1、数据库系统中，若事务并发执行的调度是可串行化的，即认为该并发结果是正确的，为什么？(8 分)

事务并发执行的调度是可串行化的，也就是说对于该事务并发执行的调度与该事务的串行调度等价；对于串行调度，各事务的操作没有交叉，没有互相干扰，因此不会产生并发执行时的冲突问题，因此与之等价的事务并发调度也不会产生冲突，即并发结果是正确的。

2、设数据库中包含下列关系:

course(cno, cname, dame)/*课程号，课程名，开课系名*/

enroll(sid, grade, cno, dname1, sectno, pname, dname2)/*学号，成绩， 课程号，开课系名，分班号，任课教师名，任课教师所在系名*/

student(sid, sname, sex, age, year, gpa)/*学号，姓名，性别，年龄，年级，成绩点*/

仔细观察以下各表的实例数据，给出各关系表的主键和外键。(10 分)

course

cno	Cname	Dnamc
302	Programming	Computer Sei
310	Thermodynamics	Chemical Eng
310	Intro to Garbage	Sanitary Eng
365	City Planning	Civil Eng
375	Highway Eng	Civil Eng
461	Geometry	Mathematics

student

sid	sname	Sex	age	Year	gpa
1	Jacobs, T.	M	29	4	3.60
2	Pierson. E.	F	32	2	3.50
3	Zeene. Ben N.	M	21	2	3.90
4	Sulfate, Barry	M	19	2	2.80
5	Form, Clara O.	F	18	1	3.30
6	Scott, Kim	M	20	1	3.80

enroll

Sid	Grade	Cno	dnamel	sectno	Pname	dname2
1	3.00	310	Chemical Eng	1	Brian, C.	Chemical Eng
1	3.00	302	Computer Sci	2	Brown, S.	Computer Sci
3	3.50	375	Civil Eng	1	Clark, E.	Civil Eng
4	4.00	461	Mathematics	1	Clark, E.	Civil Eng
4	3.00	514	Industrial Eng	1	Brian, C.	Chemical Eng
4	3.50	302	Computer Sci	2	Brown, S.	Computer Sci
5	4.00	302	Computer Sci	1	Brian, C.	Chemical Eng
6	4.00	302	Computer Sci	1	Clark, E.	Civil Eng

① course 关系表中，主键为{cno，dname}，没有外键；

② student 关系表中，主键为{sid}，没有外键；

③ enroll 关系表中，主键为{sid，cno，dname1}，外键有{sid}，{cno，dname1}

3、针对第 2 题的数据模式，完成以下查询(每个查询只能用单条 SQL 语句完成)
(1) 查询选修 Computer Sci 系所开课程，且课程成绩在 3 分以上的学生学号和姓名(6 分);
(2) 查询只有一人选修的课程(6 分);
(3) 查询 Computer Sci 系所开设各门课程的平均成绩(6 分);
(4) 查询选了 Computer Sci 系所开设全部课程的学生姓名及学号(6 分);
(5) 查询 Computer Sci 系所开设各门课程的最高分及取得最高分学生的学号(6 分)

(1)
```
SELECT S.sid, S.sname FROM student S, enroll E
WHERE E.dname1=’Computer Sci’ AND E.grade>3 AND S.sid=E.sid
```
(2)
```
SELECT * FROM course C WHERE C.cno IN
(SELECT E.cno FROM enroll E GROUP BY E.cno HAVING COUNT(*)=1)
```
(3)
```
SELECT AVG(E.grade) FROM enroll E
GROUP BY E.cno, E.dname1 HAVING E.dname1=’Computer Sci’
```
(4)
```
SELECT S.sname, S.sid FROM student S
WHERE NOT EXISTS
(SELECT C.cno FROM course C WHERE C.dname=’Computer Sci’
AND NOT EXISTS
(SELECT E.cno FROM enroll E WHERE E.sid=S.sid AND E.cno=C.cno))
```
(5)
```
SELECT E.grade, E.sid FROM enroll E,
(SELECT MAX(E1.grade) AS maxgrade, E1.cno FROM enroll E1
WHERE E1.dname1=’Computer Sci’ GROUP BY E1.cno) AS TEMP
WHERE E.dname1=’Computer Sci’
AND E.cno=TEMP.cno AND E.grade=TEMP.maxgrade

SELECT E.grade, E.sid FROM enroll E
WHERE dname1=’Computer Sci’ AND E.grade=
(SELECT MAX(E1.grade) FROM enroll E1
WHERE dname1=’Computer Sci’
GROUP BY E1.cno HAVING E1.cno=E.cno)
```

