中国科学技术大学

KDELab

数据库系统及应用 习题课

2022年春季学期

刘高聪 余卓翰 韦宇霏 王灿 汪洪韬

余卓翰

HOMEWORK #1

hw1

1、为什么在 ANSI/SPARC 体系结构中，外模式需要设计成多个？这么做有什么好处？

- 外模式：单个用户所看的局部数据的逻辑结构和特征的描述，用户与数据库系统的数据接口。保证数据安全性，每个用户只能看到外模式中的数据，其余数据不可见。

- 重点：简洁方便，安全性，灵活性。

hw1

2、什么是数据库的逻辑数据独立性？请举例说明。

- 当概念模式发生改变时，只要修改外模式/ 模式映象，可保持外模式不变，从而保持用户应用程序不变，保证了数据与用户程序的逻辑独立性。

hw1

3、关系数据模型要求外码所引用的属性必须是候选码，我们能否放松要求让外码引用非码 属性？试给出你的分析。

- 不能，候选码唯一，非码属性不唯一。

hw1

4、现实世界中的数据约束是否都可以通过关系数据模型的三类完整性规则来表示？如果 是，请解释理由。如果不是，请给出一个反例。

- 不可以，比如优秀率不超过40%，不及格率不超过30%。
- 注：用户定义的完整性只能对某一属性设置具体的约束，不能对表的统计量进行约束。涉及多个元组的约束或单个元组多属性的约束据无法通过完整性规则来表示

hw1

5、

给定下面的关系模式：图书（图书号，书名，作者，单价，库存量），读者（读者号，姓名，工作单位，地址），借阅（图书号，读者号，借期，还期，备注）

请使用关系代数表达式回答下列查询（注意：字符串须用单引号括起来）：

(1) 检索读者 Rose 的工作单位和地址；

$$\Pi_{工作单位，地址}\left(\sigma_{姓名='Rose'}\left(读者\right)\right)$$

hw1

5、

给定下面的关系模式：图书（图书号，书名，作者，单价，库存量），读者（读者号，姓名，工作单位，地址），借阅（图书号，读者号，借期，还期，备注）

请使用关系代数表达式回答下列查询（注意：字符串须用单引号括起来）：

(2) 检索读者 Rose 所借阅过的图书的图书名和借期；

$$\Pi_{书名，借期}\left(\sigma_{姓名='Rose'}\left(图书 \bowtie 读者 \bowtie 借阅\right)\right)$$

hw1

5、

给定下面的关系模式：图书（图书号，书名，作者，单价，库存量），读者（读者号，姓名，工作单位，地址），借阅（图书号，读者号，借期，还期，备注）

请使用关系代数表达式回答下列查询（注意：字符串须用单引号括起来）：

(3) 检索未借阅图书的读者姓名；

$$\Pi_{姓名}((\Pi_{读者号}(读者) - \Pi_{读者号}(借阅)) \bowtie 读者)$$

$\Pi_{姓名}(读者) - \Pi_{姓名}(读者 \bowtie 借阅)$//错误，姓名非主属性，可能存在相同值

hw1

5、

给定下面的关系模式：图书（图书号，书名，作者，单价，库存量），读者（读者号，姓名，工作单位，地址），借阅（图书号，读者号，借期，还期，备注）

请使用关系代数表达式回答下列查询（注意：字符串须用单引号括起来）：

(4) 检索 Ullman 所写的书的书名和单价；

$$\Pi_{书名，单价}\left(\sigma_{作者='Ullman'}\left(图书\right)\right)$$

hw1

6、

现有关于供应商、零件、工程的关系数据库，其模式如下（加下划线为主码）：

供应商：S (S#, sname, status, city)

零件：P (P#, pname, color, weight, city)

工程：J(J#, jname, city)

供应：SPJ (S#, J#, P#, QTY)　表示供应商 S#为工程 J#供应了 QTY 数量的零件 P#

请写出下列查询的关系代数表达式：

hw1

供应商：S (S#, sname, status, city)

零件：P (P#, pname, color, weight, city)

工程：J(J#, jname, city)

供应：SPJ (S#, J#, P#, QTY)　表示供应商 S#为工程 J#供应了 QTY 数量的零件 P#

(1) 将供应商号为'S01'的供应商的 city 改为'合肥';

$$S \leftarrow \Pi_{S\#,\ sname,\ staus,\ city='合肥'}(\sigma_{S\#='S01'}(S)) \cup (S - \sigma_{S\#='S01'}(S))$$

注：使用扩展投影修改属性值。

hw1

供应商：S (S#, sname, status, city)

零件：P (P#, pname, color, weight, city)

工程：J(J#, jname, city)

供应：SPJ (S#, J#, P#, QTY)　表示供应商 S#为工程 J#供应了 QTY 数量的零件 P#

(2) 求每个供应商的供应商号以及该供应商提供了零件的工程数；

$$\gamma_{S\#,\ COUNT(distinct\ J\#)\rightarrow sum\_j}(SPJ)$$

hw1

供应商：S (S#, sname, status, city)

零件：P (P#, pname, color, weight, city)

工程：J(J#, jname, city)

供应：SPJ (S#, J#, P#, QTY)　表示供应商 S#为工程 J#供应了 QTY 数量的零件 P#

(3) 求工程号为'J01'的工程所使用的每种零件的零件号，零件名以及数量；

$$\Pi_{P\#,\ pname,\ sum\_QTY}((\gamma_{P\#,\ SUM(QTY)\rightarrow sum\_QTY}(\sigma_{J\#='J01'}(SPJ)))\bowtie P)$$

$$\Pi_{P\#,\ pname,\ sum\_QTY}(\gamma_{P\#,\ SUM(QTY)\rightarrow sum\_QTY}(\sigma_{J\#='J01'}(SPJ\bowtie P)))//\text{错误，缺失属性}\ pname$$

注：分组操作之后得到的属性，对于每一组，产生一个如下内容的元组：该组的分组属性值，该组中所有元组对列表L的属性聚集操作的结果。

hw1

供应商：S (S#, sname, status, city)

零件：P (P#, pname, color, weight, city)

工程：J(J#, jname, city)

供应：SPJ (S#, J#, P#, QTY)　表示供应商 S#为工程 J#供应了 QTY 数量的零件 P#

(4) 求为工程（包括不同的工程）累计供应了 10 种以上不同零件的供应商名称列表；

$$\Pi_{sname}\left(\sigma_{sum\_P\#>10}\left(\gamma_{S\#,\ COUNT(P\#)\rightarrow sum\_P\#}(SPJ)\bowtie S\right)\right)$$

注：题目要求是10种以上零件种类，不是10个以上零件数目。

hw1

供应商：S (S#, sname, status, city)

零件：P (P#, pname, color, weight, city)

工程：J(J#, jname, city)

供应：SPJ (S#, J#, P#, QTY)　表示供应商 S#为工程 J#供应了 QTY 数量的零件 P#

(5) 求为所有位于'Hefei'的工程都提供了零件的供应商号和供应商名称。

$$\Pi_{S\#,\ sname}((\Pi_{S\#,\ J\#}(SPJ) \div \Pi_{J\#}(\sigma_{city='Hefei'}(J))) \bowtie S)$$

