

数据库课内实验

计算机93	李云广	2193712575
-------	-----	------------

1. 实验要求

- 1. 在openGauss中创建MYDB数据库，并在MYDB中创建学生、课程、选课三个表。
- 2. 将相应数据加入相应的表中。
- 3. 完成相应的增删改查操作。
- 4. 生成数据并插入数据库中。
- 5. 恢复其他同学的数据库，并简单分析。

2. 实验过程

2.0 前奏

打开数据库

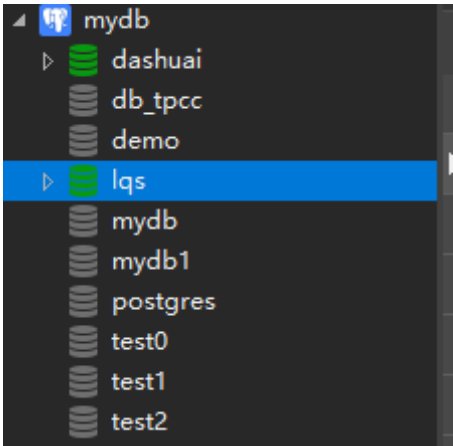
使用虚拟机安装openEuler系统，按照指导书要求安装openGuass数据库，安装完成后尝试开始使用。

```
1 su - omm
2 gs_om -t start
```

使用MobaXterm连接虚拟机并开启数据库：

```
[omm@lygdb ~]$ gs_om -t start
Starting cluster.
=====
Successfully started.
[omm@lygdb ~]$ gs_om -t status
-----
cluster_name      : dbCluster
cluster_state     : Normal
redistributing    : No
-----
[omm@lygdb ~]$
```

新建用户my_root，并使用navicat连接数据库：



在建立表之前想知道opengauss的数据类型特点

```
test0=> create table
t1(k char(1) not null,
test0(> k2 varchar(1) not null);
CREATE TABLE
test0=>
test0=>
test0=>
test0=> insert into t1 values('a', 'b');
INSERT 0 1
test0=> insert into t1 values('a', 'bc');
ERROR:  value too long for type character varying(1)
CONTEXT:  referenced column: k2
test0=> insert into t1 values('aa', 'bb');
ERROR:  value too long for type character(1)
CONTEXT:  referenced column: k
```

char(n)表示存n个字符，而且对于英文来说的。

```
test0=> insert into t1 values('李','a');
ERROR:  value too long for type character(1)
CONTEXT:  referenced column: k
test0=>
```

一般使用utf-8编码的话，每个汉字占三个字节，也就是3个字符？

我们来尝试一下

```
test0=> create table t2(k1 char(2), k2 char(3), k3 ch
ar(4), k4 char(5), k5 char(6));
CREATE TABLE
test0=> insert into t2 values('李')
test0-> ;
ERROR:  value too long for type character(2)
CONTEXT:  referenced column: k1
```

发现两个字符空间并不能存储一个汉字

```
test0=> insert into t2 values(null, '李')
;
INSERT 0 1
```

三个字符空间却成功了，说明一个汉字就占三个字节。

```
test0=> insert into t2 values(null, '李',null,null,'
李云')
;
INSERT 0 1
```

2.1 建表

先建立一个数据库mydb1。

```
1 | create database mydb1;
```

建表要考虑到数据结构的使用

对于学号要使用char，对于姓名可以使用varchar(150)考虑到有些外国人名字长，性别一般只有一个汉字可以使用char(3)，生日使用Date，身高使用float(2)，宿舍使用 $5 + 3 + 3 = 11$ 即可。

课程号使用char(5)，课程名最长为77，使用varchar(77)即可，学时最长为480，所以使用smallint(-32768 ~ 32768)即可，学分最多为50学分，并且可以为小数，使用numeric(3, 1)即可，教师名直接使用varchar(150)。

对于成绩，要比较精确，直接使用numeric(6, 3)。

```
1 | CREATE TABLE S575 (  
2 |     Sno CHAR(8) NOT NULL,  
3 |     SNAME VARCHAR(150) NOT NULL,  
4 |     SEX CHAR(3) NOT NULL,  
5 |     BDATE DATE NOT NULL,  
6 |     HEIGHT FLOAT(2) DEFAULT 0.0,  
7 |     DORM CHAR(15),  
8 |     PRIMARY KEY (Sno)  
9 | );  
10 | CREATE TABLE C575 (  
11 |     Cno CHAR(5) NOT NULL,  
12 |     CNAME VARCHAR(77) NOT NULL,  
13 |     PERIOD SMALLINT NOT NULL,  
14 |     CREDIT NUMERIC(3, 1) NOT NULL,  
15 |     TEACHER VARCHAR(150) NOT NULL,  
16 |     PRIMARY KEY (Cno)  
17 | );  
18 | CREATE TABLE SC575 (  
19 |     Sno CHAR(8) NOT NULL,  
20 |     Cno CHAR(5) NOT NULL,  
21 |     GRADE NUMERIC(6, 3) DEFAULT NULL,  
22 |     PRIMARY KEY (Cno, Sno),  
23 |     FOREIGN KEY (Sno)  
24 |         REFERENCES S575 (Sno)  
25 |         ON DELETE CASCADE,  
26 |     FOREIGN KEY (Cno)  
27 |         REFERENCES C575 (Cno)  
28 |         ON DELETE RESTRICT  
29 | );
```

```


mydb1=# CREATE TABLE S575 (
    Sno CHAR(8) NOT NULL,
    SNAME VARCHAR(150) NOT NULL,
    SEX CHAR(3) NOT NULL,
    BDATE DATE NOT NULL,
    HEIGHT FLOAT(2) DEFAULT 0.0,
    DORM CHAR(15),
    PRIMARY KEY (Sno)
);
CREATE TABLE C575 (
    Cno CHAR(5) NOT NULL,
    CNAME VARCHAR(77) NOT NULL,
    PERIOD SMALLINT NOT NULL,
    CREDIT NUMERIC(3 , 1 ) NOT NULL,
    TEACHER VARCHAR(150) NOT NULL,
    PRIMARY KEY (Cno)
);
CREATE TABLE SC575 (
    Sno CHAR(8) NOT NULL,
    Cno CHAR(5) NOT NULL,
    GRADE NUMERIC(6 , 3 ) DEFAULT NULL,
    PRIMARY KEY (Cno , Sno),
    FOREIGN KEY (Sno)
        REFERENCES S575 (Sno)
        ON DELETE CASCADE,
    FOREIGN KEY (Cno)
        REFERENCES C575 (Cno)
        ON DELETE RESTRICT
);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "s575_pkey" for table "s575"
CREATE TABLE

NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "c575_pkey" for table "c575"
CREATE TABLE


NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "sc575_pkey" for table "sc575"
CREATE TABLE

```



2.1.1 S表

名	类型	长度	小数点	不是 null	键	注释
▶ sno	char	8	0	<input checked="" type="checkbox"/>	 1	
sname	varchar	150	0	<input checked="" type="checkbox"/>		
sex	char	3	0	<input checked="" type="checkbox"/>		
bdate	timestamp	0	0	<input checked="" type="checkbox"/>		
height	float4	24	0	<input type="checkbox"/>		
dorm	char	15	0	<input type="checkbox"/>		

2.1.2 C表

名	类型	长度	小数点	不是 null	键
▶ cno	char	5	0	<input checked="" type="checkbox"/>	 1
cname	varchar	77	0	<input checked="" type="checkbox"/>	
period	int2	16	0	<input checked="" type="checkbox"/>	
credit	numeric	3	1	<input checked="" type="checkbox"/>	
teacher	varchar	150	0	<input checked="" type="checkbox"/>	

2.1.3 SC表

名	类型	长度	小数点	不是 null	键
I sno	char	8	0	<input checked="" type="checkbox"/>	 2
cno	char	5	0	<input checked="" type="checkbox"/>	 1
grade	numeric	6	3	<input type="checkbox"/>	

2.2 插入数据

这里使用一种较笨的方法插入数据，使用python的字符串操作生成insert语句，写入数据库中：

```
>>>
===== RESTART: D:\XJTU\大三下专业课\数据库实验\实验过程\python\S.py =====
insert into S575 values('01032010','王涛','男','2002-4-5',1.72,'东14舍221')
insert into S575 values('01032023','孙文','男','2003-6-10',1.80,'东14舍221')
insert into S575 values('01032001','张晓梅','女','2003-11-17',1.58,'东1舍312')
insert into S575 values('01032005','刘静','女','2002-1-10',1.63,'东1舍312')
insert into S575 values('01032112','董蔚','男','2002-2-20',1.71,'东14舍221')
insert into S575 values('03031011','王倩','女','2003-12-20',1.66,'东2舍104')
insert into S575 values('03031014','赵思扬','男','2001-6-6',1.85,'东18舍421')
insert into S575 values('03031051','周剑','男','2001-5-8',1.68,'东18舍422')
insert into S575 values('03031009','田婷','女','2002-8-11',1.60,'东2舍104')
insert into S575 values('03031033','蔡明明','男','2002-3-12',1.75,'东18舍423')
insert into S575 values('03031056','曹子衿','女','2003-12-15',1.65,'东2舍305')
insert into C575 values('CS-01','数据结构',60,3,'张军')
insert into C575 values('CS-02','计算机组成原理',80,4,'王亚伟')
insert into C575 values('CS-04','人工智能',40,2,'李蕾')
insert into C575 values('CS-05','深度学习',40,2,'崔均')
insert into C575 values('EE-01','信号与系统',60,3,'张明')
insert into C575 values('EE-02','数字逻辑电路',100,5,'胡海东')
insert into C575 values('EE-03','光电子学与光子学',40,2,'石韬')
insert into SC575 values('01032010','CS-01',82.0)
insert into SC575 values('01032010','CS-02',91.0)
```

检查数据，发现插入成功。

mydb mydb1

运行 停止 解释

1 select * from s575;

信息 结果 1

sno	sname	sex	bdate	height	dorm
01032010	王涛	男	2002-04-05 00:0	1.72	东14舍221
01032023	孙文	男	2003-06-10 00:0	1.8	东14舍221
01032001	张晓梅	女	2003-11-17 00:0	1.58	东1舍312
01032005	刘静	女	2002-01-10 00:0	1.63	东1舍312
01032112	董蔚	男	2002-02-20 00:0	1.71	东14舍221
03031011	王倩	女	2003-12-20 00:0	1.66	东2舍104
03031014	赵思扬	男	2001-06-06 00:0	1.85	东18舍421
03031051	周剑	男	2001-05-08 00:0	1.68	东18舍422
03031009	田婷	女	2002-08-11 00:0	1.6	东2舍104
03031033	蔡明明	男	2002-03-12 00:0	1.75	东18舍423
03031056	曹子衿	女	2003-12-15 00:0	1.65	东2舍305

2.3 增删改查操作

2.3.1 查询

(1)

查询电子工程系（EE）所开课程的课程编号、课程名称及学分数。

```
1 SELECT
2     Cno,
3     CNAME,
4     CREDIT
5 FROM
6     c575
7 WHERE
8     Cno LIKE 'EE%';
```

信息		结果 1	
	cno	cname	credit
▶	EE-01	信号与系统	3.0
	EE-02	数字逻辑电路	5.0
	EE-03	光电子学与光子学	2.0

(2)

查询未选课程“CS-01”的女生学号及其已选各课程编号、成绩。

```
1 SELECT
2     sc1.Sno,
3     sc1.Cno,
4     sc1.Grade
5 FROM
6     sc575 sc1
7 WHERE
8     NOT EXISTS (
9         SELECT
10             *
11         FROM
12             sc575 sc2,
13             s575 s
14         WHERE
15             s.SEX = '男'
16             AND s.Sno = sc1.Sno
17             OR sc2.Cno = 'CS-01'
18             AND sc1.Sno = sc2.Sno
19     ) UNION
20 SELECT
21     s.Sno,
22     NULL,
23     NULL
24 FROM
25     s575 s
```

```

26 WHERE
27     s.SEX = '女'
28     AND s.sno NOT IN ( SELECT sno FROM sc575 );

```

这里要注意有一部分女生是一门课都没有选的！

sno	cno	grade
03031011	EE-02	86.000
03031011	EE-01	91.000
▶ 03031009	EE-01	88.000
03031009	EE-02	78.500
03031056	(Null)	(Null)

(3)

查询2000年～2001年出生的学生的基本信息。

```

1 SELECT
2     *
3 FROM
4     s575
5 WHERE
6     BDATE LIKE '2001%'
7     OR Bdate LIKE '2000%';

```

信息		结果 1				
sno	sname	sex	bdate	height	dorm	
▶ 03031014	赵思扬	男	2001-06-06 00:0	1.85	东18舍421	
03031051	周剑	男	2001-05-08 00:0	1.68	东18舍422	

(4)

