute("UPDATE test SET id=%s,name=%s;DELETE FROM test;SELECT
'result';",
                            params=(2,'world'), multi_stmt=True)
        for i in iters:
            try:
                print i.fetchall()
            except GBaseError.InterfaceError:
                pass
        cur.execute("DROP TABLE IF EXISTS test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

4.2.2获取结果集

当使用游标执行返回结果集的 SQL 语句后，可以使用游标的 fetchone 方法获取结果集中的一行数据、fetchmany 方法获取结果集中的多行数据、fetchall 方法获取结果集中的所有数据。

4.2.2.1获取一行数据

使用游标的 fetchone 方法获取结果集中的一行数据。fetchone 接口可以循环调用，直到结果集中的数据获取完毕。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
```

```
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("DROP TABLE IF EXISTS test")
        cur.execute("CREATE TABLE IF NOT EXISTS test(id INT, name VARCHAR(50))")
        cur.execute("INSERT INTO test VALUES(1,'hello'), (2,'world')")
        cur.execute("SELECT * FROM test")
        row = cur.fetchone()
        while row is not None:
            print row
            row = cur.fetchone()
        cur.execute("DROP TABLE IF EXISTS test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

4.2.2.2获取多行数据

使用游标的 fetchmany 方法获取结果集中的多行数据。fetchmany 接口可以循环调用，直到结果集中的数据获取完毕。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
        conn = connect()
```

```
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("DROP TABLE IF EXISTS test")
        cur.execute("CREATE TABLE test (COL1 INT, COL2 VARCHAR(20))")
        opfmt = "INSERT INTO test(COL1, COL2) VALUES(%s, %s)"
        rows = []
        for i in xrange(0, 100):
            rows.append((i, "row" + str(i)))
        cur.executemany(opfmt, rows)
        cur.execute("SELECT * FROM test")

        row = cur.fetchmany(3)
        while row:
            print row
            row = cur.fetchmany(4)

        cur.execute("DROP TABLE IF EXISTS test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

4.2.2.3获取所有行数据

使用游标的 fetchall 方法获取结果集中的所有行数据。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
```

```
        cur.execute("DROP TABLE IF EXISTS test")
        cur.execute("CREATE TABLE test (COL1 INT, COL2 VARCHAR(20))")
        opfmt = "INSERT INTO test(COL1, COL2) VALUES(%s, %s)"
        rows = []
        for i in xrange(0, 100):
            rows.append((i, "row" + str(i)))
        cur.executemany(opfmt, rows)
        cur.execute("SELECT * FROM test")

        print cur.fetchall()

        cur.execute("DROP TABLE IF EXISTS test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

4.2.3获取多个结果集数据

当有多个结果集时，可使用如下方法获取多个结果集的数据。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("DROP TABLE IF EXISTS test")
        cur.execute("CREATE TABLE test (COL1 INT, COL2 VARCHAR(20))")
        opfmt = "INSERT INTO test(COL1, COL2) VALUES(%s, %s)"
        rows = []
```

```
        for i in xrange(0, 100):
            rows.append((i, "row" + str(i)))
        cur.executemany(opfmt, rows)
        iters = cur.execute("SELECT * FROM test limit 5;SELECT * FROM TEST LIMIT
10, 5", multi_stmt= True)
        for ter in iters:
            print ter.fetchall()

        cur.execute("DROP TABLE IF EXISTS test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

4.2.4获取下一个结果集

当有多个结果集返回时，可以使用游标的 nextset 方法跳过结果集后直接获取下一个结果集的数据。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("DROP TABLE IF EXISTS test")
        cur.execute("CREATE TABLE test (COL1 INT, COL2 VARCHAR(20))")
        opfmt = "INSERT INTO test(COL1, COL2) VALUES(%s, %s)"
        rows = []
        for i in xrange(0, 100):
            rows.append((i, "row" + str(i)))
        cur.executemany(opfmt, rows)
```