注：使用除操作，要注意除操作得到的属性集。

韦宇霏

HOMEWORK #2

评分

- 满分10分
- 共11小题，出错的每一小题扣0～0.5分
- 平均分7.76分

2-1、

假设 S 表中有如下记录:

S#	Name	age	Class
S1	John	17	c1
S2	Rose	18	c2
S3	Mary	20	c4

请写出下面两个语句全部执行后的输出结果。

```
Update S Set Age=20 and Class='c4' Where S#='S1';
Select * From S Where S#='S1';
```

2-2、

给定下面的基本表:

- 学生表 Student(sno: char(10), sname: varchar(50), birthdate: date)
- 课程表 Course(cno: char(10), cname: varchar(50), type: int, credit: float)
- 选课表 SC(sno, cno, score: float, term: int)

其中：type 是整型，0 表示必修课，1 表示选修课，2 表示通识课，3 表示公选课。credit 表示课程学分。term 表示第几学期，取值范围为 1-8。

2-2(1) 查询姓名中含有“科”字的学生学号和姓名

疑问点：

1. 冗余：‘%科’ ‘科%’

2-2(2) 查询学分不低于 3 分的必修课课程号和课程名

错误：

1. **缺少条件“必修课”**
2. **写成**`credit>3`
3. **写成**`score>=3`

2-2(3) 查询选修了公选课但是缺少成绩的学生学号和姓名

错误：

1. 没有去重
2. 缺少连接条件
3. 缺少条件“公选课”
4. 缺少条件“缺少成绩”
5. 使用=null判断是否为null

2-2(3) 查询选修了公选课但是缺少成绩的学生学号和姓名

参考写法1：

```
select distinct S.sno, sname
from Student S, Course C, SC
where S.sno=SC.sno and C.cno=SC.cno and type=3 and score is null;
```

参考写法2：

```
select sno, sname from Student S
where exists (select * from Course C, SC
              where S.sno=SC.sno and C.cno=SC.cno
              and type=3 and score is null);
```

2-2(4) 查询年龄大于 20 的学生学号、姓名和年龄

错误：

1. **认为今年就是2022年、认为今天就是2022年4月x日**
2. **在select子句用as取别名后，在where子句使用这个别名**
3. **认为birthdate就是年龄**
4. **写成age>=20**

疑问点：

1. **写成year(curdate())-year(birthdate)，不扣分**

2-2(4) 查询年龄大于 20 的学生学号、姓名和年龄

参考写法1：

```
select sno, sname, timestampdiff(YEAR,birthdate,now()) as age
from Student
where timestampdiff(YEAR,birthdate,now())>20;
```

参考写法2：

```
select sno, sname, timestampdiff(YEAR,birthdate,now()) as age
from Student
having age>20;
```

2-2(5) 查询已选必修课总学分大于 16 并且所选通识课成绩都大于 75 分的学生姓名

错误：

1. **计算已选必修课总学分时没有去重**
2. **未考虑学生可能重名，用sname而不是sno做intersect或minus**

疑问点：

1. **判断通识课成绩都大于75分时，认为一门课程有一次超过75即可**
2. **最终结是否加distinct都正确（参考写法未加）**

2-2(5) 查询已选必修课总学分大于 16 并且所选通识课成绩都大于 75 分的学生姓名

参考写法：

```
select sname from Student
where sno in (select sno
              from (select distinct sno, cno from SC)disSC, Course C
              where type=0 and disSC.cno=C.cno
              group by sno
              having sum(credit)>16
             )
and sno not in (select sno from SC, Course C
                where type=2 and SC.cno=C.cno and score<=75
               );
```

2-2(6) 查询已经修完所有必修课且成绩合格的学生学号和姓名

错误：

1. **用count判断是否修完所有必修课，但没有考虑存在重修，未去重**
2. **连接条件不完整**

2-2(6) 查询已经修完所有必修课且成绩合格的学生学号和姓名

参考写法1（认为只需要有一次合格就算合格）：

```
select sno, sname
from Student S
where not exists
    (select * from Course C
     where type=0 and not exists
         (select * from SC
          where S.sno=SC.sno and C.cno=SC.cno and score>=60
         )
    );
```

2-2(6) 查询已经修完所有必修课且成绩合格的学生学号和姓名

参考写法2（认为每次都合格才算合格）：

```
select sno, sname
from Student S
where not exists
    (select * from Course C
     where type=0 and not exists
         (select * from SC
          where S.sno=SC.sno and C.cno=SC.cno))
and not exists
    (select * from Course C, SC
     where S.sno=SC.sno and C.cno=SC.cno and type=0 and (score is
null or score<60));
```

2-2(6) 查询已经修完所有必修课且成绩合格的学生学号和姓名

参考写法3（认为只需要有一次合格就算合格，用count判断修完所有必修课）：

```
select sno, sname
from Student S
where (select count(*) from
          (select C.cno from Course C, SC
           where type=0 and S.sno=SC.sno and C.cno=SC.cno
           group by C.cno
           having max(score)>=60)temp)
      =(select count(*) from Course where type=0);
```

2-2(7) 查询每门课程的课程名、课程类型、平均成绩和不及格率，要求结果按通识课、必修课、选修课、公选课顺序排列

错误：

1. 出现最高分、最低分字段
2. 使用子查询统计不及格数时，缺少不及格率为0的课程
3. 使用sum、count来统计不及格数，但计算方式错误
4. 认为分数为null就是0分、不及格或及格

疑问点：

1. 计算不及格率时，分母使用count(*)，包含了成绩为null的记录，没有扣分
2. 最终结果是否加distinct都正确（参考写法未加）
3. 计算平均成绩和不及格率时，按人次还是按人算都正确（参考写法按人次）
4. 用select选择没有出现在group by中、但函数依赖于group by子句的列，没有扣分（SQL 99, only_full_group_by）
5. 通过除以零得到null的，没有扣分（error_for_division_by_zero）

2-2(7) 查询每门课程的课程名、课程类型、平均成绩和不及格率，要求结果按通识课、必修课、选修课、公选课顺序排列

参考写法1（使用子查询统计不及格数）：

```
select cname, type, avgscore, failnum/nullif(allnum,0) as
failrate
from Course C,
(select cno, avg(score) as avgscore, count(score) as allnum
from SC group by cno)A,
((select cno, count(score) as failnum from SC where
score<60 group by cno) union (select cno, 0 from SC group
by cno having min(score)>=60))F
where C.cno=A.cno and A.cno=F.cno
order by decode(type,2,1,0,2,1,3,4); #由Oracle提供
```

2-2(7) 查询每门课程的课程名、课程类型、平均成绩和不及格率，要求结果按通识课、必修课、选修课、公选课顺序排列

参考写法2（使用count统计不及格数）：

```
select cname, type, avg(score) as avgscore,
count(case when score<60 then 2022 else null end)
/nullif(count(score),0) as failrate
from Course C, SC
where C.cno=SC.cno
group by C.cno, cname, type
order by case type when 2 then 1
                  when 0 then 2
                  when 1 then 3
                  when 3 then 4 end;
```

2-2(7) 查询每门课程的课程名、课程类型、平均成绩和不及格率，要求结果按通识课、必修课、选修课、公选课顺序排列

参考写法3（使用sum统计不及格数）：

```
select cname, type, avg(score) as avgscore,
sum(score<60)/nullif(count(score),0) as failrate
from Course C, SC
where C.cno=SC.cno
group by C.cno, cname, type
order by field(type,2,0,1,3); #由MySQL提供
```