查询每位学生的学号、学生姓名及其已选课程的学分总数。

```

1 SELECT
2     s575.sno,
3     sname,
4     SUM ( credit ) AS sum_credit
5 FROM
6     s575,
7     c575,
8     sc575
9 WHERE
10    s575.sno = sc575.sno
11    AND c575.cno = sc575.cno
12 GROUP BY
13    s575.sno
14 UNION
15
16 SELECT

```

```

17      s575.sno,
18      sname,
19      0 AS sum_credit
20 FROM
21      ( s575 LEFT OUTER JOIN sc575 ON ( s575.sno = sc575.sno ) )
22 WHERE
23      sc575.sno IS NULL;

```

信息	结果 1	
sno	sname	sum_credit
▶ 01032023	孙文	9
03031033	蔡明明	8
01032010	王涛	9
03031011	王倩	8
03031051	周剑	8
01032005	刘静	9
03031056	曹子衿	0
01032001	张晓梅	9
03031014	赵思扬	8
01032112	董蔚	11
03031009	田婷	8

这里注意有一个没有选任何课程的同学，所以在后面多加了一个unoin。

(5)

查询选修课程“CS-02”的学生中成绩第二高的学生学号。

```

1  SELECT
2      sno
3  FROM
4      sc575 sc1
5  WHERE
6      sc1.cno = 'CS-02'
7      AND sc1.grade = ( SELECT grade FROM sc575 sc2 WHERE sc2.cno = 'CS-02'
                        ORDER BY grade DESC LIMIT 1, 1 );

```

信息	结果 1
sno	
01032010	

(6)

查询平均成绩超过“王涛”同学的学生学号、姓名和平均成绩，并按学号进行降序排列。

```
1 SELECT
2     s1.sno,
3     s1.sname,
4     AVG ( sc1.grade ) AS avg_grade
5 FROM
6     s575 s1,
7     sc575 sc1
8 WHERE
9     s1.sno = sc1.sno
10 GROUP BY
11     s1.sno
12 HAVING
13     avg_grade > ALL (
14         SELECT AVG
15             ( sc2.grade ) AS avg2
16         FROM
17             sc575 sc2,
18             s575 s2
19         WHERE
20             sc2.sno = s2.sno
21             AND s2.sname = '王涛'
22         GROUP BY
23             s2.sno
24     )
25 ORDER BY
26     s1.sno DESC;
```

29

3

信息	结果 1
----	------

sno	sname	avg_grade
▶ 03031033	蔡明明	91.0000000000000000
03031011	王倩	88.5000000000000000
01032112	董蔚	88.5000000000000000

这里考虑是不是有重名的王涛，所以使用了一个 `> all`。

降序要使用一个DESC。

(7)

查询选修了计算机专业全部课程（课程编号为“CS-xx”）的学生姓名及已获得的学分总数。

```
1 SELECT
2     s1.sname,
3     SUM ( credit )
4 FROM
5     c575 c1,
6     s575 s1,
7     sc575 sc1
8 WHERE
9     c1.cno = sc1.cno
```

```

10     AND sc1.sno = s1.sno
11     AND NOT EXISTS (
12         SELECT
13             *
14         FROM
15             c575 c2
16         WHERE
17             c2.cno LIKE 'CS-%'
18             AND NOT EXISTS ( SELECT * FROM sc575 sc3 WHERE sc3.cno = c2.cno AND
sc3.sno = s1.sno )
19     )
20     AND sc1.grade > 60
21 GROUP BY
22     s1.sno;

```

信息	结果 1
sname	sum
董蔚	9

(8)

查询选修了3门以上课程（包括3门）的学生中平均成绩最高的同学学号及姓名。

```

1  SELECT
2      s1.sno,
3      s1.sname
4  FROM
5      s575 s1,
6      sc575 sc1
7  WHERE
8      s1.sno = sc1.sno
9  GROUP BY
10     s1.sno
11  HAVING
12     COUNT ( * ) >= 3
13     AND AVG ( sc1.GRADE ) >= (
14         SELECT MAX
15             ( avg_grade )
16         FROM
17             ( SELECT AVG ( sc2.GRADE ) AS avg_grade FROM sc575 sc2 GROUP BY sc2.sno
18             HAVING COUNT ( * ) >= 3 ) AS table0
19     );

```

sno	sname
01032112	董蔚

2.3.2 添加、删除、修改记录

1. 分别在S×××和C×××表中加入记录('01032005', '刘竞', '男', '1993-12-10', 1.75, '东14舍312')及('CS-03', '离散数学', 64, 4, '陈建明')。

```
1 insert into s575 values ('01032005','刘竞','男','1993-12-10','1.75','东14舍312');
2 insert into c575 values ('CS-03','离散数学',64,4,'陈建明');
```

```
信息
insert into s575 values ('01032005','刘竞','男','1993-12-10','1.75','东14舍312')
> ERROR: duplicate key value violates unique constraint "s575_pkey"
DETAIL: Key (sno)=(01032005) already exists.
> 时间: 0.005s
```

刘竞的学号已经存在了，并且作为主键，所以不能插入。

第二句插入成功：

```
信息
insert into c575 values ('CS-03','离散数学',64,4,'陈建明')
> Affected rows: 1
> 时间: 0.001s
```

2. 将S×××表中已修学分数大于60的学生记录删除。

```
1 delete from S575
2 where Sno in
3 (select SC575.Sno
4 from SC575,C575
5 where SC575.Cno = C575.Cno
6 and SC575.Grade is not null
7 group by Sno
8 having sum(C575.Credit) > 60
9 );
```

```
信息
delete from S575
where Sno in
(select SC575.Sno
from SC575,C575
where SC575.Cno = C575.Cno
and SC575.Grade is not null
group by Sno
having sum(C575.Credit) > 60
)
> Affected rows: 0
> 时间: 0.001s
```

3. 将“张明”老师负责的“信号与系统”课程的学时数调整为64，同时增加一个学分。

```
1 update C575
2 set Credit = Credit + 1, PERIOD = 64
3 where Cname = '信号与系统'
4 and Teacher = '张明';
```

信息

```
update C575
set Credit = Credit + 1, PERIOD = 64
where Cname = '信号与系统'
and Teacher = '张明'
> Affected rows: 1
> 时间: 0.003s
```

2.3.3 视图操作

1. 居住在“东18舍”的男生视图，包括学号、姓名、出生日期、身高等属性。

```
1 create view dong_18_she_nan as
2 select *
3 from s575
4 where s575.DORM like '东18舍%' and sex = '男';
```

信息

```
create view dong_18_she_nan as
select *
from s575
where s575.DORM like '东18舍%' and sex = '男'
> OK
> 时间: 0.011s
```

```
6 select * from dong_18_she_nan;
```

信息

结果 1

sno	sname	sex	bdate	height	dorm
03031014	赵思扬	男	2001-06-06 00:0	1.85	东18舍421
03031051	周剑	男	2001-05-08 00:0	1.68	东18舍422
03031033	蔡明明	男	2002-03-12 00:0	1.75	东18舍423

2. “张明”老师所开设课程情况的视图，包括课程编号、课程名称、平均成绩等属性。

```
1 create view zhangming_c as
2 select c575.cno, c575.cname, avg(grade) as avg_grade
3 from c575, sc575
4 where
5 c575.cno = sc575.cno and
6 teacher = '张明'
7 group by sc575.cno
8 ;
```

```
create view zhangming_c as
select c575.cno, c575.cname, avg(grade) as avg_grade
from c575, sc575
where
c575.cno = sc575.cno and
teacher = '张明'
group by c575.cno
> OK
> 时间: 0.008s
```

11 select * from zhangming_c;		
信息	结果 1	
cno	cname	avg_grade
▶ EE-01	信号与系统	85.8000000000000000

3. 所有选修了“人工智能”课程的学生视图，包括学号、姓名、成绩等属性。

```

1 create view renzhi_s as
2 select s575.sno, s575.sname, sc575.grade
3 from sc575, s575, c575
4 where sc575.sno = s575.sno
5 and sc575.cno = c575.cno
6 and c575.cname = '人工智能';

```

```

create view renzhi_s as
select s575.sno, s575.sname, sc575.grade
from sc575, s575, c575
where sc575.sno = s575.sno
and sc575.cno = c575.cno
and c575.cname = '人工智能'
> OK
> 时间: 0.003s

```

8 select * from renzhi_s;		
信息	结果 1	
sno	sname	grade
▶ 01032010	王涛	83.5
01032001	张晓梅	83.0
01032005	刘静	82.0
01032023	孙文	76.0
01032112	董蔚	86.0

2.4 生成随机数据并插入并分析效率

对于课程，我使用Python对于教务处的全校课程表进行了爬取，获得了本学期学校的所有课程信息，共计3910条，但是问题是学校课程记录的CK好像不仅有Cno一个，例如这门国防教育：

```

MILI100554##国防教育##32.0##2.0##问鸿滨,闫忠林,张赞
MILI100554##国防教育##32.0##2.0##初阔林,李科,张昊
MILI100554##国防教育##32.0##2.0##徐宇春,刘玉青,王志朋
MILI100554##国防教育##32.0##2.0##闫忠林,问鸿滨,张赞
MILI100554##国防教育##32.0##2.0##李科,初阔林,张昊
MILI100554##国防教育##32.0##2.0##刘玉青,王志朋,徐宇春
MILI100554##国防教育##32.0##2.0##张赞,问鸿滨,闫忠林
MILI100554##国防教育##32.0##2.0##张昊,初阔林,李科
MILI100554##国防教育##32.0##2.0##王志朋,徐宇春,刘玉青
MILI100554##国防教育##32.0##2.0##问鸿滨,闫忠林,张赞
MILI100554##国防教育##32.0##2.0##初阔林,李科,张昊

```

他们的CK都是一样的，于是为了契合本次实验的要求，我对于这些相同Cno的课程均使用第一条记录作为此课程记录。

2.4.2 第一次插入 + 效率分析

```

连接数据库...
5月 31, 2022 7:34:27 下午 org.postgresql.core.v3.ConnectionFactoryImpl openConnectionImpl
信息: [d7fef935-39e3-401b-9470-d08961b42745] Try to connect. IP: 192.168.56.102:26000
5月 31, 2022 7:34:27 下午 org.postgresql.core.v3.ConnectionFactoryImpl openConnectionImpl
信息: [192.168.56.1:57997/192.168.56.102:26000] Connection is established. ID: d7fef935-39e3-401b-9470-d08961b42745
5月 31, 2022 7:34:27 下午 org.postgresql.core.v3.ConnectionFactoryImpl openConnectionImpl
信息: Connect complete. ID: d7fef935-39e3-401b-9470-d08961b42745
实例化Statement对象...
yes
9000
49505
Goodbye!

```

```

+ - ✓ ✕ C ■
select * from sc575
查询时间: 0.021s
第 1 条记录 (共 9026 条)

```

插入成功！

查询(5)

写法1

```

1 SELECT
2     sno
3 FROM
4     sc575 sc1
5 WHERE
6     sc1.cno = 'CS-02'
7     AND sc1.grade = ( SELECT grade FROM sc575 sc2 WHERE sc2.cno = 'CS-02'
                        ORDER BY grade DESC LIMIT 1, 1 );

```

已知通过explain可以分析一个sql语句的优劣，如下：

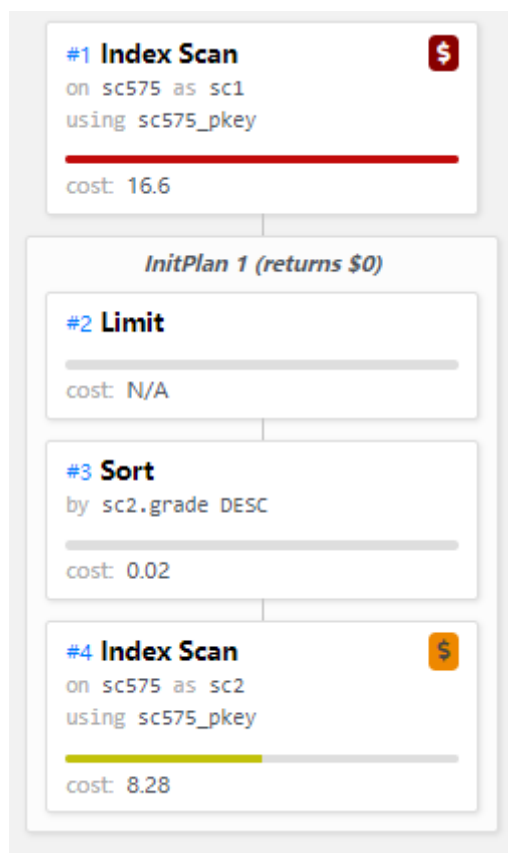
```

ORDER BY grade DESC LIMIT 1, 1);
                                QUERY PLAN
-----
Index Scan using sc575_pkey on sc575 sc1 (cost=8.30..16.58 rows=1 width=9)
  Index Cond: (cno = 'CS-02'::bpchar)
  Filter: (grade = $0)
  InitPlan 1 (returns $0)
    -> Limit (cost=8.30..8.30 rows=1 width=6)
      -> Sort (cost=8.29..8.30 rows=1 width=6)
          Sort Key: sc2.grade DESC
          -> Index Scan using sc575_pkey on sc575 sc2 (cost=0.00..8.28
rows=1 width=6)
              Index Cond: (cno = 'CS-02'::bpchar)
(9 rows)

EXPLAIN
mydb1=#

