Nicolas Carchio
Homework 4
12/2/19

19.2

1. List all Keys for R

CD	ABE	

ACD, CDE, BCD

2. Is R in 3NF?
   Yes, all attributes in R are prime, and thus it is in 3NF
3. Is R in BCNF?
   No, all three functional dependencies' superkey does not determine the other attributes

19.3

1. X → Y, Z → Y
   For each unique value of X and Z, we get the same value of Y
2. If we change a Z(3) to a Z(2) we still get the two functional dependencies from before: X → Y, Z → Y so nothing changes.

19.5

R(ABCDEFGHI)

1. R1(ABCDE) A → B, C → D
   a. Cannot find the primary key for these functional dependencies
      The strongest normal form for R1 is 1NF, ACE+ = ACEBD
   b. A → B, C → D

AC		BDE

ACE+ is the key, so AB and CD are the BCNF decomposition

2. R2(ABF) AC → E, B → F
   a. Can't identify the primary key with the functional dependencies above, so R2 is in 1NF
   b. AC → E, B → F
      AB, BF is the decomposition

3. R3(AD) D → G, G → H
   a.

AD	G	H

   D+ is the super key since G and H are not present in R3 (or its keys), so R3 is in BCNF

   b. It is already in BCNF, so don't have to decompose
4. R4(DCHG) A → I, I → A

Nicolas Carchio
Homework 4
12/2/19

a.

	AI	

A and I aren't present in R4 and therefore the relation R4 cannot be decomposed with respect to the A → I and I → A functional dependencies. Therefore, it is in BCNF.

b. Already in BCNF, so don't have to decompose

5. R(AICE) no dependencies
   a. No dependencies, so it is in BCNF
   b. Already in BCNF, so don't have to decompose

19.6

1. (1,2,3), (4,2,3), (5,3,3)
   a. A → B, holds
   b. BC → A, does not hold, an example is the two tuples (1,2,3) & (4,2,3)
   c. B → C, holds
2. No, we cannot say that there are functional dependencies that hold over S because in order for a functional dependency to hold for a relation, it must make a statement about all allowable instances of the relation that we are referring to. (in this case, it is S) Basically a dependency holds over all possible instances, and we only have 3 so we cannot determine this.

19.7

R (ABCD)

a. Candidate keys
b. Strongest NF
c. If not in BCNF, decompose it

1. C → D, C → A, B → C

B	C	AD

   a. B
   b. 2NF but not 3NF (all keys are singletons but aren't all prime)
   c. CD, CA and BC (it is lossless)

2. B → C, D → A

BD		AC

   a. BD
   b. 1NF but not 2NF
   c. BD, BC, AD (lossless)

3. ABC → D, D → A

BC	AD	

Nicolas Carchio
Homework 4
12/2/19

a. ABC, BCD
b. 3NF but not BCNF
c. If we split R into AD and BCD we can't preserve ABC → D so there is no BCNF decomposition

4. A → B, BC → D, A → C

A	BC	D

a. A
b. 2NF but not 3NF (BC → D is not prime)
c. ABC, BCD

5. AB → C, AB → D, C → A, D → B

	ABCD	

a. AB, CD, AD, CB
b. 3NF but not BCNF
c. C to A and D to B are violations. We cannot decompose to BCNF due to AC, BCD but we lose the AB ones. So we can try AC, BD, CD but we are going to lose Functional Dependencies so this won't be in BCNF form. Therefore, there is no BCNF decomposition.

19.10

a. candidate keys
b. is decomposition good or no?

1. B → C, D → A: into BC and AD

BC		CA

a. BD
b. BC and AD is not ideal because it is lossy due to the join of BC and AD (BC * AD can be bigger than ABCD). This could ultimately create a result set that contains items not in ABCD.

2. AB → C, C → A, C → D: ACD and BC

B	AC	D

a. AB, BC
b. AB → C is not preserved, ABC will then be added to the decomposition to attempt to avoid the loss of that dependency. However, by adding ABC, we introduce redundancies to some of the relations. Therefore, the current decomposition is not enough.

3. A → BC, C → AD : ABC and AD

	AC	BD

Nicolas Carchio
Homework 4
12/2/19

a. A, C
b. Already in BCNF, so don't decompose. No need for ABC and AD because it is already in BCNF. With the decomposition they have given, C → AD is not preserved.

4. A→ B, B → C, C → D: AB, ACD

A	BC	D

a. A
b. This is a lossless decomposition due to the candidate key of A. However, this is not dependency preserving because B → C is not preserved.

5. A → B, B→ C, C → D: AB, AD, CD

A	BC	D

a. A
b. This is a lossless decomposition but not dependency preserving because B → C is not preserved, causing redundancy through some of the relations. Therefore it is not a great BCNF decomposition.