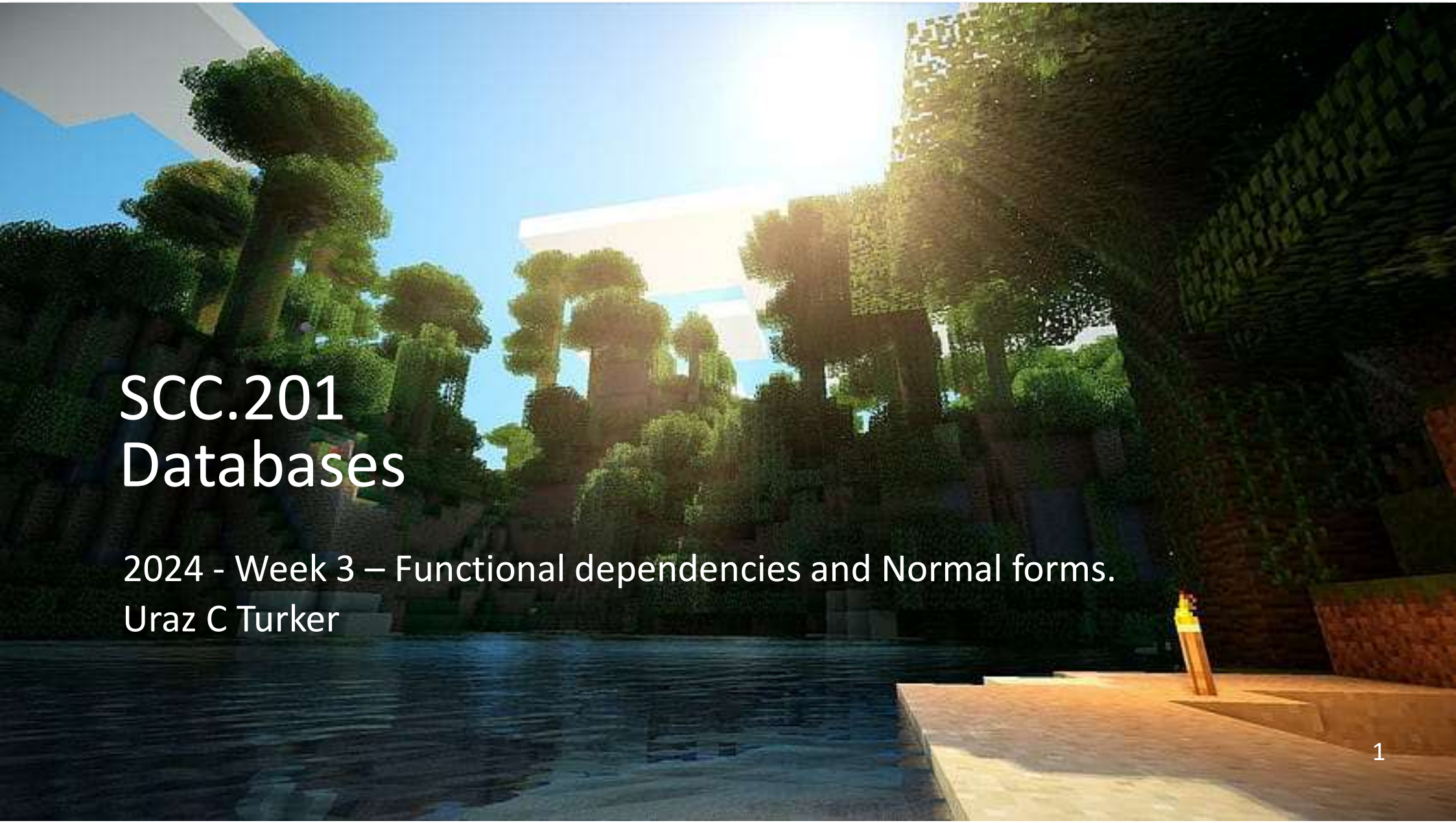


SCC.201 Databases

2024 - Week 3 – Functional dependencies and Normal forms.

Uraz C Turker

From you…

Lancaster University

Any other comments

- Not all of these Question forms need to be required. Incase someone has only 1 thing to say

- Multi-valued should not be used really if want it to be in normal form as first normal form states single valued attributes, so adding this can be used for the first few weeks and then changing might confuse some people

From you…

What aspect of the module did you find most difficult?

