实习二：数据库约束设计

成员：陈嘉2000013094 马佳媛2000013121 黄粤2000013095

In [1]:
```
%load_ext sql
import pymysql
pymysql.install_as_MySQLdb()
%sql mysql://stu2000013094:stu2000013094@162.105.146.37:43306
%sql use stu2000013094;
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
0 rows affected.
```

Out[1]: []

In [2]:
```
#查看所有表
%sql show tables;
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
21 rows affected.
```

Out[2]:

Tables_in_stu2000013094
aggResult
article
barrage
coin
collect
comment
comment_post
creation
fans
favorites

本次实习的目标是体验如何在数据库中利用各种手段完成数据库约束设计，以及如何使用触发器完成类似物化视图的数据一致性维护。主要包括如下两个练习：

练习一 约束设计

1、按照题目要求进行建表：

employees(eno, ename, dno, salary, level, email)

各属性约束如下：
eno：整型，主码；
ename：长度为100的字符型，非空；
dno：整型，非空，外码，与department表中的dno相互参照；
salarv：整型，非空；

In [3]:
```
%%sql
/*若多个表之间存在外键约束，不考虑顺序直接drop会失败，因此我们先取消外键约束的检查*/
set @@foreign_key_checks=0;

# 如果数据表已经存在则删除表
# 创建employees表
drop table if exists employees;

CREATE TABLE employees
(
    emp_eno INT PRIMARY KEY,
    emp_ename VARCHAR(100) NOT NULL,
    emp_dno INT NOT NULL,
    emp_salary INT NOT NULL,
    emp_level INT NOT NULL,
    emp_email VARCHAR(20) UNIQUE NOT NULL
);
set @@foreign_key_checks=1;
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[3]: []

department(dno, dname, manager, budget)

各属性约束如下：
dno：整型，主码；
dname：{‘销售部’、’财务部’，’人事部’}之一，非空；
manager：整型，非空，外码，与employee表中的eno相互参照；
budget：整型，非空；
cost：整型，默认为0

In [4]:
```
%%sql
#若多个表之间存在外键约束，不考虑顺序直接drop会失败，因此我们先取消外键约束的检查
set @@foreign_key_checks=0;

# 如果数据表已经存在则删除表
drop table if exists department;

CREATE TABLE department
(
    dep_dno INT PRIMARY KEY,
    dep_dname ENUM('销售部','财务部','人事部') NOT NULL,
    dep_manager INT NOT NULL,
    dep_budget INT NOT NULL,
    dep_cost INT DEFAULT(0)

);
set @@foreign_key_checks=1;
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[4]: []

2、生成样例数据，插入一些条目

In [5]:
```
%%sql

delete from employees;
delete from department;

#department
INSERT INTO department(dep_dno,dep_dname,dep_manager,dep_budget,dep_cost)
VALUES(1,'销售部',7,10000,0);
INSERT INTO department(dep_dno,dep_dname,dep_manager,dep_budget,dep_cost)
VALUES(2,'财务部',3,20000,0);
INSERT INTO department(dep_dno,dep_dname,dep_manager,dep_budget,dep_cost)
VALUES(3,'人事部',5,15000,0);
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
0 rows affected.
0 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

Out[5]: []

In [6]:
```
%%sql

select * from department;
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
3 rows affected.
```

Out[6]:

dep_dno	dep_dname	dep_manager	dep_budget	dep_cost
1	销售部	7	10000	0
2	财务部	3	20000	0
3	人事部	5	15000	0

建立触发器，在每次插入新员工时更新对应的cost

In [7]:
```
%%sql
drop trigger if exists cost_by_now;
create trigger cost_by_now after insert
    on employees for each row
    update department set department.dep_cost=
    (select sum(emp_salary) from employees where emp_dno=new.emp_dno)
    where dep_dno = new.emp_dno;
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
0 rows affected.
0 rows affected.
```

Out[7]: []

In [8]:
```
%%sql

INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(1,'员工1号',1,1500,1,'12345@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(2,'员工2号',1,1700,1,'12346@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(3,'员工3号',3,3000,2,'12347@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(4,'员工4号',1,3500,3,'12348@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(5,'员工5号',3,4500,4,'12349@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(6,'员工6号',2,5500,5,'12325@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(7,'员工7号',2,1000,1,'12335@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(8,'员工8号',3,1500,1,'12315@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(9,'员工9号',2,2500,2,'12375@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES(10,'员工10号',2,3500,3,'12395@qq.com');
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

