Abel Marin

March 4, 2020

1. Problem 1
   1. We begin by creating a matrix with a row for each relation and a column for each attribute. We then set each entry to bij where i is the relation number and j is the attribute number. This is the initial state:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | b11 | b12 | b13 | b14 |
| R2 | b21 | b22 | b23 | b24 |
| R3 | b31 | b32 | b33 | b34 |

We then go through each entry and check to see if the relation contains the attribute in which case, we change the entry to be aj. Thus, this would be the state of the matrix after this step:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | b23 | a4 |
| R3 | b31 | a2 | b33 | a4 |

Next we look at each functional dependency to see if multiple rows agree on the same determinant. If they do, then we can set the row which agree to also agree on the dependent attributes. We first look at A1 -> A3 and we notice that R1 and R2 both agree on A1. Therefore, we can set these two rows to agree on A3. This would be the resulting matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | b31 | a2 | b33 | a4 |

We then look at A2 -> A3 and we notice that there are no two rows which agree on this, so we move on with the same matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | b31 | a2 | b33 | a4 |

We then look at A4 -> A1 and we notice that R2 and R3 both agree on A4. From here we set these two rows to agree on A1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | a1 | a2 | b33 | a4 |

We then repeat this process over again. If nothing changes then we exit the loop. We look at the functional dependency A1 -> A3 and notice that R1, R2, and R3 all agree on A1. Therefore, we set these rows to all agree on A3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | a1 | a2 | a3 | a4 |

We then look at the functional dependency A2 -> A3. Still no two relations agree on A2 so we move on with the same matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | a1 | a2 | a3 | a4 |

We next look at the functional dependency A4 -> A1. R2 and R3 agree on A4 but they already agree on A1, so we leave the matrix unchanged:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | a1 | a2 | a3 | a4 |

We then repeat this process over again. If nothing changes then we exit the loop. We look at the functional dependency A1 -> A3 and notice that R1, R2, and R3 all agree on A1. However, all three already agreed on A3 so we continue on with the matrix unchanged:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | a1 | a2 | a3 | a4 |

We then look at the functional dependency A2 -> A3. Still no two relations agree on A2, so we move on with the same matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | a1 | a2 | a3 | a4 |

We next look at the functional dependency A4 -> A1. R2 and R3 agree on A4 but they already agree on A1, so we leave the matrix unchanged:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | a1 | a2 | a3 | a4 |

Now that we have gone through the loop we exit and end up with this as our resulting matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 | A2 | A3 | A4 |
| R1 | a1 | b12 | a3 | b14 |
| R2 | a1 | b22 | a3 | a4 |
| R3 | a1 | a2 | a3 | a4 |

Finally, we check to see if there are any rows which are all aj’s and we notice that R3 has all aj’s. Thus, this decomposition has the lossless join property.

* 1. This decomposition has the lossless join property since the final matrix has a row which is made up entirely of aj’s. That row is R3 and for A1 it has a1, for A2 it has a2, for A3 it has a3, and for A4 it has a4.

1. In order to find the minimal basis, we must start by splitting up each relation such that there is only one attribute on the right. This would mean we split up b -> c,d into b -> c and b -> d. We get: {a -> b; a, d -> c; b -> c; b -> d; c -> d}.

We next look at the functional dependencies which have more than one attribute on the left and check to see if we can split them up. We look at a,d -> c which split up would be a -> c and d -> c. We keep a -> c and get rid of d -> c because we need this a -> b and b ->c. We get: {a -> b; a -> c; b ->c; b->d; c->d}.

Next, we must replace all dependencies which can derived from other from a combination of other dependencies as long as the resulting set is equivalent. We can remove a -> c through a combination of a->b and b->c. We can remove b -> d through a combination of b -> c and c -> d. We get a minimum basis of: {a -> b; b ->c; c -> d}.

1. Problem 3
   1. Candidate Keys: {SID}, {SSN}
   2. Student is not in 3NF because in order for it to be in 3NF, all attributes on the left side are superkeys or each attribute on the right is a part of some candidate key. The relation First -> Class and Major -> Dept do not satisfy either of these conditions. First and Major are not super keys and Class and Dept are not part of any of the candidate keys.
   3. In order to decompose Student into relations in 3NF we must start by finding the minimal basis of the functional dependencies. This can be achieved by splitting up SID -> SSN, Major and SSN -> First, Last, SID such that there is only one attribute on the right side. We would also need to remove SID -> Dept since SID determines Major and Major already determines Dept. The resulting set is the minimal basis of F. We get: {SID -> SSN; SID -> Major; SSN -> First; SSN -> Last; SSN -> SID; Major -> Dept; First -> Class}

Next we have to take each determinant and create a relation containing the determinant itself and the dependencies of said determinant. We would get:

STUDENT1(SID, SSN, Major)

STUDENT2(SSN, First, Last, SID)

STUDENT3(Major, Dept)

STUDENT4(First, Class)

We next check to see if any of the relations contain a candidate key. STUDENT1 and STUDENT2 each do so we move on. Lastly, we check to see if there are any redundant relations. That is, a relation is redundant if all of its attributes are included in another relation. There are none so we are done. The relations preserve the dependency preservation property and the lossless join property since the 3NF algorithm automatically preserve this property. The resulting decomposition is:

STUDENT1(SID, SSN, Major)

STUDENT2(SSN, First, Last, SID)

STUDENT3(Major, Dept)

STUDENT4(First, Class)