```

为了让explain的结果更加清晰，我们这里使用expalin.dalibo.com来辅助我们分析：



可以看到这个语句，自底向上依次分析：

1. 先使用Index scan，利用主键上的索引查找CS02课程的记录，cost有8.28。
2. 再使用top-N sort对grade排序，找到最大的两条记录，cost很低，估计并没有对所有的记录进行排序。
3. 使用limit函数找到第二大的记录，没有cost。
4. 再使用index scan，利用索引找到所有成绩与第二高成绩相同的记录，cost最高为16.6，因为需要对每条记录进行判断。

写法2

```

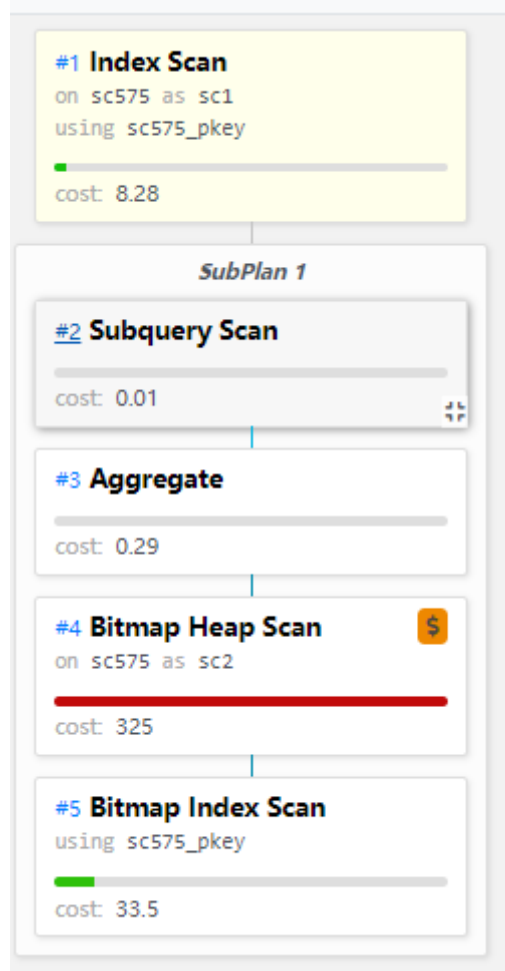
1 | SELECT
2 |   sno
3 | FROM
4 |   sc575 sc1
5 | WHERE

```

```

6      sc1.cno = 'CS-02'
7      AND EXISTS (
8      SELECT
9          *
10     FROM
11         ( SELECT COUNT ( * ) AS count_hi FROM sc575 sc2 WHERE sc2.cno =
sc1.cno AND sc2.grade > sc1.grade )
12     WHERE
13         count_hi = 1
14 );

```



这个语句的执行过程：

1. Bitmap Index scan对sc1进行索引扫描，cost为33.5。
2. Bitmap Heap Scan对sc2进行扫描（一次性将满足条件的索引项全部取出，并在内存中进行排序, 然后根据取出的索引项访问表数据）cost为325最高。
3. 利用Aggregate判断count(*) = 1，cost很低。
4. Index scan找到满足以上子查询的所有查询结果，cost很低。

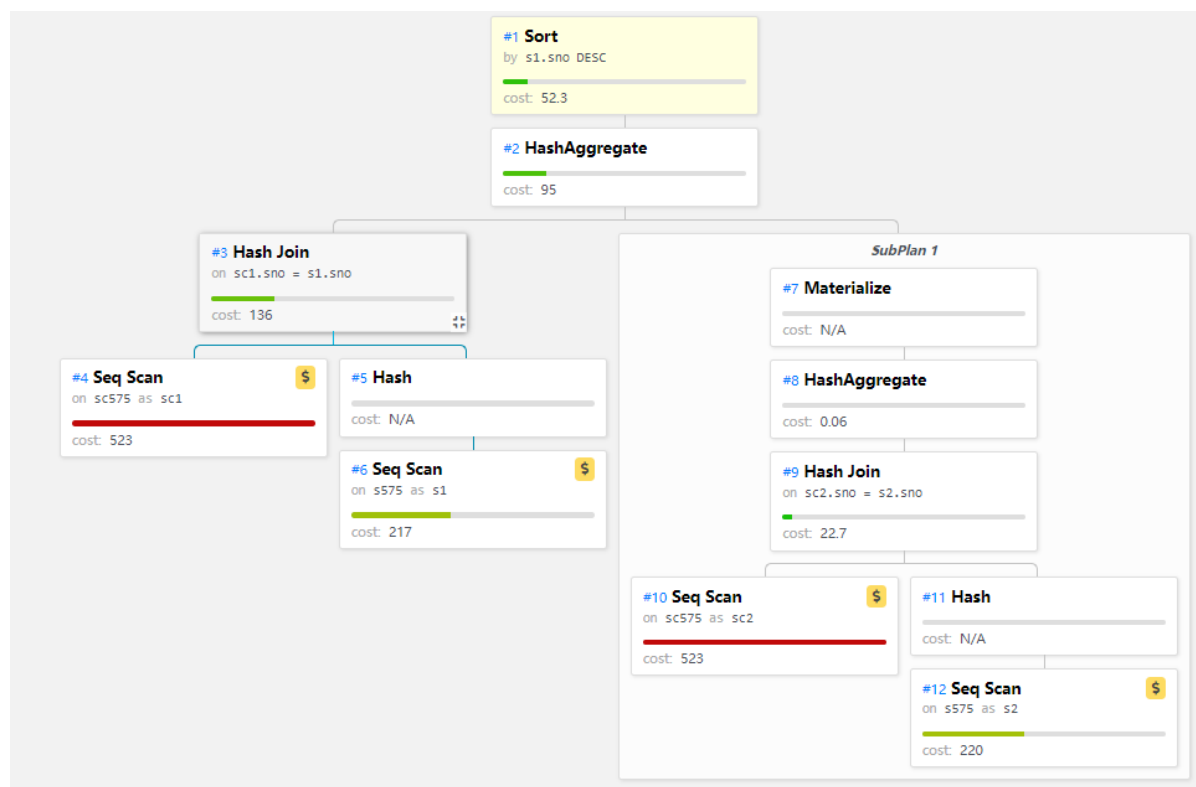
实际上对于每条sc1记录都执行了一遍子查询，最终返回了sc1的记录，cost较低。

对比这两个查询，还是第一个查询效率更高一些。

查询(6)

写法1

```
1  SELECT
2      s1.sno,
3      s1.sname,
4      AVG ( sc1.grade ) AS avg_grade
5  FROM
6      s575 s1,
7      sc575 sc1
8  WHERE
9      s1.sno = sc1.sno
10 GROUP BY
11     s1.sno
12 HAVING
13     avg_grade > ALL (
14     SELECT AVG
15         ( sc2.grade ) AS avg2
16     FROM
17         sc575 sc2,
18         s575 s2
19     WHERE
20         sc2.sno = s2.sno
21         AND s2.sname = '王涛'
22     GROUP BY
23         s2.sno
24 )
25 ORDER BY
26     s1.sno DESC;
```

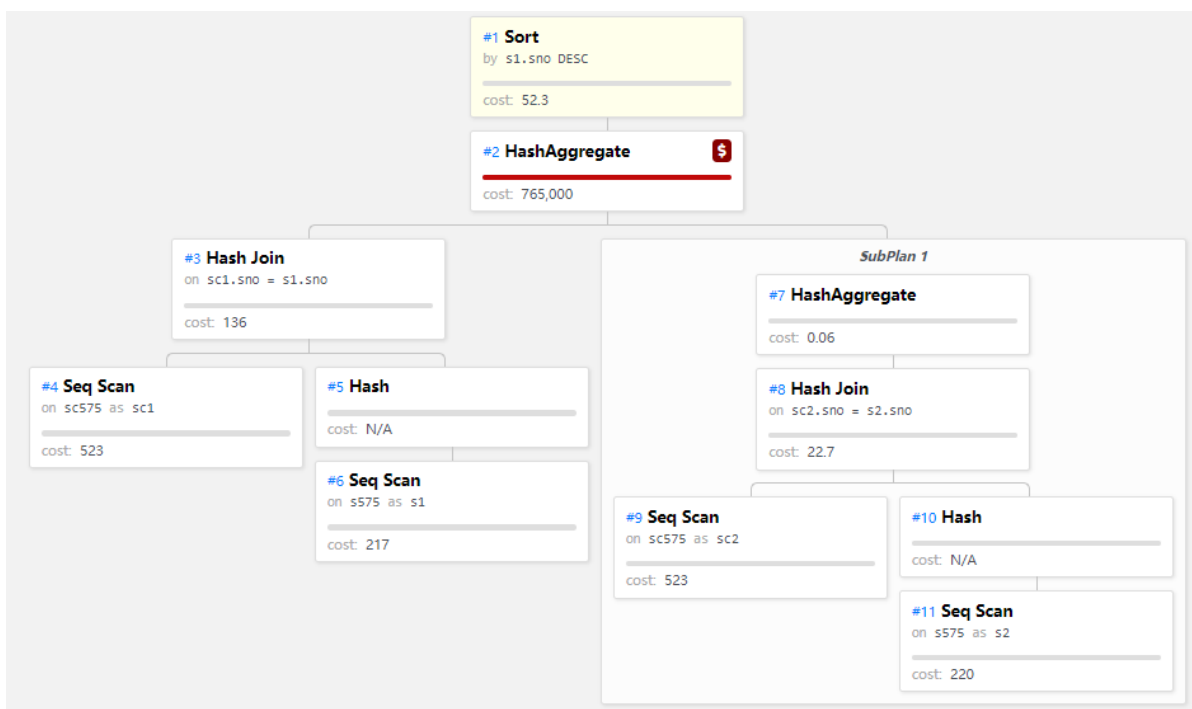


1. Seq Scan顺序扫描找到王涛，和他的成绩，然后连接，组成了子查询，cost大约为800。
2. 外层查询中也是使用的Seq Scan，cost差不多。

```

1  SELECT
2      s1.sno,
3      sname,
4      AVG ( sc1.grade ) AS avg_grade
5  FROM
6      s575 s1,
7      sc575 sc1
8  WHERE
9      s1.sno = sc1.sno
10 GROUP BY
11     s1.sno
12 HAVING
13     EXISTS (
14         SELECT
15             s2.sno
16         FROM
17             s575 s2,
18             sc575 sc2
19         WHERE
20             s2.sno = sc2.sno
21             AND s2.sname = '王涛'
22         GROUP BY
23             s2.sno
24         HAVING
25             AVG ( sc1.grade ) > AVG ( sc2.grade )
26     )
27 order by s1.sno DESC;

```



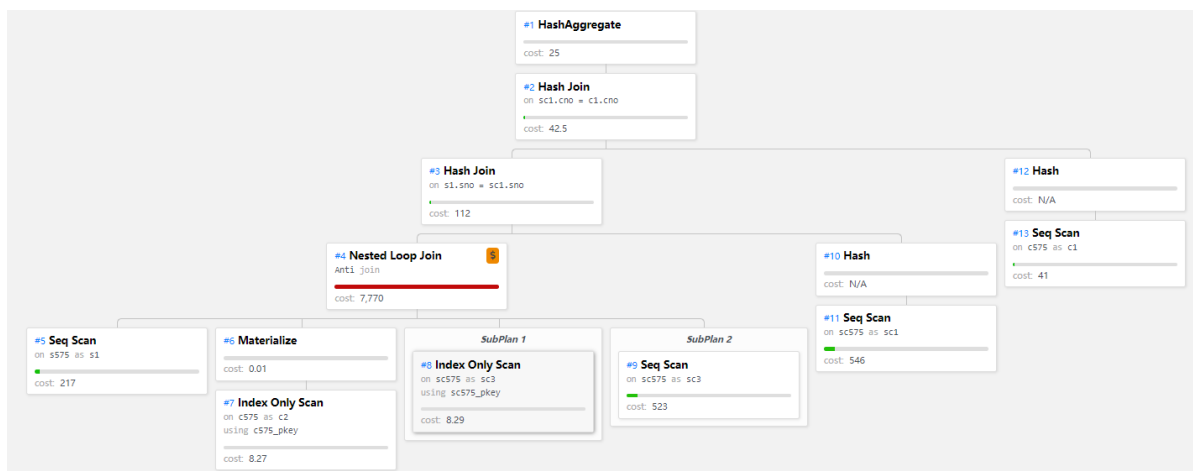
1. 每个自查询中都是使用Seq Scan，效率与写法1差不多。
2. 但是外层使用的not exists，因此每个外层的记录都需要里面查询一遍，这样cost就指数增长，因此这个HashAggregate的cost非常高。