Out[8]: []

In [9]:
```
%%sql

select * from employees;
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
10 rows affected.
```

Out[9]:

emp_eno	emp_ename	emp_dno	emp_salary	emp_level	emp_email
1	员工1号	1	1500	1	12345@qq.com
2	员工2号	1	1700	1	12346@qq.com
3	员工3号	3	3000	2	12347@qq.com
4	员工4号	1	3500	3	12348@qq.com
5	员工5号	3	4500	4	12349@qq.com
6	员工6号	2	5500	5	12325@qq.com
7	员工7号	2	1000	1	12335@qq.com
8	员工8号	3	1500	1	12315@qq.com
9	员工9号	2	2500	2	12375@qq.com

In [10]:
```
%%sql

select * from department;
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
3 rows affected.
```

Out[10]:

dep_dno	dep_dname	dep_manager	dep_budget	dep_cost
1	销售部	7	10000	6700
2	财务部	3	20000	12500
3	人事部	5	15000	9000

3、添加check约束限制取值范围

各约束描述如下：

外键约束：

emp_dno:需要存在于department表中的dep_dno中
dep_mananger:需要存在于employees表中的emp_eno中

格式约束：

emp_email:正则表达式格式

范围限制：

dep_dname:要求是{'销售部'、'财务部'，'人事部'}之一
emp_level:取值为1到5的整数
emp_salary: emp_salary的取值范围为[emp_level * 1000, emp_level * 1000 +1000],同时一个

部门所有员工的salary总和不能超过该部门的budget

In [11]:
```
%%sql


ALTER TABLE employees ADD CONSTRAINT FK_emp_d FOREIGN KEY (emp_dno)
REFERENCES department(dep_dno);
#添加外键约束，manager存在于eno中
ALTER TABLE department ADD CONSTRAINT FK_dep_m FOREIGN KEY (dep_manager)
REFERENCES employees(emp_eno);
#email限制正则表达式格式
ALTER TABLE employees ADD CONSTRAINT CK_email CHECK
(emp_email REGEXP '^[A-Za-z0-9\u4e00-\u9fa5]+@[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)+$');
#level限制1-5之间
ALTER TABLE employees ADD CONSTRAINT CK_level CHECK
(emp_level >= 1 and emp_level <= 5);
#salary为正整数,且受到level的限制
ALTER TABLE employees ADD CONSTRAINT CK_salary CHECK
(emp_salary >= 1000*emp_level and emp_salary <= (1000*emp_level+1000));
#budget为正整数,且高于已使用的预算值
ALTER TABLE department ADD CONSTRAINT CK_budget CHECK
(dep_budget >= dep_cost);
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
10 rows affected.
3 rows affected.
10 rows affected.
10 rows affected.
10 rows affected.
3 rows affected.
```

Out[11]: []

4、对添加的约束条件进行验证

(1)验证emp_dno是否受dep_dno约束:

输入超出范围的dno值会报错

In [12]:
```
%%sql

INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (11,'员工11号',5,1700,1,'22346@qq.com');
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306

---------------------------------------------------------------------------
IntegrityError                            Traceback (most recent call last)
/usr/local/share/anaconda3/envs/py37/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_context(self, dialect, constructor, statement, parameters, execution_options, *args, **kw)
   1808                     self.dialect.do_execute(
-> 1809                         cursor, statement, parameters, context
   1810                     )

/usr/local/share/anaconda3/envs/py37/lib/python3.7/site-packages/sqlalchemy/engine/default.py in do_execute(self, cursor, statement, parameters, context)
    731     def do_execute(self, cursor, statement, parameters, context=None):
--> 732         cursor.execute(statement, parameters)
    733

/usr/local/share/anaconda3/envs/py37/lib/python3.7/site-packages/pymysql/cursors.py in execute(self, query, args)
    147
```

dno范围内的值可以被正常插入

In [13]:
```
%%sql
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (11,'员工11号',1,1700,1,'22346@qq.com');
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
1 rows affected.
```