2-2(8) 查询存在课程重修不及格情况的学生学号、姓名以及重修不及格的课程号和课程名

错误：

1. 认为不及格才能重修
2. 认为是指第一次不及格且有重修
3. 认为没有分数就是0分或不及格
4. 多统计了第一次修读就不及格的情况
5. 错误的嵌套查询顺序导致回答的重数既不是1次，也不是重修不及格次数

疑问点：

1. 回答的重数是重修不及格次数
2. 认为是仅指最后一次不及格

2-2(8) 查询存在课程重修不及格情况的学生学号、姓名以及重修不及格的课程号和课程名

参考写法：

```
select distinct S.sno, sname, C.cno, cname
from Student S, Course C, SC SC1
where S.sno=SC1.sno and C.cno=SC1.cno and
score<60 and exists
(select * from SC SC2 where SC1.sno=SC2.sno
and SC1.cno=SC2.cno and SC1.term>SC2.term);
```

2-3、

现有一个员工表 E(no, name, level)，其中每个员工有个 int 型的 level 字段 (1:一般员工，2:经理，3:顾问)，现在要求输出员工的 no，name 和 level，并且按照"经理，一般员工，顾问"的顺序输出结果，请使用 SQL 语句给出解答。

参考写法：

```
select * from E
order by field(level,2,1,3);
```

2-4、

给定关系模式 R(A,B)、S(B,C)和 T(C,D)，已知有下面的关系代数表达式，其中 p 是仅涉及属性 R.A 的谓词，q 是仅涉及 R.B 的谓词，m 是涉及仅 S.C 的谓词。请写出与此关系代数表达式对应的 SQL 查询语句:

$$\pi_D((\sigma_{(p \wedge q \wedge m)}(R \bowtie S)) \bowtie T)$$

错误：

1. 最终结果没有去重
2. 随意改变运算顺序
3. 使用多条select语句回答
4. 认为p、q、m是属性或值
5. 使用数学符号或逗号代替and

2-4、

参考写法1：

```
select distinct D
from (select A, R.B, C from R, S
      where R.B=S.B and p(R.A) and q(R.B) and m(S.C)
     )pqmRS, T
where pqmRS.C=T.C;
```

参考写法2：

```
select distinct D
from (select * from R natural join S where p(A) and
q(B) and m(C))pqmRS natural join T;
```

其他问题

1. 不给from中的select子查询起别名
2. 给from中的表起别名后，仍然使用原名
3. 使用多表的同名属性时，不加表名前缀
4. 以为用join、inner join、outer join就是自然连接，不写连接条件
5. 使用视图，但创建前不检查是否存在同名视图，且用后也不销毁
6. 试图通过直接先后执行多个select语句或通过union、union all来排序
7. 使用case，最后缺少end
8. 混淆exists和in
9. 试图select没有出现在group by中、也不函数依赖于group by子句的列

王灿

HOMEWORK #3

HW3

- 1、已知有关系模式 R(A, B, C, D, E)，R 上的一个函数依赖集如下：F={A→BC, B→CE, A→B, AB→C, AC→DE, E→A }
- （1）求出 F 的最小函数依赖集（要求写出求解过程）
  - 将右边写成单属性并去除重复FD(分解率)
    - F={A→B,A → C, B→C,B → E, A→B, AB→C, AC→D, AC→ E, E→A }
    - F={A→B,A → C, B→C,B → E, AB→C, AC→D, AC→ E, E→A }
  - 消去左部冗余属性
    - A→B,AB → C可推出A → C
    - A→C,AC → D,AC →E可推出A → D, A → E
    - F={A→B,A → C, B→C,B → E,A→D, A→ E, E→A }
  - 消去冗余函数依赖
    - A → C, A→E冗余
    - F={A→B, B→C,B → E,A→D, E→A }

（2）求 R 的候选码，并给出证明

X → U ∈F+，则X是R的一个超码，如果同时不存在X的真子集Y，使得Y → U成立，则X是R的一个候选码。

因为A → B，A → C, A → D, A → E，并且不存在A的真子集Y，使得Y→ U成立，所以A为候选码。

因为E → A，所以E → ABCDE，并且不存在E的真子集Y，使得Y→ U成立，所以E为候选码。同理B为候选码。

所以A、E、B为候选码。

- 2、已知有关系模式$R(A, B, C, D, E)$，R上的一个函数依赖集如下：
- F={A→BD, BC→D, DCE→A, D→B, E→D }
- （1）求出 F 的最小函数依赖集
  - 将右边写成单属性
    - F ={A→B,A→D, BC→D, DCE→A, D→B, E→D }
  - 消去左部冗余属性
    - E→D,DCE → A可推出CE →A
    - F ={A→B,A→D, BC→D, CE→A, D→B, E→D }
  - 消去冗余函数依赖
    - A → B冗余
    - F ={A→D, BC→D, CE→A, D→B, E→D }

（2）求 R 的候选码

由 CE→A,A→D, D→B可得CE→U，

并且不存在CE的真子集Y，使得Y→U成立，所以CE为候选码

（3）R 属于第几范式？为什么？

因为B局部依赖E ，所以R属于 1NF

□（4）请将 R 无损连接并且保持函数依赖地分解到 3NF。

最小函数依赖集：F ={A→D, BC→D, CE→A, D→B, E→D }

对F按相同左部分组，得到q={R1(A,D),R2(B,C,D),R3(A,C,E),R4(E,D)}

主码为{C,E}

P =q U R(C,E)={R1(A,D),R2(B,C,D),R3(A,C,E),R4(E,D)}

- **3、现有关系模式: R(A, B, C, D, E, F, G)，R 上的一个函数依赖集： F={AB→E, A→B, B→C, C→D}**
- **（1）该关系模式满足第几范式? 为什么?**

  最小函数依赖集为F={A→B, B →C, C →D, A →E}

  候选码为{A,F,G}，有局部依赖，比如B局部依赖A

  所以是1NF.

（2）如果将关系模式 R 分解为： R1(A, B, E），R2(B, C, D)，R3(A, F, G)，该数据库模式最高 满足第几范式?