经过比较，还是写法1更好。

查询(7)

写法1

```
1  SELECT
2      s1.sname,
3      SUM ( credit )
4  FROM
5      c575 c1,
6      s575 s1,
7      sc575 sc1
8  WHERE
9      c1.cno = sc1.cno
10     AND sc1.sno = s1.sno
11     AND NOT EXISTS (
12         SELECT
13             *
14         FROM
15             c575 c2
16         WHERE
17             c2.cno LIKE 'CS-%'
18             AND NOT EXISTS ( SELECT * FROM sc575 sc3 WHERE sc3.cno = c2.cno AND
19                             sc3.sno = s1.sno )
20     )
21     AND sc1.grade > 60
22 GROUP BY
23     s1.sno;
```



这个主要的cost就在Nested Loop Join这里就是因为这里使用了一个not exists语句，所以cost指数增大为7770。

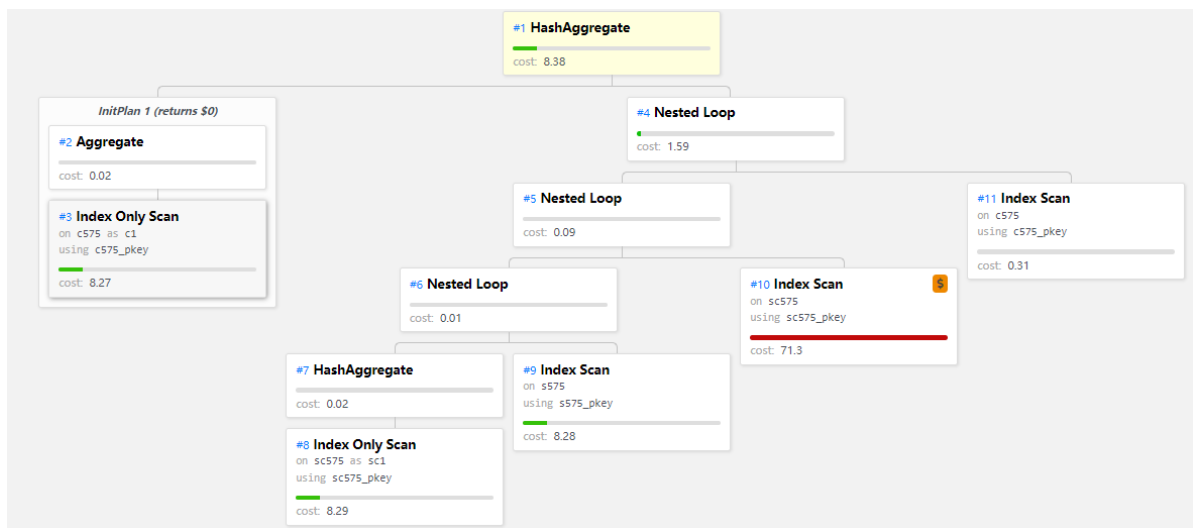
写法2

```
1  select sname, sum(credit)
2  from s575, sc575, c575
3  where s575.sno = sc575.sno and c575.cno = sc575.cno
4  and sc575.grade >= 60
5  and s575.sno in
6  (
7      select sno
8      from
9      (
10         select sc1.sno, count(*) as count_cs
11         from sc575 sc1
```

```

12 where sc1.cno like 'CS-%'
13 group by sc1.sno
14 ) as t1
15 where count_cs =
16 (
17 select count(*)
18 from (
19 select c1.cno
20 from c575 c1
21 where c1.cno like 'CS-%'
22 )
23 )
24 )
25 group by s575.sno
26 ;

```



这里直接求取cs课程的数量，对比cs课程数量的多少，cost较少。

应该是写法2更好，但是感觉有点投机取巧的意思。

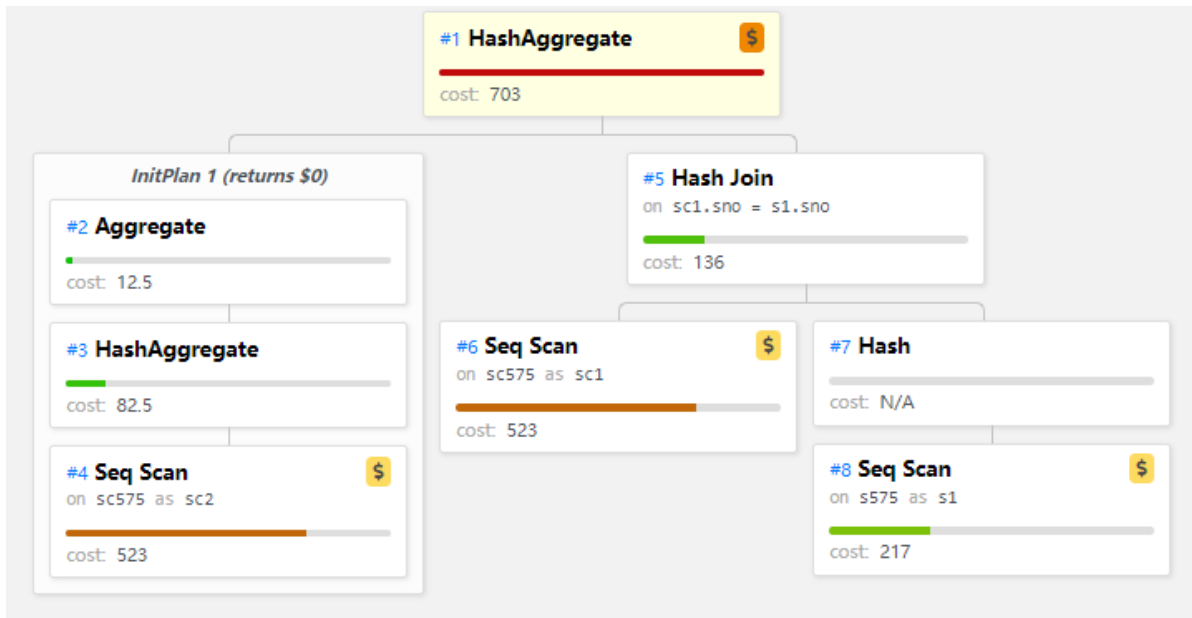
查询(8)

写法1

```

1  SELECT
2      s1.sno,
3      s1.sname
4  FROM
5      s575 s1,
6      sc575 sc1
7  WHERE
8      s1.sno = sc1.sno
9  GROUP BY
10     s1.sno
11  HAVING
12     COUNT ( * ) >= 3
13     AND AVG ( sc1.GRADE ) >= (
14         SELECT MAX
15             ( avg_grade )
16         FROM
17             ( SELECT AVG ( sc2.GRADE ) AS avg_grade FROM sc575 sc2 GROUP BY sc2.sno
18             HAVING COUNT ( * ) >= 3 ) AS table0
19     );

```

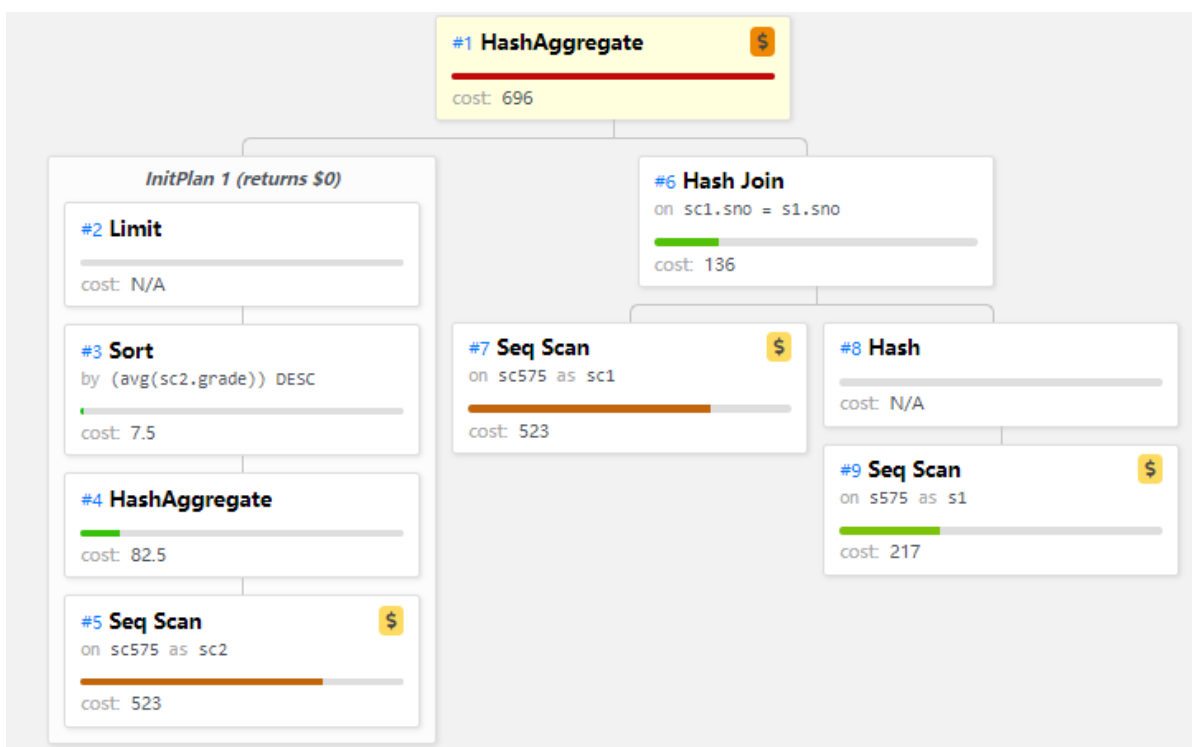


主要的cost在于HashAggregate这里也就是group by，以及两个Seq Scan。

写法2

```

1  select s1.sno ,s1.sname
2  from s575 s1, sc575 sc1
3  where s1.sno = sc1.sno
4  group by s1.sno
5  having count(*) >= 3
6  and avg(sc1.grade) = (
7  select avg(sc2.grade) as avg_grade
8    from sc575 sc2
9    group by sc2.sno
10   having count(*) >= 3
11   order by avg_grade DESC
12   limit 1
13 );
  
```



二者很类似，基本没有区别。

2.4.2 第二次插入 + 效率分析

查询(5)

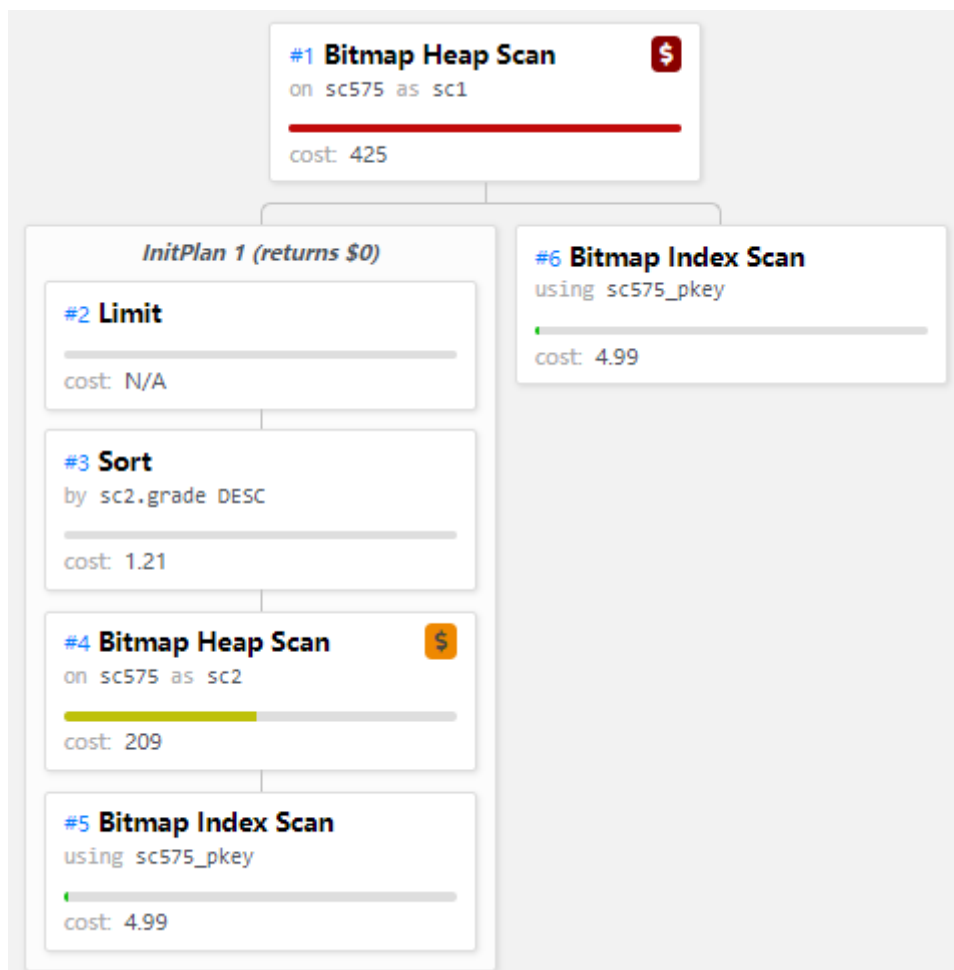
写法1

```
1 SELECT
2     sno
3 FROM
4     sc575 sc1
5 WHERE
6     sc1.cno = 'CS-02'
7     AND sc1.grade = ( SELECT grade FROM sc575 sc2 WHERE sc2.cno = 'CS-02'
                        ORDER BY grade DESC LIMIT 1, 1 );
```

