针对某一个硬盘分区（例如E盘），导入到数据库中，并做如下统计:

1.一共有多少普通文件，多少目录夹

2.所有普通文件及目录共占用了多少字节、所占用空间（数据块单位512字节）

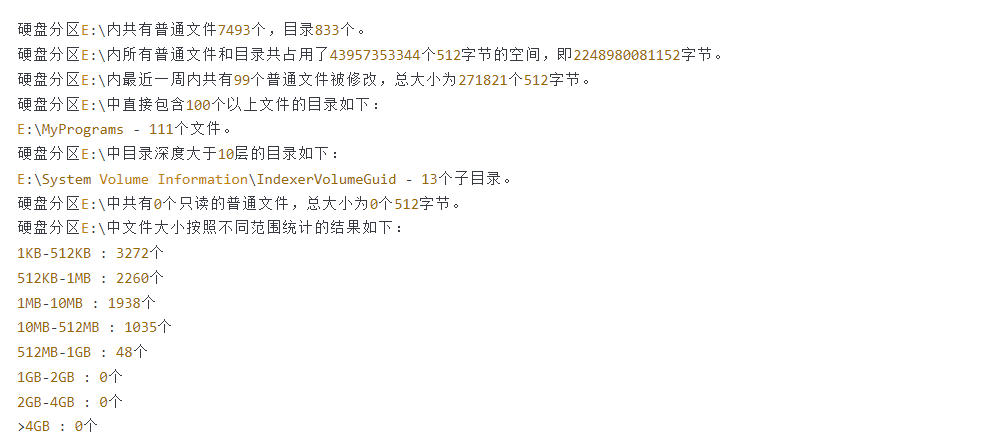
3.统计最近1周内修改的普通文件的文件名称、文件大小、以及文件总数

4.统计直接包含100普通文件以上的目录的目录名称以及文件数量

5.统计目录深度大于10层的目录名称

6.统计所有只读的普通文件数量、所占字节数

7.统计所有文件的的大小，按照1KB，512KB，1MB，10MB，512MB，1GB,2GB，4GB，以及以上统计文件数量



代码：

import os

import time

# 分类统计所有文件和目录

def count\_files\_and\_dirs(path):

files = 0

dirs = 0

for root, \_, filenames in os.walk(path):

for filename in filenames:

files += 1

for \_ in root.split(os.sep):

dirs += 1

return (files, dirs)

# 统计所有普通文件和目录占用空间

def count\_size(path):

size = 0

for root, \_, filenames in os.walk(path):

for filename in filenames:

filepath = os.path.join(root, filename)

size += os.stat(filepath).st\_size

size += os.path.getsize(root)

return (size // 512, size)

# 统计最近1周修改过的普通文件

def find\_recent\_files(path):

recent\_files = []

for root, \_, filenames in os.walk(path):

for filename in filenames:

filepath = os.path.join(root, filename)

time\_diff = time.time() - os.path.getmtime(filepath)

if time\_diff <= 7 \* 24 \* 60 \* 60: # 1 week

recent\_files.append((filename, os.path.getsize(filepath)))

return recent\_files

# 统计直接包含100普通文件以上的目录信息

def find\_large\_dirs(path):

large\_dirs = []

for root, \_, filenames in os.walk(path):

file\_count = len(filenames)

if file\_count >= 100:

large\_dirs.append((root, file\_count))

return large\_dirs

# 统计目录深度大于10层的目录信息

def find\_deep\_dirs(path):

deep\_dirs = []

for root, \_, \_ in os.walk(path):

depth = len(root.split(os.sep))

if depth > 10:

deep\_dirs.append(root)

return deep\_dirs

# 统计只读文件数量和占用字节数

def count\_readonly\_files(path):

readonly\_files = 0

size = 0

for root, \_, filenames in os.walk(path):

for filename in filenames:

filepath = os.path.join(root, filename)

if not os.access(filepath, os.W\_OK):

readonly\_files += 1

size += os.path.getsize(filepath)

return (readonly\_files, size)