- R1(A,B,E)，F1={AB→E，A→B} ={A→E，A→B}，为BCNF
- R2(B,C,D)，F2={B→C，C→D}，主码为B，而B →C,C →D,D传递依赖于B，所以不满足3NF，故最高为2NF.
- R3(A,F,G)，F3=Φ，为BCNF。
- 所以最高满足第二范式。

（3）请将关系模式 R 无损连接并且保持函数依赖地分解到3NF，要求给出具体步骤。

最小FD集：F={A→B, B →C, C →D, A →E}

把所有不在F中出现的属性组成一个关系模式R′={F,G}

对F按相同左部分组，得到q={R1(A,B,E),R2(B,C),R3(C,D),R′(F,G)}

主码为{A,F,G}

P =q U R(A,F,G)={R1(A,B,E),R2(B,C),R3(C,D),R(A,F,G)}

- （4）请将关系模式 R 无损连接地分解到 BCNF，要求给出步骤
- A->E 是独立的，先后消除不影响，先消除它，得到 R1(A,E),R2(A,B,C,D,F,G)
- A->B,B->C,C->D是连续的传递依赖，消除的先后顺序会影响最终结果。
- 先消除R2中的A->B:Rx(A,C,D,F,G) 依赖变成A->C,C->D
  - 再消除A->C:R1(A,E),R2(A,B),R3(A,C),R4(A,D),R5(A,F,G)
  - 再消除C->D:R1(A,E),R2(A,B),R3(C,D),R4(A,C),R5(A,F,G)
- 先消除R2中的B->C:Rx(A,B,D,F,G) 依赖变成A->B,B->D.
  - 再消除A->B:R1(A,E),R2(B,C),R3(A,B),R4(A,D),R5(A,F,G)
  - 再消除B->D:R1(A,E),R2(B,C),R3(B,D),R4(A,B),R5(A,F,G)
- 先消除R2中的C->D:Rx(A,B,C,F,G) 依赖变成A->B,B->C
  - 再消除A->B:R1(A,E),R2(C,D),R3(A,B),R4(A,C),R5(A,F,G)
  - 再消除B->C:R1(A,E),R2(C,D),R3(B,C),R4(A,B),R5(A,F,G)
- 综上，BCNF分解总共有6种不同的答案。
- 第三问的答案也满足BCNF，也可为答案。

汪洪韬

HOMEWORK #4

4.1

1、假设我们在数据库中设计了如下基本表来存储文献：paper(id: int, title: varchar(200), abstract: varchar(1000))。最常见的文献查询可以描述为“查询 title 中同时包含给定关键词的文献”，关键词可以是一个，也可以是多个。请回答下面问题（假设所有文献都是英文文献）：

1) 假如在 title 上创建了 B+-tree 索引，能不能提高此查询的效率（须解释理由）？

2) 由于文献 title 的关键词中存在很多重复词语，因此上述文献查询可以借鉴我们课上讲述的支持重复键值的辅助索引技术来进一步优化。请基于此思想画出一种优化的索引结构，简要说明该索引上的记录插入过程以及文献查询过程。

4.1 (1)

1) 假如在 title 上创建了 B+-tree 索引，能不能提高此查询的效率（须解释理由）？

不能。

文献查询并不是精确(whole-key)点查询，而是范围(partial-key)查询，查询的单词可能出现在title的任意位置。由于B+树是根据键的字典序排序的，仅支持前缀查询，不支持中缀和后缀范围查询。因此使用B+树结构仍要扫描所有数据项。

4.1（2）

2) 由于文献 title 的关键词中存在很多重复词语，因此上述文献查询可以借鉴我们课上讲述的支持重复键值的辅助索引技术来进一步优化。请基于此思想画出一种优化的索引结构，简要说明该索引上的记录插入过程以及文献查询过程。

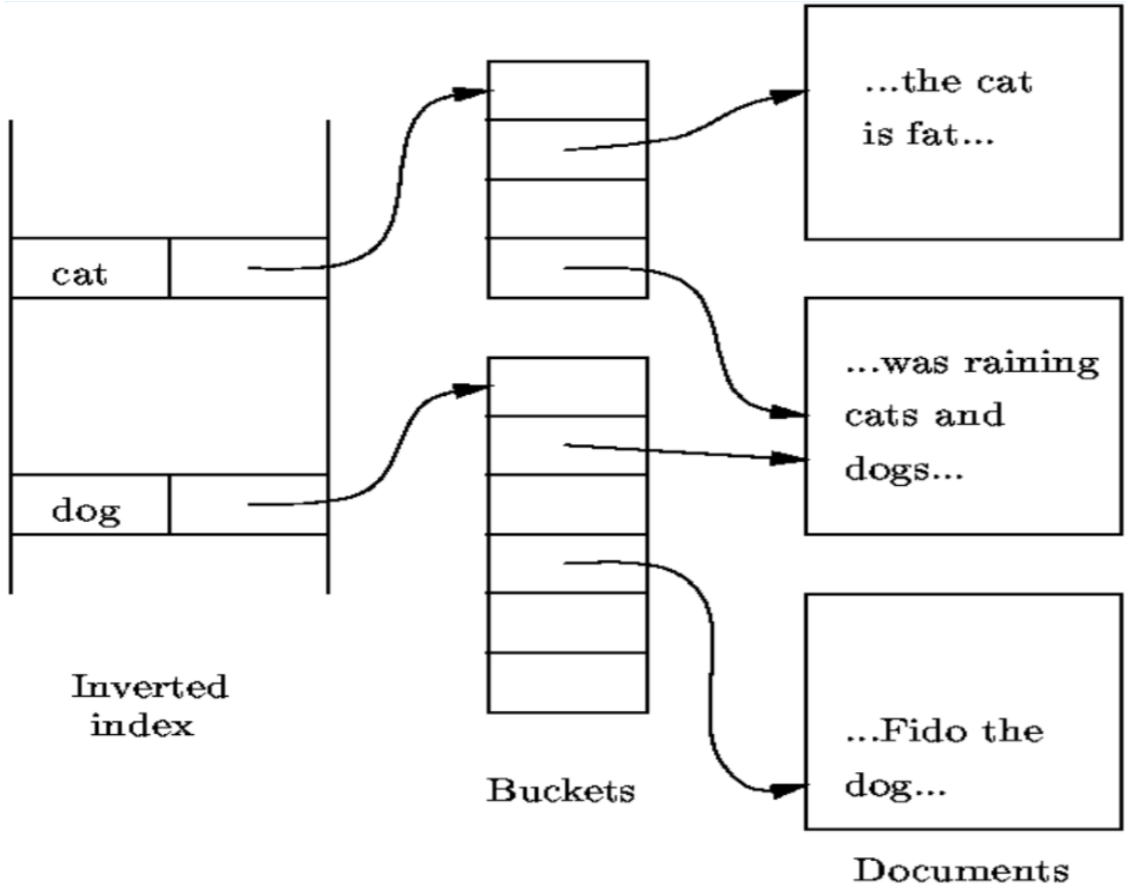
使用倒排索引。倒排索引根据单词索引到title包含该单词的所有文档。

相比于B+树结构，采用倒排索引优点如下：

(a). 避免重复单词的存储

(b). 对于文献查询不需要扫描所有数据项，仅通过少数几次索引查询即可完成搜索。

4.1 (2)



插入：

对文档的title进行分词，然后将各个关键词插入到索引中。

（1）查询关键词是否已经存在，若存在则直接将指针插入对应的桶中；

（2）若不存在，则插入关键词，增加对应的桶以及指向文档的指针；

查询：

（1）单个关键词；返回得到的文档集合即可；

（2）多个关键词；返回各个关键词查询结果的文档集合的交集；

4.2

2. 假设有如下的键值，现用 5 位二进制序列来表示每个键值的 hash 值。回答问题：

A [11001]　　B [00111]　　C [00101]　　D [00110]　　E [10100]　　F [01000]　　G [00011]

H [11110]　　I [10001]　　J [01101]　　K [10101]　　L [11100]　　M [01100]　　N [11111]

1）如果将上述键值按 A 到 N 的顺序插入到可扩展散列索引中，若每个桶大小为一个磁盘块，每个磁盘块最多可容纳 3 个键值，且初始时散列索引为空，则全部键值插入完成后该散列索引中共有几个桶？并请写出键值 E 所在的桶中的全部键值。

