SCC.221
Data Engineering

2024 - Week 3 – Functional dependencies and Normal forms.

Uraz C Turker

Learning Outcomes

Lancaster University

- You will learn
  - Functional Dependencies and how to reveal them.
  - Normal forms (1st,2nd, 3rd, and Boyce-Codd), their role, and methods of implementing them.
- After this week, you will be able to normalise given relational model databases and tell the normalisation level of each relation.

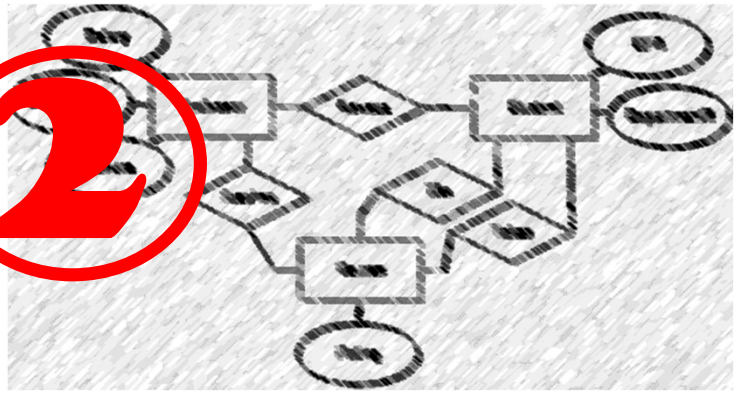
Contrary to some popular conceptions of the Vikings, they were not a "race" linked by ties of common ancestry or patriotism, and could not be defined by any particular sense of "Viking-ness." Most of the Vikings whose activities are best known come from the areas now known as Denmark, Norway and Sweden, though there are mentions in historical records of Finnish, Estonian and Saami Vikings as well. Their common ground–and what made them different from the European peoples they confronted–was that they came from a foreign land, they were not "civilized" in the local understanding of the word and–most importantly–they were not Christian.

- We received a work plan in plain English.
-
-
-

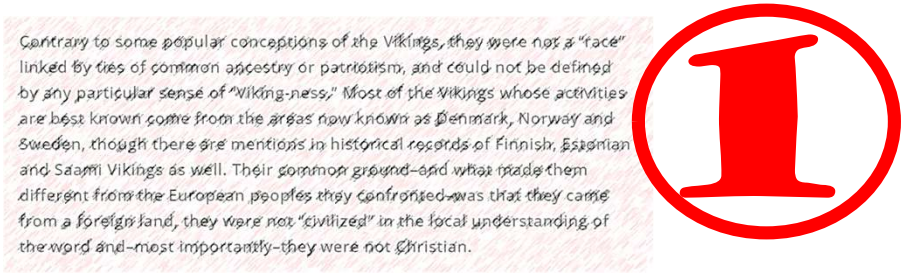
Contrary to some popular conceptions of the Vikings, they were not a "race" linked by ties of common ancestry or patriotism, and could not be defined by any particular sense of "Viking-ness." Most of the Vikings whose activities are best known come from the areas now known as Denmark, Norway and Sweden, though there are mentions in historical records of Finnish, Estonian and Saami Vikings as well. Their common ground–and what made them different from the European peoples they confronted–was that they came from a foreign land, they were not "civilized" in the local understanding of the word and–most importantly–they were not Christian.

