guobin second commit `c7fe008` on Oct 17, 2017

0 contributors

116 lines (92 sloc) | 2.92 KB

```
# -*-coding=utf-8-*-
import sys
import os
import pymysql


def command_line_args(args):
    if len(args) == 0:
        return False
    argsDict = dict()
    if '--help' in args:
        argsDict['help'] = 1
        return argsDict
    for arg in args:
        params = arg.split('=')
        if len(params) != 2:
            return False

        key = params[0]
        if not key.startswith('--'):
            return False
        key = key[2:]
        argsDict[key] = params[1]
    return argsDict


def help():
    print '''help desc:
        --host=(mysql host)
        --user=(mysql database user)
        --password=(mysql database user)
        --port=(value or default 3306)
        --database=(databases name)
        --back-path=(backpath)'''


def get_tables(args):
    db = args['database']
    conn = pymysql.connect(host=args['host'],
                           port=args.has_key('port') and args['port'] or 3306,
                           user=args['user'],
                           passwd=args['password'],
                           db=db
                           )
    cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)
    cursor.execute('SELECT table_name FROM information_schema.tables ' +
                   "WHERE table_schema='" + db + "' and table_type='base table'")
    rets = cursor.fetchall()
    conn.close()
    return rets


def get_lock_tables_sql(tables):
    sql = 'LOCK TABLES '
    for val in tables:
        sql += val['table_name'] + ' READ,'
    sql = sql[:-1] + ';'
    return sql

```

```
60
61  def valid_back_path(args):
62      if not args.has_key('back-path'):
63          print 'back-path is required'
64          sys.exit(1)
65
66      if not os.path.isdir(args['back-path']):
67          print 'back-path is not valid path'
68          sys.exit(1)
69
70
71  def lock_tables_dump_data(args):
72      valid_back_path(args)
73      db = args['database']
74      db_tables = get_tables(args)
75      lock_sql = get_lock_tables_sql(db_tables)
76
77      conn = pymysql.connect(host=args['host'],
78                             port=args.has_key('port') and args['port'] or 3306,
79                             user=args['user'],
80                             passwd=args['password'],
81                             db=db
82                             )
83      cursor = conn.cursor()
84      cursor.execute(lock_sql)
85
86      command_dump_mysql_db(args)
87
88      cursor.execute('UNLOCK TABLES;')
89      cursor.close()
90
91
92  def command_dump_mysql_db(args):
93      cmd = '/usr/local/bin/mydumper -t 8 -u %s -p %s -h %s -P %s --long-query-guard 300 --no-locks -B %s -c -o %s' % (
94          args['user'],
95          args['password'],
96          args['host'],
97          args.has_key('port') and args['port'] or 3306,
98          args['database'],
99          args['back-path']
100     )
101     output = os.popen(cmd)
102     print output.read()
103
104
105 if __name__ == '__main__':
106     args = command_line_args(sys.argv[1:])
107     if not args:
108         print ("params is error")
109         sys.exit(1)
110
111     if args.has_key('help'):
112         help()
113         sys.exit(0)
114
115     lock_tables_dump_data(args)
```