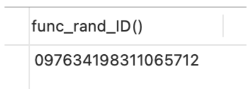
作业6-10185102142-李泽浩

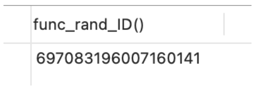
(1)编写函数func_rand_ID，可随机生成长度为18位的仿身份证号码，第1-6位为0-9之间的数字，第7-14位为19600101到20200101之间的某个有效日期，第15-17位为0-9之间的数字，第18位为0-9之间的数字或大写X。

```
DROP FUNCTION IF EXISTS func_rand_ID;

CREATE FUNCTION func_rand_ID() RETURNS VARCHAR(30)
READS SQL DATA
BEGIN
  DECLARE result VARCHAR(18) DEFAULT '';
  DECLARE first_num VARCHAR(6) DEFAULT '';
  DECLARE birthday VARCHAR(10) DEFAULT '';
  DECLARE second_num VARCHAR(3) DEFAULT '';
  DECLARE last_num VARCHAR(1) DEFAULT '';
  DECLARE charset VARCHAR(11) DEFAULT '';
  DECLARE interday INT DEFAULT 0;
  DECLARE numset varchar(10) DEFAULT '';
  DECLARE i INT DEFAULT 0;
  DECLARE j INT DEFAULT 0;

  SET numset = "0123456789";
  SET charset = "0123456789X";

  #处理1-6位置上
  WHILE i<6 DO
    SET first_num = CONCAT(first_num, substring(numset,FLOOR(1+RAND()*10),1));
    SET i = i + 1;
  END while;
  #SET first_num = rand()*100000;

  #处理生日
  SET interday = floor(RAND()*21915);#时间间隔
  SET birthday = DATE_ADD("1960-01-01",INTERVAL interday day);
  SELECT REPLACE(birthday,'-','') INTO birthday;#去除横线

  #处理15-17位置上
  WHILE j<3 DO
    SET second_num = CONCAT(second_num,substring(numset,FLOOR(1+RAND()*10),1));
    SET j = j + 1;
  END while;
  #SET second_num = floor(RAND()*100);

  #处理最后一位
  SET last_num = substring(charset,FLOOR(1+RAND()*11),1);

  #合并结果
  SET result = CONCAT(result, first_num, birthday, second_num, last_num);

  RETURN result;
END;

SELECT func_rand_ID();
```