- Coming back to reinforce my learning is difficult

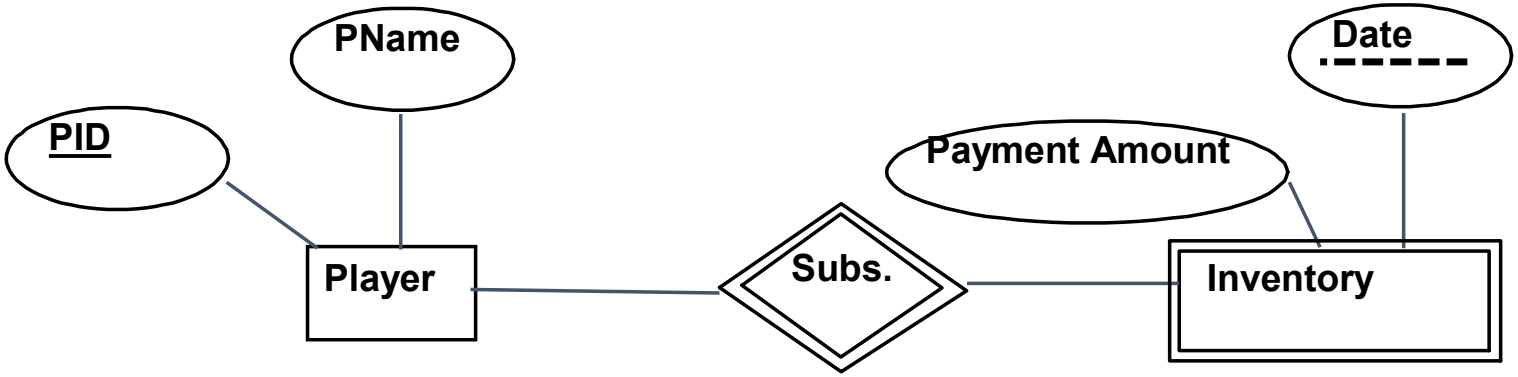
From you…

Lancaster University

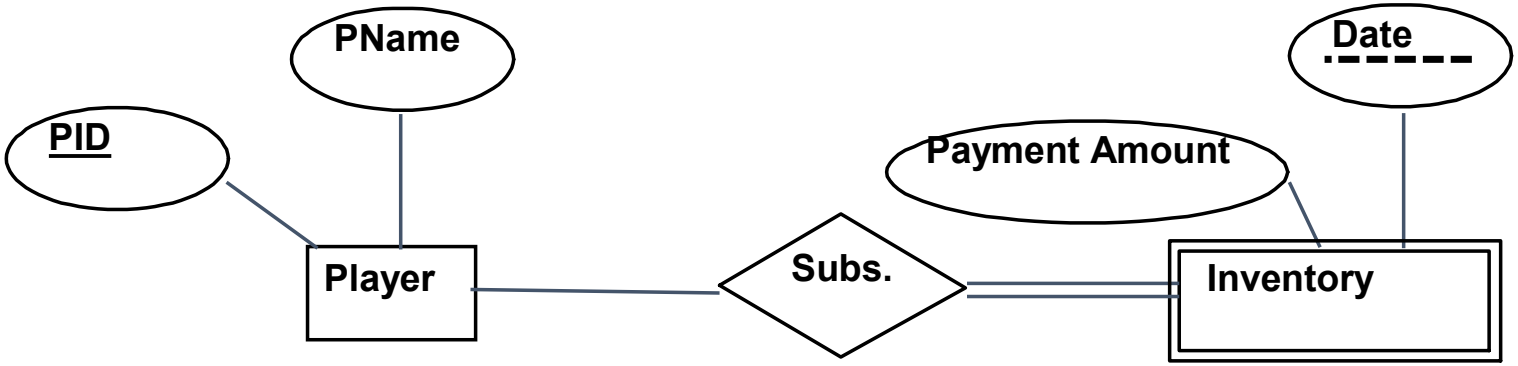
What can be improved to help with the module?

- At the beggining of each lecture, show which chapters of what book we can use to help ourselves