4、稠密索引是否一定能够提高针对索引属性查询的效率?为什么? (8 分)
稠密索引不一定能提高针对索引属性的查询效率。
① 如果是查询小文件中的全部或相当多的记录时，使用索引并不能提高查询效率，反而会因为索引增加开销；
② 如果稠密索引为次索引，但不是簇集索引，也就是说一个键值对应的多条记录分散在不同的物理块中；当一个键值对应的记录较多时，取这些记录时访问物理课的 I/O 开反而会降低查询的效率。

5、对第二题数据库，
(1)其中的关系模式的设计考虑哪些问题？（6 分）
(2)如果实际应用中还需要表达每门课的先修课情况，应该如何调整数据库的设计？
请给出你的设计方案并说明理由。(注意每门课的先修课可能不止一门，而且不同课程先修课的数目是不同的，可以修改现有表的设计也可以增加新表) (6 分)
(1)
考虑了完整性约束和规范化的问题。
① 数据库中各个关系表都有主键，主键的值是唯一的且不为空，满足了实体完整性约束。同时 enroll 表中定义了对 course 和 student 表的外键，满足了引用完整性约束。
② 关系中的属性都是原子的，因此满足第一范式的要求；同时各关系表中不存在部分函数依赖和传递函数依赖，因此该数据库的设计也满足第二和第三范式的要求。
(2)
先修课程包括课程号、课程名、开课院系等属性。如果将先修课程情况直接作为属性附加到 course 关系表中，会导致关系中出现部分函数依赖，不符合二范式的要求。而且先修课程可能不止一门，上述的做法还会产生大量冗余。因此需要新增一个关系表表达先修课程情况。关系表设计如下：

cno	dname	cno1	dname1

其中 cno 和 dname 表示当前课程的课程号和开课院系，cno1 和 dname1 表示先修课程的课程号和开课院系。cno 和 dname、cno1 和 dname1 都是对 course 表的外键。

6、相较层次和网状数据库系统，查询优化对关系数据库系统更为重要。你认为这句话对么?给出理由。(8 分)
我认为这句话是对的。
层次和网状数据库的数据模型使用指针表示属性之间的关系，这样的结构也固定了这两种数据库的查询路径，进行优化的空间有限。
关系数据理论和关系数据查询语言提供了查询优化的空间。关系型数据库的抽象程度深，其查询语言一般是非过程语言，仅表达查询要求，不说明查询的过程。对于同一个查询语句，对应的关系代数等价的不同表达式的查询效率有着很大的差异；集合操作不同的执行规则和策略也对查询效率有着很大的影响；同时关系数据在物理存储形式和存取方式和路径上都没有限制。因此对于关系数据库系统来说，查询优化就更为重要，它对系统的性能有着很大的影响。

7、现代数据库系统通常采用后备副本结合运行记录的方法实现系统的故障恢复，发生介质失效时，CTL 表中最近一次 CP 点以前提交的事务是否要做 redo 操作?(需回答原因)什么时候可以清空 CTL 表? (14 分)
发生介质失效时，需要在加载最近后备副本后，根据运行记录中的后像，重做最近后备副本以后提交的所有更新事务。因此最近一次 CP 点以前提交的事务也要做 redo 操作。因为发生介质失效将丢失所有数据，包括所有 CP 点写入失效介质

中的数据。
在取后备副本后，以前的运行记录就失去价值，可以清空 CTL 表。

8、试编写一个触发器，监视上面数据库中 enroll 表上的 INSERT 操作，对每条 INSERT 语句，判断其插入元组的 grade 值是否小于 3.0，将这样的元组自动插入到 failedcourse 表中(设 filedcourse 表与 enroll 表的模式完全相同)。(10 分)

```
CREATE TRIGGER insert_grade_check
AFTER INSERT ON enroll
REFERENCING NEW TABLE AS NE
FOR EACH STATEMENT
WHEN (EXISTS(SELECT * FROM NE
                    WHERE grade<3.0))
INSERT
    INTO failedcourse
    SELECT * FROM NE
    WHERE NE.grade<3.0
```