1

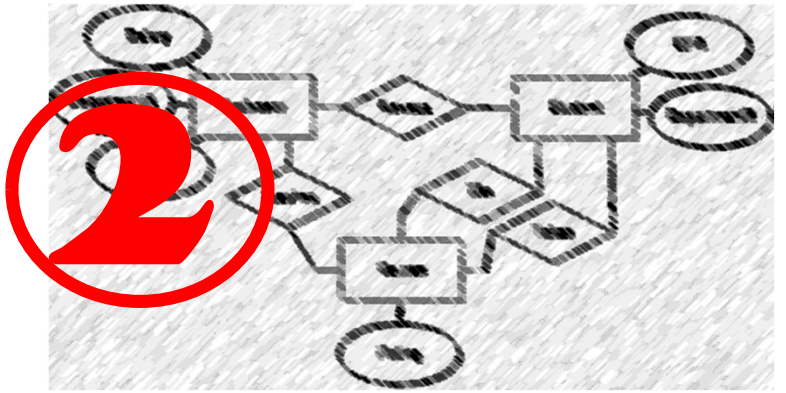
2



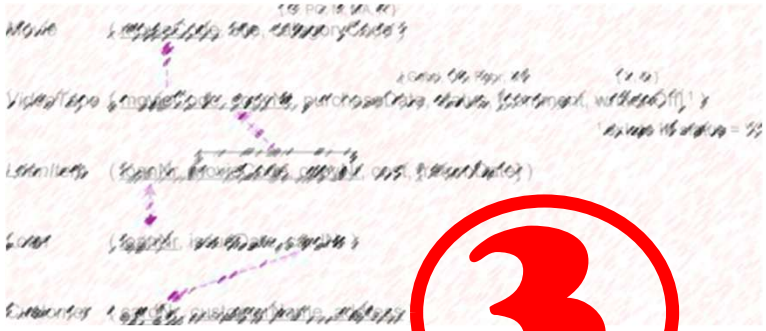
- We received a work plan in plain English.
- We derived its ER diagram.
-
-



1

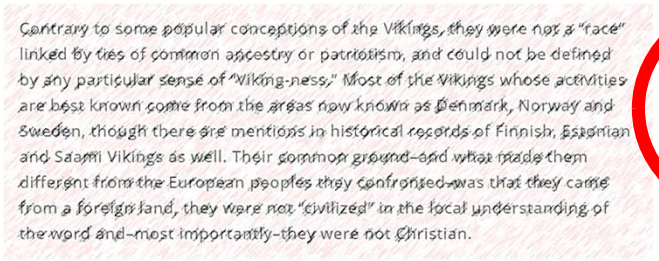


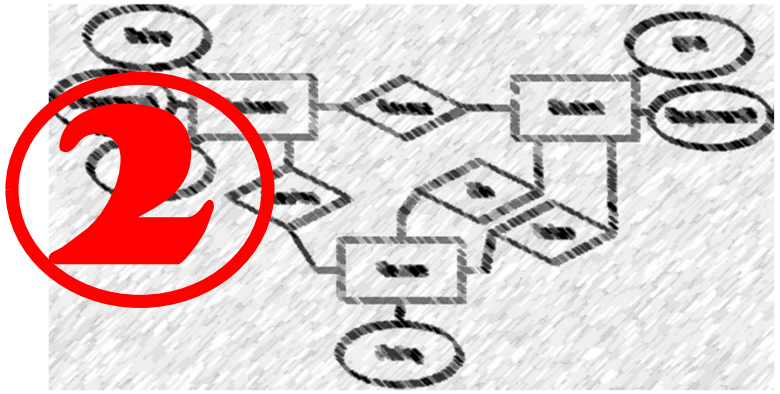
2

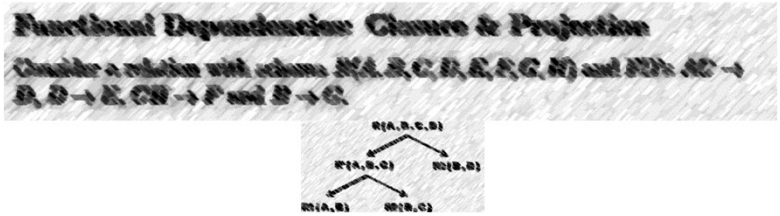


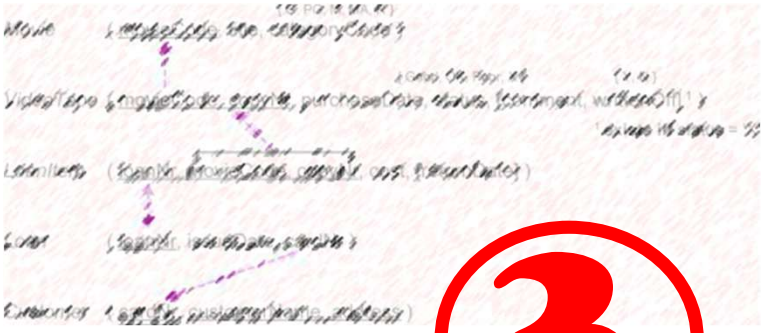
3

- We received a work plan in plain English.
- We derived its ER diagram.
- We created its Relational Schema and ICs.
-


Contrary to some popular conceptions of the Vikings, they were not a "race" linked by ties of common ancestry or patriotism, and could not be defined by any particular sense of "Viking-ness." Most of the Vikings whose activities are best known come from the areas now known as Denmark, Norway and Sweden, though there are mentions in historical records of Finnish, Estonian and Saami Vikings as well. Their common ground–and what made them different from the European peoples they confronted–was that they came from a foreign land, they were not "civilized" in the local understanding of the word and–most importantly–they were not Christian.