```
        cur.execute("SELECT * FROM test limit 5;SELECT * FROM TEST LIMIT 10, 5;SELECT
* FROM TEST LIMIT 15, 5", multi_stmt= True)
        cur.nextset(2)
        print cur.fetchall()

        cur.execute("DROP TABLE IF EXISTS test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

注意：获取下一个结果集方法 nextset 不能在遍历过程中使用，如下面的例子。

```
iters = cur.execute("SELECT * FROM test limit 5;SELECT * FROM TEST LIMIT 10, 5;SELECT
* FROM TEST LIMIT 15, 5", multi_stmt= True)
        for ter in iters:
            cur.nextset()
            print cur.fetchall()
```

4.2.5 执行存储过程

使用游标的 callproc 方法可以调用 GBase 数据库的存储过程。存储过程返回的结果集通过 4.2.2 小节中描述的方法获取。

如果存储过程包含输出参数，则可以通过游标的 save_param_val 方法获取输出参数值。

1） 下面的样例代码使用 callproc 方法调用存储过程，并获取存储过程结果集。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
```

```
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("DROP PROCEDURE IF EXISTS TESTCALLPROC2")
        cur.execute("CREATE PROCEDURE TESTCALLPROC2() \
                BEGIN \
                select 3; \
                select 4; \
                END")
        res = cur.callproc("TESTCALLPROC2")
        for r in res:
            print r.fetchall()
        cur.execute("DROP PROCEDURE IF EXISTS TESTCALLPROC2")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

2) 下面的样例代码使用 callproc 方法调用存储过程，并通过 save_param_val 方法获取存储过程返回值。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("DROP PROCEDURE IF EXISTS TESTCALLPROC1")
        cur.execute("CREATE PROCEDURE TESTCALLPROC1(\
                    IN pFac1 INT, IN pFac2 INT, OUT pProd INT) \
                    BEGIN \
                    SET pProd := pFac1 * pFac2; \
                    select 1; \
```

```
                    select 2; \
                    END")
        res = cur.callproc("TESTCALLPROC1", (3,4,0))
        for r in res:
            print r.fetchall()
        print cur.save_param_val()
        cur.execute("DROP PROCEDURE IF EXISTS TESTCALLPROC2")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

4.2.6执行事务

通过设置连接参数中的 autocommit=False 参数，即可控制事务，可通过连接的 commit 方法提交事务、rollback 方法回滚事务。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test',
              'autocommit':False,}
    try:
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("DROP PROCEDURE IF EXISTS test")
        cur.execute("CREATE TABLE IF NOT EXISTS test(id INT, val VARCHAR(20))")
        cur.execute("INSERT INTO test VALUES(1,'hello')")
        conn.rollback()
        cur.execute("INSERT INTO test VALUES(2,'world')")
        conn.commit()
        cur.execute("SELECT * FROM test")
        print cur.fetchall()
```

```
        cur.execute("DROP TABLE IF EXISTS test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

或者使用 cur.execute("set autocommit=false")来开启一个事物。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
        conn = connect()
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("DROP PROCEDURE IF EXISTS test")
        cur.execute("CREATE TABLE IF NOT EXISTS test(id INT, val VARCHAR(20))")
        cur.execute("set autocommit=false")
        cur.execute("INSERT INTO test VALUES(1,'hello')")
        conn.rollback()
        cur.execute("INSERT INTO test VALUES(2,'world')")
        conn.commit()
        cur.execute("SELECT * FROM test")
        print cur.fetchall()
        cur.execute("DROP TABLE IF EXISTS test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

4.2.7 插入二进制数据

使用游标的 execute 可以插入二进制数据内容。请参考下面的样例。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
        conn = connect()
        conn.connect(**config)
        fobj = open("c:\\1.gif",'rb')
        fbytes = fobj.read()
        fobj.close()
        cur = conn.cursor()
        cur.execute("CREATE TABLE IF NOT EXISTS test(f blob)")
        cur.execute("INSERT INTO test VALUES(_binary%s)",(fbytes,))
        cur.execute("DROP TABLE IF EXISTS test")
        cur.close()
    except (GBaseError.DatabaseError, IOError), err:
        print err
    finally:
        conn.close()
```

4.2.8批量插入数据

当使用 insert 语句插入数据时，通过调用游标的 executemany 方法可以实现数据批量插入。参见如下样例代码。

