INFS 614 – Quiz #4
11/27/07

Name: ________________________________ G#: ______________________

1. Let F = {A→B, B→C}. Is C→A in F^+? Why or why not?

   Alternative 1: We can compute F+ from F. It's clear that C→A cannot be inferred from F, so it's not in F^+.

   Alternative 2: We can compute C^+ and see if A is in C^+:

   ***Result* = C**

   A is NOT in Result so we don't consider A→B

   B is NOT in Result so we don't consider B→C

   We've looked at all the FDs in F, so C^+ = C. Since A is NOT in C^+, C→A is NOT in F^+.

2. In the previous question, list the candidate key(s). Show your work.

   We need to find the minimal set of attributes whose attribute closure contains all the attributes. We have to compute at least A+, B+ and C+ (they are the minimal "set" of attributes). We already know C+, which does NOT contain all the attributes (i.e., C is not a candidate key). So we need to compute A+ and B+, and go from there. If *both* A and B are candidate keys, then we're done. Otherwise, we need to compute more attribute closures.

   1) Let's try A^+.

   ***Result* = A**

   Consider A→B:
   A is in *Result* so add B to *Result*: *Result* = AB.

   Consider B→C:

B is in *Result* so add C to *Result*: *Result* = ABC.

Done, and A^+ = {ABC}. So A is a candidate key.

2) Now try B^+.

***Result* = B**

Consider B→C first:
B is in *Result* so add C to *Result*: *Result* = BC.

Consider A→B:
A is NOT in *Result* so do nothing.

B^+ = {BC}. So B is NOT a candidate key.

So far we only have A as the candidate key. It's clear that BC (which is also a minimal subset, since neither B nor C is a candidate key) is not a candidate key either, from the FD B→C (i.e. B is not a key, and since C is determined by B, BC could not be a key either). You can also compute BC+ to verify. Note that we don't need to check AB or AC, since they both contain A (therefore not minimal subsets).

3. For the relation R(A, B, C, D, E), let F = {A→B, C→D}. Is R in BCNF? If not, decompose it into a collection of BCNF relations.

   The candidate key for R is ACE (see lecture slides for the logic; you can also figure this out by computing the attribute closures).

   For A→B: It's a non-trivial FD, and A is not a superkey, so A→B violates BCNF.
   For C→D: It's a non-trivial FD, and C is not a superkey, so C→D violates BCNF.

   Note, as long as there is one FD that violates BCNF, then R is not in BCNF.

   To decompose:

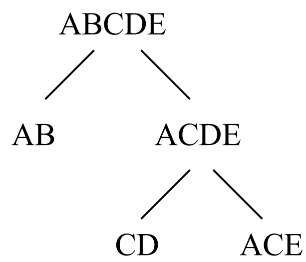
   We can choose any violating FD as a starting point. Let's say we start with A→B.

   We'll decompose R into AB and ACDE (i.e. ABCDE – B).

Check if ACDE is in BCNF: No, because C→D violates BCNF condition(s). We need to decompose ACDE.

We'll decompose it into CD and ACE (i.e., ACDE – D).

None of the FDs violate BCNF for ACE. So we're done. The following is the summary:



4. What are the two conditions for a decomposition to be considered a "good" decomposition?

   Lossless join
   Dependency preserving