已知通过explain可以分析一个sql语句的优劣。

```
mydb1=# explain SELECT
      sno
FROM
      sc575 sc1
WHERE
      sc1.cno = 'CS-02'
      AND sc1.grade = ( SELECT grade FROM sc575 sc2 WHERE sc2.cno = 'CS-02' OR
DER BY grade DESC LIMIT 1, 1 );
               QUERY PLAN
-----
Bitmap Heap Scan on sc575 sc1  (cost=220.35..429.97 rows=1 width=9)
  Recheck Cond: (cno = 'CS-02'::bpchar)
  Filter: (grade = $0)
  InitPlan 1 (returns $0)
    -> Limit  (cost=215.36..215.36 rows=1 width=6)
      -> Sort  (cost=215.36..215.60 rows=97 width=6)
        Sort Key: sc2.grade DESC
        -> Bitmap Heap Scan on sc575 sc2  (cost=5.01..214.39 rows=97 w
idth=6)
          Recheck Cond: (cno = 'CS-02'::bpchar)
          -> Bitmap Index Scan on sc575_pkey  (cost=0.00..4.99 row
s=97 width=0)
            Index Cond: (cno = 'CS-02'::bpchar)
    -> Bitmap Index Scan on sc575_pkey  (cost=0.00..4.99 rows=97 width=0)
        Index Cond: (cno = 'CS-02'::bpchar)
  (13 rows)

EXPLAIN
```



数据量大了之后查询plan显然发生了变化：

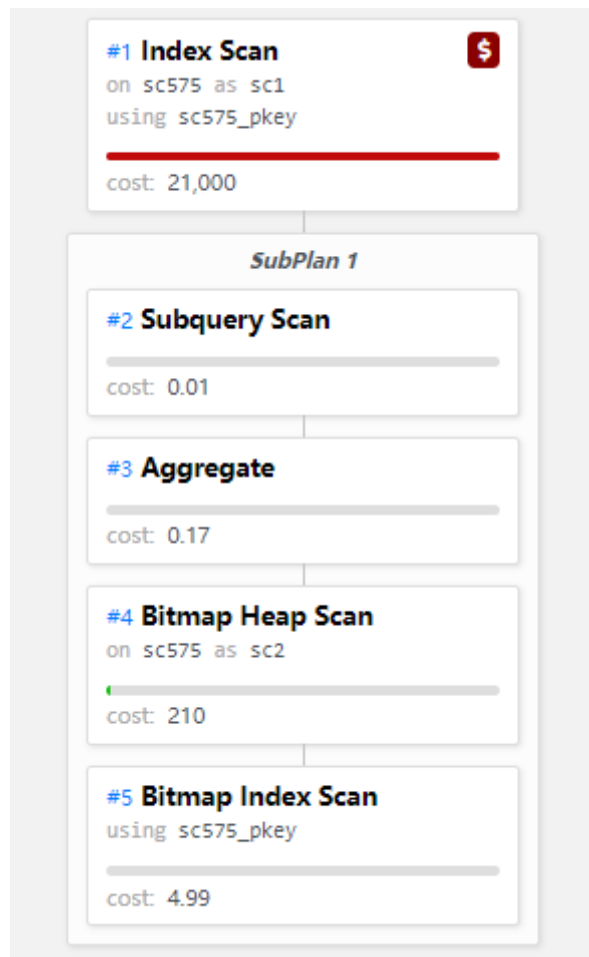
之前是使用的Seq Scan，这里使用的Bitmap Index Scan，对于大量记录更为有效。

写法2

```

1      SELECT
2      sno
3  FROM
4      sc575 sc1
5  WHERE
6      sc1.cno = 'CS-02'
7      AND EXISTS (
8          SELECT
9              *
10         FROM
11             ( SELECT COUNT ( * ) AS count_hi FROM sc575 sc2 WHERE sc2.cno =
12               sc1.cno AND sc2.grade > sc1.grade )
13         WHERE
14             count_hi = 1
15     );

```



这个查询与数据量小时候的plan是一致的，这里的外层查询显然cost要高很多。

这里显然写法1效率要更高。

查询(6)

写法1

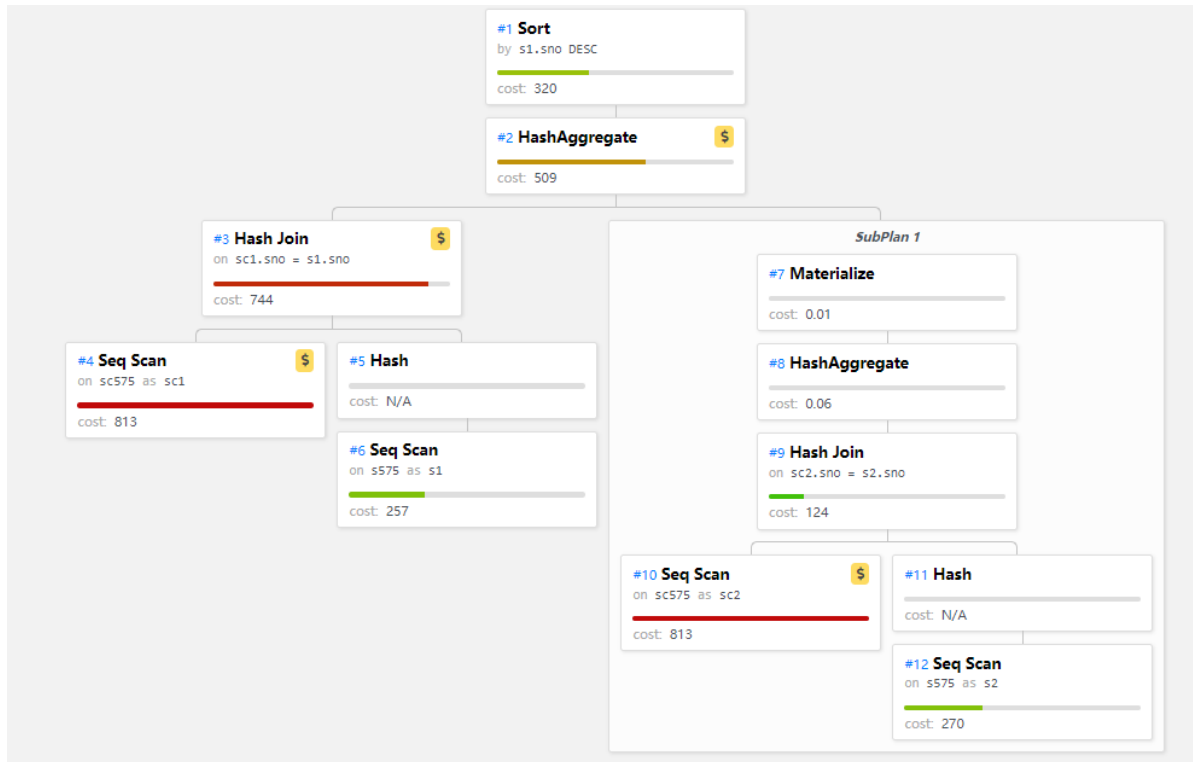
```
1  SELECT
2      s1.sno,
3      s1.sname,
4      AVG ( sc1.grade ) AS avg_grade
5  FROM
6      s575 s1,
7      sc575 sc1
8  WHERE
9      s1.sno = sc1.sno
10 GROUP BY
11     s1.sno
12 HAVING
13     avg_grade > ALL (
14     SELECT AVG
15         ( sc2.grade ) AS avg2
16     FROM
17         sc575 sc2,
18         s575 s2
19     WHERE
20         sc2.sno = s2.sno
21         AND s2.sname = '王涛'
22     GROUP BY
23         s2.sno
```



```

24 )
25 ORDER BY
26 s1.sno DESC;

```

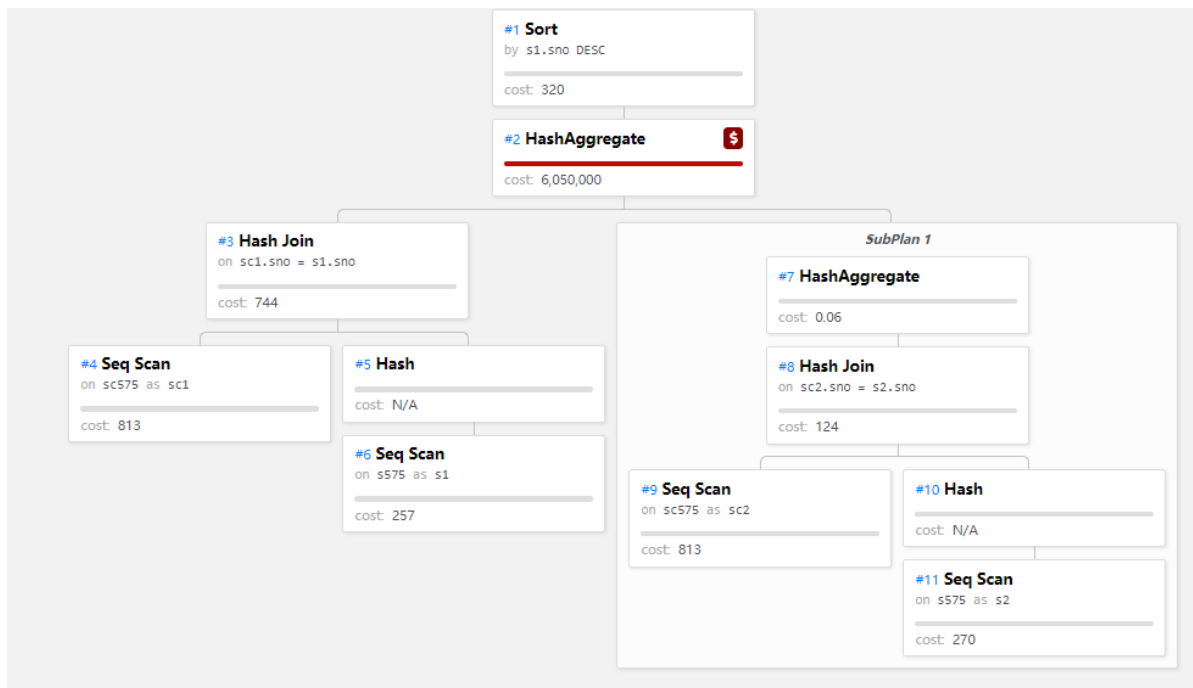


写法2

```

1  SELECT
2      s1.sno,
3      sname,
4      AVG ( sc1.grade ) AS avg_grade
5  FROM
6      s575 s1,
7      sc575 sc1
8  WHERE
9      s1.sno = sc1.sno
10 GROUP BY
11     s1.sno
12 HAVING
13     EXISTS (
14         SELECT
15             s2.sno
16         FROM
17             s575 s2,
18             sc575 sc2
19         WHERE
20             s2.sno = sc2.sno
21             AND s2.sname = '王涛'
22         GROUP BY
23             s2.sno
24         HAVING
25             AVG ( sc1.grade ) > AVG ( sc2.grade )
26     )
27 ORDER BY s1.sno DESC;