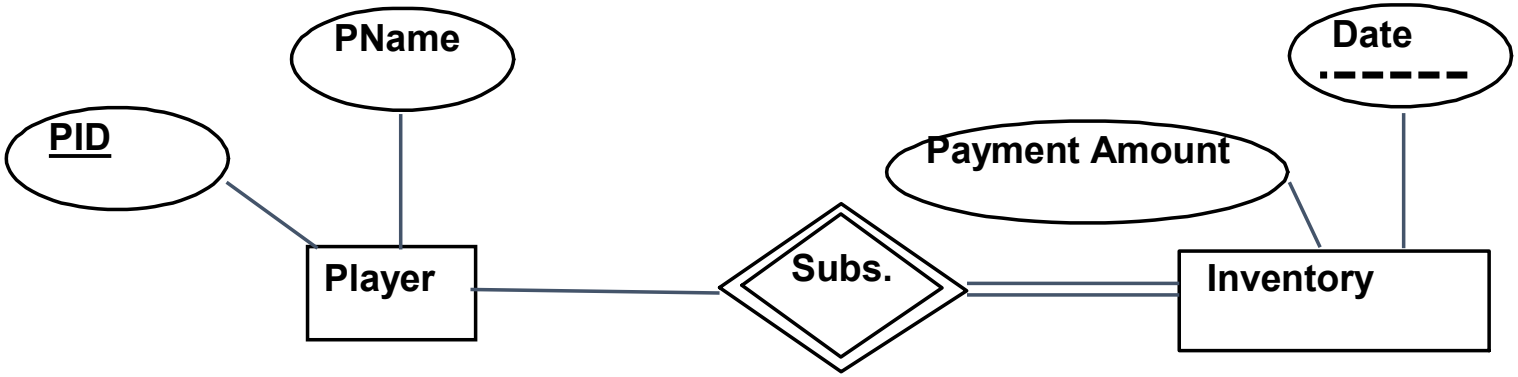
Lancaster University



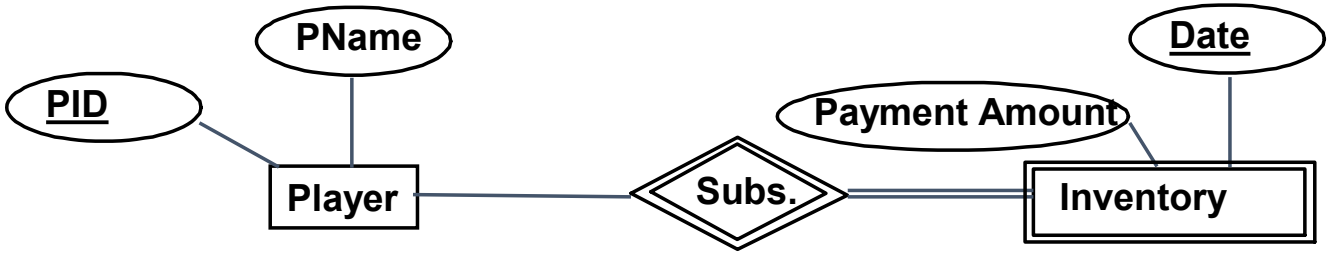
Lancaster University



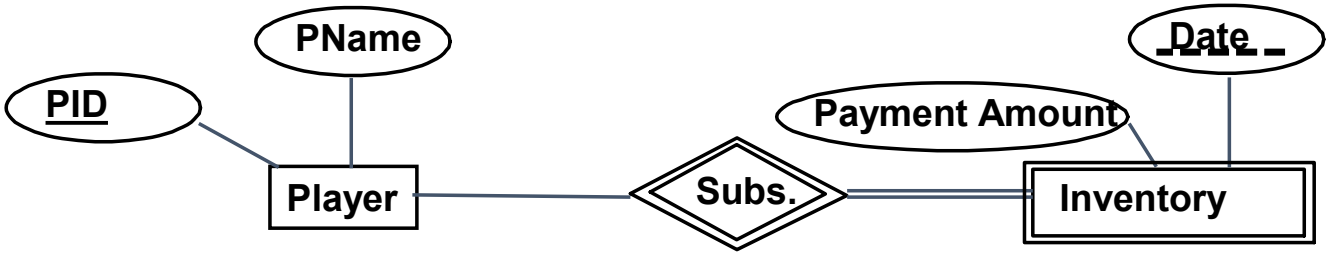
Lancaster University



Lancaster University



Lancaster University



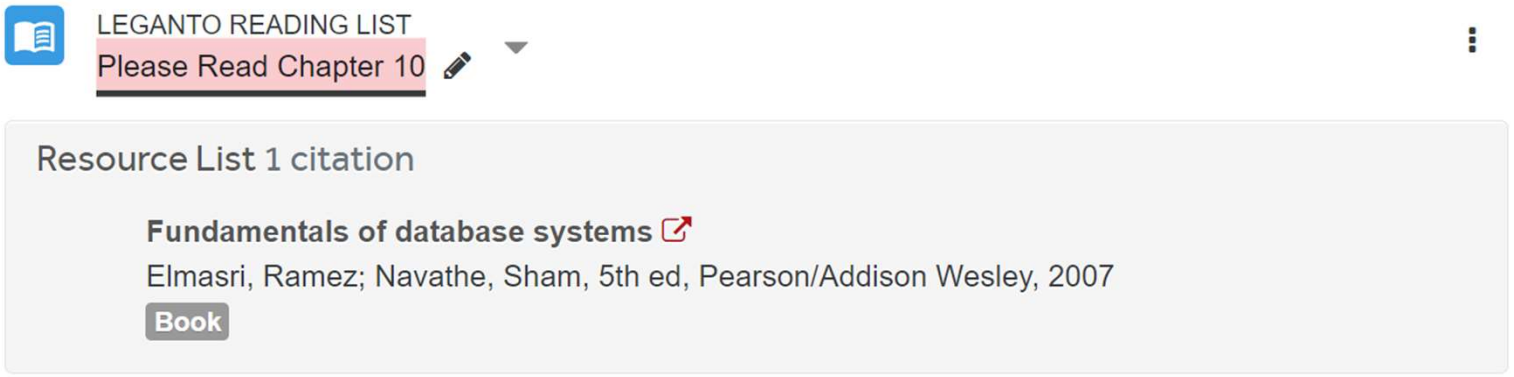
Week 1	Lecture 1	Introduction to the module, Why do we need Databases? Entity Relationship Model
	Lecture 2	Entity Relationship Model (ERM)
	Lab	NO LAB
Week2	Lecture 1	Relational Model (RM)
	Lecture 2	ER to RM
	Lab	ER diagrams.
Week 3	Lecture 1	Functional Dependencies
	Lecture 2	1st, 2nd, 3rd and Boyce-Cott Normal Forms
	Lab	ER to Relational Model.
Week 4	Lecture 1	Relational Algebra
	Lecture 2	Relational Algebra
	Lab	Functional dependencies and Normal forms
Week 5	Lecture 1	SQL Scripts
	Lecture 2	Advanced SQL Scripts
	Lab	Relational algebra
Week 6	Lecture 1	JDBC
	Lecture 2	Review
	Lab	Advanced SQL Scripting
Week 7	Lecture 1	Coursework, Record Search - B-Trees
	Lecture 2	Record Search - B-Trees (Cont.)
	Lab	Project
Week 8	Lecture 1	Search - Hashing
	Lecture 2	Physical Storage - record files, Storage - secondary files
	Lab	Project
Week 9	Lecture 1	Concurrency - Transaction Processing (cont)
	Lecture 2	Durability of Transactions and Crash Recovery
	Lab	Project
Week 10	Lecture 1	Advanced SQL - schemas, views, access control
	Lecture 2	Review
	Lab	Project

Lancaster University

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.



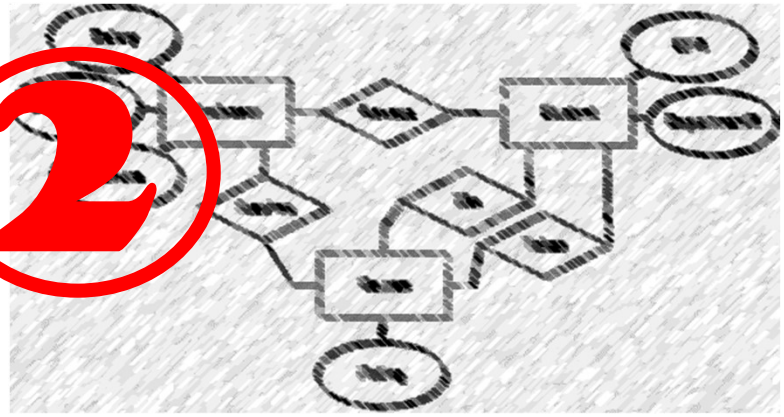
Contrary to some popular conceptions of the Vikings, they were not a "race" linked by ties of common ancestry or patriotism, and could not be defined by any particular sense of "Viking-ness." Most of the Vikings whose activities are best known come from the areas now known as Denmark, Norway and Sweden, 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

