2020 DB Exam

一

1. B
2. A
3. D
4. B
5. A

二

1. T
2. T
3. F
4. **TODO**
5. F
6. F
7. F **TOCHECK**
8. T
9. F
10. T

三

1. 分别是
   - 外模式: 视图
   - 概念模式: 表
   - 内模式: 文件及其磁盘存储
2. 逻辑独立性: 数据库的概念模式做修改的时候只需要同时修改外模式/模式映像即可保持外模式不变, 使得用户程序不便, 这就保证了数据和用户程序的独立性.

四

1. 不必. 诸如 select 等查询操作不会修改数据库的内容, 因此对数据库恢复毫无影响.
2. 嵌套事务的潜在问题. 若事务 A 内执行事务 B:

    - 若 B 已经提交, A 需要 rollback, 此时 B 修改的持久性已经生效, A 的 rollback 无法撤销 B 的作用, 从而破坏了事务 A 的一致性和原子性.
    - 若 A 和 B 都要修改同一数据, 如果采用了锁机制, 可能会导致死锁.
3. 朴素的 Redo 日志和 脏读有个 P 关系?

五

1. F 的最小函数依赖集
    1. 右边化为单属性并消去冗余:
       $F = \{A \to B, A \to D, BC \to D, DCE \to A, D \to B, E \to D\}$
    2. 消除冗余函数依赖:
       $F = \{A \to D, BC \to D, DCE \to A, D \to B, E \to D\}$
    3. 消除冗余左部属性:
       $F = \{A \to D, BC \to D, CE \to A, D \to B, E \to D\}$
2. R 的候选码: 显然 $\{C, E\}$ 是一个超码, 且 C 和 E 单独均不能成为候选码, 故候选码可以为 $\{C, E\}$
3. D 只依赖于 E, 因此不满足 2NF, 是 1NF
4. 分解:
    1. 保持函数依赖地分解到 3NF
        1. 分类: AD, BCD, ACE, BD, DE
        2. 去掉子集: R1(AD), R2(BCD), R3(ACE), R4(DE)
    2. 无损连接且保持函数依赖地分解到 3NF
        1. R1(AD), R2(BCD), R3(ACE), R4(DE), R5(CE) 中 R5 的属性是 R3 子集
        2. 故最终是 R1(AD), R2(BCD), R3(ACE), R4(DE)

六

1. ```
   select f.fname from faculty f, course c, department d
   where f.fid = c.fid and c.room = '3C102'
       and f.fname like '赵%'
       and f.did = d.did and d.dname = '计算机';
   ```

2. ```
   select s.sid, s.sname from student s
   where not exists (
       select * from course c, faculty f, SC sc
       where c.fid = f.fid and c.cid = sc.cid and s.sid = sc.sid
           and f.fname <> '张三'
   );
   ```

3.
```sql
select d.dname as 'department', count(distinct sc.sid) as count
from department d, course c, SC sc
where c.cid = sc.cid and c.cname = 'DB' and sc.score is null
group by department
order by count DESC;
```