Out[13]: []

(2)验证dep_manager是否受emp_eno约束:

输入超出范围的dep_manager值会报错

In [14]:
```
%%sql

INSERT INTO department(dep_dno,dep_dname,dep_manager,dep_budget,dep_cost)
VALUES(4,'销售部',15,10000,0);
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306

---------------------------------------------------------------------------
IntegrityError                            Traceback (most recent call last)
/usr/local/share/anaconda3/envs/py37/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_context(self, dialect, constructor, statement, parameters, execution_options, *args, **kw)
   1808                     self.dialect.do_execute(
-> 1809                         cursor, statement, parameters, context
   1810                     )

/usr/local/share/anaconda3/envs/py37/lib/python3.7/site-packages/sqlalchemy/engine/default.py in do_execute(self, cursor, statement, parameters, context)
    731     def do_execute(self, cursor, statement, parameters, context=None):
--> 732         cursor.execute(statement, parameters)
    733

/usr/local/share/anaconda3/envs/py37/lib/python3.7/site-packages/pymysql/cursors.py in execute(self, query, args)
    147
```

(3)验证dep_dname是否受取值范围约束：

输入超出范围的dep_dname值会报错

In [15]:
```
%%sql

INSERT INTO department(dep_dno,dep_dname,dep_manager,dep_budget,dep_cost)
VALUES(4,'随便部',2,10000,0);
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306

---------------------------------------------------------------------------
DataError                                 Traceback (most recent call last)
/usr/local/share/anaconda3/envs/py37/lib/python3.7/site-packages/sqlalchemy/engine/base.py in _execute_context(self, dialect, constructor, statement, parameters, execution_options, *args, **kw)
   1808                     self.dialect.do_execute(
-> 1809                         cursor, statement, parameters, context
   1810                     )

/usr/local/share/anaconda3/envs/py37/lib/python3.7/site-packages/sqlalchemy/engine/default.py in do_execute(self, cursor, statement, parameters, context)
    731     def do_execute(self, cursor, statement, parameters, context=None):
--> 732         cursor.execute(statement, parameters)
    733

/usr/local/share/anaconda3/envs/py37/lib/python3.7/site-packages/pymysql/cursors.py in execute(self, query, args)
    147
```

(4)验证emp_level是否受取值范围约束：

输入超出范围的emp_level值会报错

In [16]:
```
%%sql

INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (12,'员工12号',1,1700,11,'22346@qq.com');
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
(pymysql.err.OperationalError) (3819, "Check constraint 'CK_level' is violated.")
[SQL: INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_ema
il) VALUES (12,'员工12号',1,1700,11,'22346@qq.com');]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqlalche.me/
e/14/e3q8))
```

(5)验证emp_salary是否受取值范围约束：

输入超出范围的emp_salary值会报错

In [17]:
```
%%sql

INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (13,'员工13号',1,1700,2,'32346@qq.com');
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
(pymysql.err.OperationalError) (3819, "Check constraint 'CK_salary' is violated.")
[SQL: INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_ema
il) VALUES (13,'员工13号',1,1700,2,'32346@qq.com');]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqlalche.me/
e/14/e3q8))
```

(6)验证emp_email是否符合正则表达式：

输入格式不符的emp_email值会报错

In [18]:
```
%%sql

INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (14,'员工14号',1,1700,1,'32346');
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
(pymysql.err.OperationalError) (3819, "Check constraint 'CK_email' is violated.")
[SQL: INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_ema
il) VALUES (14,'员工14号',1,1700,1,'32346');]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqlalche.me/
e/14/e3q8))
```

(7)验证一个部门的员工salary总和是否不能超过该部门的budget：

输入员工salary超出时会报错

In [19]:
```
%%sql

INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (15,'员工15号',3,1000,1,'52346@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (16,'员工16号',3,1000,1,'62346@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (17,'员工17号',3,2000,2,'72346@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (18,'员工18号',3,2000,2,'82346@qq.com');
INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_email)
VALUES (19,'员工19号',3,1700,1,'92346@qq.com');
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
(pymysql.err.OperationalError) (3819, "Check constraint 'CK_budget' is violated.")
[SQL: INSERT INTO employees(emp_eno,emp_ename,emp_dno,emp_salary,emp_level,emp_ema
il) VALUES (19,'员工19号',3,1700,1,'92346@qq.com');]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqlalche.me/
e/14/e3q8))
```