随机生成两个身份证号如下：



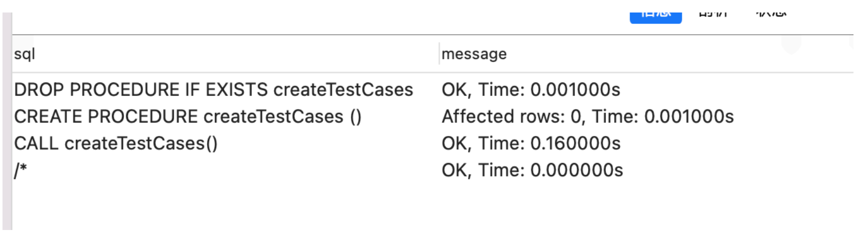
func_rand_ID()
097634198311065712



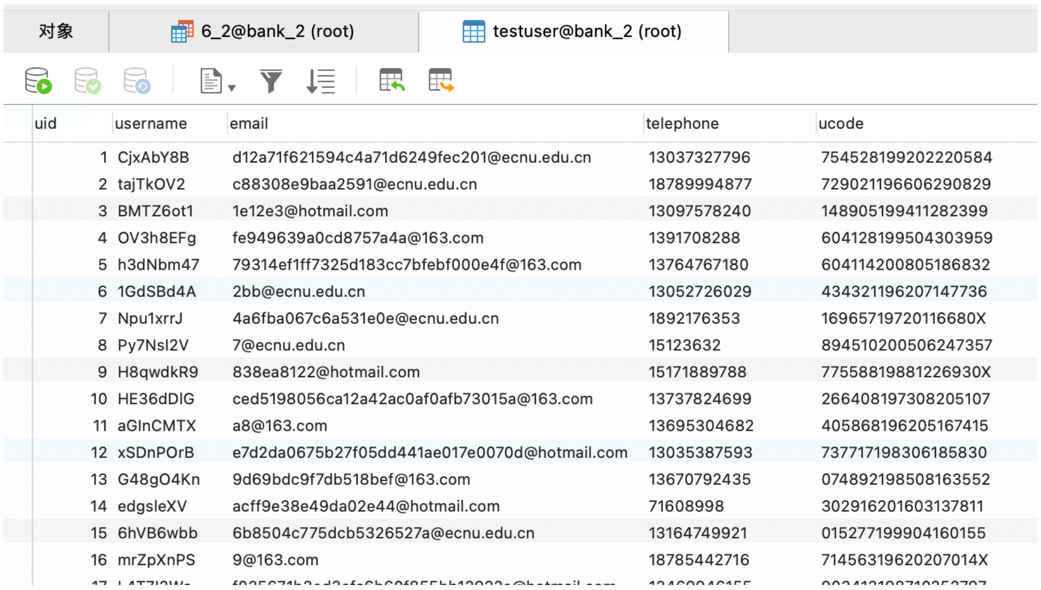
func_rand_ID()
697083196007160141

（2）使用下面的语句新建一张测试表testUser，编写存储过程createTestCases，往testUser表中插入100条测试数据，其中username是随机生成的长度为8的字符串（符号可包括a-z、A-Z、0-9），email由函数func_rand_email生成，telephone由函数func_rand_telnum生成，ucode由函数func_rand_ID生成。

```
DROP PROCEDURE IF EXISTS createTestCases;

CREATE PROCEDURE createTestCases ()
BEGIN
  DECLARE i INT DEFAULT 0;
  DECLARE id INT DEFAULT 0;
  DECLARE cnt INT DEFAULT 0;
  DECLARE name VARCHAR(8) DEFAULT '';
  DECLARE mail VARCHAR(75) DEFAULT '';
  DECLARE phone VARCHAR(11) DEFAULT '';
  DECLARE code VARCHAR(30) DEFAULT '';
  DECLARE charset VARCHAR(70);

  SET charset =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';

  WHILE cnt < 100 DO
    SET name = '';
    SET i = 0;
    SELECT cnt INTO id;#uid
    SELECT func_rand_email() into mail;#email
    SELECT func_rand_telnum() into phone;#telephone
    SELECT func_rand_ID() into code;#ucode
    #生成username
    WHILE i<8 DO
      SET name = concat(name,substring(charset,floor(1+RAND()*62),1));
      SET i=i+1;
    END WHILE;
    INSERT INTO testuser VALUES (id,name,mail,phone,code);

    SET cnt = cnt + 1;
  END WHILE;

END;

CALL createTestCases();
```



```
sql                                          message
DROP PROCEDURE IF EXISTS createTestCases     OK, Time: 0.001000s
CREATE PROCEDURE createTestCases ()          Affected rows: 0, Time: 0.001000s
CALL createTestCases()                       OK, Time: 0.160000s
/*                                           OK, Time: 0.000000s
```