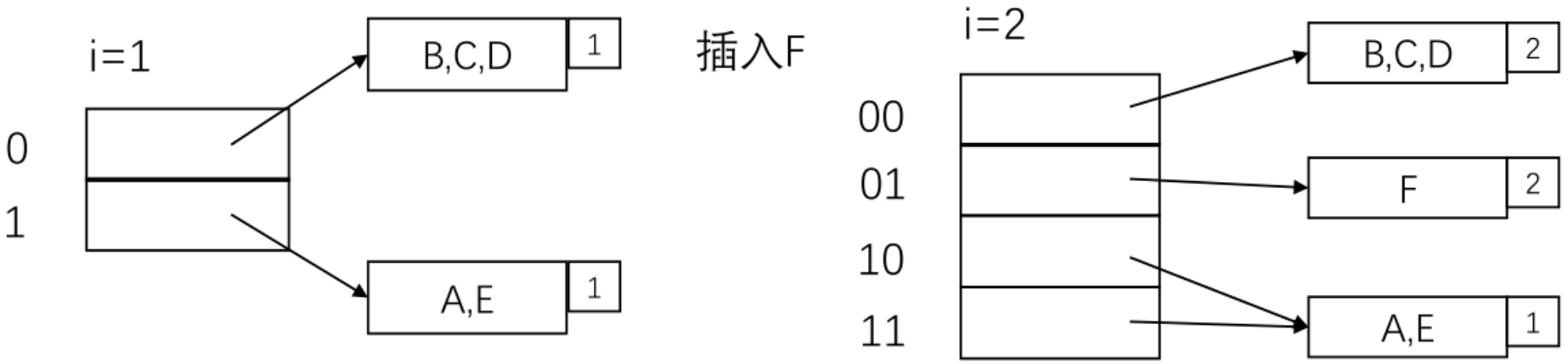
2）前一问题中，如果换成线性散列索引，其余假设不变，同时假设只有当插入新键值后空间利用率大于 80%时才增加新的桶，则全部键值按序插入完成后该散列索引中共有几个桶？并请写出键值 B 所在的桶中的全部键值（包括溢出块中的键值）。

4.2 (1)

2. 假设有如下的键值，现用 5 位二进制序列来表示每个键值的 hash 值。回答问题：

A [11001]	B [00111]	C [00101]	D [00110]	E [10100]	F [01000]	G [00011]
H [11110]	I [10001]	J [01101]	K [10101]	L [11100]	M [01100]	N [11111]

1）如果将上述键值按 A 到 N 的顺序插入到可扩展散列索引中，若每个桶大小为一个磁盘块，每个磁盘块最多可容纳 3 个键值，且初始时散列索引为空，则全部键值插入完成后该散列索引中共有几个桶？并请写出键值 E 所在的桶中的全部键值。

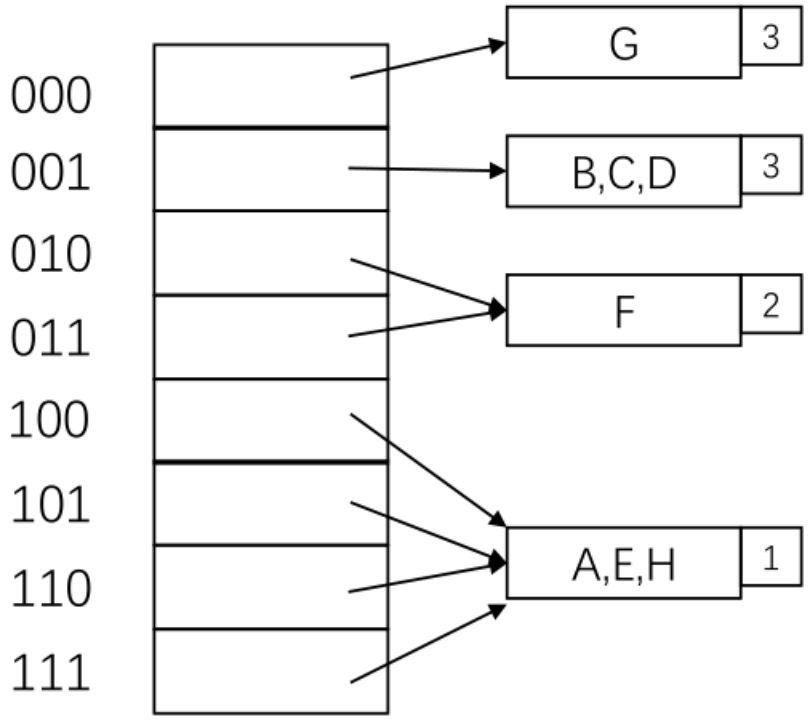


4.2 (1)

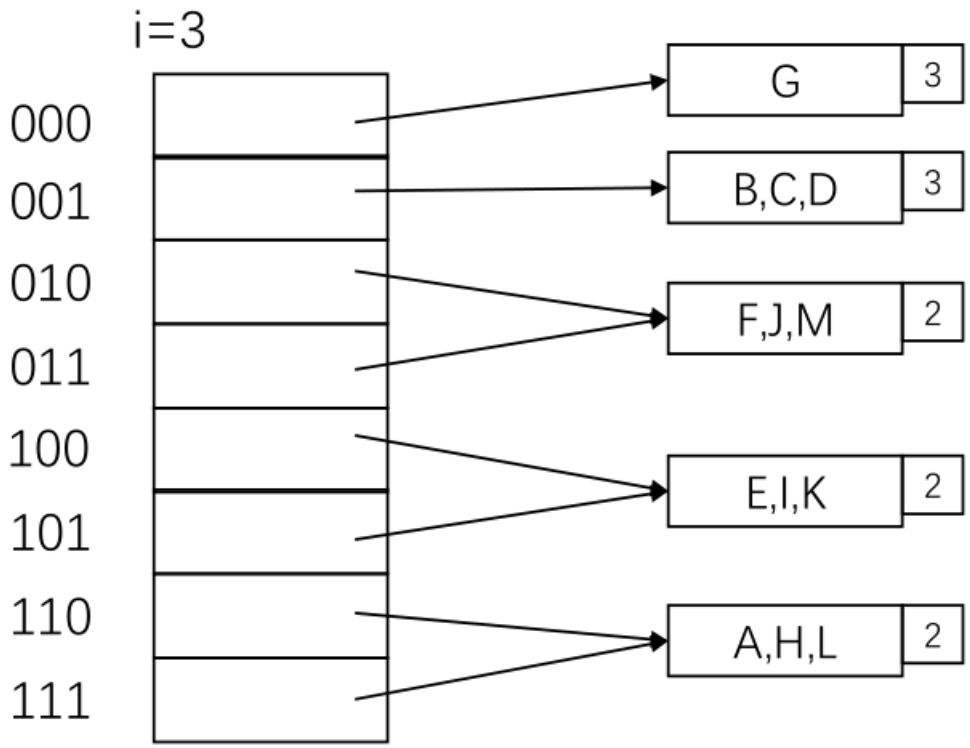
A [11001] B [00111] C [00101] D [00110] E [10100] F [01000] G [00011]

H [11110] I [10001] J [01101] K [10101] L [11100] M [01100] N [11111]

