`

1a)

i)

SELECT Lecturer.lname

FROM (Enrolled INNER JOIN Course ON Enrolled.cname = Course.name) INNER JOIN Lecturer ON Course.lid = Lecturer.lid

GROUP BY Lecturer.lid

HAVING Count(Enrolled.snum) < 5;

SELECT lname

FROM Lecturer

WHERE lid IN (

SELECT lid8i8

FROM Course

WHERE name

IN (

SELECT cname AS name

FROM Enrolled

GROUP BY name

HAVING count(snum) < 5

)

)

SELECT numStudents, lecturer.lname

FROM lecturer,

(SELECT count(cname) as numStudents, course.lid as lectID

FROM enrolled JOIN course on course.name = enrolled.cname

GROUP BY lectID)

WHERE numStudents < 5 AND lectID = lecturer.lid

^^^

I tested the above solution on similar things in the coursework database and it seemed to work it’s pretty much the same as the first solution

SELECT lname

FROM lecturer NATURAL LEFT JOIN (student JOIN enrolled ON name = cname)

GROUP BY lid, lname # Group by lid also to handle the case where two (distinct) lecturers have the same name

HAVING COUNT(snum) < 5 # If lecturer teaches a course with no enrolment, or if a lecturer doesn’t teach any courses

ii)

SELECT degreeyear

, Avg(age) FROM Student

GROUP BY degreeyear;

iii)

SELECT degreeyear, Avg(age) FROM Student

GROUP BY degreeyear

HAVING degreeyear <> 1;

iv)

SELECT Student.sname FROM Enrolled

INNER JOIN Student USING (snum)

GROUP BY Student.snum, Student.sname

HAVING COUNT(Enrolled.cname) = (SELECT COUNT(\*) FROM Course);

In case we are looking for maximum as in not all the courses:

select sname

from (select snum, max(cname)

from (select snum, count(cname)

from enrolled

group by snum)) join student

*(considering the comment above from Max,*

*but there may be a way to not write the inner query twice)*

SELECT sname

FROM student NATURAL JOIN (SELECT snum, count(cname) AS n\_course

FROM enrolled

GROUP BY snum)

WHERE n\_course = (SELECT MAX(n\_course)

FROM (SELECT snum, count(cname) AS n\_course

FROM enrolled

GROUP BY snum))

v)

~~SELECT MAX(degreeyear), age FROM Student~~

~~GROUP BY age;~~

Michal Pándy’s solution:

SELECT age, degreeyear FROM

(SELECT age, degreeyear, count(sname) as numStd

FROM Student

GROUP BY age, degreeyear)  
GROUP BY age  
HAVING numStd = max(numStd);

Alternative solution:

select x.degreeyear, x.age, count(\*) from Student as x

group by x.degreeyear, x.age

having count(\*) =

(select count(\*) from Student as y

where x.degreeyear = y.degreeyear

group by y.degreeyear, y.age

order by count(\*) desc

limit 1);

Another way (confirmed to work):

SELECT s.age, s.degreeyear

FROM student AS s

GROUP BY age, degreeyear

HAVING COUNT(\*) >= ALL (SELECT COUNT(\*)

FROM student AS d

WHERE s.age = d.age

GROUP BY age, degreeyear);

1b)

interior(name)

room\_int(interior\_name)

interior\_name **references** interior.name

hotel\_has(hotel\_name, interior\_name, price)

hotel\_name **references** hotel.name

interior\_name **references** hotel\_int.name

hotel\_int(interior\_name)

interior\_name **references** interior.name

hotel(name, type, description, categorie, telephone, zip, street, number)

zip, city, street, number ===fk===> address(zip, city, street, number)

hotel\_photo(hotel\_name, photo)

hotel\_name **references** hotel.name **on delete cascade**

room\_type\_has(type, name)

type **references** room\_type.type **on delete cascade**

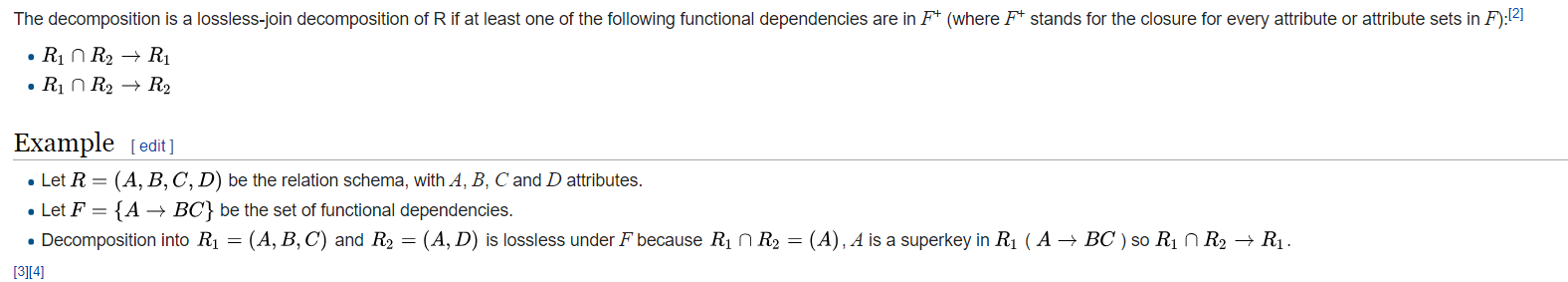
name **references** room\_int.interior\_name **on delete cascade**

room\_type(type, hotel\_name,photo, description, count, price)