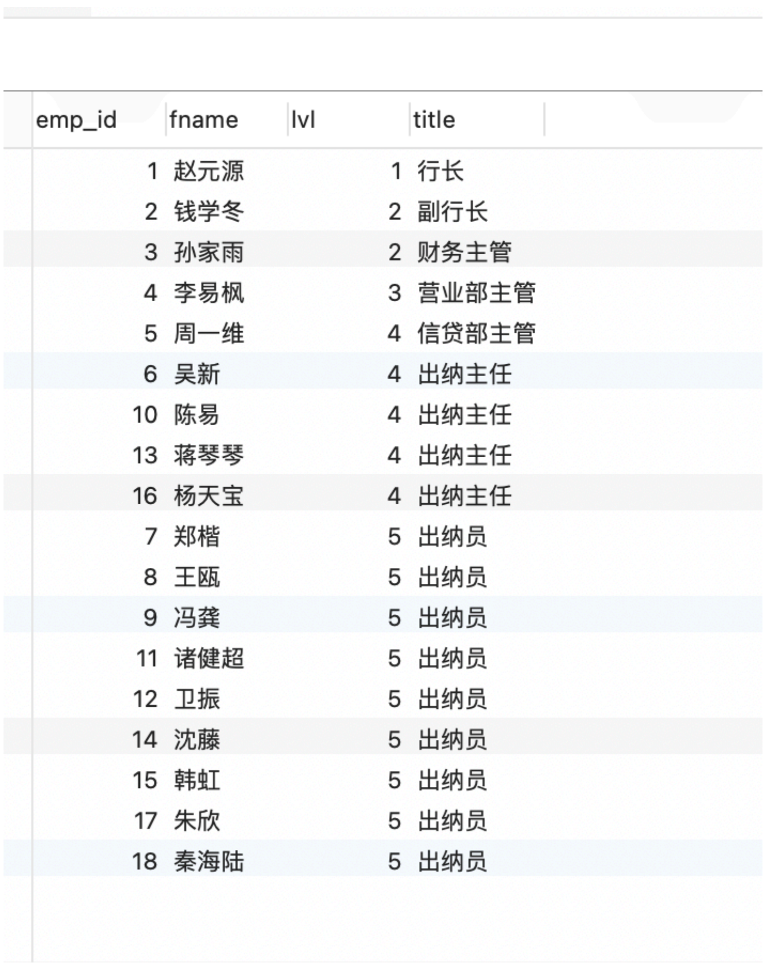


uid	username	email	telephone	ucode
1	CjxAbY8B	d12a71f621594c4a71d6249fec201@ecnu.edu.cn	13037327796	754528199202220584
2	tajTkOV2	c88308e9baa2591@ecnu.edu.cn	18789994877	729021196606290829
3	BMTZ6ot1	1e12e3@hotmail.com	13097578240	148905199411282399
4	OV3h8EFg	fe949639a0cd8757a4a@163.com	1391708288	604128199504303959
5	h3dNbm47	79314ef1ff7325d183cc7bfebf000e4f@163.com	13764767180	604114200805186832
6	1GdSBd4A	2bb@ecnu.edu.cn	13052726029	434321196207147736
7	Npu1xrrJ	4a6fba067c6a531e0e@ecnu.edu.cn	1892176353	16965719720116680X
8	Py7NsI2V	7@ecnu.edu.cn	15123632	894510200506247357
9	H8qwdkR9	838ea8122@hotmail.com	15171889788	77558819881226930X
10	HE36dDIG	ced5198056ca12a42ac0af0afb73015a@163.com	13737824699	266408197308205107
11	aGInCMTX	a8@163.com	13695304682	405868196205167415
12	xSDnPOrB	e7d2da0675b27f05dd441ae017e0070d@hotmail.com	13035387593	737717198306185830
13	G48gO4Kn	9d69bdc9f7db518bef@163.com	13670792435	074892198508163552
14	edgsleXV	acff9e38e49da02e44@hotmail.com	71608998	302916201603137811
15	6hVB6wbb	6b8504c775dcb5326527a@ecnu.edu.cn	13164749921	015277199904160155
16	mrZpXnPS	9@163.com	18785442716	71456319620207014X

（3）在employee表中有引用EMP_ID字段的SUPERIOR_EMP_ID字段，代表上级领导编号，行长没有上级领导，因此其SUPERIOR_EMP_ID的值为NULL，其余员工的SUPERIOR_EMP_ID值均不为NULL。创建存储过程getEmpStructure，在employee表上利用递归CTE实现员工层级的计算，行长层级为1，副行长为2，依此类推，该存储过程输出员工编号（EMP_ID）、员工姓名（LAST_NAME和FIRST_NAME拼接）、层级（命名为LEVEL）、职位名称（TITLE）。

```
#非procedure方法
WITH RECURSIVE employee_level AS(
    SELECT EMP_ID, CONCAT(LAST_NAME, FIRST_NAME) as fname, 1 lvl, TITLE
    FROM employee
    WHERE SUPERIOR_EMP_ID IS NULL
      UNION ALL
    SELECT e.EMP_ID, CONCAT(e.LAST_NAME,e.FIRST_NAME) as fname,  lvl+1, e.TITLE
    FROM employee as e  INNER JOIN employee_level as el on e.SUPERIOR_EMP_ID =
el.EMP_ID
  )
  SELECT emp_id, fname, lvl, e.title
  FROM employee_level as el INNER JOIN employee as e USING(EMP_ID)
  ORDER BY lvl;
```

```
DROP PROCEDURE IF EXISTS getEmpStructure;

CREATE PROCEDURE getEmpStructure()
BEGIN
    WITH RECURSIVE employee_level AS(
    SELECT EMP_ID, CONCAT(LAST_NAME,FIRST_NAME) as fname, 1 lvl, TITLE
    FROM employee
    WHERE SUPERIOR_EMP_ID IS NULL
      UNION ALL
    SELECT e.EMP_ID, CONCAT(e.LAST_NAME,e.FIRST_NAME) as fname,  lvl+1, e.TITLE
    FROM employee as e  INNER JOIN employee_level as el on e.SUPERIOR_EMP_ID =
el.EMP_ID
  )
  SELECT * FROM employee_level;

END;

CALL getEmpStructure();
```