插入G i=3

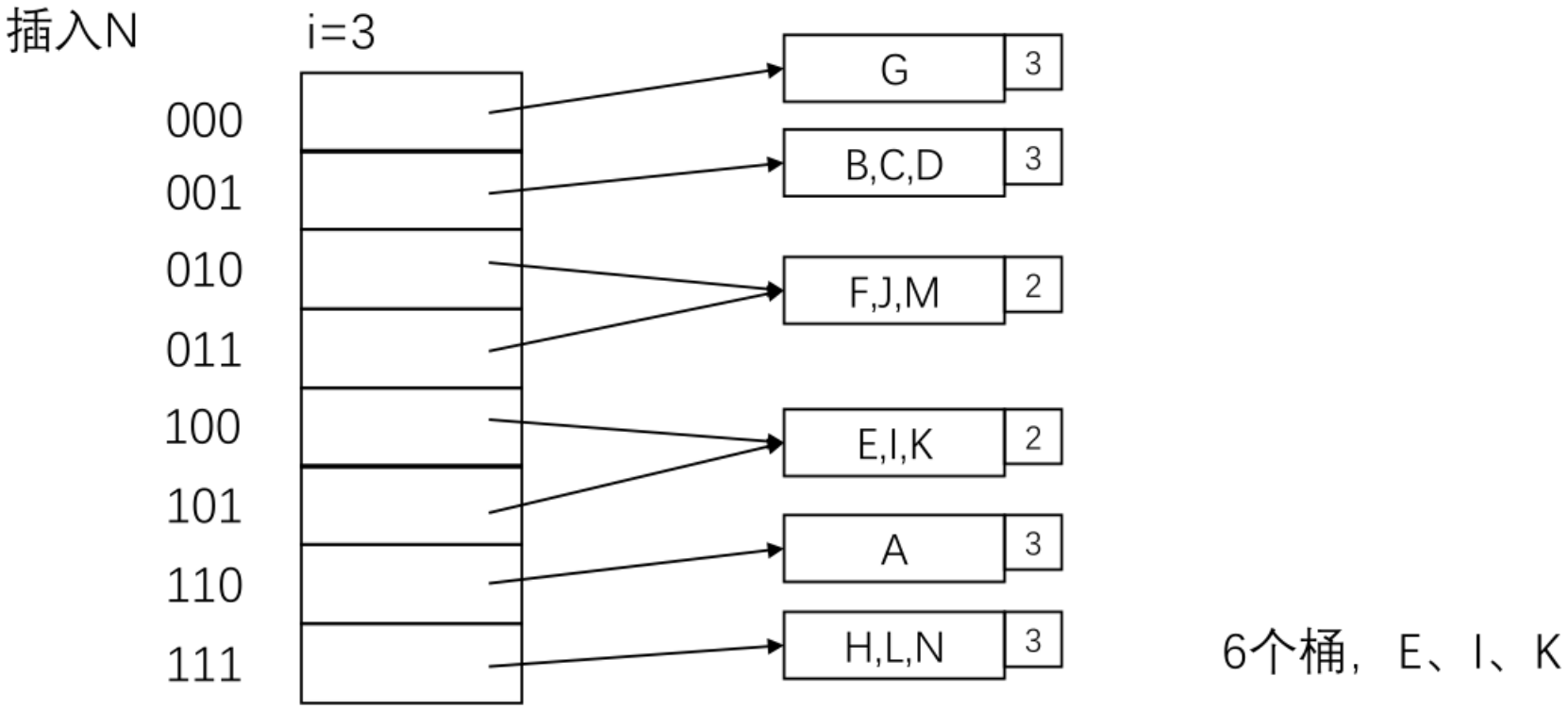


插入I i=3



4.2 (1)

A [11001]	B [00111]	C [00101]	D [00110]	E [10100]	F [01000]	G [00011]
H [11110]	I [10001]	J [01101]	K [10101]	L [11100]	M [01100]	N [11111]



6个桶，E、I、K

4.2 (2)

2. 假设有如下的键值，现用 5 位二进制序列来表示每个键值的 hash 值。回答问题：

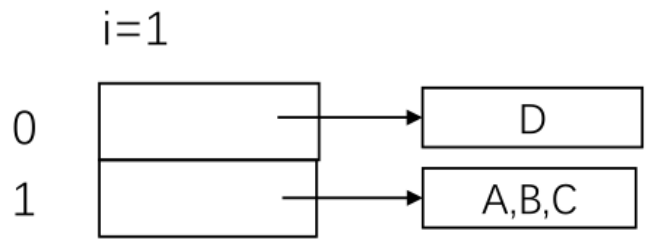
A [11001]　　B [00111]　　C [00101]　　D [00110]　　E [10100]　　F [01000]　　G [00011]

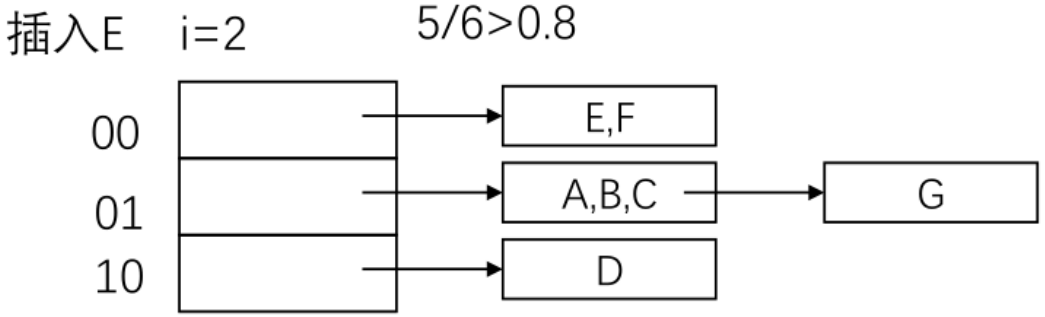
H [11110]　　I [10001]　　J [01101]　　K [10101]　　L [11100]　　M [01100]　　N [11111]

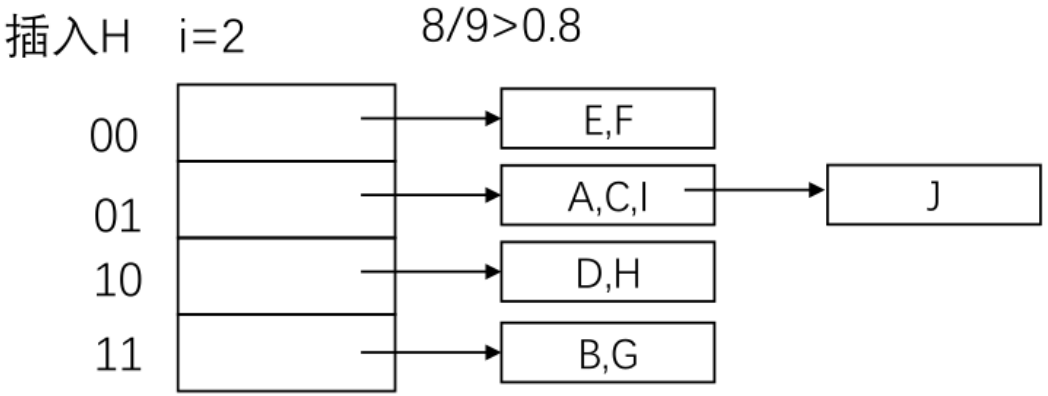
2）前一问题中，如果换成线性散列索引，其余假设不变，同时假设只有当插入新键值后空间利用率大于 80%时才增加新的桶，则全部键值按序插入完成后该散列索引中共有几个桶？并请写出键值 B 所在的桶中的全部键值（包括溢出块中的键值）。