# 统计所有文件的大小分类信息

def count\_size\_distribution(path):

size\_distribution = [0] \* 10

for root, \_, filenames in os.walk(path):

for filename in filenames:

filepath = os.path.join(root, filename)

filesize = os.path.getsize(filepath)

if filesize < 1024:

size\_distribution[0] += 1

elif filesize < 512 \* 1024:

size\_distribution[1] += 1

elif filesize < 1024 \* 1024:

size\_distribution[2] += 1

elif filesize < 10 \* 1024 \* 1024:

size\_distribution[3] += 1

elif filesize < 512 \* 1024 \* 1024:

size\_distribution[4] += 1

elif filesize < 1024 \* 1024 \* 1024:

size\_distribution[5] += 1

elif filesize < 2 \* 1024 \* 1024 \* 1024:

size\_distribution[6] += 1

elif filesize < 4 \* 1024 \* 1024 \* 1024:

size\_distribution[7] += 1

else:

size\_distribution[8] += 1

return size\_distribution

# 测试

path = 'E:\\'

print('1. 普通文件数量：%d，目录数量：%d' % count\_files\_and\_dirs(path))

print('2. 所有文件共占用空间：%d 字节，占用块数：%d' % count\_size(path))

print('3. 最近1周内修改的普通文件信息：', find\_recent\_files(path))

print('4. 包含100个以上普通文件的目录信息：', find\_large\_dirs(path))

print('5. 目录深度大于10层的目录名称：', find\_deep\_dirs(path))

print('6. 只读文件数量：%d，占用字节数：%d' % count\_readonly\_files(path))

print('7. 文件大小分类统计：\n 1KB-512KB：%d\n 512KB-1MB：%d\n 1MB-10MB：%d\n 10MB-512MB：%d\n 512MB-1GB：%d\n 1GB-2GB：%d\n 2GB-4GB：%d\n 4GB以上：%d' % tuple(count\_size\_distribution(path)))

import os

import time

import pymysql

# 分类统计所有文件和目录

def count\_files\_and\_dirs(path):

files = 0

dirs = 0

for root, \_, filenames in os.walk(path):

for filename in filenames:

files += 1

for \_ in root.split(os.sep):

dirs += 1

return (files, dirs)

# 统计所有普通文件和目录占用空间

def count\_size(path):

size = 0

for root, \_, filenames in os.walk(path):

for filename in filenames:

filepath = os.path.join(root, filename)

size += os.stat(filepath).st\_size

size += os.path.getsize(root)

return (size // 512, size)

# 连接数据库

conn = pymysql.connect(host='127.0.0.1', user='myuser', password='Lp200211', database='dascalb2018', charset=’uf8’)

# 创建游标对象

cursor = conn.cursor()

# 创建数据表

sql\_create\_table = '''

CREATE TABLE IF NOT EXISTS disk\_info (

id INT AUTO\_INCREMENT PRIMARY KEY,

path VARCHAR(255) NOT NULL,

file\_count INT NOT NULL,

dir\_count INT NOT NULL,

size INT NOT NULL,

size\_blocks BIGINT NOT NULL,

create\_time DATETIME DEFAULT CURRENT\_TIMESTAMP

)

'''

cursor.execute(sql\_create\_table)

conn.commit()

# 获取硬盘分区信息

path = 'E:\\'

file\_count, dir\_count = count\_files\_and\_dirs(path)

size, size\_blocks = count\_size(path)

# 插入数据

sql\_insert = '''

INSERT INTO disk\_info (path, file\_count, dir\_count, size, size\_blocks)

VALUES (%s, %s, %s, %s, %s)

'''

data = (path, file\_count, dir\_count, size, size\_blocks)

cursor.execute(sql\_insert, data)

conn.commit()

# 关闭游标和连接

cursor.close()

conn.close()