In [20]:
```
%%sql
select * from employees;
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
15 rows affected.
```

Out[20]:

emp_eno	emp_ename	emp_dno	emp_salary	emp_level	emp_email
1	员工1号	1	1500	1	12345@qq.com
2	员工2号	1	1700	1	12346@qq.com
3	员工3号	3	3000	2	12347@qq.com
4	员工4号	1	3500	3	12348@qq.com
5	员工5号	3	4500	4	12349@qq.com
6	员工6号	2	5500	5	12325@qq.com
7	员工7号	2	1000	1	12335@qq.com
8	员工8号	3	1500	1	12315@qq.com
9	员工9号	2	2500	2	12375@qq.com
10	员工10号	2	3500	3	12395@qq.com
11	员工11号	1	1700	1	22346@qq.com
15	员工15号	3	1000	1	52346@qq.com
16	员工16号	3	1000	1	62346@qq.com
17	员工17号	3	2000	2	72346@qq.com
18	员工18号	3	2000	2	82346@qq.com

In [21]:
```
%%sql
select * from department;
```

```
 * mysql://stu2000013121:***@162.105.146.37:53306
3 rows affected.
```

Out[21]:

dep_dno	dep_dname	dep_manager	dep_budget	dep_cost
1	销售部	7	10000	8400
2	财务部	3	20000	12500
3	人事部	5	15000	15000

练习二 触发器设计

创建所需要用到的表

In [20]:
```
%%sql
SET @@foreign_key_checks=0;

DROP TABLE IF EXISTS originData;

CREATE TABLE originData
(
    id SERIAL PRIMARY KEY,
    value INT NOT NULL
);

SET @@foreign_key_checks=1;
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[20]: []

In [21]:
```
%%sql
SET @@foreign_key_checks=0;

DROP TABLE IF EXISTS sum_slidingWin;

CREATE TABLE sum_slidingWin
(
    id INT PRIMARY KEY,
    value INT NOT NULL
);

SET @@foreign_key_checks=1;
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[21]: []

In [22]:
```
%%sql
SET @@foreign_key_checks=0;

DROP TABLE IF EXISTS max_slidingWin;

CREATE TABLE max_slidingWin
(
    id INT PRIMARY KEY,
    value INT NOT NULL
);

SET @@foreign_key_checks=1;
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[22]: []

In [23]:
```
%%sql
SET @@foreign_key_checks=0;

DROP TABLE IF EXISTS aggResult;

CREATE TABLE aggResult
(
    sumRes INT,
    maxRes INT
);

SET @@foreign_key_checks=1;

# 根据文档，aggResult表初始存在一行(0, 0)
INSERT INTO aggResult(sumRes, maxRes) VALUES(0, 0);
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
0 rows affected.
0 rows affected.
0 rows affected.
0 rows affected.
1 rows affected.
```

Out[23]: []

连接数据库

In [24]:
```
# 预设数据库参数
host = '162.105.146.37'
user = 'stu2000013094'
pswd = 'stu2000013094'
port = 43306
db = 'stu2000013094'

# 连接数据库
db = pymysql.connect(host=host, user=user, password=pswd, port=port, db=db)
cursor = db.cursor()
```

数据生成函数

In [25]:
```
# 向originData插入随机生成的行
# 一共生成K行，value取[lwb, upb]中随机整数
import random

def gen_row(K, lwb, upb):
    sql = '''INSERT INTO originData(value) VALUES(%s);'''