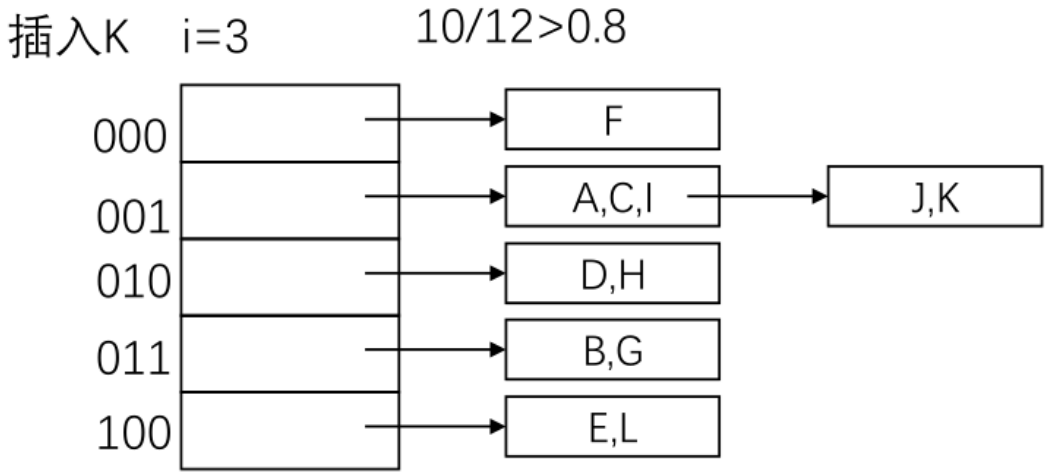
A [11001] B [00111] C [00101] D [00110] E [10100] F [01000] G [00011]

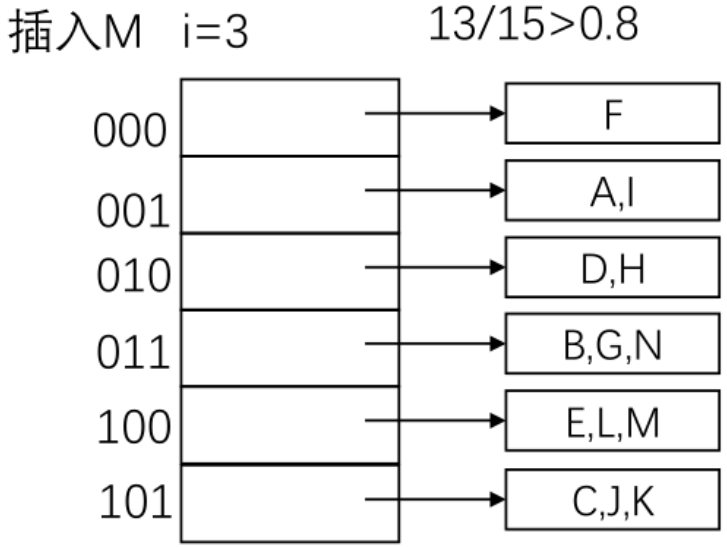
H [11110] I [10001] J [01101] K [10101] L [11100] M [01100] N [11111]











6个桶，B、G、N

4.3

3、对于 B+树，假设有以下的参数：

参数	含义	参数	含义
N	记录数	S	读取一个磁盘块时的寻道时间
n	B+树的阶，即节点能容纳的键数	T	读取一个磁盘块时的传输时间
R	读取一个磁盘块时的旋转延迟	m	在内存的 m 条记录中查找 1 条记录的时间（线性查找）

假设所有磁盘块都不在内存中。现在我们考虑一种压缩 B+树，即对 B+树的节点键值进行压缩存储。假设每个节点中的键值压缩 1 倍，即每个节点在满的情况下可压缩存储 $2n$ 个压缩前的键值和 $2n+1$ 个指针。额外代价是记录读入内存后必须解压，设每个压缩键值的内存解压时间为 c。给定 N 条记录，现使用压缩 B+树进行索引，请问在一棵满的 n 阶压缩 B+树中查找给定记录地址的时间是多少？（使用表格中的参数表示，$n+1$ 或 $n-1$ 可近似表示为 n）？

4.3

3、对于 B+树，假设有以下的参数：

参数	含义	参数	含义
N	记录数	S	读取一个磁盘块时的寻道时间
n	B+树的阶，即节点能容纳的键数	T	读取一个磁盘块时的传输时间
R	读取一个磁盘块时的旋转延迟	m	在内存的 m 条记录中查找 1 条记录的时间（线性查找）

只有叶子节点存放数据，假设层数为k，则有$(2n+1)^{k-1}\times 2n = N$，可得$k = \log_{2n} N$
故时间为：$(R + S + T + 2nc + 2n)\times \log_{2n} N$

刘高聪

HOMEWORK #5

HW 5

- 1. 什么是事务的 ACID 性质？请给出违背事务 ACID 性质的具体例子，每个性质举一个例子。
  - **原子性 Atomicity**
    - 事务是不可分的原子，其中的操作要么都做，要么都不做
    - 反例：转账中A余额减去100并且写入磁盘，但此时系统崩溃，且未做恢复。
  - **一致性 Consistency**
    - 事务的执行保证数据库从一个一致状态转到另一个一致状态（满足所有完整性约束）
    - 反例：举一个违反完整性约束的例子，如上例转账。

HW 5

- 1. 什么是事务的 ACID 性质？请给出违背事务 ACID 性质的具体例子，每个性质举一个例子。
  - **隔离性 Isolation**
    - 多个事务一起执行时相互独立
    - 反例：同一个事务对同一个数据的两次读的值不同。
  - **持久性 Durability**
    - 事务一旦成功提交，就在数据库永久保存
    - 反例：事务已提交，而改变的数据仍然还在磁盘缓冲区中排队等待写入磁盘中，导致系统重启后commit的结果无效。
    - 反例：磁盘损坏导致数据丢失，且没有备份。

HW 5

- 2. 目前许多 DBMS 例如 MySQL 都默认不支持嵌套事务（即在一个事务内部又开始另一个事务），请分析一下：如果 DBMS 支持嵌套事务，将面临哪些问题（至少写出2 点以上并且要给出自己的分析）？

  假设事务A调用事务B

  - **原子性和持久性矛盾。**
    - 如果B commit，但之后A rollback,此时A的原子性与B的持久性冲突。
  - **一致性影响**
    - B开始的时候可能不处于一致性状态。

HW 5

- 2. 目前许多 DBMS 例如 MySQL 都默认不支持嵌套事务（即在一个事务内部又开始另一个事务），请分析一下：如果 DBMS 支持嵌套事务，将面临哪些问题（至少写出2 点以上并且要给出自己的分析）？