Hotel\_name **references** hotel.name **on delete cascade**

address(zip, city, street, number, hotel\_name)

hotel\_name references hotel.name

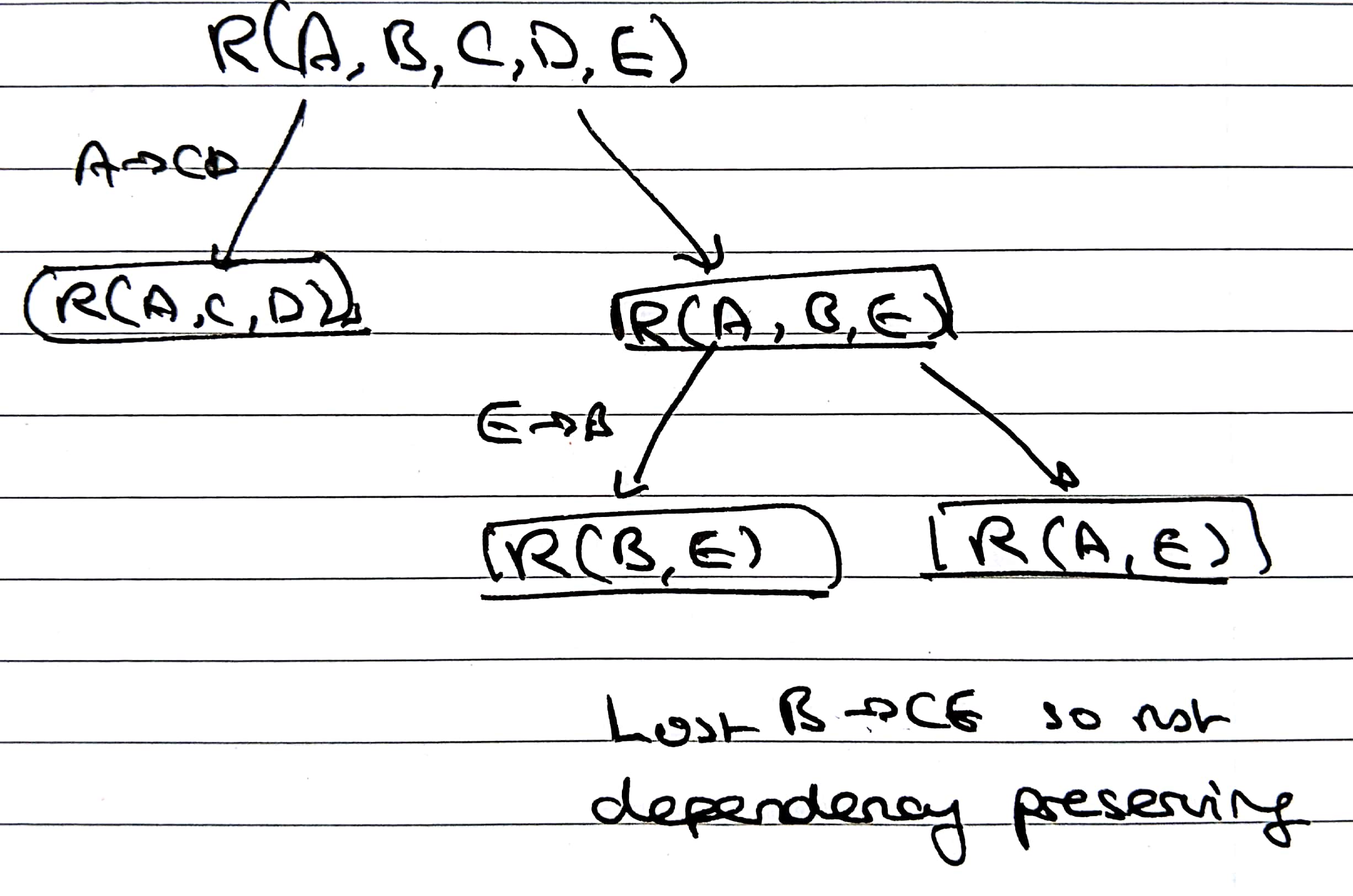
2a)

From Wikipedia

(A, B, C) intersect (A, D, E) = (A)

By A ->BC, A -> ABC holds. Therefore, this decomposition is a lossless-join relation.