Functional Dependencies: Closure & Projection



- We received a work plan in plain English.
- We derived its ER diagram.
- We created its Relational Schema and ICs.
- We optimise the tables.

- Consider this table.
- Can you observe relations between values of different attributes?

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“Hero”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	98

- Consider this table.
- Can you observe relations between values of different attributes?
  - allay implies ALLAY

https://minecraft.fandom.com/wiki/Entity_format

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“Hero”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98

https://minecraft.fandom.com/wiki/Entity_format

- Consider this table.
- Can you observe relations between values of different attributes?
  - allay implies ALLAY
  - arrow implies ARROW
  - .....

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“Hero”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98

https://minecraft.fandom.com/wiki/Entity_format

- Consider this table.
- Can you observe relations between values of different attributes?
  - ID -> NAME, “ID implies NAME”, “NAME functionally dependent on ID”, “there is a FUNCTIONAL DEPENDENCY”.

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“Hero”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98

- Issues we may encounter if we keep table as of now. (Insertion/Update/Modification Anomaly)
  - How would I check If I entered the correct NAME value for a given ID value?

https://minecraft.fandom.com/wiki/Entity_format

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“Hero”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98
42	“Angora”	1	33	54	1	Boat_Chest	BOATH WITH GUN	1	88

- Issues we may encounter if we keep table as of now. (Insertion/Update/Modification Anomaly)
  - How would I check If I entered the correct NAME value for a given ID value?
  - Should I have to Check all tuples in the table?

https://minecraft.fandom.com/wiki/Entity_format

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“Hero”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98
42	“Angora”	1	33	54	1	Boat_Chest	BOATH WITH GUN	1	88

- Since ID->NAME (or lets say I->N for short)

- Since ID->NAME (or lets say I->N for short)
- I can **decompose** the table into two

- Since ID->NAME (or lets say I->N for short)
- I can decompose the table into two
- Where I keep the main table without the implied attribute **N** and create another table having implying and the implied attributes **I,N**.

- Since ID->NAME (or lets say I->N for short)
- Then I can decompose the table into two
- Where I keep the main table without N and create another table having I,N.

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	"VERRDO"	1	34	44	1	allay	ALLAY	0	45
234	"Hero"	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	"Zappara"	1	22	33	0	arrow	ARROW	1	22
0	"owned"	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	"uraz"	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	"Wirdo"	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98

- Since ID->NAME (or lets say I->N for short)
- Then I can decompose the table into two
- Where I keep the main table without N and create another table having I,N.

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“Hero”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	0	45
234	“Hero”	0	22	12	0	Magma_cube	0	12
12	“Zappara”	1	22	33	0	arrow	1	22
0	“owned”	1	-20	-20	0	Boat_chest	0	12
215	“uraz”	0	21	562	0	Magma_cube	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	0	98

- Since ID->NAME (or lets say I->N for short)
- Then I can decompose the table into two
- Where I keep the main table without N and create another table having I,N.

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“Hero”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	0	45
234	“Hero”	0	22	12	0	Magma_cube	0	12
12	“Zappara”	1	22	33	0	arrow	1	22
0	“owned”	1	-20	-20	0	Boat_chest	0	12
215	“uraz”	0	21	562	0	Magma_cube	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	0	98

- Since ID->NAME (or lets say I->N for short)
- Then I can decompose the table into two
- Where I keep the main table without N and create another table having I,N.

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“Hero”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“owned”	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98

DECOMPOSITION

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	0	45
234	“Hero”	0	22	12	0	Magma_cube	0	12
12	“Zappara”	1	22	33	0	arrow	1	22
0	“owned”	1	-20	-20	0	Boat_chest	0	12
215	“uraz”	0	21	562	0	Magma_cube	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	0	98

ID	NAME
allay	ALLAY
Magma_cube	MAGMA CUBE
arrow	ARROW
Boat_chest	BOAT WITH CHEST

- Since ID->NAME (or lets say I->N for short)
- Then I can decompose the table into two
- Where I keep the main table without N and create another table having I,N.