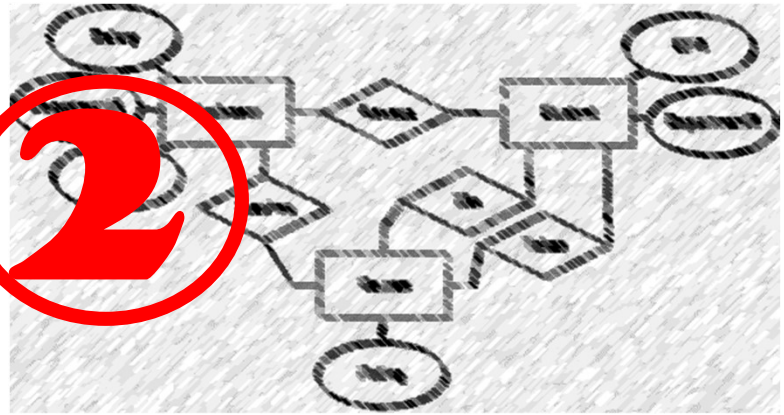
2



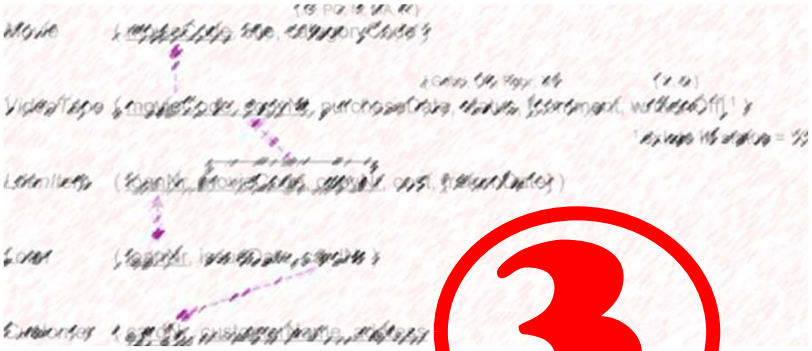
- We received a work plan in plain English.
- We derived its ER diagram.
-
-

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

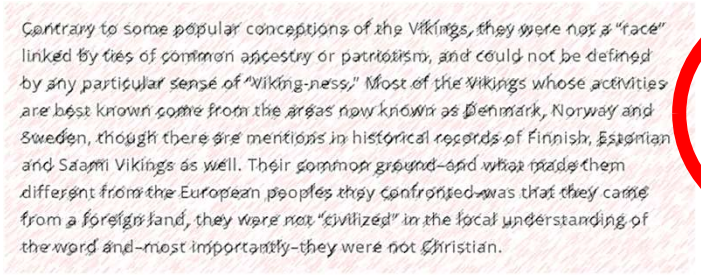


2

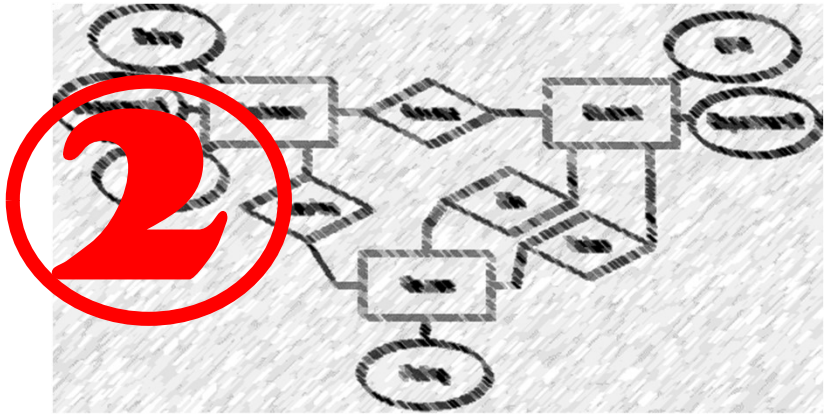


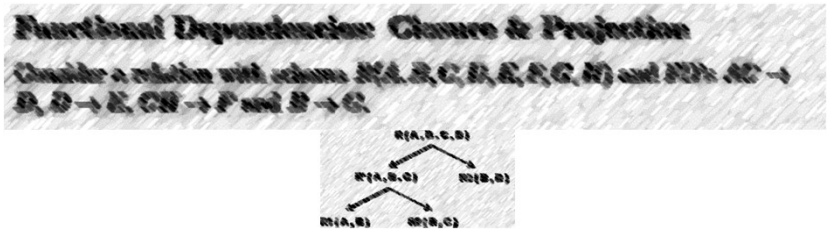
3

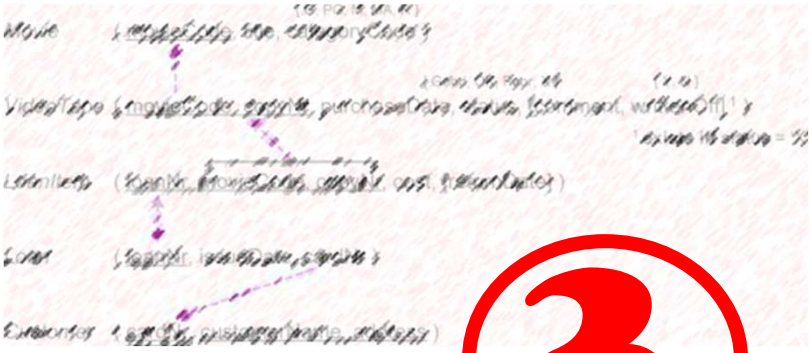
- We received a work plan in plain English.
- We derived its ER diagram.
- We created its Relational Schema and ICs.
-



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.







- 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. (Deletion Anomaly)
  - How would I retrieve NAME value if I delete all the tuples having ID=Boat_chest

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

- Issues we may encounter if we keep table as of now. (Deletion Anomaly)
  - How would I retrieve NAME value if I delete all the tuples having ID=Boat_chest
  - What was the NAME when ID = Boat_Chest?

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
215	“uraz”	0	21	562	0	Magma_cube	MAGMA CUBE	0	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?

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.

DECOMPOSITION

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

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.

Functional Dependencies (FORMAL)

Lancaster University

- A functional dependency between attributes X, Y ($X \longrightarrow Y$) holds over relation R if, for every allowable instance *r* of R:
  - $t1 \in r,\ t2 \in r,$ *such that* $\prod_X (t1) = \prod_X (t2)$ implies $\prod_Y (t1) = \prod_Y (t2)$
  - i.e., given two tuples in *r*, if the ID values agree, then the NAME values must also agree.

