V\

C \c

ca

(Really unsure about answers, feel free to edit/comment)

1. SELECT customer\_name, amount

FROM borrower NATURAL JOIN loan

1. (SELECT customer\_name FROM depositor)

EXCEPT (SELECT customer\_name FROM borrower)

3) SELECT DISTINCT branch\_name

FROM (customer NATURAL JOIN borrower) NATURAL JOIN loan

WHERE customer\_city = ‘London’

1. SELECT branch.branch\_name FROM Loan

WHERE loan\_number IN (

SELECT borrower.loan\_number FROM (customer JOIN borrower)

WHERE customer.customer\_city = ‘London’)

3) SELECT loan.branch\_name FROM borrower JOIN custom on borrower.customer\_name = customer.customer\_name JOIN loan ON borrower.loan\_number = loan.loan\_number WHERE customer.customer\_city = ‘London’

1. SELECT customer.customer\_city, SUM(loan.amount)

FROM (customer NATURAL JOIN borrower) NATURAL JOIN loan

GROUP BY customer\_city

4) SELECT SUM(loan.amount)

FROM customer JOIN borrower ON customer.customer\_name = borrower.customer\_name

JOIN loan ON borrower.loan\_number = loan.loan\_number

GROUP BY customer.customer\_city

1. SELECT borrower.customer\_name FROM (borrower JOIN depositor)

Alt 5) SELECT borrower.customer\_name FROM borrower

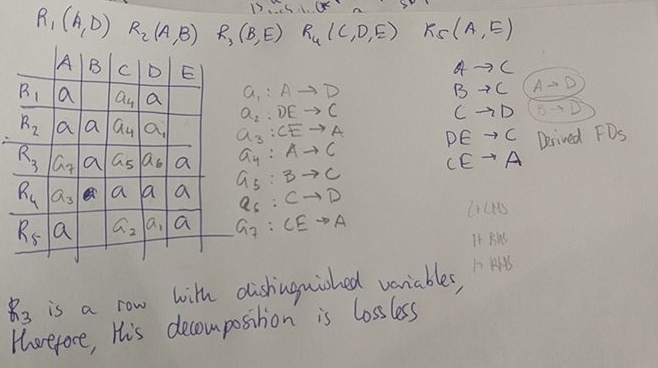
INTERSECT SELECT depositor.customer\_name FROM depositor

2a)

Ny w

Lossless decomposition proof

* See <https://www.youtube.com/watch?v=TykMe1A2u6U> for method



Thomas said: if one can recursively join the relations pairwise (with the given rule) in a particular sequence to get to R

2b)

R(A B C D E F G H I J)

Key = {A B}

The fd set is canonical

Decomposes into:

R1(A B C) (contains key)

R2(A D E)

R3(B F)

R4(F G H)

R5(D I J)

2c)

canonical cover:

A -> B

B -> C

2d)

Decomposes into:

R1(B C D)

R2(E B)

R3(A C)

R4(A E)

With all decompositions in bcnf

R(A B C D E)T

With steps:

We have R(A B C D E).

BC is not a superkey in R so we split on the FD BC -> D, giving:

R1(B C D) R2(A B C E)

A -> C applies in R2, but A is not a superkey in R2 so we split on the FD A -> C, giving:

R1(B C D) R2(A C) R3(A B E)

E -> B applies in R3 but E is not a superkey in R3 so we split on the FD E -> B, giving:

R1(B C D) R2(A C) R3(E B) R4(A E)

Cannot be decomposed further. (The decomposition is not dependency preserving)

//another solution (not sure if it is right)

Key:AD, ABC, ...

R1(B D) (D->B and left with A CＤE on the right)

R2(D E) (D ->E and left with A C D on the right)

R3(A C) (A ->C and left with A D (which is superkey) on the right)

R4(A D)