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	0	45
234	“Hero”	0	22	12	0	Magma_cube	0	12
12	“Zappara”	1	22	33	0	arrow	1	22
0	“owned”	1	-20	-20	0	Boat_chest	0	12
215	“uraz”	0	21	562	0	Magma_cube	0	88
0	“Wirdo”	1	-20	67	0	Boat_chest	0	98

ID	NAME
allay	ALLAY
Magma_cube	MAGMA CUBE
arrow	ARROW
Boat_chest	BOAT WITH CHEST

Decomposition

Lancaster University

Decompositions are done to remove redundant data that can lead to anomalies.

- There are levels of redundancy which are determined by Normal Forms.

Decomposition

Lancaster University

Decompositions are done to remove redundant data that can lead to anomalies.

- There are levels of redundancy which are determined by Normal Forms.
- Normal forms are set by revealing Functional Dependencies between attributes.

Functional Dependencies

Lancaster University

Does the following relation instance satisfy FD NAME->MOTION ?

ID	NAME	MOTION
allay	ALLAY	23
Magma_cube	MAGMA CUBE	44
allay	ALLAY	23
Boat_chest	BOAT WITH CHEST	44

Example

Lancaster University

- Some FDs on Minecraft table:
  - *CUSTOMNAMEVISIBLE* is the key: C $\longrightarrow$ CFfGI
  - *ID* determines *GLOWING*: I $\longrightarrow$ G

Did you notice anything wrong with the following instance ?

CUSTOMNAMEVISIBLE	FALLDISTANCE	fIRE	GLOWING	ID
			1	allay
			0	Magma_cube
			0	arrow
			0	Boat_chest
			1	Magma_cube
			0	Boat_chest

How do we find FD's?

Lancaster University

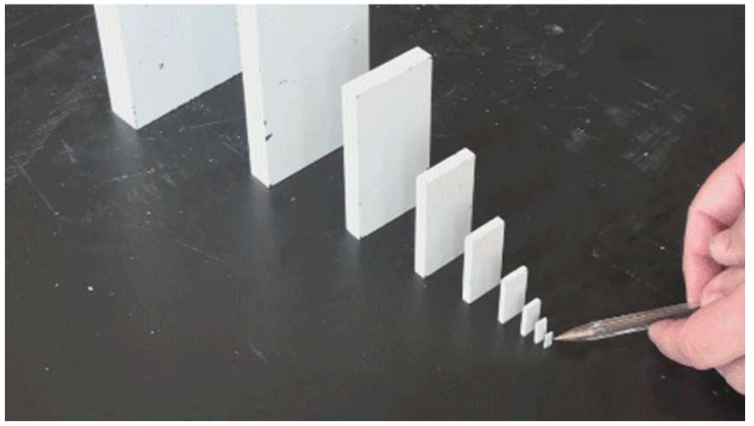
ID->NAME, What else?

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	"VERRDO"	1	34	44	1	allay	ALLAY	0	45
234	"Hero"	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	"Zappara"	1	22	33	0	arrow	ARROW	1	22
0	"owned"	1	-20	-20	0	Boat_chest	BOAT WITH CHEST	0	12
215	"uraz"	0	21	562	0	Magma_cube	MAGMA CUBE	0	88
0	"Wirdo"	1	-20	67	0	Boat_chest	BOAT WITH CHEST	0	98

Finding keys using $F+$

Lancaster University

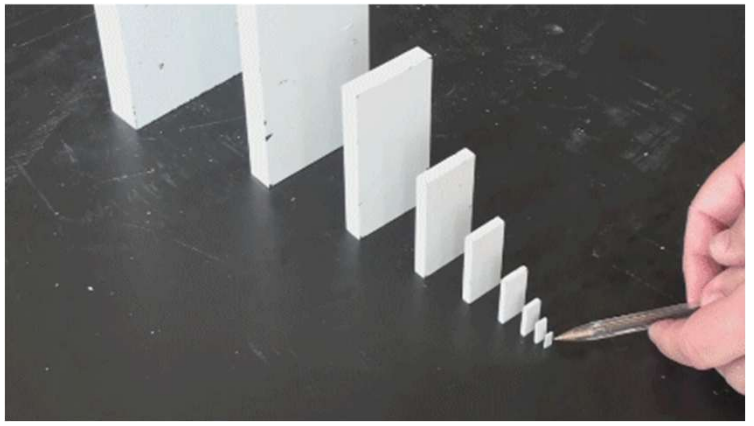
- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG.
-
-



Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
-



Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done using $F+$

Reasoning About FDs

Lancaster University

- F^{+} = *closure of F* is the set of all FDs that are implied by *F*.
-
  -
  -
  -
  -
  -
-

Reasoning About FDs

Lancaster University

- F^+= *closure of F* is the set of all FDs that are implied by *F*.
- Armstrong's Axioms (X, Y, Z are sets of attributes):
  -
  -
  -
  -
  -
-

Reasoning About FDs

Lancaster University

- F^{+} *closure of F* is the set of all FDs that are implied by *F*.
- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - *Reflexivity*: If X $\in$ Y, then Y -> X (a trivial FD) "IF X is IN Y" comment ☺
  -
  -
  -
  -
-

Reasoning About FDs

Lancaster University

- F^{+} *closure of F* is the set of all FDs that are implied by *F*.
- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - *Reflexivity*: If X $\in$ Y, then Y -> X (a trivial FD) "IF X is IN Y" comment ☺
  - *Augmentation*: If X -> Y, then XZ ->Z for any Z
  -
  -
  -
-

Reasoning About FDs

Lancaster University

- F^+ *closure of F* is the set of all FDs that are implied by *F*.
- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - *Reflexivity*: If X $\in$ Y, then Y -> X (a trivial FD) "IF X is IN Y" comment ☺
  - *Augmentation*: If X -> Y, then XZ ->Z for any Z
  - *Transitivity*: If X -> Y and Y->Z, then X->Z
  -
  -
-

Reasoning About FDs

Lancaster University

- F^{+} *closure of F* is the set of all FDs that are implied by *F*.
- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - *Reflexivity*: If X $\in$ Y, then Y -> X (a trivial FD) "IF X is IN Y" comment ☺
  - *Augmentation*: If X -> Y, then XZ ->Z for any Z
  - *Transitivity*: If X -> Y and Y->Z, then X->Z
  - *Union*: If X -> Y and X -> Z, then X -> YZ
  -
-

Reasoning About FDs

Lancaster University

- F^+ = *closure of F* is the set of all FDs that are implied by *F*.
- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - *Reflexivity*: If X $\in$ Y, then Y -> X (a trivial FD) "IF X is IN Y" comment ☺
  - *Augmentation*: If X -> Y, then XZ ->YZ for any Z
  - *Transitivity*: If X -> Y and Y->Z, then X->Z
  - *Union*: If X -> Y and X -> Z, then X -> YZ
  - *Decomposition*: If X -> YZ, then X -> Y and X -> Z

- These are *sound* and *complete* inference rules for FDs!

Reasoning About FDs

Lancaster University

For example, in the given schema, if

AIR, GLOWING -> AIR is a trivial FD because {AIR,GLOWING} is a superset of {AIR}

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION

Reasoning About FDs

Lancaster University

For example, in the given schema, if we are given

{AIR, GLOWING} -> FIRE as a FD, then {AIR, GLOWING, CUSTOMNAME} ->{FIRE, CUSTOMNAME} (by Augmentation)
or
AIR -> FIRE and FIRE -> GLOWING as a FDs, then AIR -> GLOWING (by Transitivity)

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION

Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs:

Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs:
  - In the above A->B, B->C this implicitly means that A->C ?

Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG, A->C
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs :
  - In the above A->B, B->C this implicitly means that A->C ?

Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG, A->C.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs :
  - In the above A->B, B->C this implicitly means that A->C ?
  - If BC->D and if D->EFG then BC->EFG so BCD->DEFG then BCD->DEFG BC->DEFG ?

Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG, A->C, BC->DEFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs :
  - In the above A->B, B->C this implicitly means that A->C ?
  - If BC->D and if D->EFG then BC->EFG so BCD->DEFG then BCD->DEFG BC->DEFG ?

Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG, A->C, BC->DEFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs :
  - In the above A->B, B->C this implicitly means that A->C ?
  - If BC->D and if D->EFG then BC->EFG so BCD->DEFG then BCD->DEFG BC->DEFG ?
  - If BC->DEFG then ABC->ADEFG?

Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG, A->C, BC->DEFG, ABC->ADEFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs :
  - In the above A->B, B->C this implicitly means that A->C ?
  - If BC->D and if D->EFG then BC->EFG so BCD->DEFG then BCD->DEFG BC->DEFG ?
  - If BC->DEFG then ABC->ADEFG?
  - If ABC -> ADEFG then ABBCC - > ABCDEFG?
  -
  -
  -
  -

Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG, A->C, BC->DEFG, ABC->ADEFG, ABC->ABCDEFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs :
  - In the above A->B, B->C this implicitly means that A->C ?
  - If BC->D and if D->EFG then BC->EFG so BCD->DEFG then BCD->DEFG BC->DEFG ?
  - If BC->DEFG then ABC->ADEFG?
  - If ABC -> ADEFG then ABBCC - > ABCDEFG?
  - So ABC -> ABCDEFG?
  - So ABC is a key… What else?
  -
  -

Finding keys using $F+$

Lancaster University

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG, A->C, BC->DEFG, ABC->ADEFG, ABC->ABCDEFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs :
  - In the above A->B, B->C this implicitly means that A->C ?
  - If BC->D and if D->EFG then BC->EFG so BCD->DEFG then BCD->DEFG BC->DEFG ?
  - If BC->DEFG then ABC->ADEFG?
  - If ABC -> ADEFG then ABBCC - > ABCDEFG?
  - So ABC -> ABCDEFG?
  - So ABC is a key… What else?
  -
  -

Finding keys using $F+$

Lancaster University

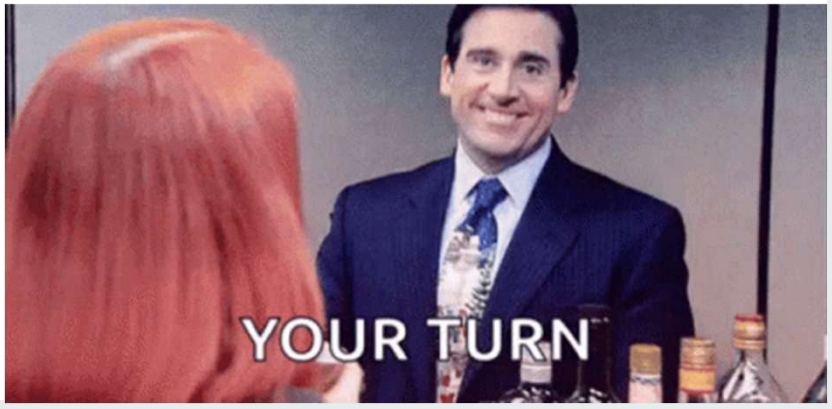
- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG, A->C, BC->DEFG, ABC->ADEFG, ABC->ABCDEFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs :
  - In the above A->B, B->C this implicitly means that A->C ?
  - If BC->D and if D->EFG then BC->EFG so BCD->DEFG then BCD->DEFG BC->DEFG ?
  - If BC->DEFG then ABC->ADEFG?
  - If ABC -> ADEFG then ABBCC - > ABCDEFG?
  - So ABC -> ABCDEFG?
  - So ABC is a key… What else?
  - A->B and A->C so AAA-> ABC
  - .

Finding keys using $F+$

- Let us assume that we are given a relation R={A,B,C,D,E,F,G}
- Also let us assume that we are given a set of FDs
  - A->B , B->C, BC->D, D->EFG, A->C, BC->DEFG, ABC->ADEFG, ABC->ABCDEFG, A->ABCDEFG.
- Whenever we are to optimise a relation, we have to analyse the FDs to reveal hidden dependencies.
- This can be done AAs :
  - In the above A->B, B->C this implicitly means that A->C ?
  - If BC->D and if D->EFG then BC->EFG so BCD->DEFG then BCD->DEFG BC->DEFG ?
  - If BC->DEFG then ABC->ADEFG?
  - If ABC -> ADEFG then ABBCC - > ABCDEFG?
  - So ABC -> ABCDEFG?
  - So ABC is a key… What else?
  - A->B and A->C so AAA-> ABC
  - So A->ABCDEFG and A is a key.