```
from GBaseConnector import connect, GBaseError
if __name__ == '__main__':
    config = {'host':'172.16.0.131',
              'user':'root',
              'passwd':'111111',
              'port':5258,
              'db':'test'}
    try:
        conn = connect()
```

```
        conn.connect(**config)
        cur = conn.cursor()
        cur.execute("DROP TABLE IF EXISTS test")
        cur.execute("CREATE TABLE test (COL1 INT, COL2 VARCHAR(20))")
        opfmt = "INSERT INTO test(COL1, COL2) VALUES(%s, %s)"
        rows = []
        for i in xrange(0, 100):
            rows.append((i, "row" + str(i)))
        cur.executemany(opfmt, rows)
        cur.execute("DROP TABLE IF EXISTS test")
    except GBaseError.DatabaseError, err:
        print err
    finally:
        conn.close()
```

5 GBase Python 常见问题

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

5.1 SQL 语句执行后，长时间没返回的处理

客户端应用使用接口时如遇 SQL 语句过于复杂，导致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），客户端会抛出超时异常。若避免此情况的发生可在连接参数中指定 connection_timeout 参数为一个较大的值。

6 GBase Python 客户端类

接口驱动基于 Python Database API 2.0 标准，并实现了标准中定义的类、属性、方法。请看如下表 6-1 介绍。

表格 6-1 GBase Python 客户端类

类	描 述
GBaseConnection	连接类。负责接口和 GBase 数据库端之间的命令传输和数据接收。
GBaseCursor	游标类。游标类对外提供了操作 GBase 数据库主要接口。包括执行 SQL 查询，获取结果集，执行存储过程，获取存储过程结果集功能。
GBaseError	异常类，定义了 GBase 数据库和 Python 标准异常之间的映射关系。

6.1 连接类

此类可以打开和管理一个到 GBase Server 的连接。通常用于发送 SQL 命令和读取结果。

6.1.1 属性

6.1.1.1 user

此属性用来获取连接参数中定义的用户名。

类型	值	默认值	可读写
string	非空字符串	“	read

6.1.1.2 host

此属性用来获取连接参数中定义的主机地址。

类型	值	默认值	可读写
string	有效的 IPV4 地址	127.0.0.1	read

6.1.1.3 port

此属性用来获取连接参数中定义的主机端口号。

类型	值	默认值	可读写
int	有效的端口号	5258	read

6.1.1.4 database

此属性用来设置获取连接参数中定义的数据库。

类型	值	默认值	可读写
string	非空字符串	“	read/write

6.1.1.5 autocommit

此属性用来设置获取自动提交模式，此模式在支持事务的产品中有效。为 True 时，自动提交模式开启，执行任何 DML 语句时数据的更改立刻生效。为 False 时，自动提交模式关闭，受影响数据不会立刻生效。

类型	值	默认值	可读写
bool	True / False	True	read/write

6.1.2 方法

6.1.2.1 GBaseConnection

GBaseConnection 类构造函数。

函数名称	返回值	返回值类型	函数参数	参数含义
GBaseConnectio	instanc	GBaseConnect	**kwargs	连接参数，以字

n	e	ion	(dict)	典方式传入构造函数。如下示例: config = { 'host': '172.16.31.10', 'user': 'gbase', 'port': 5258, 'database': 'test' }

样例代码:

```
code
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = GBaseConnection(**config)

# 使用下面的方法初始化连接，也返回 GBaseConnection 对象
conn = connect(**config)
```

6.1.2.2 close

关闭连接方法。此方法调用后，当前 GBaseConnection 对象与 GBase Server 的连接断开。

函数名称	返回值	返回值类型	函数参数	参数含义
close	无	无	无	无

样例代码:

```
code
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = connect(**config)
conn.close()
```

6.1.2.3 connect

实现与 GBase Server 建立通讯连接。此方法如果不单独调用，则默认在 GBaseConnection 构造函数中调用。

函数名称	返回值	返回值类型	函数参数	参数含义
connect	无	无	**kwargs (dict)	连接参数,以字典方式传入构造函数。如下示例: config = { 'host': '172.16.31.10', 'user': 'gbase', 'port': 5258, 'database': 'test' }

样例代码:

```
code
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = GBaseConnection()
```

```
conn.connect(**config)
conn.close()
```

6.1.2.4 commit

实现事务提交功能，与事务回滚 rollback 方法对应。当连接事务开启后(连接参数 autocommit 配置为 false)，并且已经执行了 DML 语句，如：insert into [table] values('value1')。需要调用此方法提交事务，已使 DML 语句生效。

函数名称	返回值	返回值类型	函数参数	参数含义
commit	无	无	无	无

样例代码：

code

```
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test',
          'autocommit': False}
conn = connect(**config)
cur = conn.cursor()
cur.execute("insert into test values(1,'aaa')")
conn.commit()
conn.close()
```

6.1.2.5 cursor

此方法返回与 GBaseConnection 对象关联的游标对象，游标对象可用来执行 DML/DDL 语句、获取结果集、执行存储过程等。

函数名称	返回值	返回值类型	函数参数	参数含义
cursor	无	无	无	无

样例代码：

```
code
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
cur.execute("insert into test values(1,'aaa')")
conn.close()
```

6.1.2.6 rollback

实现事务回滚功能，与事务提交 commit 方法对应。当连接事务开启后（连接参数 autocommit 配置为 false），并且已经执行了 DML 语句，如：insert into [table] values('value1')。需要调用此方法可回滚事务，已使 DML 语句取消。

函数名称	返回值	返回值类型	函数参数	参数含义
commit	无	无	无	无

样例代码：

```
code
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test',
          'autocommit': False}
conn = connect(**config)
cur = conn.cursor()
cur.execute("insert into test values(1,'aaa')")
conn.rollback()
conn.close()
```

6.2 游标类

此类可以操作 GBase Server 数据库资源，执行 DML/DDL 语句，调用存储过程，获取结果集功能。

6.2.1 属性

6.2.1.1 description

此属性用来描述执行 SQL 查询后的结果集列信息，以 LIST 方式存储，并包含以下数据内容。

- name：列名
- code：列类型编码
- display_size：None
- internal_size：None
- precision：None
- scale：None
- null_ok：是否为空

类型	值	默认值	可读写
list	结果集列信息列表，列表中为元组，存储上述描述的内容	None	read

6.2.1.2 rowcount

此属性用来描述 SQL 查询结果后的结果集行数量，或者为 UPDATE、DELETE、INSERT 后影响的数据行数。

类型	值	默认值	可读写
int	行数	-1	read

6.2.1.3 arraysize

此属性使用 fetchmany 获取结果集大小，fetchmany 的 size 参数为 None 时生效。

类型	值	默认值	可读写
int	结果集行数	1	read/write

6.2.1.4 Info

当执行 INSERT、UPDATE、DELETE、MERGE 和 LOAD 语句后，数据库会以字符串形式返回 info 信息，info 属性功能同 CAPI 的 gbase_info 方法，用以获取该字符串。

类型	值	默认值	可读写
string	数据库返回的信息	“”	read

6.2.2方法

6.2.2.1 GBaseCursor

GBaseCursor 类构造函数。

函数名称	返回值	返回值类型	函数参数	参数含义
GBaseCursor	instance	GBaseCursor	connection (GBaseConnection)	与 GBase Server 已经建立连接的

				GBaseConnection 对象。

样例代码:

```
code
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
conn.close()
```

6.2.2.2 execute

该方法用来执行单条或多条 DML/DDL 语句，如果为多条语句则返回迭代器。

函数名称	返回值	返回值类型	函数参数	参数含义
execute	多语句迭代器	iterator	1.operation(string) 2.params(tuple) 3.multi_stmt(bool)	1.要执行的 SQL 语句 2.SQL 语句参数 3.是否为多条 SQL 语句

样例代码:

```
code 1
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
operation = "SELECT 1; INSERT INTO test VALUES(1,'a'); SELECT 2"
```

```
for result in cur.execute(operation,multi_stmt=True):
    if result.has_rows:
        print result.fetchall()
    else:
        print 'no result set'
conn.close()
```

code 2

```
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
operation = "INSERT INTO test VALUES(%s,%s)"
cur.execute(operation,(1,'a'))
conn.close()
```

6.2.2.3 executemany

批量执行多条 SQL 语句，并针对 insert 语句进行优化执行。

函数名称	返回值	返回值类型	函数参数	参数含义
executemany	无	无	1.operation(string) 2.seq_of_params(list)	1.SQL 语句 2.SQL 语句参数 LIST

样例代码:

code

```
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
```

```
            'password' : 'gbase20110531',
            'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
operation = "INSERT INTO test VALUES(%s,%s)"
cur.executemany(operation, [(1,'a'),(2,'b')])
conn.close()
```

6.2.2.4 callproc

执行存储过程。可执行带 out 参数的存储过程。

函数名称	返回值	返回值类型	函数参数	参数含义
callproc	无	返回游标类型的迭代	1.procname(string) 2.params(tuple)	1.存储过程名 2.存储过程参数列表

样例代码:

code 1 (不带 OUT 参数)

```
DELIMITER |
CREATE PROCEDURE `test`.`test_proc`(in id int, in name varchar(50))
BEGIN
    select id,name;
END |
```

```
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
            'port' : 5258,
            'user' : 'gbase',
            'password' : 'gbase20110531',
            'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
res = cur.callproc("test_proc", (1,'test'))
for rs in res:
    print rs.fetchall()
```

```
conn.close()
```

code 2 (带 OUT 参数)

```
CREATE PROCEDURE `test`.`test_proc`(OUT param_name int)
BEGIN
     set param_name = 100;
END |
```

```
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
            'port' : 5258,
            'user' : 'gbase',
            'password' : 'gbase20110531',
            'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
for rs in cur.callproc('test_proc', ('',)):
    pass
print cur.save_param_val()
conn.close()
```

6.2.2.5 fetchone

该方法在执行 SQL 查询语句后，获取一行结果集。

函数名称	返回值	返回值类型	函数参数	参数含义
fetchone	包含一行结果集的元组	tuple	无	无

样例代码:

code

```
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
            'port' : 5258,
            'user' : 'gbase',
            'password' : 'gbase20110531',
            'database' : 'test'}
conn = connect(**config)
```

```
cur = conn.cursor()
cur.execute("select * from test")
print cur.fetchone()
conn.close()
```

6.2.2.6 fetchmany

该方法在执行 SQL 查询语句后，获取多行结果集。

函数名称	返回值	返回值类型	函数参数	参数含义
fetchmany	结果集 LIST	list	size(int)	size 如果为 None，则返回 arraysize 行数据；否则返回 size 行数据。

样例代码:

code

```
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
cur.execute("select * from test")
print cur.fetchmany(2)
conn.close()
```

6.2.2.7 fetchall

该方法在执行 SQL 查询语句后，获取所有行结果集。

函数名称	返回值	返回值类型	函数参数	参数含义
fetchall	结果集 LIST	list	无	无

样例代码：

```
code
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
cur.execute("select * from test")
print cur.fetchall()
conn.close()
```

6.2.2.8 nextset

该方法在执行多条 SQL 查询语句，且只需要多个查询结果中的一个时，可通过该方法跳过其他结果集，从而简化结果集获取过程。

函数名称	返回值	返回值类型	函数参数	参数含义
nextset	如果没有更多的结果集，这个方法返回 None。否则将返回 true。	bool None	next_number	跳过结果集的数量

样例代码：