emp_id	fname	lvl	title
1	赵元源	1	行长
2	钱学冬	2	副行长
3	孙家雨	2	财务主管
4	李易枫	3	营业部主管
5	周一维	4	信贷部主管
6	吴新	4	出纳主任
10	陈易	4	出纳主任
13	蒋琴琴	4	出纳主任
16	杨天宝	4	出纳主任
7	郑楷	5	出纳员
8	王瓯	5	出纳员
9	冯龚	5	出纳员
11	诸健超	5	出纳员
12	卫振	5	出纳员
14	沈藤	5	出纳员
15	韩虹	5	出纳员
17	朱欣	5	出纳员
18	秦海陆	5	出纳员

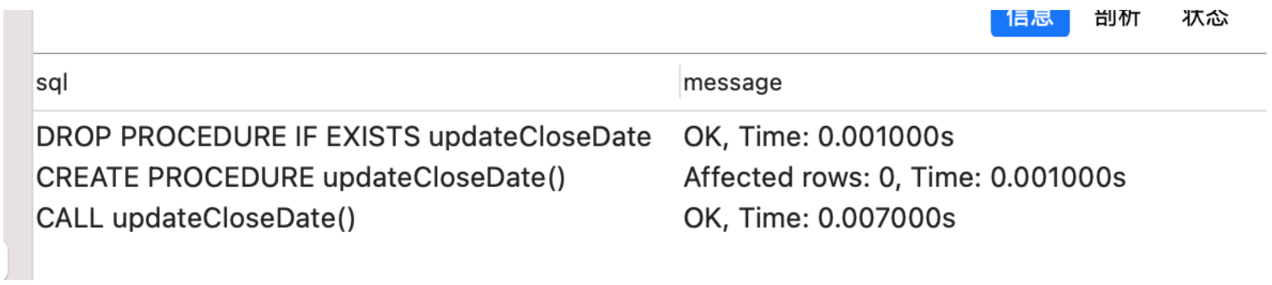
（4）创建存储过程updateCloseDate，该存储过程更新account表中的关闭日期（CLOSE_DATE），根据产品编号做不同的更新操作，要求使用游标：

a.对产品编号对应类型为存款的账户（即产品编号PRODUCT_CD对应PRODUCT_TYPE_CD为ACCOUNT），如开户日期（OPEN_DATE）在2015-01-01之前（含）的，设置其关闭日期（CLOSE_DATE）为开户日期加20年，否则为开户日期加30年；

b.对产品编号对应类型为贷款的账户（即产品编号PRODUCT_CD对应PRODUCT_TYPE_CD为LOAN），如可用余额（AVAIL_BALANCE）少于100000的（含），设置其关闭日期（CLOSE_DATE）为开户日期加20年，否则为开户日期加30年；