	ID	NAME	MOTION
t1	allay	ALLAY	23
	Magma_cube	MAGMA CUBE	44
	arrow	ARROW	1
t2	allay	ALLAY	54

Whenever ID value of a tuple is *allay*, NAME value of the same tuple is *ALLAY*
Whenever ID value of a tuple is *arrow*, NAME value of the same tuple is *ARROW*

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

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
Magma_cube	MAGMA CUBE	46

Example

Lancaster University

- Some FDs on Entities of Minecraft :
  - *CUSTOMNAMEVISIBLE* is the key: C $\rightarrow$ CFfGI
  - *ID* determines *GLOWING*: I $\rightarrow$ 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

Example

Lancaster University

- Some FDs on Entities of Minecraft :
  - *CUSTOMNAMEVISIBLE* is the key: C $\rightarrow$ CFfGI
  - *ID* determines *GLOWING*: I $\rightarrow$ 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

Example

Lancaster University

- Some FDs on Entities of Minecraft :
  - *CUSTOMNAMEVISIBLE* is the key: C $\rightarrow$ CFfGI
  - *ID* determines *GLOWING*: I $\rightarrow$ 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
			0	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 $F+$

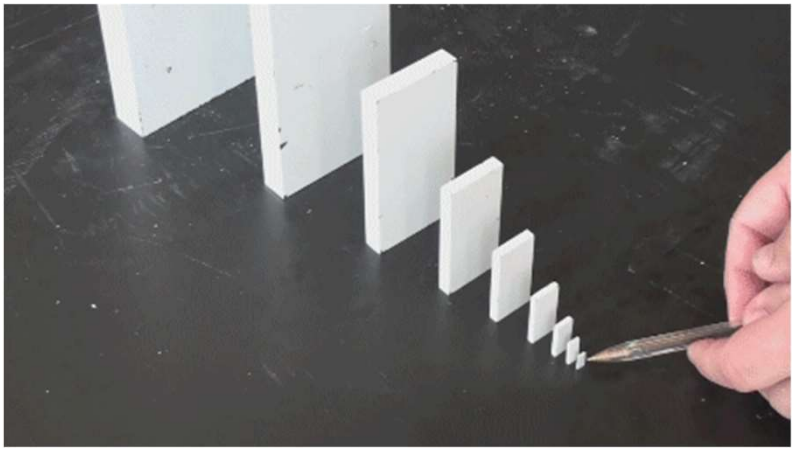
Lancaster University

- 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) "XY->Y"
  -
  -
  -
  -
-

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) "XY->Y"
  - *Augmentation*: If X -> Y, then XZ ->YZ 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) "XY->Y"
  - *Augmentation*: If X -> Y, then XZ ->YZ 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) "XY->Y"
  - *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
  -
-

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) "XY->Y"
  - *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
since {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)

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->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->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->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->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->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->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->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 $F+$

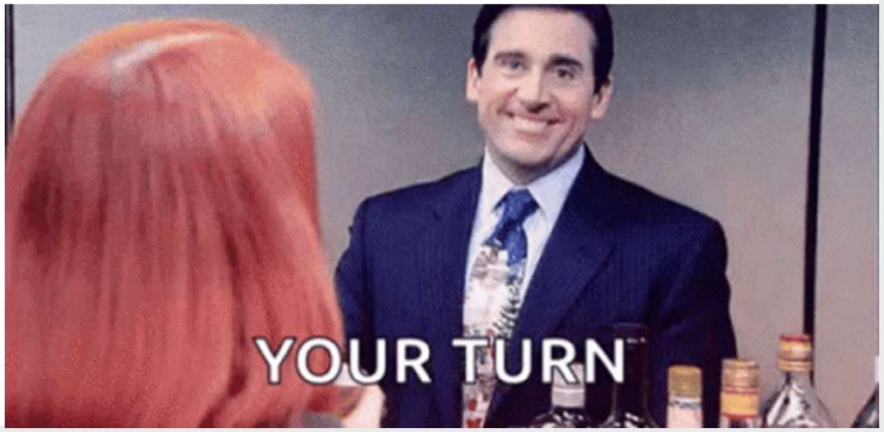
Lancaster University

- 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, 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->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 AA-> ABC
  - So A->ABCDEFG and A is a key.

Reasoning About FDs

Lancaster University

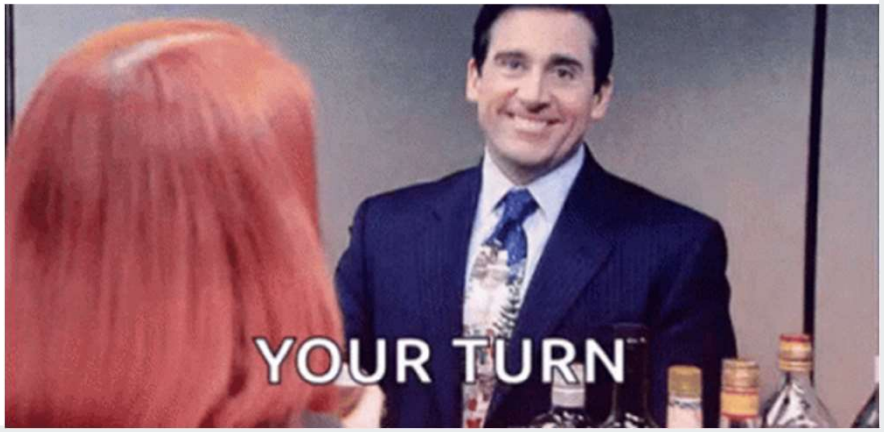
- Example: Contracts(*cid,sid,jid,did,pid,qty,value*), and given FDs:
  - C is the key: C $\longrightarrow$ CSJDPQV
  - JP -> C
  - SD -> P then prove that SDJ is a KEY:



Reasoning About FDs

Lancaster University

- Example: Contracts(*cid,sid,jid,did,pid,qty,value*), and given FDs:
  - C is the key: C $\longrightarrow$ CSJDPQV
  - JP -> C
  - SD -> P then prove that SDJ is a KEY:
- SD -> P implies SDJ -> JP (by Augmentation)

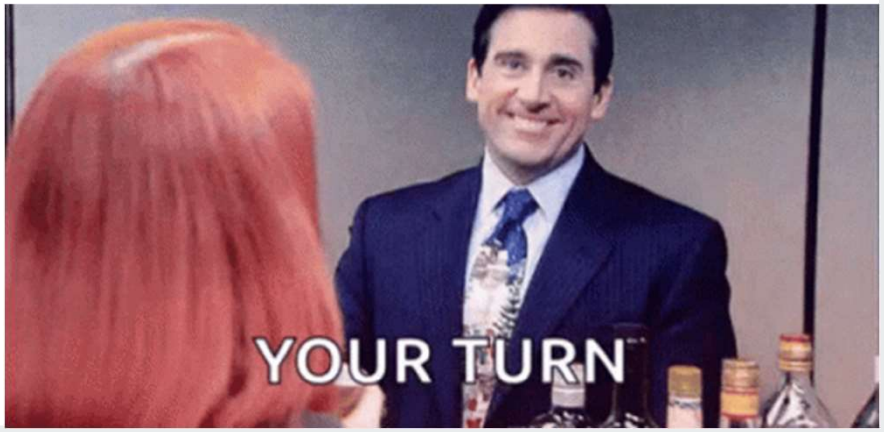


Reasoning About FDs

Lancaster University

- Example: Contracts(*cid,sid,jid,did,pid,qty,value*), and given FDs:
  - C is the key: C $\longrightarrow$ CSJDPQV
  - JP -> C
  - SD -> P then prove that SDJ is a KEY:
- SD -> P implies SDJ -> JP (by Augmentation)

- SDJ -> JP, JP -> CSJDPQV imply SDJ -> CSJDPQV



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

1) Flatten the multivalued attributes to atomic attributes.

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

1) Flatten the multivalued attributes to atomic attributes.

AIR	NAMES	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	INVULNERABLE	MOTION
5500	“yUMA”..	1	34	44	1	allay	0	45
5500	“VERRDO”	1	34	44	1	allay	0	45
234	“Hero”	0	22	12	0	Magma_cube	0	12
234	”ZEBUR”	0	22	12	0	Magma_cube	0	12
12	“Zappara”	1	22	33	0	arrow	1	22
12	”OLUMU”	1	22	33	0	arrow	1	22
0	“owned”	1	-20	-20	0	Boat_chest	0	12
…	…	..	..	..	..	…	…	..

Normal Forms

Lancaster University

1) Flatten the multivalued attributes to atomic attributes.
2) Extend the Primary key so that it contains the new atomic attribute.

AIR	NAMES	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	INVULNERABLE	MOTION
5500	“yUMA”..	1	34	44	1	allay	0	45
5500	“VERRDO”	1	34	44	1	allay	0	45
234	“Hero”	0	22	12	0	Magma_cube	0	12
234	”ZEBUR”	0	22	12	0	Magma_cube	0	12
12	“Zappara”	1	22	33	0	arrow	1	22
12	”OLUMU”	1	22	33	0	arrow	1	22
0	“owned”	1	-20	-20	0	Boat_chest	0	12
…	…	..	..	..	..	…	…	..

Normal Forms

Lancaster University

1) Flatten the multivalued attributes to atomic attributes.
2) Extend the Primary key so that it contains the new atomic attribute.

AIR	NAMES	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	INVULNERABLE	MOTION
5500	“yUMA”..	1	34	44	1	allay	0	45
5500	“VERRDO”	1	34	44	1	allay	0	45
234	“Hero”	0	22	12	0	Magma_cube	0	12
234	”ZEBUR”	0	22	12	0	Magma_cube	0	12
12	“Zappara”	1	22	33	0	arrow	1	22
12	”OLUMU”	1	22	33	0	arrow	1	22
0	“owned”	1	-20	-20	0	Boat_chest	0	12
…	…	..	..	..	..	…	…	..

Lancaster University

Lets start

Previously

Lancaster University

- Functional dependency between attributes.

Previously

Lancaster University

- A functional dependency between attributes X, Y (X Y) holds over relation R if, for every allowable instance *r* of R:
  - *t1 -> r, t2 -> r, such that* $\prod_X (t1) = \prod_X (t2)$ implies $\prod_Y (t1) = \prod_Y (t2)$
  - i.e., given two tuples in *r*, if the ID values agree, then the NAME values must also agree.

Previously

Lancaster University

- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - *Reflexivity*: If $X \in Y$, then Y -> X (a trivial FD) "XY->Y"
  - *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

Previously

Lancaster University

1st Normal Form

Previously

Lancaster University

1st Normal Form

All attributes are atomic

Previously

Lancaster University

1st Normal Form

All attributes are atomic.

Flatten the multi-valued attribute

Extend the Primary Key to the flattened attribute.

Normal Forms

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“VERRDO”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“URAZ”	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

- 2nd Normal form
- Prime attribute: any attribute that is part of a CANDIDATE KEY
- Non-prime attributes: The rest of the attributes

Normal Forms

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“VERRDO”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“URAZ”	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

- 2nd Normal form
- Prime attribute: any attribute that is part of a CANDIDATE KEY
- Non-prime attributes: The rest of the attributes

Let us assume that there are **two** Candidate Keys which are {(CUSTOMNAME, ID), (CUSTOMNAME, NAME)}.

Normal Forms

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“VERRDO”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“URAZ”	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

- 2nd Normal form
- Prime attribute: any attribute that is part of a CANDIDATE KEY
- Non-prime attributes: The rest of the attributes

Let us assume that there are **two** Candidate Keys which are {(CUSTOMNAME, ID), (CUSTOMNAME, NAME)}.