```



这里由于使用了exists所以叠加起来的cost较大。

因此写法1更好点。

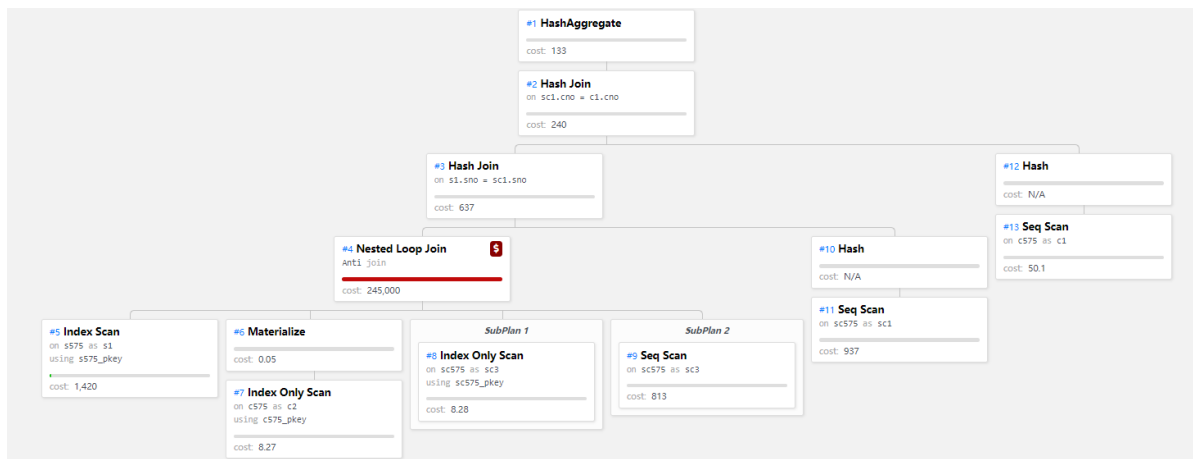
查询(7)

写法1

```

1  SELECT
2      s1.sname,
3      SUM ( credit )
4  FROM
5      c575 c1,
6      s575 s1,
7      sc575 sc1
8  WHERE
9      c1.cno = sc1.cno
10     AND sc1.sno = s1.sno
11     AND NOT EXISTS (
12         SELECT
13             *
14         FROM
15             c575 c2
16         WHERE
17             c2.cno LIKE 'CS-%'
18             AND NOT EXISTS ( SELECT * FROM sc575 sc3 WHERE sc3.cno = c2.cno AND
19                             sc3.sno = s1.sno )
20     )
21     AND sc1.grade > 60
22  GROUP BY
23     s1.sno;

```



写法2

```

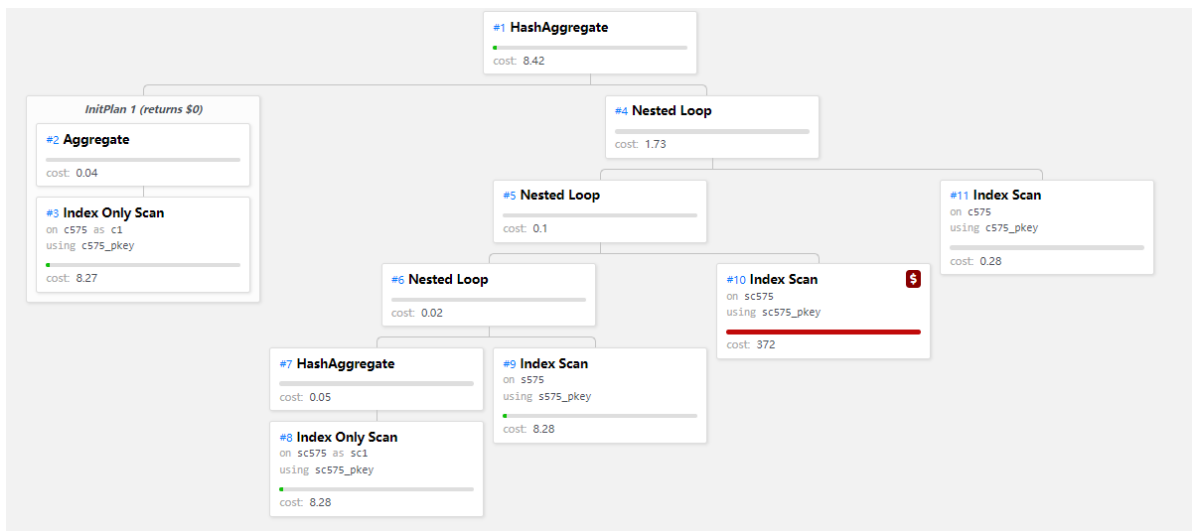
1  select sname, sum(credit)
2  from s575, sc575, c575
3  where s575.sno = sc575.sno and c575.cno = sc575.cno
4  and sc575.grade >= 60
5  and s575.sno in
6  (
7  select sno
8  from
9  (
10 select sc1.sno, count(*) as count_cs
11  from sc575 sc1
12  where sc1.cno like 'CS-%'
13  group by sc1.sno
14  ) as t1
15  where count_cs =
16  (
17  select count(*)
18  from (
19  select c1.cno
20  from c575 c1
21  where c1.cno like 'CS-%'
22  )
23  )
24  )
25  group by s575.sno
26  ;

```

```

QUERY PLAN
-----
HashAggregate (cost=91.38..91.44 rows=6 width=53)
  Group By Key: s575.sno
  InitPlan 1 (returns $0)
    -> Aggregate (cost=4.25..4.26 rows=1 width=8)
      -> Seq Scan on c575 c1 (cost=0.00..4.25 rows=1 width=0)
        Filter: (cno ~ 'CS-%'::text)
    -> Nested Loop (cost=8.28..87.09 rows=6 width=21)
      -> Nested Loop (cost=8.28..85.39 rows=6 width=23)
        -> Nested Loop (cost=8.28..16.59 rows=1 width=26)
          -> HashAggregate (cost=8.28..8.30 rows=1 width=25)
            Group By Key: sc1.sno
            Filter: (count(*) = $0)
          -> Index Only Scan using sc575_pkey on sc575 sc1 (cost=0.00..8.28 rows=1 width=9)
            Index Cond: ((cno >= 'CS-'::bpchar) AND (cno < 'CS.'::bpchar))
            Filter: (cno ~ 'CS-%'::text)
          -> Index Scan using s575_pkey on s575 (cost=0.00..8.27 rows=1 width=17)
            Index Cond: (sno = sc1.sno)
        -> Index Scan using sc575_pkey on sc575 (cost=0.00..68.71 rows=9 width=15)
          Index Cond: (sno = s575.sno)
          Filter: (grade >= 60::numeric)
      -> Index Scan using c575_pkey on c575 (cost=0.00..0.27 rows=1 width=10)
        Index Cond: (cno = sc575.cno)
(22 rows)

```



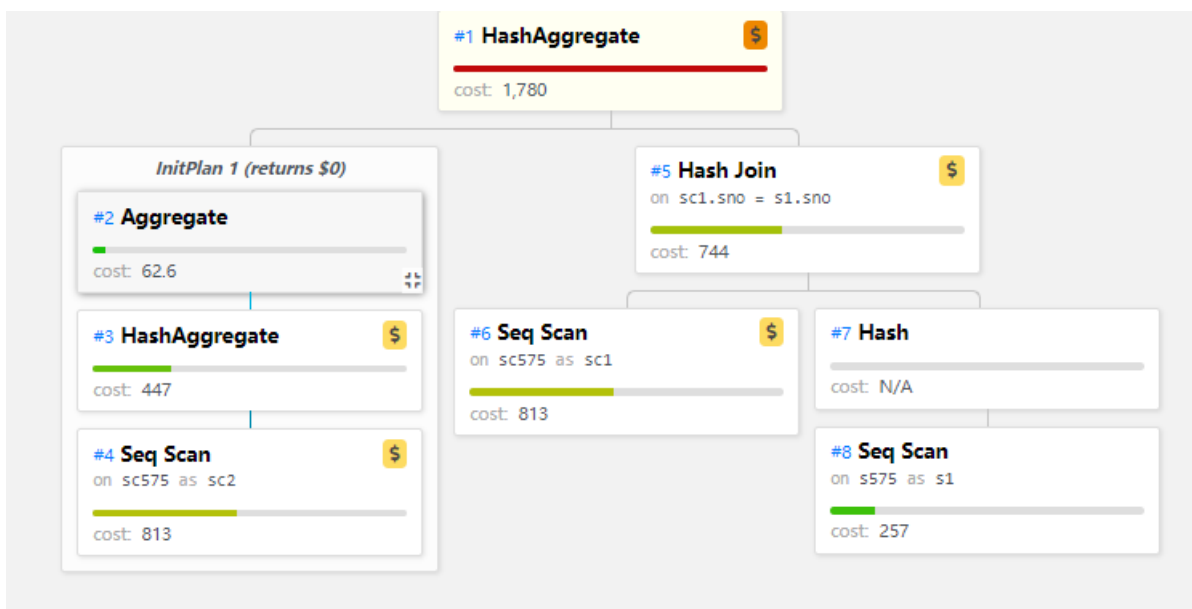
这个查询的情况与数据量小的时候类似。

查询(8)

写法1

```

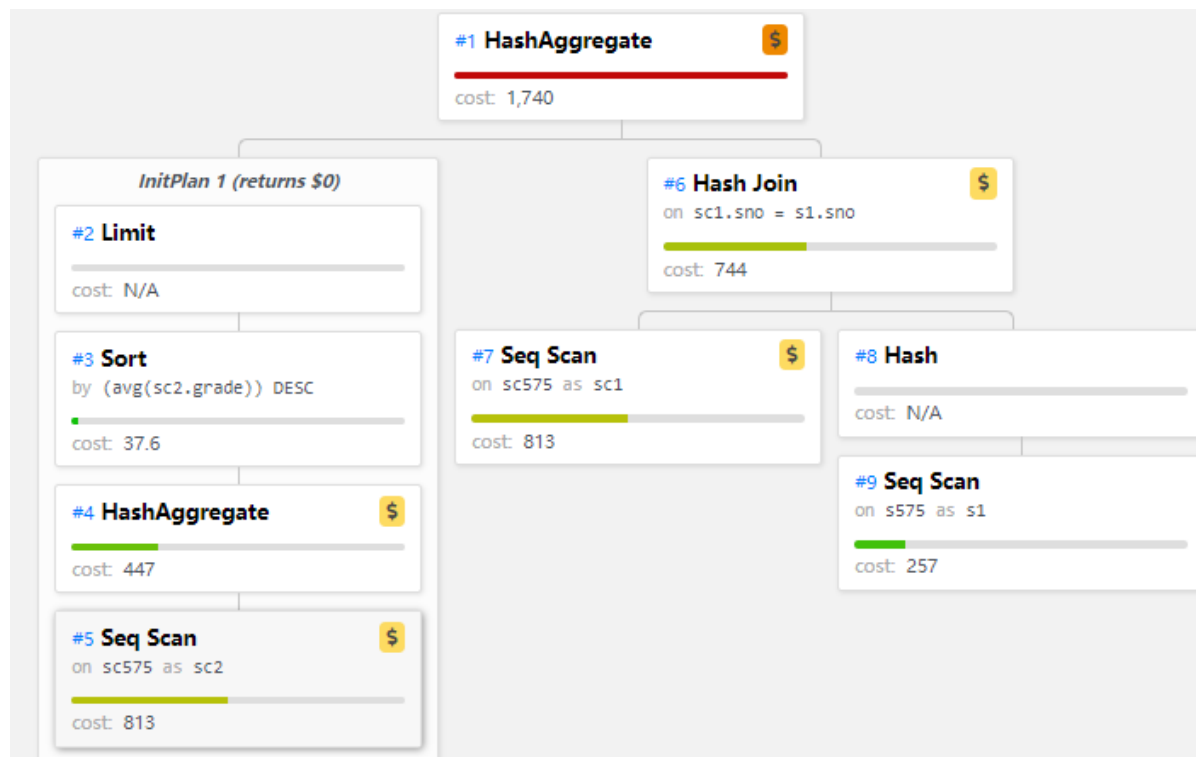
1  SELECT
2      s1.sno,
3      s1.sname
4  FROM
5      s575 s1,
6      sc575 sc1
7  WHERE
8      s1.sno = sc1.sno
9  GROUP BY
10     s1.sno
11  HAVING
12     COUNT ( * ) >= 3
13     AND AVG ( sc1.GRADE ) >= (
14         SELECT MAX
15             ( avg_grade )
16         FROM
17             ( SELECT AVG ( sc2.GRADE ) AS avg_grade FROM sc575 sc2 GROUP BY sc2.sno
18             HAVING COUNT ( * ) >= 3 ) AS table0
19     );
  
```



```

1  select s1.sno ,s1.sname
2  from s575 s1, sc575 sc1
3  where s1.sno = sc1.sno
4  group by s1.sno
5  having count(*) >= 3
6  and avg(sc1.grade) = (
7  select avg(sc2.grade) as avg_grade
8    from sc575 sc2
9    group by sc2.sno
10   having count(*) >= 3
11   order by avg_grade DESC
12   limit 1
13 );

