Answers of Problem 1:

(1)

$\prod_{pName}(project \bowtie teacher \bowtie (\sigma_{dName=\text{“Computer Science”}}(department))$

(2)

```
CREATE TABLE project
    (pId    char(10),
    pName    varchar(20),
    tId    char(10),
    startTime    date,
    endTime    date,
    primary key (pId),
    foreign key (tId) references teacher);
CREATE TABLE participate
    (pId    char(10),
    sId    char(10),
    role    varchar(20),
    primary key (pId, sId),
    foreign key (pId) references project,
    foreign key (sId) references student,
    check (role=“leader” or role=“member”));
```

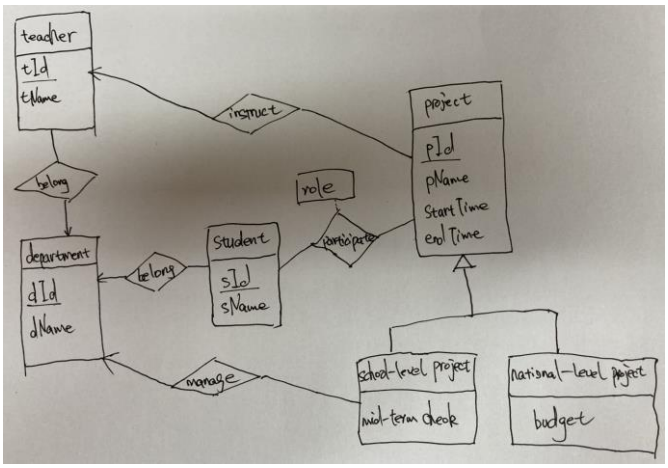
(3)

```
select distinct tName
from project, teacher
where project.tId=teacher.tId and startTime between '2020-01-01' and '2020-12-31'
```

(4)

```
select sName
from student
where sId in
    (select sId
    from participate
    group by sId
    having count(pId) > 2)
```

Answers of Problem 2:



Answers of Problem 3:

(1)

AD AEF

(2)

A->B, B->C, D->EF, EF->D

(3)

There are different decomposition results and the following is just an example.

R1=(A, B), R2=(A, C, D, E, F) (A->B)

R21=(A, C), R22=(A, D, E, F) (A->C)

R221=(D, E), R222(A, D, F) (D->E)

R2221(D, F), R2222(A, D) (D->F)

This decomposition is not dependency preserving (e.g., B->C is not preserved).

Following is another solution:

R1=(B, C), R2=(A, B, D, E, F) (B->C)

R21=(A, B), R22=(A, D, E, F) (A->B)

R221=(D, E, F), R222(A, D) (D->EF)

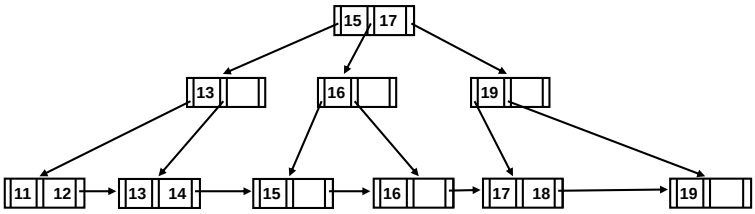
This decomposition is dependency preserving, because A->B can be checked on R21, B->C can be checked on R1, D->EF and EF->D can be checked on R221.

Answers of Problem 4:

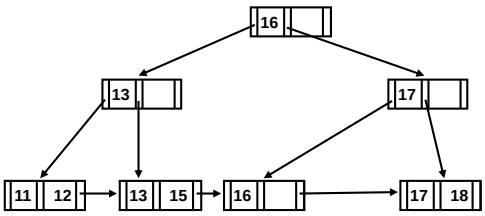
(1)

The built index is not a primary index, as the search key of the index is not the search key of the sequentially ordered data file.

(2)



(3)



Answers of Problem 5:

(1)

5 partitions, as the number of partitions is M-1.

(2)

Relation *s*, as relation *s* is smaller than relation *r*.

(3)

Recursive partition is not needed, as the size of the partitions of relation *s* (i.e., 4) is less than or equal to M-2 (i.e., 4).

(4)

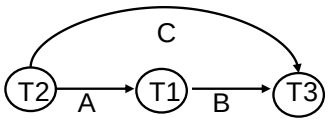
Number of block transfers: 3×(100+20)+4×5

Note: 4×5 is not necessary, which considers partially filled blocks.

Number of seeks: 2×(100+20)+2×5

Answers of Problem 6:

(1)

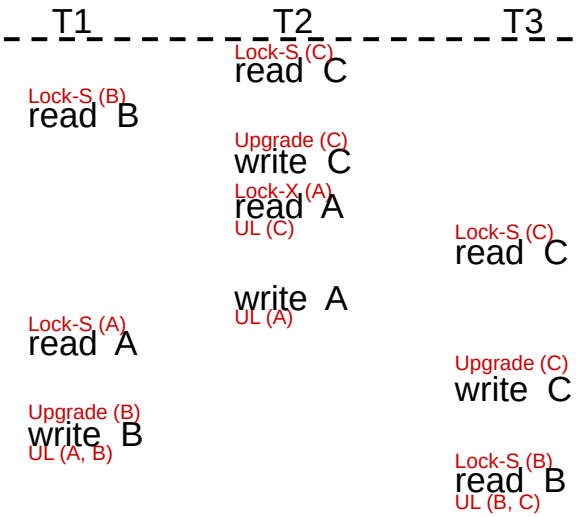


(2)

The schedule is conflict serializable, as the precedence graph is acyclic.

(3)

It is possible that the schedule is generated by the 2PL protocol with lock conversions.



(4)

T1 must commit before T3 does.

T2 must commit before T1 does.

T2 must commit before T3 does.

Answers of Problem 7:

(1)

B=2100

C= 300 or 400 or 700

(2)

redo: T_0 and T_2 undo: T_1

(3)

redo: 7-14 undo: 14-3

(4)

<T_1, C, 600>

<T_1, O_1, operation-abort>

<T_1, B, 2050>

<T_1, abort>