Reasoning About FDs

Lancaster University



Normal Forms

Lancaster University

- Normal forms are standards for a good DB schema (introduced by Codd in 1972)
- If a relation is in a particular normal form (such as BCNF, 3NF etc.), it is known that certain kinds of problems are avoided/minimised.
- Normal forms help us decide if decomposing a relation helps.

Normal Forms

Lancaster University

- First Normal Form: No set valued attributes (only atomic values)

AIR	NAMES	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	INVULNERABLE	MOTION
5500	“VERRDO”,” yUMA”..	1	34	44	1	allay	0	45
234	“Hero”,”ZEB UR”	0	22	12	0	Magma_cube	0	12
12	“Zappara”,” OLUMU”	1	22	33	0	arrow	1	22
0	“owned”	1	20	-20	0	Boat_chest	0	12
215	“uraz”,”GER ZO”	0	21	562	0	Magma_cube	0	88
0	“Wirdo”, “DONE”	1	-20	67	0	Boat_chest	0	98

Normal Forms

Lancaster University

- Candidate Key (a list of key attributes): A minimal set of attributes that can be chosen as the primary key for the table.
- Prime attribute: Any attribute that is part of a candidate key.
- Non-prime attribute: Any attribute that is not part of any candidate key.

- Let us assume that there are two candidate keys for a relation R(Age, CustomName, Name, Custom):
  - {(Age)} and {(CustomName, Name)}.
- Therefore, Age, CustomName, and Name are prime attributes.
- The remaining attribute(s), such as Custom, are non-prime attributes.

Normal Forms

Lancaster University

Second Normal Form: Table is in 1NF, and Every non-prime attribute should be ***fully functionally dependent*** on the **whole** part of primary key.

Normal Forms

Lancaster University

Second Normal Form: Table is in 1NF, and Every non-prime attribute should be ***fully functionally dependent*** on the **whole** part of primary key.

What does Fully Functional Dependent mean?

What does Fully Functional Dependent mean?

- There are two candidate keys for a relation R(Age, CustomName, Name, Custom):
  - {(Age)} and {(CustomName, Name)}.

- Let's assume the designer picks (CustomName, Name) as the primary key, and we know that Custom is a non-prime attribute.
- Then, if Name->Custom holds, then attribute Name partially determines attribute Custom. So, according to 2NF, Name->Custom introduces redundancy.
-
-

What does Fully Functional Dependent mean?

Lancaster University

- Let's assume the designer picks (CustomName, Name) as the primary key, and we know that Custom is a non-prime attribute.
- Then, if Name->Custom holds, then attribute Name partially determines attribute Custom. So, according to 2NF, Name->Custom introduces redundancy.
- If CustomName,Name->Custom holds, then Custom is ***fully functionally dependent*** on primary key.
- **Second Normal Form:** Every non-prime attribute should fully functionally depend on the primary key. (NOTE THAT WE MAY HAVE OTHER FDs !!!!)

Normal Forms

Lancaster University

Second Normal Form: Table is in 1NF, and Every non-prime attribute should be ***fully functionally dependent*** on the **whole** part of primary key.

A relation is in 2NF if it is in 1NF, and **every non-prime attribute of the relation depends on the whole of the primary key**. Note that it does not restrict the non-prime to non-prime attribute dependency.

Lancaster University

Second Normal Form:

To check:

Lancaster University

Second Normal Form:

To check:

1 Find the primary key,

Lancaster University

Second Normal Form:

To check:

1 Find the primary key,

2 Check if a non-prime attribute functionally depends on some parts of the primary key.

2nd Normal Form

Lancaster University

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?

2nd Normal Form

Lancaster University

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes

2nd Normal Form

Lancaster University

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find KEY
  -
-
  -
-

2nd Normal Form

Lancaster University

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find KEY
  - { (Su,P) }
-
  -
-

2nd Normal Form

Lancaster University

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find KEY
  - { (Su,P) }
- Check if a non-prime attribute is functionally dependent on some parts of the key.
  -
-

2nd Normal Form

Lancaster University

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find KEY
  - { (Su,P) }
- Check if a non-prime attribute is functionally dependent on some parts of the key.
  - Su -> L
-

2nd Normal Form

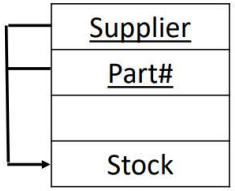
Lancaster University

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find KEY
  - { (Su,P) }
- Check if a non-prime attribute is functionally dependent on some parts of the key.
  - Su -> L
- Not in 2nd normal form.

