Brent Cahill

CS 121

Problem Set 6

1. 1. There would be no non-trivial dependencies in this case.
   2. Here, B→A would be non-trivial since it is a one-to-many mapping.
   3. Here, A→B would be non-trivial since it is a many-to-one mapping.
   4. Here, both A→B and B→A would be non-trivial.
2. 1. Union
      1. For each functional dependency :
         1. Apply the augmentation rule to the given , using a, thus we know holds.
         2. Now, if we apply the augmentation rule again to the given using , then we get
         3. Thus, we use the transitivity rule to replace the from part i. with just , giving us , as desired.
   2. Decomposition
      1. For each functional dependency :
         1. Apply the reflexivity rule to the given , giving us
         2. Now, apply the transitivity rule to the above, and we arrive at , or
      2. Similarly,
         1. Apply the reflexivity rule to the given , giving us
         2. Now, apply the transitivity rule to the above, and we arrive at , or
      3. Now we have arrived at and , as desired.
   3. Pseudotransivity
      1. For each functional dependency :
         1. Apply the augmentation rule to the given , giving us .
         2. Apply the transitivity rule to the above, and we see , or , as desired.
3. In this case, we will list all candidate keys and explain why they are superkeys for R.
   1. A is a candidate key.
      1. We know that we can get B and C from A using the decomposition rule:
      2. We can get D from A using the transitivity rule and .
      3. Now we use the pseudotransitivity rule to say that since and , then .
      4. Now, we use the transitivity rule to say that , so .
      5. Thus, we have proven that A is a candidate key.
   2. E is a candidate key.
      1. Since we have already proven A is a candidate key, we simply use the transitivity rule on to say that E is also a candidate key.
   3. Now, we list all candidate keys that also include A or E. We do not need to prove these, since we already proved that A and E are candidate keys.
      1. AB, AC, AD, AE, ABC, ABD, ABE, ACD, ACE, ADE, ABCD, ABCE, ABDE, ACDE, ABCDE, EB, EC, ED, EBC, EBD, ECD, and EBCD.
   4. BC is a candidate key.
      1. We know that we can get D from BC using the decomposition rule: .
      2. Now we can use the transitivity rule get E, since: , which we know is a candidate key.
      3. We use the transitivity rule now to say that BC is a candidate key.
   5. BCD is also a candidate key, since it includes BC, a candidate key.
   6. CD is a candidate key:
      1. We know that we can reach E, a candidate key, using the transitivity rule, thus we know that CD is a candidate key.
4. No. Because if at any point during the multivalued dependency there is a single valued dependency: , but a multivalued dependency to C, which is allowed, and still results in then the result would be: , not .
5. * 1. Apply the Union rule to and : Left with
     2. Here, we know that the D in is extraneous, because and . So now we are left with .
     3. We know that the B in is extraneous because and , and using the transitivity rule, we get , so we get .
     4. Now we apply the union rule to and , so we are left with .
     5. Now, we use the transitivity rule to say that since , A is extraneous in , so we are left with our final answer:
     6. .
   1. BC is a candidate key for R:
      1. We know that we can reach A from .
      2. We know that we already have B.
      3. We know that we already have C.
      4. We know that we can reach D from .
      5. We know that we can reach E from .
      6. We know that we can reach G using the transitivity rule: , so . Thus, we have proven BC is a candidate key for R.
   2. with .
      1. Using , we can say that
      2. Now, we know that
      3. Using , we create
      4. Now we say that
      5. Using , we create
      6. Now we create
      7. So, our final BCNF is , where A and C are superkeys and the schemas are determined via B and C.
      8. In this BCNF, the dependency is not preserved.
   3. with .
      1. Using , we create
      2. Now,
      3. Using , we create
      4. Now,
      5. Using , we create
      6. Now,
      7. So, our final BCNF is
      8. In this BCNF, the dependency is not preserved.
   4. We use the that we created before to create the relations as follows:
      * 1. Note: we get R5 from the fact that we know BC is a candidate key.
6. 1. 1. is a candidate key.
         1. Note that we can reach *course\_id, section\_id, term,* and *year* obviously.
         2. We can reach *meet\_time, room, num\_students,* and *instructor\_id* from the second relation. Thus, we have shown that is a candidate key.
      2. is a candidate key.
         1. Note that we can reach *room, meet\_time, term,* and *year*, obviously.
         2. We can reach  *instructor\_id, course\_id,* and *section\_id* from the third relation.
         3. We can reach *dept, units,* and *course\_level* from the first relation using the decomposition rule.
         4. We can reach *meet\_time, room, num\_students,* and *instructor\_id* from the second relation.
   2. 1. We know that *instructor\_id* is extraneous in because when we look at the second and third relations, we can use the transitivity rule and the decomposition rule to arrive at
      2. In this case, we say that rather than *instructor\_id* being extraneous in the second relation, we say that it is extraneous in the third. Using similar logic, we arrive at
      3. The reason why we can remove *instructor\_id* is because it can be “reached” using either the second *or* third relation, so we do not need it in both. It is the only element for which this is true.
      4. The second cover makes more sense, since it is more likely that a database would be easy to understand if the *instructor\_id* was in the same relation as the course for which he taught, rather than the room which he taught it in, which could change, and is not related. Since *instructor\_id* is the only thing that changes throughout the covers, this is the only thing that need to be taken into consideration.
   3. Third Normal Form would be most appropriate in this case. This is because there is a very limited amount of records in this database. The dependency would be as follows:
      1. (course\_id, dept, units, course\_level), (course\_id, section\_id, term, year, meet\_time, room, num\_students, instructor\_id), (room, meet\_time, term, year, instructor\_id, course\_id, section\_id).
      2. We would then create the following:
         1. (course\_id, dept, units, course\_level), (course\_id, section\_id, term, year, meet\_time, room, num\_students, instructor\_id)
      3. In this case, course\_id is a foreign key to the first relation, and {room, meet\_time, term, year} is a candidate key.
7. 1. Schema:
      1. email\_info(email\_id, send\_date, from\_addr, to\_addr, subject, email\_body)
      2. correspondence(from\_addr, to\_addr)
      3. attachments(email\_id, attachment\_name, attachment\_body)
   2. We know that i. is in 4NF because is a superkey.
   3. We know that ii. is in 4NF because is trivial
   4. As above, iii. is in 4NF because is a superkey.