Alexander Miller
hw9.pdf

R = {A,B,C,D,E,F,G,H,I,J,K,L,M,N}

F = {
A -> BEJKLM
D -> BF
DG -> BCEGHIJ
ABDFGHL -> IM
}

1. 5 Non-trivial Elements of F+:

- ADGN -> ABCDEFGHIJKLMN
- AD -> ABDEFJKLM
- ADG -> ABCDEFGHIJKLM
- ABDFGHL -> ABCDEFGHIJKLM
- DG -> BCEFGHIJ

2. What is the candidate key of R?

- ADGN -> ABCDEFGHIJKLMN

Use functional dependencies and Armstrong axioms to prove this.

- A -> BEJKLM [given] [line1]
- D -> BF [given] [line2]
- DG -> BCEGHIJ [given] [line3]
- A -> A [Reflexivity] [line4]
- D -> D [Reflexivity] [line5]
- G -> G [Reflexivity] [line6]
- N -> N [Reflexivity] [line7]
- AD -> ABDEFJKLM [Union; lines 1,2,4,5] [line8]
- ADG -> ABCDEFGHIJKLM [Union; lines 3,6,8] [line9]
- ADGN -> ABCDEFGHIJKLMN [Union; lines 7,9] [line10]
- So ADGN -> R, so ADGN is a super key
- N is required: It is not determined by any other attribute
- A is required: It is not determined by any other attribute
- D is required: It is not determined by any other attribute
- G is required: It is not determined by any other attribute
- Since A, N, D, and G are all required, we cannot eliminate any from the key and get a valid super key
- therefore, ADGN is a candidate key.
- q.e.d.

Are there any other candidate keys for R? Why or why not?

- No.
- Since none of A,N,D, and G are determined by other attributes, all are required in a candidate key

- Since all of these are required by any candidate key, there exists no candidate key without these attributes
- There exists no smaller candidate key, since every other candidate key has to have at least these attributes
- There exists no larger candidate key, since the addition of attributes would make it a super key
- Since there is no smaller candidate key, there is no larger candidate key, and any other candidate key must have the same attributes, any other candidate key would be the same candidate key
- *Therefore, the candidate key ADGN is unique*
- q.e.d.

3. Update anomaly for R

- E is determined by A
- consider tuples T1 and T2, which have the same value for A
- if you update E in T1, you also have to update E in T2 to be the same, otherwise you have an update anomaly

Update anomaly example

- for clarity and simplicity let's only consider attributes A and E
- T1 = {0001, "John Doe"}
- T2 = {0001, "John Doe"}
- user updates T2; changes E to "Jane Smith"
- therefore, T2 = {0001, "Jane Smith"}
- Now we don't know if the E field determined by A value 0001 should be "John Doe" or "Jane Smith"

Insertion Anomaly for R

- an insertion anomaly might occur if you had information that A determines, but maybe not the information that D determines
- since D is part of the primary key, you would not be able to create a tuple to store the data

Insertion Anomaly example

- Let A = PartId, let D = SupplierId
- We want to put a new part in our database that we will be selling, but we don't yet know which supplier we will buying the input materials from
- In this schema, we would be unable to add the part to the database

Deletion Anomaly for R

- a deletion anomaly would occur if the removal of some information had collateral loss
- this also applies to the A and D relationship mentioned above

Deletion Example

- Let's say our supplier [field D] from the previous example lost their certification with the regulatory agency
- We will no longer be buying supplies from that vender, but we will still want to produce product [field A]

- Unfortunately, once we delete the supplier [D field], we will have an illegal tuple, so we will either have to delete the entire row (and lose information about product A), or we can't get rid of that supplier from our database

4. Compute a minimal cover M for F

- A -> EJKLMB, so A->E, A->J, A->K, A->L, A->M, A->B
- D -> BF, so D->B, D -> F
- DG -> HICBJGE, so DG ->H, DG->I, DG->C, DG->B, DG->J,DG->G,DG->E
- LADHBFG->MI, so LADHBFG->M, LADHBFG->I
- So M = {A->E, A->J, A->K, A->L, A->M, A->B, D->B, D->F, DG ->H, DG->I, DG->C, DG->B,DG->J,DG->G,DG->E, LADHBFG->M, LADHBFG->I}
- which reduces to:

M ={ A->E, A->J, A->K, A->L, A->M, A->B, D->F, DG ->H, DG->I}

5. Decompose R into BCNF database schema

- R1 = {<u>A</u>,B,E,J,K,L,M} [A->EJKLMB]
- R2 = {B,C,<u>D</u>,E,F,<u>G</u>,H,I,J} [D->BF, DG->HICBJGE]
- R3 = {<u>N</u>}

Is your decomposition lossless?

- R1 ∪ R2 ∪ R3 = {A,B,C,D,E,F,G,H,I,J,K,L,M,N} = R [requirement of all decompositions]
- R1 -> R1, therefore R1 ∩ R2 ∩ R3 -> R1
- R2 -> R2, therefore R1 ∩ R2 ∩ R3 -> R2
- R3 -> R3, therefore R1 ∩ R2 ∩ R3 -> R3
- So the decomposition is lossless

Does your decomposition preserve the original dependencies?

- Since A is the primary key of R1, A determines EJKLMB
  * *therefore A->EJKLMB is preserved*
- Since DG is the primary key of R2, D determines BF and DG determines HICBJGE
  * *therefore D -> BF is preserved*
  * *therefore DG -> HICBJGE is preserved*
- A-> EJKLMB is preserved, therefore A->BML
  * DG -> HICBJGE is preserved, therefore DG -> HIG
  * D -> BF is preserved, therefore D-> F
  * Since A->M and DG-> I, ADG->MI
  * if ADG-> MI, then by augmentation rule, LADHBFG ->MI
  * *So LADHBFG -> MI is preserved*
- So all dependencies of F have been preserved