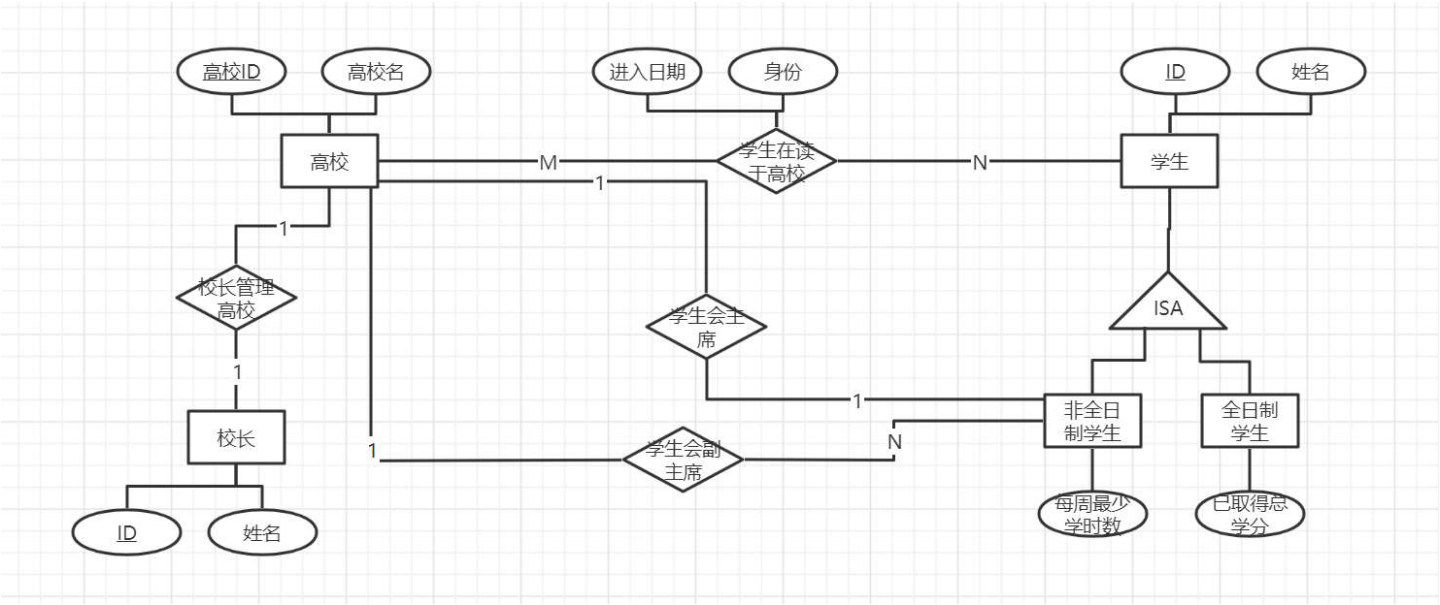
4.
```sql
select s.sid as sid, s.sname as sname
from student s, course ai_c, course db_c, SC ai_sc, SC db_sc
where s.sid = ai_sc.sid and s.sid = db_sc.sid
    and ai_c.cid = ai_sc.cid and ai_c.cname = 'AI'
    and db_c.cid = db_sc.cid and db_c.cname = 'DB'
    and db_sc.score > ai_sc.score;
```

5.
```sql
select s.sname, count(distinct c.cno) as course_count, avg(sc.score) avg_score
from student s, course c, SC sc
group by s.sid
having course_count >= 4 and min(sc.score) >= 95;
```

七

1.

ER 模型图如下



2.

转化为关系模型:

1. 转换实体
    - 高校(高校ID, 名称)
    - 学生(学生ID, 姓名)

- 校长(校长ID, 姓名)
- 非全日制学生(学生ID, 姓名, 每周最少学时)
- 全日制学生(学生ID, 姓名, 已取得总学分)

2. 转换联系
   - 高校:校长 (1:1): 高校(高校ID, 名称, 校长ID)
   - 学生:高校 (M:N): 学生在读于高校(学生ID, 高校ID, 进入日期, 身份)
   - 学生会主席: 非全日制学生:高校 (1:1): 学生会主席(学生ID, 高校ID)
   - 学生会副主席: 非全日制学生:高校 (N:1): 非全日制学生(学生ID, 姓名, 每周最少学时, 高校ID)
3. 从而得到:
   - 高校(高校ID, 名称, 校长ID)
   - 学生(学生ID, 姓名)
   - 校长(校长ID, 姓名)
   - 非全日制学生(学生ID, 姓名, 每周最少学时, 高校ID)
   - 全日制学生(学生ID, 姓名, 已取得总学分)
   - 学生会主席(学生ID, 高校ID)
   - 学生在读于高校(学生ID, 高校ID, 进入日期, 身份)