Therefore, CUSTOMNAME, ID and NAME are PRIME ATTRIBUTEs.

Normal Forms

AIR	CUSTOMNAME	CUSTOMNAMEVISIBLE	FALLDISTANCE	FIRE	GLOWING	ID	NAME	INVULNERABLE	MOTION
5500	“VERRDO”	1	34	44	1	allay	ALLAY	0	45
234	“VERRDO”	0	22	12	0	Magma_cube	MAGMA CUBE	0	12
12	“Zappara”	1	22	33	0	arrow	ARROW	1	22
0	“URAZ”	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

- 2nd Normal form
- Prime attribute: any attribute that is part of a CANDIDATE KEY
- Non-prime attributes: The rest of the attributes

Let us assume that there are **two** Candidate Keys which are {(CUSTOMNAME, ID), (CUSTOMNAME, NAME)}.

Therefore, CUSTOMNAME, ID and NAME are PRIME ATTRIBUTEs.

And the REST ARE NONPRIME 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?

Lancaster University

- Prime attribute: any attribute that is part of a candidate key
- Non-prime attributes: rest of the attributes
-
-
-

What does Fully Functional Dependent mean?

Lancaster University

- Prime attribute: any attribute that is part of a candidate key
- Non-prime attributes: rest of the attributes
- If AB is a key, and C is a non-prime attribute, then if A->C holds then A ***partially determines*** C (there is a partial functional dependency to a key) “then A->C introduces redundancy”
-
-

What does Fully Functional Dependent mean?

Lancaster University

- Prime attribute: any attribute that is part of a candidate key
- Non-prime attributes: rest of the attributes
- If AB is a key, and C is a non-prime attribute, then if A->C holds then A ***partially determines*** C (there is a partial functional dependency to a key) “then A->C introduces redundancy”
- If C is a non-prime attribute, and if AB->C holds then C ***fully functionally dependent*** on AB.
-

What does Fully Functional Dependent mean?

Lancaster University

- Prime attribute: any attribute that is part of a candidate key
- Non-prime attributes: rest of the attributes
- If AB is a key, and C is a non-prime attribute, then if A->C holds then A ***partially determines*** C (there is a partial functional dependency to a key) “then A->C introduces redundancy”
- If C is a non-prime attribute, and if AB->C holds then C ***fully functionally dependent*** on AB.
- Second Normal Form: Every non-prime attribute should fully functionally depend on the primary key. (NOTE THAT WE MAY HAVE OTHER FDs !!!!)

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

-
-
-
-
  -
  -
-

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

2nd Normal form

Lancaster University

- Let us assume that there is a Primary Key, which is { (ID, NAME) } (composite key).
-
-
-
  -
  -
-

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

2nd Normal form

Lancaster University

- Let us assume that there is a Primary Key, which is { (ID, NAME) } (composite key).
- Therefore, ID and NAME are PRIME ATTRIBUTEs.
-
-
  -
  -
-

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

2nd Normal form

Lancaster University

- Let us assume that there is a Primary Key, which is { (ID, NAME) } (composite key).
- Therefore, ID and NAME are PRIME ATTRIBUTEs.
- And the INVULNERABLE is a NONPRIME ATTRIBUTE.
-
  -
  -
-

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

2nd Normal form

Lancaster University

- Let us assume that there is a Primary Key, which is { (ID, NAME) } (composite key).
- Therefore, ID and NAME are PRIME ATTRIBUTEs.
- And the INVULNERABLE is a NONPRIME ATTRIBUTE.
- Observe:
  -
  -
-

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

2nd Normal form

Lancaster University

- Let us assume that there is a Primary Key, which is { (ID, NAME) } (composite key).
- Therefore, ID and NAME are PRIME ATTRIBUTEs.
- And the INVULNERABLE is a NONPRIME ATTRIBUTE.
- Observe:
  - If ID = {allay} or {Boat_chest}, INVULNERABLE is 0,
  -
-

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

2nd Normal form

Lancaster University

- Let us assume that there is a Primary Key, which is { (ID, NAME) } (composite key).
- Therefore, ID and NAME are PRIME ATTRIBUTEs.
- And the INVULNERABLE is a NONPRIME ATTRIBUTE.
- Observe:
  - If ID = {allay} or {Boat_chest}, INVULNERABLE is 0,
  - Else If ID=arrow, INVULNERABLE is 1,
-

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

2nd Normal form

Lancaster University

- Let us assume that there is a Primary Key, which is { (ID, NAME) } (composite key).
- Therefore, ID and NAME are PRIME ATTRIBUTEs.
- And the INVULNERABLE is a NONPRIME ATTRIBUTE.
- Observe:
  - If ID = {allay} or {Boat_chest}, INVULNERABLE is 0,
  - Else If ID=arrow, INVULNERABLE is 1,
- There is PARTIAL DEPENDENCY (ID/NAME implies INVULNERABLE), so the relation is not in the 2nd Normal Form.

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

Normalise

Lancaster University

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

ID	NAME
allay	ALLAY
arrow	ARROW
arrow	ARROW
Boat_chest	BOAT WITH CHEST
allay	ALLAY
Boat_chest	BOAT WITH CHEST

ID	INVULNERABLE
allay	0
arrow	1
Boat_chest	0

Normalise (OR)

Lancaster University

ID	NAME	INVULNERABLE
allay	ALLAY	0
arrow	ARROW	1
arrow	ARROW	1
Boat_chest	BOAT WITH CHEST	0
allay	ALLAY	0
Boat_chest	BOAT WITH CHEST	0

ID	NAME
allay	ALLAY
arrow	ARROW
arrow	ARROW
Boat_chest	BOAT WITH CHEST
allay	ALLAY
Boat_chest	BOAT WITH CHEST

NAME	INVULNERABLE
ALLAY	0
ARROW	1
BOAT WITH CHEST	0

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

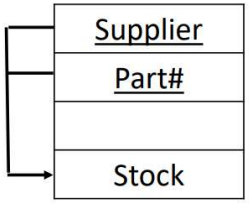
Lancaster University

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.