    try:
        for i in range(K):
            value = random.randint(lwb, upb)
            cursor.execute(sql, (value))
            db.commit()
    except:
        db.rollback()
```

数据清理函数

In [48]:
```
# 清理表中数据，使表回到初始状态
def clear_all():
    try:
        cursor.execute('''TRUNCATE TABLE originData;''')
        cursor.execute('''TRUNCATE TABLE sum_slidingWin;''')
        cursor.execute('''TRUNCATE TABLE max_slidingWin;''')
        cursor.execute('''TRUNCATE TABLE aggResult;''')
        cursor.execute('''INSERT INTO aggResult(sumRes, maxRes) VALUES(0, 0);''')
        db.commit()
    except:
        db.rollback()
```

1.通过触发器维护sum_slidingWin和max_slidingWin

In [26]:
```
# 创建sum_slidingWin的维护触发器
def create_sum_win(N):
    sql = '''CREATE TRIGGER sum_win AFTER INSERT
             ON originData FOR EACH ROW
             BEGIN
                 IF NEW.id >= (SELECT min(id) FROM sum_slidingWin) + %s THEN
                 BEGIN
                     DELETE FROM sum_slidingWin WHERE id + %s = NEW.id;
                     INSERT INTO sum_slidingWin(id, value) VALUES(NEW.id, NEW.value);
                 END;
                 ELSE
                     INSERT INTO sum_slidingWin(id, value) VALUES(NEW.id, NEW.value);
                 END IF;
             END;'''

    try:
        cursor.execute('''DROP TRIGGER IF EXISTS sum_win;''', ())
        cursor.execute(sql, (N, N))
        db.commit()
    except:
        db.rollback()
```

In [27]:
```
# 创建max_slidingWin的维护触发器
def create_max_win(N):
    sql = '''CREATE TRIGGER max_win AFTER INSERT
             ON originData FOR EACH ROW
             BEGIN
                 IF NEW.id >= (SELECT min(id) FROM max_slidingWin) + %s THEN
                 BEGIN
                     DELETE FROM max_slidingWin WHERE id + %s = NEW.id;
                     INSERT INTO max_slidingWin(id, value) VALUES(NEW.id, NEW.value);
                 END;
                 ELSE
                     INSERT INTO max_slidingWin(id, value) VALUES(NEW.id, NEW.value);
                 END IF;
             END;'''

    try:
        cursor.execute('''DROP TRIGGER IF EXISTS max_win;''', ())
        cursor.execute(sql, (N, N))
        db.commit()
    except:
        db.rollback()
```

2.增量更新aggResult

对于窗口未满的情况，上述触发器会进行一次INSERT操作，对于窗口已满的情况，上述触发器还会进行一次DELETE操作，我们对这两个操作设置触发器就可以完成增量更新

In [28]:
```
def create_sum_agg_ins():
    sql = '''CREATE TRIGGER sum_agg_ins AFTER INSERT
            ON sum_slidingWin FOR EACH ROW
                UPDATE aggResult SET sumRes = sumRes + NEW.value;'''

    try:
        cursor.execute('''DROP TRIGGER IF EXISTS sum_agg_ins;''', ())
        cursor.execute(sql, ())
        db.commit()
    except:
        db.rollback()
```

In [29]:
```
def create_sum_agg_del():
    sql = '''CREATE TRIGGER sum_agg_del AFTER DELETE
            ON sum_slidingWin FOR EACH ROW
                UPDATE aggResult SET sumRes = sumRes - OLD.value;'''

    try:
        cursor.execute('''DROP TRIGGER IF EXISTS sum_agg_del;''', ())
        cursor.execute(sql, ())
        db.commit()
    except:
        db.rollback()
```

In [30]:
```
def create_max_agg_ins():
    sql = '''CREATE TRIGGER max_agg_ins AFTER INSERT
            ON max_slidingWin FOR EACH ROW
            BEGIN
                IF NEW.value > (SELECT maxRes FROM aggResult) THEN
                    UPDATE aggResult SET maxRes = NEW.value;
                END IF;
            END;'''

    try:
        cursor.execute('''DROP TRIGGER IF EXISTS max_agg_ins;''', ())
        cursor.execute(sql, ())
        db.commit()
    except:
        db.rollback()
```

In [31]:
```
def create_max_agg_del():
    sql = '''CREATE TRIGGER max_agg_del AFTER DELETE
             ON max_slidingWin FOR EACH ROW
             BEGIN
                 IF OLD.value = (SELECT maxRes FROM aggResult) THEN
                     UPDATE aggResult SET maxRes =
                         (SELECT max(value) FROM max_slidingWin);
                 END IF;
             END;'''

    try:
        cursor.execute('''DROP TRIGGER IF EXISTS max_agg_del;''', ())
        cursor.execute(sql, ())
        db.commit()
    except:
        db.rollback()
```

至此，本练习的基础功能已经完成，可以进行简单的测试

In [55]:
```
# 测试参数
N = 20
K = 1000
lwb = 1
upb = 100

