中科云

2018年4月23日 14:29

基础知识:

```
MySQL III 試題
     有两个表定义如下。
    CREATE TABLE stations (
      'id' int(11) NOT NULL AUTO_INCREMENT.
      code varchar(28) NOT NULL,
      province varchar(20) DEFAULT NULL,
      city varchar(40) DEFAULT NULL,
     name varchar(45) DEFAULT NULL,
    PRIMARY KEY ("10"),
    UNIQUE KEY uniq code ( code ).
    KEY 'ix_province' ('province'),
    KEY 'ix_city' ('city'),
  ) ENGINE=InnoDB AUTO_INCREMENT=2631 DEFAULT CHARSET=utf8;
 CREATE TABLE history (
    scode varchar(20) NOT NULL,
   time datetime NOT NULL,
   a smallint(11) DEFAULT NULL,
   'b' smallint(6) DEFAULT NULL,
   c float DEFAULT NULL,
  PRIMARY KEY ('code', 'time'),
  CONSTRAINT '1bfk_1' FOREIGN KEY ('scode') REFERENCES 'stations' ('code')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

(1) small int和int类型的区别, 其它还有什么整数类型?

smallint: 2字节, 大整数值

int:4字节,大整数值

整数类型: tinyint, mediumint, bigint

(2) auto_increment-2631是什么意思

自增id从2631开始

(3) uniq_code, ix_province, ix_city是什么意思, 有什么作用? uniq_code, ix_province, ix_city联合唯一索引, 加速查找, 不能重复

数据查询

- (1) 从history表中查询2016年8月份每个station的a(smallint),b(smallint),c(float)的平均值 select * from history
- (2) 从history表中和stations表中查询2016年8月份每个station每一天的a不为空的记录数量,要求输出包含站点code, province, city
- (3)在同一地区A和B两个站点,取同一时间各项数据的平均值作为该地区该时间的数值;从history表中查询该地区2017年的a,b,c的平均值
 - (4) 请简述left join, right join, inner join, outer join的区别 inner join: 只连接匹配的行

left join: 左连接: 优先显示左表的全部记录

right join: 右连接, 优先显示右表的全部记录

outer join包含left join, right join, full join(mysql不支持)

数据插入

(1) 请编写查询,往history表中插入两条数据,scode=CC1234 insert into stations(code, province, city, name) values ('CC1234','陕西','西安','aa');

insert into history(scode, time, a, b, c) values ('CC1234', '2016. 8. 1', 11, 12, 11. 12), ('CC1234', '2016. 8. 2', 11, 12, 11. 12);

(2) 现有另外一个history_another表,和history表结构完全一样,请编写查询从history_another表中往history表中迁移数据 create table history_another like history; insert into history_another select * from history;

(3) 请分析问题(2) 中的SQL语句是否会执行失败? 在什么情况下执行失败? 是否有办法避免?

```
以下代码的输出是什么? 请给出答案并解释
  class Parent(object):
     x=1
  class Child1(Parent):
     pass
  class Child2(Parent):
     pass
  print Parent. x, Child1. x, Child2. x
  Child. x=2
  print Parent. x, Child1. x, Child2. x
  Parent. x=3
  print Parent. x, Child1. x, Child2. x
```

```
1 1 1
1 2 1
3 2 3
```

以下代码的输出是什么?请给出答案并解释 def multipliers():

return [lambda x:i*x for i in range(4)] print([m(2) for m in multipliers()]) 输出[6,6,6,6] 请修改multipliers的定义来产生期望的结果 def multipliers(): return (lambda x:i*x for i in range(4))

有0<x<=10,10<x<=20,20<x<=30,……,200<x,这样的21个区间分别对应1-21二十一个级别,请编写一个函数level(x)根据输入值返回对应级别

有一个数据结构如下所示, 请编写一个函数从该结构数据中返回由指定

```
的字段和对应的值组成的字典。如果指定字段不存在, 则跳过该字
段。(10分)
data: {"time": "2016-08-05T13:13:05", "some_id": "ID1234",
 "grp1": { "fld1":1,
 "fld2":2},
 "xxx2": { "fld3":0,
 "fld5":0.4},
 "fld6":11,
 "fld7":7,
"fId46":8}
fields:由"|"连接的以"fld"开头的字符串,如:fld2|fld3|fld|fld19
def select(data, fields):
   result={}
   field_lst=fields.split('|')
   for key in data:
     if key in field 1st:
       result[key]=data[key]
     elif type(data[key]) == dict:
       res=select(data[key], fields)
```

result.update(res)
return result

fields='fld2|fld3|fld7|fld19'
r=select(data, fields)
print(r)