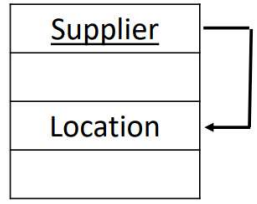
Normalise

- 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.
  - 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 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

We must find all possible Keys using F+ Closure for a given FD F.

Normalisation (3 STEPS)

Lancaster University

We must find all possible Keys using F+ Closure for a given FD F.

Then, using the Normalisation rules, we decide on the Normalisation level.

Normalisation (3 STEPS)

Lancaster University

We must find all possible Keys using F+ Closure for a given FD F.

Then, using the Normalisation rules, we decide on the Normalisation level.

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

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E }
  - What is the **Normalisation level**?
-
-
-
-

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E }
  - What is the **Normalisation level**?
- AC->BE…….AC->B, AC->E (decomposition)
-
-
-

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E , AC->B, AC->E}
  - What is the **Normalisation level?**
- AC->BE…….AC->B, AC->E (decomposition)
-
-
-

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E,AC->B,AC->E }
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC (Augmentation)
-
-

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC}
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC (Augmentation)
-
-

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC }
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D (Transitivity)
-

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D }
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D (Transitivity)
-

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D }
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D
- AC is the candidate key as AC->BE, AC->D, by union and augmentation AC→ABCDE.

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D }
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D
- AC is the candidate key as AC->BE, AC->D, by union and augmentation AC→ABCDE.
- {A,C} are PRIME ATTRIBUTES & {B,D,E} are NONPRIME ATTRIBUTES.

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= {BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D}
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D
- AC is the candidate key as AC->BE, AC->D, by union and augmentation AC→ABCDE.
- {A,C} are PRIME ATTRIBUTES & {B,D,E} are NONPRIME ATTRIBUTES.

Observation 1: Does any prime attribute determine a nonprime attribute?

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= {BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D }
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D

Observation 1: Does any prime attribute determine a nonprime attribute? (think on BC->D) So NOT IN 2NF.

- AC is the candidate key as AC->BE, AC->D, by union and augmentation AC→ABCDE.
- {A,C} are PRIME ATTRIBUTES & {B,D,E} are NONPRIME ATTRIBUTES.

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= {BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D }
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D

The answer is 1NF.

- AC is the candidate key as AC->BE, AC->D, by union and augmentation AC→ABCDE.
- {A,C} are PRIME ATTRIBUTES & {B,D,E} are NONPRIME ATTRIBUTES.

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= {BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D }
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D
- AC is the candidate key as AC->BE, AC->D, by union and augmentation AC→ABCDE.
- {A,C} are PRIME ATTRIBUTES & {B,D,E} are NONPRIME ATTRIBUTES.

To make it for 2NF: Decompose into ABCE and BCD.

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= {BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D}
  - What is the **Normalisation level**?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D

Based on ABCE and BCD
Observation 2: B determines E, so the ABCE is not in 3NF and not in BCNF.

- AC is the candidate key as AC->BE, AC->D, by union and augmentation AC→ABCDE.
- {A,C} are PRIME ATTRIBUTES & {B,D,E} are NONPRIME ATTRIBUTES.

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= {BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D}
  - How do we enhance its normalisation to 3NF?
- AC->BE……..AC->B, AC->E **Solution:**
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D
- AC is the candidate key as AC->BE, AC->D, by union and augmentation AC→ABCDE.
- {A,C} are PRIME ATTRIBUTES & {B,D,E} are NONPRIME ATTRIBUTES.

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= {BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D}
  - How do we enhance its normalisation to 3NF?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D

Solution: As B->E, the nonprime attribute implies the nonprime attribute. We decompose ABCE to ABC and BE.

- AC is the candidate key as AC->BE, AC->D, by union and augmentation ABC→ABCDE.
- {A,C} are PRIME ATTRIBUTES & {B,D,E} are NONPRIME ATTRIBUTES.

Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= {BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D}
  - How do we enhance its normalisation to BCNF?
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D
- AC is the candidate key as AC->BE, AC->D, by union and augmentation AC→ABCDE.
- {A,C} are PRIME ATTRIBUTES & {B,D,E} are NONPRIME ATTRIBUTES.

Solution: Finally we have three tables ABC, BE, and BCD. This is in 3NF, and BCNF.

More examples are in Lab material Next Week.

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.

Question 2

Functional Dependencies and Normal forms

2.a. Let ***R(A,B,C,D)*** be a relation with functional dependencies ***{A->C, C->D}***. By using the Armstrong's Axioms prove that {***AB->ABCD***}. Show all the steps involved.

[5 marks]

A->D by Transitivity, AAA -> ACD by augmentation, A->ACD, AB->ABCD.

2.b. What is the normalisation level of relation *R*?

[2 marks]

A and B are prime numbers, but a Part of the key (A) implies C (D) so it is in 1NF.

2.c. By decomposing the relation R, create relations in the Boyce-Codd normal form and show that they are indeed in the Boyce-Codd normal form.

[7 marks]

AB, AC and CD. In all tables, the only dependencies are superkey dependencies (AB), (A->C), and (C->D).

2.d. Fill in the blanks

I. A …… between attributes X, Y (X -> Y) holds over relation R if, for every allowable instance *r* of R given two tuples in *r*, if the X values agree, then the Y values must also agree.

[2 marks]

Functional dependency

II. If a table is……., and decomposed into smaller tables, it is known that certain kinds of problems are avoided/minimised.

[2 marks]

Normalised

III. ……. Normal Form: No set-valued attributes.

[2 marks]

First

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

Lancaster University

2.e. The above given instance of relation ***R(EMP_ID,PROJECT_ID, MANAGER)*** has a primary key **{EMP_ID,PROJECT_ID}**. Identify:

I. Give the Functional dependencies of R. **[2.5 marks]**

Project_ID-> MANAGER

II. Print the non-prime attributes of R. **[2.5 marks]**

MANAGER

Nextweek

Lancaster University