  假设事务A调用事务B

  - **隔离性**
    - A事务需要先修改一条数据v再调用B，B事务也需要修改数据v。如果系统采用了锁机制，则会陷入死锁。否则如果B读的是事务A修改后的数据，则发生了脏读；如果读的是修改前的数据，则发生了脏写，两种情况均破坏了隔离性。
  - **死循环**
    - AB相互调用，可能陷入死循环。

HW 5

3. 下面是一个数据库系统开始运行后的日志记录，该数据库系统支持检查点。

1) <T1, Begin Transaction>
2) <T1, A, 49, 20>
3) <T2, Begin Transaction>
4) <T1, B, 250, 20>
5) <T1, A, 75, 49>
6) <T2, C, 35, 20>
7) <T2, D, 45, 20>
8) <T1, Commit Transaction>
9) <T3, Begin Transaction>
10) <T3, E, 55, 20>
11) <T2, D, 46, 45>
---------- ①
12) <T2, C, 65, 35>
13) <T2, Commit Transaction>
14) <T3, Commit Transaction>
---------- ②
15) **<CHECKPOINT>**
16) <T4, Begin Transaction>
17) <T4, F, 100, 20>
18) <T4, G, 111, 20>
19) <T4, F, 150, 100>
---------- ③
20) <T4, Commit Transaction>

设日志修改记录的格式为 <Tid, Variable, New value, Old value>，请给出对于题中所示①、②、③三种故障情形下，数据库系统恢复的过程以及数据元素 A, B, C, D, E, F 和 G 在执行了恢复过程后的值。

① T1：redo

T2、T3：undo

先undo：D=45, E=20, D=20, C=20

再redo：A=49, B=250, A=75

所以，A=75, B=250, C=20, D=20, E=20, F=20, G=20

② T1、T2、T3：redo

redo：A=49, B=250, A=75, C=35, D=45, E=55, D=46, C=65

所以，A=75, B=250, C=65, D=46, E=55, F=20, G=20

③ 检查点前：A=75, B=250, C=65, D=46, E=55, F=20, G=20

检查点后：

T4：undo

undo：F=100, G=20, F=20

所以，A=75, B=250, C=65, D=46, E=55, F=20, G=20

汪洪韬

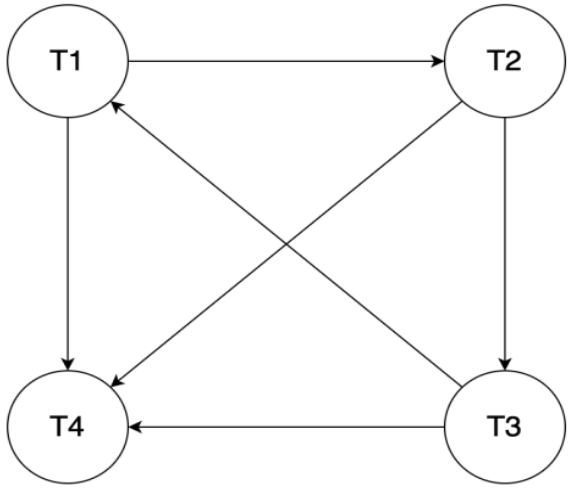
HOMEWORK #6

6.1

1. 判断下面的并发调度是否冲突可串？如果是，请给出冲突等价的串行调度事务顺序；如果不是，请解释理由。

w3(D); r1(A); w2(A); r4(A); r1(C); w2(B); r3(B); r3(A); w1(D); w3(B); r4(B); r4(C); w4(C); w4(B)

根据并发调度作出其对应的优先图：



可以看到该优先图中存在环路，因此该并发调度不是冲突可串。

6.2

2. 证明：如果一个并发调度 S 中的所有事务都遵循 2PL，则该调度必定是冲突可串调度。

反证法。如果发生了不可串调度情况，则该调度不满足冲突可串性，那么它的优先图必存在环。设其上事务为$T_1, T_2 \ldots T_n$，操作的数据为$D_1, D_2 \ldots D_n$，则有

- $uL_1(D_1) \rightarrow L_2(D_1)$
- $L_2(D_1) \rightarrow uL_2(D_2)$
- $uL_2(D_2) \rightarrow L_3(D_2)$
- **…**
- $uL_n(D_n) \rightarrow L_1(D_n)$
- $L_1(D_n) \rightarrow uL_1(D_1)$

事务T_1在释放锁之后又申请了锁，违背了2PL，故不成立。

6.3

3. 如果一个并发调度中的所有事务都遵循两阶段锁协议，该并发调度还会出现脏读问题吗？如果不会，请解释理由；如果会，请给出一个例子。

t	T1	T2
1	xL1(A)	
2	A=A-100	
3	xL1(B)	
4	Write(A)	
5	xL1(B)	
6	Unlock(A)	
7		sL2(A)
8	B=B+100	Read(A)
9	Write(B)	
10	Unlock(B)	
11	Rollback	

6.4

4. 考虑下面两个事务：

T1 = r1(A) w1(A) r1(B) w1(B)
T2 = r2(B) r2(A) w2(A) w2(B)

假设调度器插入了 X 锁动作，如下所示：
T1 = xL1(A) r1(A) w1(A) xL1(B) r1(B) w1(B) , AFTER COMMIT: U1(A) U1(B)
T2 = xL2(B) r2(B) xL2(A) r2(A) w2(A) w2(B) , AFTER COMMIT: U2(A) U2(B)

现在 T1 和 T2 并发执行，问：
1） T1，T2 并发执行能保证数据库的一致性吗？为什么？
2） 如果不使用上面的加锁方式，而是约定事务按先 A 后 B 的顺序请求锁，T1、T2 并发执行会产生死锁吗？为什么？
3） 如果使用等待—死亡(wait-die)死锁预防方案，假设 T1 先开始，那么如果产生死锁时系统将采取什么样的动作？

6.4 (1)

T1 = xL1(A) r1(A) w1(A) xL1(B) r1(B) w1(B) , AFTER COMMIT: U1(A) U1(B)

T2 = xL2(B) r2(B) xL2(A) r2(A) w2(A) w2(B) , AFTER COMMIT: U2(A) U2(B)

现在 T1 和 T2 并发执行，问：

1） T1，T2 并发执行能保证数据库的一致性吗？为什么？

可以，因为这是两阶段锁，由6.2知两阶段锁的调度一定是可串化调度，因此可以保证一致性。(死锁由系统处理)

6.4 (2)

2） 如果不使用上面的加锁方式，而是约定事务按先 A 后 B 的顺序请求锁，T1、T2 并发执行会产生死锁吗？为什么？

不会。约定顺序之后，T1和T2都需要先申请A上的锁，无论哪个事务申请成功，另一个事务都只能等待A锁的释放，而不能申请其他锁。等申请成功的事务提交释放A锁后，另一个事务才能继续执行，因此不会产生死锁。

6.4 (3)

3） 如果使用等待－死亡(wait-die)死锁预防方案，假设 T1 先开始，那么如果产生死锁时系统将采取什么样的动作？

如果采用wait-die方案，最初T1获得A上的锁，T2获得B上的锁，T1申请B上的锁发现被T2占用，因为T1先开始，所以T1进入等待状态，当T2申请A上的锁发现被T1占用，开始回滚，此时T1获得B上的锁顺利运行，T2在T1提交以后获得B上的锁继续执行。

谢 谢！