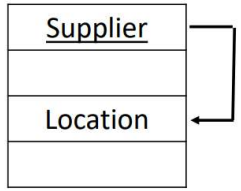
Normalise

- As a part (S) of a key (Su,P) implies NON-PRIME ATTRIBUTE {L}, we need to normalise the relation by introducing relations (Su,P,St) and (Su,L).



Supplier	Part#	Stock
Acme	1	17
Acme	2	25
Acme	3	17
Ajax	1	25
Ajax	3	18
Amco	1	3
Amco	2	22
Jamco	1	3

We give each functional dependency its own relation.



Supplier	Location
Acme	London
Ajax	Bristol
Amco	Glasgow
Jamco	Glasgow

Third Normal Form (3NF)

Lancaster University

- Relation R is in 3NF if it is in 2NF and for all $X \rightarrow A$
  - $A \in X$ (called a *trivial* FD), or
  - X contains a key (***superkey***) for R, or
  - A is part of some key for R. (A is a prime attribute)
  - In other words, **there should not be the case that another non-prime attribute determines a non-prime attribute.**
- If R is in 3NF, some redundancy still is possible. i.e. an attribute determines some other attribute. "A non-prime Attribute determines a prime attribute". This will be eliminated by BCNF.

*A set of attributes (S) is called a **superkey** if there exists a subset S' of S such that S' is a candidate key.*

3NF

Lancaster University

- E# is the key and Candidate key is {(E#)}.
-
-
-

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

Lancaster University

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic
-
  -
-

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

Lancaster University

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic, and there does not exist a non-prime attribute that is partially implied by part of a key in the candidate key (So it is in 2nd NF).
-
  -
-

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

Lancaster University

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic, and there does not exist a non-prime attribute that is partially implied by part of a key in the candidate key (So it is in 2nd NF).
- How about FDs of non-prime attributes?
  -
-

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

Lancaster University

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic, and there does not exist a non-prime attribute that is partially implied by part of a key in the candidate key (So it is in 2nd NF).
- How about FDs of non-prime attributes?
  - M#->MTEL# (or MTEL#->M#)
-

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

Lancaster University

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic, and there does not exist a non-prime attribute that is partially implied by part of a key in the candidate key (So it is in 2nd NF).
- How about FDs of non-prime attributes?
  - M#->MTEL# (or MTEL#->M#)
- So it is not in 3NF.

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

Lancaster University

- To normalise, for every FD we create a new table.
  - M#->MTEL# (or MTEL#->M#)
  - E#-> N,A,J#,M#

E#	Name	Age	Job#	M#
100	J. Smith	22	22	1
101	J. Smith	45	22	1
102	A. Adams	22	17	2
103	K. Bedford	35	12	3
104	F. Bloggs	55	17	3
108	A. Adams	35	12	3
109	J. Dean	35	12	1

M#	MTEL#
1	64413
1	64413
2	37611
3	18653
3	18653
3	18653
1	64413

Boyce-Codd Normal Form (BCNF)

- Relation R is in 3NF and with FDs *F* is in BCNF if, for all $X \rightarrow A$ in F^+
  - $A \in X$ (called a *trivial* FD), or
  - X contains a key for R. (i.e., X is a superkey)
- In other words, R is in BCNF if the only non-trivial FDs that hold over R are key constraints.
  - No dependency in R that can be predicted using FDs alone.
  - To check we must know all keys.

Normalisation (3 STEPS)

Lancaster University

For a given FD F we have to find all possible Keys using F^+ Closure.

Normalisation (3 STEPS)

Lancaster University

For a given FD F we have to find all possible Keys using F^+ Closure.

And then using the Normalisation rules we decide on the Normalisation level.

Normalisation (3 STEPS)

Lancaster University

For a given FD F we have to find all possible Keys using F^+ Closure.

And then using the Normalisation rules we decide on the Normalisation level.

If required level is not satisfied we decompose the relation according to the FDs that prevents required Normal Form

Summary

Lancaster University

- After getting the RM of your ERD, you need to optimise your tables against possible redundancies. You also need to protect your tables against anomalies.
- Normalisation is the phase in which you optimise your tables by analysing the functional dependencies, which can be identified through functional closure analysis.