# 表和触发器初始化
clear_all()
create_sum_win(N)
create_max_win(N)
create_sum_agg_ins()
create_sum_agg_del()
create_max_agg_ins()
create_max_agg_del()

# 初始化检验
try:
    cursor.execute('''SELECT * FROM aggResult;''', ())
    print(cursor.fetchall()[0])
    db.commit()
except:
    db.rollback()

# 生成数据，查看结果
gen_row(K, lwb, upb)
try:
    cursor.execute('''SELECT count(*) FROM originData;''', ())
    print(cursor.fetchall()[0][0])
    cursor.execute('''SELECT * FROM sum_slidingWin;''', ())
    print(cursor.fetchall())
    cursor.execute('''SELECT * FROM max_slidingWin;''', ())
    print(cursor.fetchall())
    cursor.execute('''SELECT * FROM aggResult;''', ())
    print(cursor.fetchall()[0])
    db.commit()
except:
    db.rollback()
```

```
(0, 0)
1000
((981, 91), (982, 24), (983, 34), (984, 89), (985, 91), (986, 95), (987, 91), (98
8, 20), (989, 4), (990, 66), (991, 95), (992, 78), (993, 17), (994, 69), (995, 6),
(996, 59), (997, 3), (998, 83), (999, 77), (1000, 79))
((981, 91), (982, 24), (983, 34), (984, 89), (985, 91), (986, 95), (987, 91), (98
8, 20), (989, 4), (990, 66), (991, 95), (992, 78), (993, 17), (994, 69), (995, 6),
(996, 59), (997, 3), (998, 83), (999, 77), (1000, 79))
(1171, 95)
```

3.维护最小的max_slidingWin

对于一个最小的max_slidingWin而言，当一个新行到来时，以下两类行都要被删除：
1.过期的行，即id + N <= NEW.id的行
2.太小的行，即value <= NEW.value的行，这样的行在过期前永远不会成为窗口内的最大值
我们可以修改max_win触发器来实现这样的功能

In [32]:
```
# 创建修改后的max_slidingWin的维护触发器
# 由于可能出现一个极大的值清空其他所有max_slidingWin的行
# 需要先INSERT后DELETE以免聚合函数出错
def create_max_win_modified(N):
    sql = '''CREATE TRIGGER max_win AFTER INSERT
             ON originData FOR EACH ROW
             BEGIN
                 INSERT INTO max_slidingWin(id, value)
                     VALUES(NEW.id, NEW.value);
                 DELETE FROM max_slidingWin WHERE id + %s <= NEW.id;
                 DELETE FROM max_slidingWin
                     WHERE id < NEW.id AND value <= NEW.value;
             END;'''

    try:
        cursor.execute('''DROP TRIGGER IF EXISTS max_win;''')
        cursor.execute(sql, (N))
        db.commit()
    except:
        db.rollback()
```

In [56]:
```
# 测试参数
N = 20
K = 1000
lwb = 1
upb = 100

# 表和触发器初始化
clear_all()
create_sum_win(N)
create_max_win_modified(N)
create_sum_agg_ins()
create_sum_agg_del()
create_max_agg_ins()
create_max_agg_del()

# 初始化检验
try:
    cursor.execute('''SELECT * FROM aggResult;''', ())
    print(cursor.fetchall()[0])
    db.commit()
except:
    db.rollback()


# 生成数据，查看结果
gen_row(K, lwb, upb)
try:
    cursor.execute('''SELECT count(*) FROM originData;''', ())
    print(cursor.fetchall()[0][0])
    cursor.execute('''SELECT * FROM sum_slidingWin;''', ())
    print(cursor.fetchall())
    cursor.execute('''SELECT * FROM max_slidingWin;''', ())
    print(cursor.fetchall())
    cursor.execute('''SELECT * FROM aggResult;''', ())
    print(cursor.fetchall()[0])
    db.commit()
except:
    db.rollback()
```