```



与数据量小的时候类似。

2.4.3 提高效率

要提高效率可以

- 改变语句的逻辑或写法。
- 在相应属性添加索引。

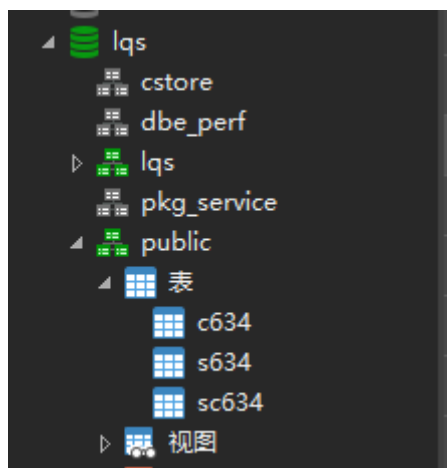
2.5 交换数据库并恢复

数据库备份来源-----计算机91 刘青帅

使用navicat直接进行备份的还原，还原的时候要注意新建一个与源数据库用户一致的用户名，以免没有建表的权限。

这里新建一个lqs用户，并且赋予管理员权限。

然后直接还原即可：



还原成功，然后简单分析一下这个表：

2.5.1 S表

名	类型	长度	小数点	不是 null	键
sno	char	10	0	<input checked="" type="checkbox"/>	1
sname	varchar	20	0	<input checked="" type="checkbox"/>	
sex	char	6	0	<input checked="" type="checkbox"/>	
bdate	timestamp	0	0	<input checked="" type="checkbox"/>	
height	numeric	3	2	<input type="checkbox"/>	
dorm	varchar	30	0	<input checked="" type="checkbox"/>	

sno长度固定且为8，这里感觉没有必要给长度为10；

sname长度为20太短了，考虑到会有留学生，长度应该加长；

2.5.2 C表

名	类型	长度	小数点	不是 null	键
cno	char	10	0	<input checked="" type="checkbox"/>	1
cname	varchar	160	0	<input checked="" type="checkbox"/>	
period	int2	16	0	<input checked="" type="checkbox"/>	
credit	numeric	2	1	<input checked="" type="checkbox"/>	
teacher	varchar	40	0	<input checked="" type="checkbox"/>	

这里我发现teacher的varchar长度仅仅给了40，我感觉这里没有考虑到会有外教，名字比较长的情况。

cno的长度固定且均为5，在这里给长度为10我感觉没有必要。

2.5.3 SC表

名	类型	长度	小数点	不是 null	键
sno	char	10	0	<input checked="" type="checkbox"/>	1
cno	char	10	0	<input checked="" type="checkbox"/>	2
grade	numeric	4	1	<input type="checkbox"/>	

SC表比较好。

2.5.4 数据质量

	cno	cname	period	credit	teacher
▶	CS-01	数据结构	60	3.0	张军
	CS-02	计算机组成原理	80	4.0	王亚伟
	CS-04	人工智能	40	2.0	李蕾
	CS-05	深度学习	40	2.0	崔均
	EE-02	数字逻辑电路	100	5.0	胡海东
	EE-03	光电子学与光子学	40	2.0	石韬
	JS-106	《左传》导读	20	1.0	许昭
	RW-107	社会组织管理	120	6.0	王社民
	JJ-108	国际市场调研与预	40	2.0	范爱莉
	RJ-109	环境系统监测	70	3.5	陈芯莹
	JX-110	工程制图	80	4.0	何平
	SM-111	遗传学实验	90	4.5	何效增
	HX-112	有机化学 I -2	100	5.0	冯骥磊
	CS-03	离散数学	64	4.0	陈建明
	EE-01	信号与系统	64	4.0	张明
	YX-113	医学微生物学	60	3.0	汪敏强
	YX-114	生物化学	30	1.5	权芳
	WG-115	日语写作实践	70	3.5	赵宏亮
	HX-116	医用有机化学实验	50	2.5	陈晓黎
	CL-117	材料科学基础-1	20	1.0	范艳蕊

21223182	冯学军	女	2002-10-08 00:0	1.74	东17舍516
14857721	何森	女	2003-05-11 00:0	1.63	北7舍723
07986625	葛金琰	女	1999-06-27 00:0	1.77	南11舍104
91230268	周志淳	女	1999-08-06 00:0	1.51	南1舍925
04569293	朱卫	男	2001-10-17 00:0	1.81	北13舍903
31970095	杜正春	女	2000-09-20 00:0	1.58	南0舍004
30648078	王瑞	女	2002-01-28 00:0	1.56	西0舍411
53407417	苏国利	女	1999-05-22 00:0	1.61	北15舍929
54085108	范俊梅	男	2000-10-19 00:0	1.62	西19舍825
93359913	何德杰	女	2001-03-03 00:0	1.59	北14舍018
69228894	朱增	男	1997-04-18 00:0	1.85	西2舍120
37210262	刘广庆	女	1998-09-21 00:0	1.79	东18舍616
36474674	周晓文	女	1998-05-13 00:0	1.65	南12舍700
67789563	苏立明	男	1998-06-06 00:0	1.84	南15舍803
40358800	赖天伟	男	2002-06-19 00:0	1.64	东4舍427
69131935	赵文振	女	2001-06-26 00:0	1.57	北14舍004
85863838	陈丽敏	女	2002-01-25 00:0	1.63	西0舍114

生成数据质量较好，课程的信息均为教务处爬取的，好像学生的信息是使用teacher的名字相互组合而成的，非常有真实性。

3 实验总结

本次实验收获了很多，列举如下：

- 学会了配置openguass环境，配置本地虚拟机以及使用navicat进行连接。
- 学会了使用navicat进行数据库的操作。
- 学会了使用JDBC连接数据库并且生成随机数据插入数据库。
- 学会了使用爬虫爬取网页内容。
- 学会了使用explain语句进行sql查询的效率分析。

4 附录

4.1 生成初始数据 --- python源码

init_data.py

```

1  f = open("s575.txt", 'r',encoding = 'utf-8')
2
3  line = f.readline()
4
5  while(line != ''):
6      y = line.split()
7
8      line = f.readline()
9
10     print("insert into s575 values('" + y[0] + "','" + y[1] + "','" + y[2] +
        "','" + y[3] + "','" + str(y[4]) + "','" + y[5] + "')")

```



```

11
12
13 f.close()
14
15
16 f = open("C575.txt", "r", encoding = 'utf-8')
17
18 line = f.readline()
19
20 while(line != ''):
21     y = line.split()
22     line = f.readline()
23     print("insert into C575 values('" + y[0] + "','" + y[1] + "','" +
24 str(y[2]) + "','" + str(y[3]) + "','" + y[4] + "')")
25 f.close()
26
27 f = open("SC575.txt", "r", encoding = 'utf-8')
28
29 line = f.readline()
30
31 while(line != ''):
32     y = line.split()
33     line = f.readline()
34     if(len(y) == 2):
35         print("insert into SC575 values('" + y[0] + "','" + y[1] + "','" +
36 "NULL" + "')")
37     else:
38         print("insert into SC575 values('" + y[0] + "','" + y[1] + "','" +
39 y[2] + "')")

```

4.2 爬取教务处课程表 --- python源码

getcourse.py

```

1 import requests
2 import json
3 req = requests.session()
4
5 dic = {}
6 # 这里的cookie是随时间变换的，每次需要使用不一样的cookie
7 with open('cookie.txt', 'r') as file_obj:
8     cookie = file_obj.read()
9 f = open("course.txt", "w")
10 for ii in range(1, 1000):
11
12
13     data = {'querySetting': ''
14 [{"name": "XNXQDM", "value": "2021-2022-
15 2", "linkOpt": "and", "builder": "equal"},
16 [{"name": "RWZTDM", "value": "1", "linkOpt": "and", "builder": "equal"},
17 {"name": "RWZTDM", "linkOpt": "or", "builder": "isNull"}]]'',
18         '*order': '+KKDWDW,+KCH,+KXH',
19         'pageSize': 1000,
20         'pageNumber': ii}
21     headers = {

```

```

19         'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.77 Safari/537.36',
20         'cookie': cookie}
21     # 课程接口
22     rep =
req.post('http://ehall.xjtu.edu.cn/jwapp/sys/kbcx/modules/qxkcb/qxfbkcck.do
', data=data,
23         headers=headers)
24     content = rep.text
25     # 生成一个字典
26     content = json.loads(content)
27     for i in range(len(content['datas']['qxfbkcck']['rows'])):
28         teacher = content['datas']['qxfbkcck']['rows'][i]['SKJS']
29         credit = content['datas']['qxfbkcck']['rows'][i]['KNZXS']
30         cno = content['datas']['qxfbkcck']['rows'][i]['KCH']
31         time = content['datas']['qxfbkcck']['rows'][i]['XNXQDM']
32         name = content['datas']['qxfbkcck']['rows'][i]['KCM']
33         period = content['datas']['qxfbkcck']['rows'][i]['XS']
34         # 取出对应的数据
35         dic['cno'] = cno
36         dic['name'] = name
37         dic['period'] = period
38         dic['credit'] = credit
39         dic['teacher'] = teacher
40         dic['time'] = time
41         if(teacher == None):
42             strs = cno + "##" + name + "##" + str(period) + "##" +
str(credit) + "##" + "Null" + '\n'
43         else:
44             strs = cno + "##" + name + "##" + str(period) + "##" +
str(credit) + "##" + teacher + '\n'
45
46         f.write(strs)
47
48     f.close()
49
50

```

4.3 JDBC生成数据 --- Java源码

getcourse.java

```

1  import java.io.*;
2  import java.util.*;
3
4  public class getcourse {
5      static ArrayList<ArrayList<String>> course;
6      static Set<String> cno;
7      static HashMap<String,String> cno_new;
8      static ArrayList<String> cno_list100;
9      static ArrayList<String> cno_list1000;
10     static ArrayList<String> insert100;
11     static ArrayList<String> insert1000;
12     private static void readFile1(File fin) throws IOException {
13         course = new ArrayList<ArrayList<String>>();
14         cno = new HashSet<String>();
15         cno_new = new HashMap<String, String>();

```

```

16         insert100 = new ArrayList<String>();
17         insert1000 = new ArrayList<String>();
18         cno_list100 = new ArrayList<String>();
19         cno_list1000 = new ArrayList<String>();
20         FileInputStream fis = new FileInputStream(fin);
21         //Construct BufferedReader from InputStreamReader
22         BufferedReader br = new BufferedReader(new InputStreamReader(fis));
23         String line = null;
24         int x = 0, y = 0, z = 1;
25         int i = 0;
26         HashSet<String> cno_re = new HashSet<>();
27         cno_re.add("CS-01");
28         cno_re.add("CS-02");
29         cno_re.add("CS-03");
30         cno_re.add("CS-04");
31         cno_re.add("CS-05");
32         cno_re.add("EE-01");
33         cno_re.add("EE-02");
34         cno_re.add("EE-03");
35         while ((line = br.readLine()) != null) {
36             String[] a = line.split("###");
37             ArrayList<String> tmp = new ArrayList<String>(Arrays.asList(a));
38             course.add(tmp);
39             if(cno.contains(a[0])){
40             }
41             else{
42                 Random rd = new Random();
43                 i++;
44                 char a0 = (char)('A' + rd.nextInt(0,25));
45                 char a1 = (char)('A' + rd.nextInt(0,25));
46                 int number = rd.nextInt(1,6);
47                 while(cno_re.contains("'" + a0 + a1 + "-0" +
String.valueOf(number))){
48                     a1 = (char)('A' + rd.nextInt(0,25));
49                 }
50                 cno_re.add("'" + a0 + a1 + "-0" + String.valueOf(number));
51                 String NewString = "'" + a0 + a1 + "-" + "0" +
String.valueOf(number);
52                 String insert_sql = "insert into C575 values(\"' + NewString
+ "\",\"' + a[1] + \"'\",\" + a[2] + \"'\",\" + a[3] + \"'\",\" + a[4] + \"'\");"
53
54                 if(cno_list100.size() < 100){
55                     cno_list100.add(NewString);
56                     insert100.add(insert_sql);
57                 }
58                 if(cno_list1000.size() < 1000){
59                     cno_list1000.add(NewString);
60                     insert1000.add(insert_sql);
61                 }
62                 cno_new.put(a[0], NewString);
63                 System.out.println(insert_sql);
64                 x += (y + (z) / 6) / 26;
65                 y = (y + (z) / 6) % 26;
66                 z = (z) % 6 + 1;
67             }
68
69             cno.add(a[0]);
70

```

```

71
72         // System.out.println(a[0].substring(1,3));
73     }
74     br.close();
75 }
76 public static void go(){
77     try {
78         File f = new
File("D:\\Project\\java_proj\\untitled\\src\\course.txt");
79         readFile1(f);
80     }catch (Exception ex){
81         ex.printStackTrace();
82     }
83 }
84 public static void main(String[] args){
85     go();
86 }
87 }