c.对产品编号对应类型为保险的账户（即产品编号PRODUCT_CD对应PRODUCT_TYPE_CD为INSURANCE），设置其关闭日期（CLOSE_DATE）为开户日期加15年。

```
DROP PROCEDURE IF EXISTS updateCloseDate;

CREATE PROCEDURE updateCloseDate()
BEGIN
  DECLARE id INT DEFAULT 0; #account_id
  DECLARE money DECIMAL(12,4) DEFAULT 0;  #AVAIL_BALANCE
  DECLARE type_cd varchar(255) DEFAULT ''; #PRODUCT_TYPE_CD
  DECLARE opendate DATE; #START_DATE
  DECLARE cursor_update CURSOR FOR
    SELECT a.ACCOUNT_ID, a.AVAIL_BALANCE, a.OPEN_DATE, p.PRODUCT_TYPE_CD
    FROM account as a INNER JOIN product as p on a.PRODUCT_CD = p.PRODUCT_CD;

  DECLARE exit HANDLER FOR NOT FOUND CLOSE cursor_update;
  OPEN cursor_update;
  REPEAT
    FETCH cursor_update INTO id, money, opendate, type_cd;
    #case a-1
    UPDATE account
      SET account.CLOSE_DATE = ADDDATE(opendate,INTERVAL 20 YEAR)
      WHERE type_cd = "ACCOUNT" AND opendate <= "2015-01-01" AND id =
account.ACCOUNT_ID;
    #case a-2
    UPDATE account
      SET account.CLOSE_DATE = ADDDATE(opendate,INTERVAL 30 YEAR)
      WHERE type_cd = "ACCOUNT" AND opendate > "2015-01-01" AND id =
account.ACCOUNT_ID;
    #case b-1
    UPDATE account
      SET account.CLOSE_DATE = ADDDATE(opendate,INTERVAL 20 YEAR)
      WHERE type_cd = "LOAN" AND money <= 100000 AND id = account.ACCOUNT_ID;
    #case b-1
    UPDATE account
      SET account.CLOSE_DATE = ADDDATE(opendate,INTERVAL 30 YEAR)
      WHERE type_cd = "LOAN" AND money > 100000 AND id = account.ACCOUNT_ID;
    #case c
    UPDATE account
      SET account.CLOSE_DATE = ADDDATE(opendate,INTERVAL 15 YEAR)
      WHERE type_cd = "INSURANCE" AND id = account.ACCOUNT_ID;
    UNTIL 0 END REPEAT;

  CLOSE cursor_update;

END;

CALL updateCloseDate();
```



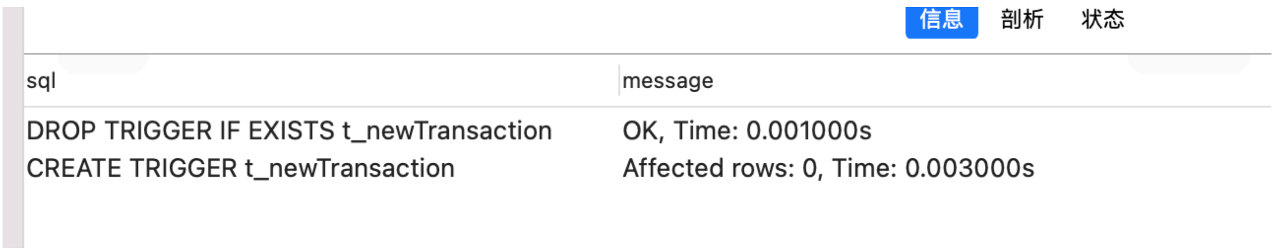
```
sql                                            message
DROP PROCEDURE IF EXISTS updateCloseDate       OK, Time: 0.001000s
CREATE PROCEDURE updateCloseDate()             Affected rows: 0, Time: 0.001000s
CALL updateCloseDate()                         OK, Time: 0.007000s
```

（5）在acc_transaction上定义触发器t_newTransaction，当往acc_transaction中插入一条数据时，依据账户编号（ACCOUNT_ID）更新account表中对应账户的可用余额（AVAIL_BALANCE）和最后活跃日期（LAST_ACTIVITY_DATE）：

a.如果插入数据的交易类型编码（TXN_TYPE_CD）为CD、TT、IC、LI则设置可用余额为当前可用余额加上交易金额、最后活跃日期为当前日期

b.如果插入数据的交易类型编码（TXN_TYPE_CD）为CW、TF则设置可用余额为当前可用余额减去交易金额、最后活跃日期为当前日期；如当前可用余额减去交易金额小于0，则撤销对acc_transaction此条数据的插入，同时输出提示"余额不足Insufficient Balance"。