2b)

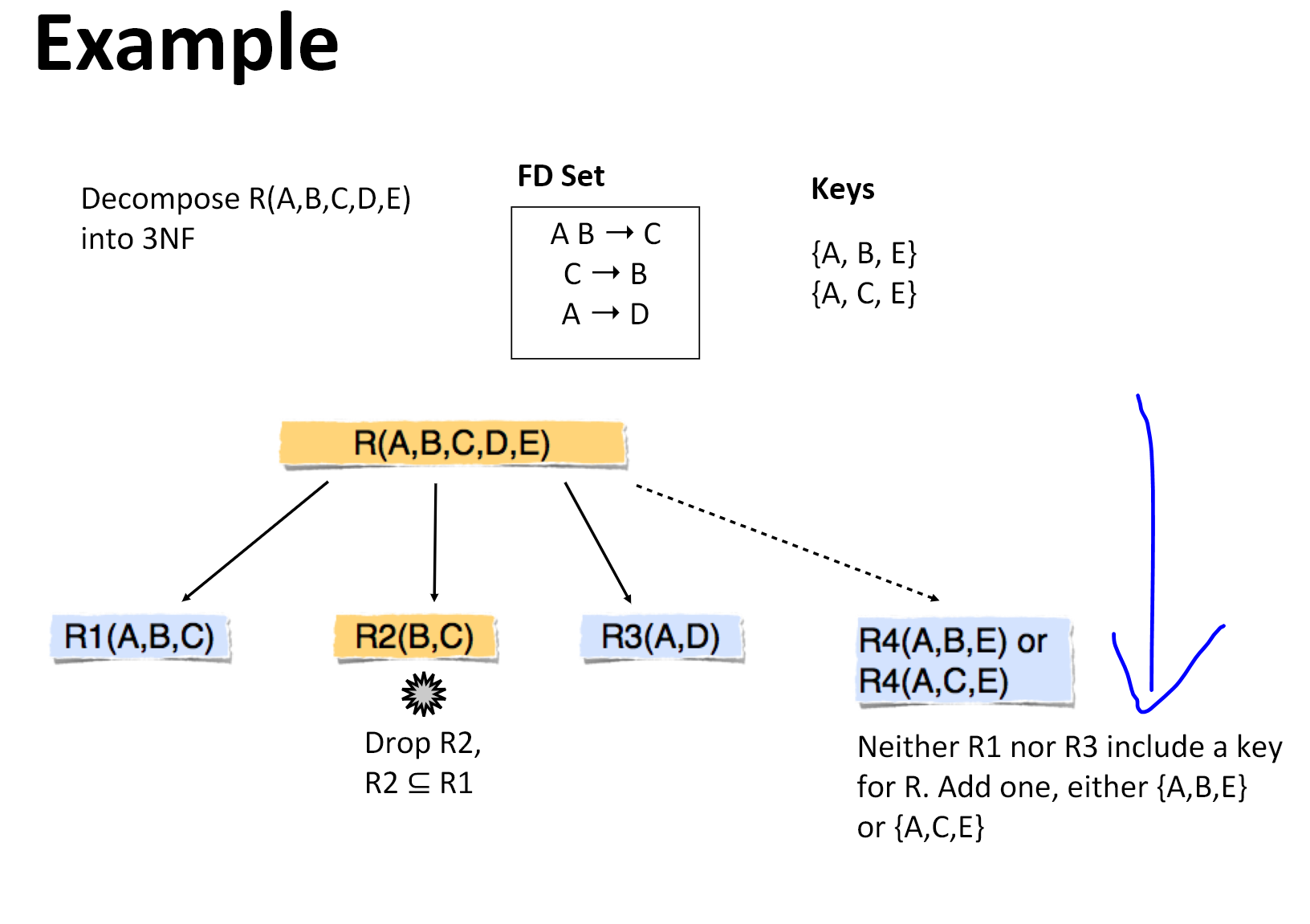


Since B->C is not preserved, add another relation (B, C). Then (A, C, D), (B, E), (A, E), and (B, C) is a 3NF decomposition which is lossless-join and preserves functional dependencies.

2c)

R(A, C, D) R(B, C, E) R(A, B)

Instead of R(A,B) any key would be okay



2d)

(The canonical cover is not unique, so you may have a different answer)

Remove A -> E as it is extraneous (included in first FD)

In, A -> BCE, C and E are extraneous, so remove C and E

In B -> CEA, no more extraneous attributes.

Canonical cover:

A -> B

B -> CEA

Alternative Canonical Cover:  
B -> A

A -> CEB

Another alternative canonical cover:

a->be

b->ac

Alternative (this is exactly the same as the previous one, just union rule it. Also this is technically a minimal cover right)

a->b

b->ca

a->e