```

getStudent.java

```

1  import javax.swing.*;
2  import java.awt.event.ActionEvent;
3  import java.io.BufferedReader;
4  import java.io.File;
5  import java.io.FileInputStream;
6  import java.io.InputStreamReader;
7  import java.util.ArrayList;
8  import java.util.HashSet;
9  import java.util.Random;
10
11 public class getStudent {
12     static ArrayList<String> insert1000;
13     static ArrayList<String> insert5000;
14     static ArrayList<String> sno1000;
15     static ArrayList<String> sno5000;
16     static ArrayList<String> ming;
17     static ArrayList<String> xing;
18     static ArrayList<String> name;
19     static ArrayList<String> sushe;
20     static void getming(){
21         try {
22             ming = new ArrayList<String>();
23             xing = new ArrayList<>();
24             name = new ArrayList<String>();
25             File file = new
File("D:\\Project\\java_proj\\untitled\\src\\ming.txt");
26             File file2 = new
File("D:\\Project\\java_proj\\untitled\\src\\xing.txt");
27
28             FileInputStream fis = new FileInputStream(file);
29             BufferedReader br = new BufferedReader(new
InputStreamReader(fis));
30             String line = null;
31             while ((line = br.readLine()) != null) {
32                 String[] a = line.split("\\ ");
33                 for(int i = 0; i < a.length; i++)ming.add(a[i]);

```

```

34     }
35
36     FileInputStream fis2 = new FileInputStream(file2);
37     BufferedReader br2 = new BufferedReader(new
InputStreamReader(fis2));
38     line = null;
39     while ((line = br2.readLine()) != null) {
40         String[] a = line.split(",");
41         for(int i = 0; i < a.length; i++)xing.add(a[i]);
42     }
43
44     for(int i = 0; i < ming.size(); i++){
45         for(int j = 0; j < ming.size(); j++){
46             name.add(ming.get(i) + ming.get(j));
47         }
48     }
49 }catch (Exception ex){
50     ex.printStackTrace();
51 }
52
53 }
54 static void getinsert1000(){
55     sushe = new ArrayList<String>();
56     insert1000 = new ArrayList<String>() ;
57     sno1000 = new ArrayList<>();
58     HashSet<Integer> sno_re = new HashSet<Integer>();
59     for(int i = 0; i < 1000; i++){
60         Random rd = new Random();
61
62         int sno = 11000000 + rd.nextInt(1,100000);
63         while(sno_re.contains(sno)){
64             sno = 11000000 + rd.nextInt(1,100000);
65         }
66         sno_re.add(sno);
67         sno1000.add(String.valueOf(sno));
68         int first_name_index = rd.nextInt(0, xing.size() - 1);
69         int last_name_index = rd.nextInt(0, ming.size() - 1);
70         int last_name_index2 = rd.nextInt(0, ming.size() - 1);
71         int ming1 = rd.nextInt(10);
72         String sname = ming1 >= 4 ? xing.get(first_name_index) +
ming.get(last_name_index) + ming.get(last_name_index2) :
xing.get(first_name_index) + ming.get(last_name_index);
73
74         Boolean west = rd.nextBoolean();
75         String w = west ? "西" : "东";
76         int dorm_no = rd.nextInt(20) + 1;
77         int floor = rd.nextInt(6) + 1;
78         int room = rd.nextInt(40) + 1;
79         String room_str = room < 10 ? "0" + String.valueOf(room) :
String.valueOf(room);
80         String dorm = w + String.valueOf(dorm_no) + "舍" +
String.valueOf(floor) + room_str;
81         //System.out.println(w + String.valueOf(dorm_no) + "舍" +
String.valueOf(floor) + room_str);
82         String sex = west ? "男" : "女";
83         int year = rd.nextInt(2000,2006);
84         int mon = rd.nextInt(12) + 1;
85         int day = rd.nextInt(28) + 1;

```

```

86         String date = String.valueOf(year) + "-" + String.valueOf(mon)
+ "-" + String.valueOf(day);
87         int cm = rd.nextInt(50,90);
88         String height = "1." + String.valueOf(cm);
89         String insert_sql = "insert into S575 values(\"'\" +
String.valueOf(sno) + "\",\"'\" + sname + "\",\"'\" + sex + "\",\"'\" + date +
+ "\",\"'\" + height + "\",\"'\" + dorm + "\")";
90         System.out.println(insert_sql);
91         insert1000.add(insert_sql);
92     }
93 }
94 static void getinsert5000(){
95     insert5000 = new ArrayList<String>();
96     sno5000 = new ArrayList<>();
97     HashSet<Integer> sno_re = new HashSet<>();
98     for(int i = 10000; i < 10000 + 5000; i++){
99         Random rd = new Random();
100
101         int sno = 10000000 + rd.nextInt(1,1000000);
102         while(sno_re.contains(sno)){
103             sno = 10000000 + rd.nextInt(1,1000000);
104         }
105         sno_re.add(sno);
106         sno5000.add(String.valueOf(sno));
107
108         int first_name_index = rd.nextInt(0, xing.size() - 1);
109         int last_name_index = rd.nextInt(0, ming.size() - 1);
110         int last_name_index2 = rd.nextInt(0, ming.size() - 1);
111         int ming1 = rd.nextInt(10);
112         String sname = ming1 >= 4 ? xing.get(first_name_index) +
ming.get(last_name_index) + ming.get(last_name_index2) :
xing.get(first_name_index) + ming.get(last_name_index);
113
114
115
116         Boolean west = rd.nextBoolean();
117         String w = west ? "西" : "东";
118         int dorm_no = rd.nextInt(20) + 1;
119         int floor = rd.nextInt(6) + 1;
120         int room = rd.nextInt(40) + 1;
121         String room_str = room < 10 ? "0" + String.valueOf(room) :
String.valueOf(room);
122         String dorm = w + String.valueOf(dorm_no) + "舍" +
String.valueOf(floor) + room_str;
123         //System.out.println(w + String.valueOf(dorm_no) + "舍" +
String.valueOf(floor) + room_str);
124         String sex = west ? "男" : "女";
125         int year = rd.nextInt(2000,2006);
126         int mon = rd.nextInt(12) + 1;
127         int day = rd.nextInt(28) + 1;
128         String date = String.valueOf(year) + "-" + String.valueOf(mon)
+ "-" + String.valueOf(day);
129         int cm = rd.nextInt(50,90);
130         String height = "1." + String.valueOf(cm);
131         String insert_sql = "insert into S575 values(\"'\" +
String.valueOf(sno) + "\",\"'\" + sname + "\",\"'\" + sex + "\",\"'\" + date +
+ "\",\"'\" + height + "\",\"'\" + dorm + "\")";
132         System.out.println(insert_sql);

```

```

133         insert5000.add(insert_sql);
134
135     }
136 }
137 public static void go(){
138     getming();
139
140     getinsert1000();
141     getinsert5000();
142 }
143 public static void main(String[] args){
144     go();
145 }
146 }

```

getSC.java

```

1  import java.util.ArrayList;
2
3  public class getSC {
4      static ArrayList<String> sc1;
5      static ArrayList<String> sc2;
6      static ArrayList<String> g;
7      static ArrayList<String> g2;
8
9      static ArrayList<String> insert_s;
10     static ArrayList<String> insert_s2;
11     static ArrayList<String> insert_c;
12     static ArrayList<String> insert_c2;
13     static ArrayList<String> insert_sc;
14     static ArrayList<String> insert_sc2;
15     public static void go(){
16         getcourse gc = new getcourse();
17         gc.go();
18         ArrayList<String> cnoLis100 = gc.cno_list100;
19         ArrayList<String> cnoLis1000 = gc.cno_list1000;
20         //for(int i = 0; i < cnoLis.size(); i++)
21         System.out.println(cnoLis.get(i));
22         //System.out.println(cnoLis1000.size());
23         getStudent gs = new getStudent();
24         gs.go();
25         ArrayList<String> snoLis1000 = gs.sno1000;
26         ArrayList<String> snoLis5000 = gs.sno5000;
27         g = new ArrayList<>();
28         g2 = new ArrayList<>();
29         sc1 = new ArrayList<>();
30         insert_c = gc.insert100;
31         insert_c2 = gc.insert1000;
32         insert_s = gs.insert1000;
33         insert_s2 = gs.insert5000;
34         insert_sc = new ArrayList<>();
35         insert_sc2 = new ArrayList<>();
36
37         for(int i = 0; i < 1000; i++){
38             int rd = (i * i + 9) % 11;
39             int k = 1;
40             while(rd < 100){

```

```

40         String[] tmp = {SnoLis1000.get(i), cnoLis100.get(rd)};
41         sc1.add(tmp);
42         rd += 11;
43         double gd = Math.random() * 60 + 40;
44         if((i * i * 7 + 2) % 4 == 0)gd = -1;
45         String grade = gd == -1 ? "NULL" : String.format("%.2f",gd);
46         g.add(grade);
47         String insert_sql = "insert into sc575 values(\"' + tmp[0] +
48         \"'\",\"'\" + tmp[1] + \"'\",\" + grade + \")";
49         System.out.println(insert_sql);
50         insert_sc.add(insert_sql);
51     }
52 }
53 sc2 = new ArrayList<>();
54 System.out.println("yes");
55 for(int i = 0;i < 5000; i++){
56     int rd = (i * i + 9) % 101;
57     int k = 1;
58     while(rd < 1000){
59         String[] tmp = {SnoLis5000.get(i), cnoLis1000.get(rd)};
60         sc2.add(tmp);
61         rd += 101;
62         double gd = Math.random() * 60 + 40;
63         if((i * i * 7 + 2) % 4 == 0)gd = -1;
64         String grade = gd == -1 ? "NULL" : String.format("%.2f",gd);
65         g2.add(grade);
66         String insert_sql = "insert into sc575 values(\"' + tmp[0] +
67         \"'\",\"'\" + tmp[1] + \"'\",\" + (grade) + \");";
68         insert_sc2.add(insert_sql);
69     }
70 }
71 }
72 System.out.println(sc1.size());
73 System.out.println(sc2.size());
74 }
75 public static void main(String[] args){
76     go();
77 }
78 }
79 }
80 }

```

4.4 JDBC连接数据库

openGauss.java

```

1  import java.sql.*;
2  import java.util.ArrayList;
3  public class openGaussDemo {
4
5      static final String JDBC_DRIVER = "org.postgresql.Driver";
6      static final String DB_URL =
7      "jdbc:postgresql://192.168.56.102:26000/mydb1?ApplicationName=app1";
8      // 数据库的用户名与密码，需要根据自己的设置
9      static final String USER = "my_root";

```



```

9      static final String PASS = "my_root@123";
10     public static void main(String[] args) {
11         Connection conn = null;
12         Statement stmt = null;
13         try{
14             // 注册 JDBC 驱动
15             Class.forName(JDBC_DRIVER);
16
17             // 打开链接
18             System.out.println("连接数据库...");
19             conn = DriverManager.getConnection(DB_URL,USER,PASS);
20
21             // 执行查询
22             System.out.println(" 实例化Statement对象...");
23             stmt = conn.createStatement();
24             String sql;
25             //sql = "SELECT * FROM s575";
26             PreparedStatement ps = null;
27             getSC gsc = new getSC();
28             getSC.go();
29             ArrayList<String> sql_s = gsc.insert_s;
30             ArrayList<String> sql_c = gsc.insert_c;
31             ArrayList<String> sql_sc = gsc.insert_sc;
32             for(int i = 0; i < sql_s.size(); i++)stmt.execute(sql_s.get(i));
33             for(int i = 0; i < sql_c.size(); i++)stmt.execute(sql_c.get(i));
34             for(int i = 0; i < sql_sc.size();
i++)stmt.execute(sql_sc.get(i));
35             //ResultSet rs = stmt.executeQuery(sql);
36             // System.out.println(rs);
37             // 展开结果集数据库
38             /*
39             while(rs.next()){
40                 // 通过字段检索
41                 int id = rs.getInt("Sno");
42                 String name = rs.getString("SNAME");
43                 String url = rs.getString("DORM");
44
45                 // 输出数据
46                 System.out.print("ID: " + id);
47                 System.out.print(", 站点名称: " + name);
48                 System.out.print(", 站点 URL: " + url);
49                 System.out.print("\n");
50             }*/
51             // 完成后关闭
52             //rs.close();
53             stmt.close();
54             conn.close();
55         }catch(SQLException se){
56             // 处理 JDBC 错误
57             se.printStackTrace();
58         }catch(Exception e){
59             // 处理 Class.forName 错误
60             e.printStackTrace();
61         }finally{
62             // 关闭资源
63             try{
64                 if(stmt!=null) stmt.close();
65             }catch(SQLException se2){

```

```
66         }// 什么都不做
67         try{
68             if(conn!=null) conn.close();
69         }catch(SQLException se){
70             se.printStackTrace();
71         }
72     }
73     System.out.println("Goodbye!");
74 }
75 }
```