```
code
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = connect(**config)
```

```
cur = conn.cursor()
cur.execute("select 1;select 2;select * from test",multi_stmt=True)
cur.nextset(2)
print cur.fetchall()
conn.close()
```

6.2.2.9 close

该方法在游标对象使用完成后，可以调用此方法关闭游标对象，并释放资源。

函数名称	返回值	返回值类型	函数参数	参数含义
close	返回执行成功或者失败的状态	bool	无	无

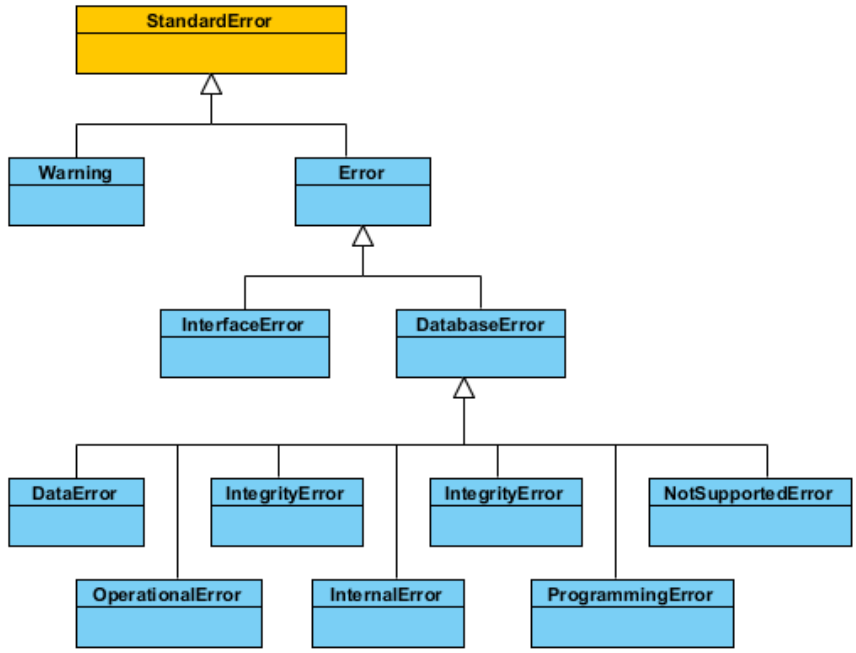
样例代码:

```
code
from GBaseConnector import connect
config = {'host' : '172.16.3.10',
          'port' : 5258,
          'user' : 'gbase',
          'password' : 'gbase20110531',
          'database' : 'test'}
conn = connect(**config)
cur = conn.cursor()
cur.execute("select * from test")
print cur.fetchall()
conn.close()
```

6.3 异常类

接口将所有异常处理类全部封装到了 GBaseError.py 文件中，所以当引用这些异常类时需引用 GBaseError.py 文件。所有的错误、异常、警告等内容的

错误处理全部经由此文件中的类完成。下图为所有异常类继承关系图。



图表 6-1 GBase Python 异常类继承图

- Warning

抛出重要的警告如当正在插入时数据截断等。

- Error

错误异常的父类，可以使用 Error 捕获所有数据库操作异常。

- InterfaceError

相关的数据库接口，不是数据库本身的错误引发的异常。

- DatabaseError

代表数据库相关的错误。

- OperationError

数据库操作错误,比如数据源未找到，事务无法处理，内存分配错误等。

- IntegrityError

数据完整性不一致错误，比如外键检查错误等。

- InternalError

数据库内部错误,比如游标无效(游标关闭后继续调用)，事务不同步等。

- ProgrammingError

编程错误，比如表不存在，SQL 语法错误等。

- NotSupportedError

数据库不支持的错误。比如在不支持事务的数据库上调用 rollback 操作等。

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