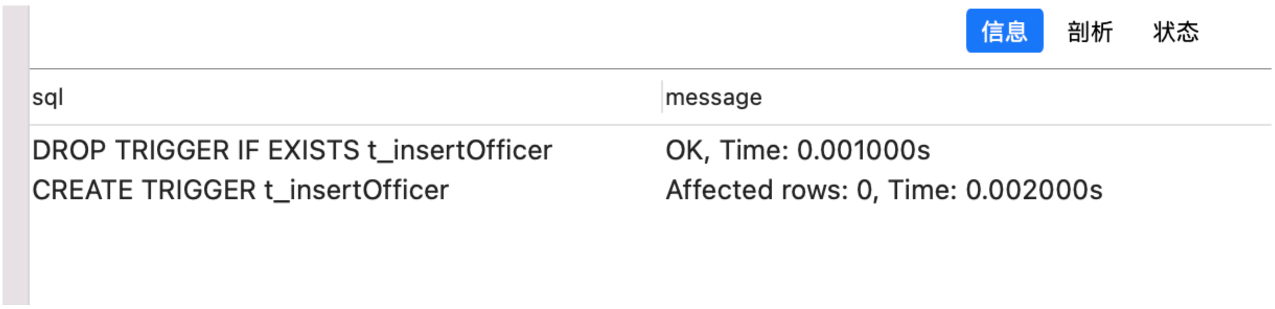
```
DROP TRIGGER IF EXISTS t_newTransaction;

CREATE TRIGGER t_newTransaction
BEFORE INSERT
ON acc_transaction FOR EACH ROW
BEGIN
  SET @message = "余额不足Insufficient Balance";

  #case a
  IF new.TXN_TYPE_CD = "CD" OR "TT" OR "IC" OR "LI" THEN
    #更新可用余额
    UPDATE account SET account.AVAIL_BALANCE = account.AVAIL_BALANCE + 
new.AMOUNT
      WHERE account.ACCOUNT_ID = new.ACCOUNT_ID;
    #更新时间
    UPDATE account SET account.LAST_ACTIVITY_DATE = CURRENT_DATE
      WHERE account.ACCOUNT_ID = new.ACCOUNT_ID;

  #case b
  ELSEIF new.TXN_TYPE_CD = "CW" OR "TF" THEN
    #当前可用余额减去交易金额小于0
    IF (account.AVAIL_BALANCE - new.AMOUNT < 0) THEN
      SIGNAL SQLSTATE 'HY000' SET MESSAGE_TEXT = @message;#抛出异常
    #当前可用余额减去交易金额大于0
    ELSE
      #更新金额
      UPDATE account SET account.AVAIL_BALANCE = account.AVAIL_BALANCE - 
new.AMOUNT
        WHERE account.ACCOUNT_ID = new.ACCOUNT_ID;
      #更新时间
      UPDATE account SET account.LAST_ACTIVITY_DATE = CURRENT_DATE
        WHERE account.ACCOUNT_ID = new.ACCOUNT_ID;
    END IF;
  END IF;
END;
```



sql	message
DROP TRIGGER IF EXISTS t_newTransaction	OK, Time: 0.001000s
CREATE TRIGGER t_newTransaction	Affected rows: 0, Time: 0.003000s

（6）使用下面的语句创建一张表officer_temp。在officer上定义触发器t_insertOfficer，当往officer中新插入一条单位联系人信息时候，检查对应的客户编号（CUST_ID）字段，如该客户编号对应的单位联系人信息在officer表中已存在，往officer_temp中插入一条数据，内容为：原有联系人编号（OFFICER_ID）、当前新联系人的START_DATE；如该客户编号对应的单位联系人信息不存在则直接插入。

```
DROP TRIGGER IF EXISTS t_insertOfficer;

CREATE TRIGGER t_insertOfficer
BEFORE INSERT
ON officer FOR EACH ROW
BEGIN
  DECLARE nid INT;

  SELECT f.OFFICER_ID INTO nid
  FROM officer as f
  WHERE f.CUST_ID = new.CUST_ID AND ISNULL(f.END_DATE)
  LIMIT 1;

  IF (nid IS NOT NULL) THEN
    INSERT INTO officer_temp VALUES(nid, new.START_DATE);
  END IF;

END;
```



sql	message
DROP TRIGGER IF EXISTS t_insertOfficer	OK, Time: 0.001000s
CREATE TRIGGER t_insertOfficer	Affected rows: 0, Time: 0.002000s