```
(0, 0)
1000
((981, 47), (982, 67), (983, 51), (984, 49), (985, 97), (986, 15), (987, 70), (98
8, 86), (989, 78), (990, 36), (991, 10), (992, 2), (993, 93), (994, 64), (995, 8
7), (996, 98), (997, 40), (998, 72), (999, 74), (1000, 54))
((996, 98), (999, 74), (1000, 54))
(1190, 98)
```

4.调整N的大小，查看性能影响

In [54]:
```
# 取N=5,10,...,50，对每个N，生成iter次数据，每次batch_size行
# 每个iter计算一次max_slidingWin的大小，取平均输出
import numpy as np

# 测试参数
batch_size = 10
iter = 100
lwb = 1
upb = 100

# 测试
result = []
for N in range(5, 51, 5):
    # 表和触发器初始化
    clear_all()
    clear_all()
    create_sum_win(N)
    create_max_win_modified(N)
    create_sum_agg_ins()
    create_sum_agg_del()
    create_max_agg_ins()
    create_max_agg_del()

    # iter轮数据生成
    temp = []
    for i in range(iter):
        gen_row(batch_size, lwb, upb)
        try:
            cursor.execute('''SELECT count(*) FROM max_slidingWin;''')
            temp.append(cursor.fetchall()[0][0])
            db.commit()
        except:
            db.rollback()

    # 计算平均值
    result.append(np.mean(temp)/N)

# 输出结果
print(result)
```

```
[0.41600000000000004, 0.28300000000000003, 0.228, 0.184, 0.1436, 0.12966666666666668, 0.12085714285714287, 0.10800000000000001, 0.08822222222222223, 0.08]
```

调试工具

这部分代码是在debug过程中使用的，其结果未及时更新

In [16]:
```
%%sql
SELECT trigger_name, action_statement FROM information_schema.triggers
WHERE (event_object_table = 'originData'
       OR event_object_table = 'sum_slidingWin'
       OR event_object_table = 'max_slidingWin');
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
6 rows affected.
```

Out[16]:

TRIGGER_NAME	ACTION_STATEMENT

sum_win	BEGIN
	IF NEW.id >= (SELECT min(id) FROM sum_slidingWin) + 20 THEN
	BEGIN
	DELETE FROM sum_slidingWin WHERE id + 20 = NEW.id;
	INSERT INTO sum_slidingWin(id, value) VALUES(NEW.id, NEW.value);
	END;
	ELSE
	INSERT INTO sum_slidingWin(id, value) VALUES(NEW.id, NEW.value);
	END IF;
	END

max_win	BEGIN
	IF NEW.id >= (SELECT min(id) FROM max_slidingWin) + 20 THEN
	BEGIN
	DELETE FROM max_slidingWin WHERE id + 20 = NEW.id;
	INSERT INTO max_slidingWin(id, value) VALUES(NEW.id, NEW.value);
	END;
	ELSE
	INSERT INTO max_slidingWin(id, value) VALUES(NEW.id, NEW.value);
	END IF;
	END

sum_agg_ins	UPDATE aggResult SET sumRes = sumRes + NEW.value
sum_agg_del	UPDATE aggResult SET sumRes = sumRes - OLD.value

max_agg_ins	BEGIN
	IF NEW.value > (SELECT maxRes FROM aggResult) THEN
	UPDATE aggResult SET maxRes = NEW.value;
	END IF;
	END

max_agg_del	BEGIN
	IF OLD.value = (SELECT maxRes FROM aggResult) THEN
	UPDATE aggResult SET maxRes = (SELECT max(value) FROM max_slidingWin);
	END IF;
	END

用sql直接插入一行，便于debug

In [23]:
```
%%sql
INSERT INTO originData(value) VALUES(12);
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
1 rows affected.
```

Out[23]: []

In [50]:
```
%%sql
SELECT * FROM originData;
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
0 rows affected.
```

Out[50]:

id	value

In [51]:
```
%%sql
SELECT * FROM sum_slidingWin;
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
0 rows affected.
```

Out[51]:

id	value

In [52]:
```
%%sql
SELECT * FROM max_slidingWin;
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
0 rows affected.
```

Out[52]:

id	value

In [53]:
```
%%sql
SELECT * FROM aggResult;
```

```
 * mysql://stu2000013094:***@162.105.146.37:43306
1 rows affected.
```

Out[53]:

sumRes	maxRes
0	0

In [ ]: