

Normalization

+
•
o

EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

EID	EN	DID	Foreign key	DID	DNAME	ADDRESS
1	A	1		1	CSE	BLOCK A
2	B	2		2	ECE	BLOCK B
3	C	3		3	ME	BLOCK C
4	A	3				
5	E	1				

EID	EN	DID	DID	DNAME	ADDRESS
1	A	1	1	CSE	BLOCK A
2	B	2	2	ECE	BLOCK B
3	C	3	3	ME	BLOCK C
4	A	3	3	ME	BLOCK C
5	E	1	1	CSE	BLOCK A

EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

EID	EN	DID	DNAME	ADDRESS
1	A	1	CSE	BLOCK A
2	B	2	ECE	BLOCK B
3	C	3	ME	BLOCK C
4	A	3	ME	BLOCK C
5	E	1	CSE	BLOCK A
6	E	1	CSE	BLOCK A
7	E	1	CSE	BLOCK A
8	E	1	CSE	BLOCK A

Redundancy

EID	EN	DID	Foreign key	DID	DNAME	ADDRESS
1	A	1		1	CSE	BLOCK A
2	B	2		2	ECE	BLOCK B
3	C	3		3	ME	BLOCK C

EID	EN	DID
4	A	3
5	E	1

EID	EN	DID	DNAME	ADDRESS
1	A	1	CSE	BLOCK A
2	B	2	ECE	BLOCK B
3	C	3	ME	BLOCK C
4	A	3	ME	BLOCK C
5	E	1	CSE	BLOCK A
6	m	1	CSE	BLOCK A
7	n	1	CSE	BLOCK A
8	n	1	CSE	BLOCK A
Null	null	4	AE	BLOCK F

INSERTION ANAMOLY



Redundancy



EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

DELETION ANAMOLY

EID	EN	DID	DNAME	ADDRESS
1	A	1	CSE	BLOCK A
2	B	2	ECE	BLOCK B
3	C	3	ME	BLOCK C
4	A	3	ME	BLOCK C
5	E	1	CSE	BLOCK A
6	m	1	CSE	BLOCK A
7	n	1	CSE	BLOCK A
8	n	1	CSE	BLOCK A
9	o	1	ECE	BLOCK A

Redundancy

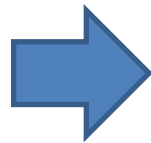
EID	EN	DID
1	A	1
2	B	2
3	C	3
4	A	3
5	E	1

Foreign key

DID	DNAME	ADDRESS
1	CSE	BLOCK A
2	ECE	BLOCK B
3	ME	BLOCK C

EID	EN	DID	DNAME	ADDRESS
1	A	1	CS	BLOCK A
2	B	2	ECE	BLOCK B
3	C	3	ME	BLOCK C
4	A	3	ME	BLOCK C
5	E	1	CS	BLOCK A
6	m	1	CS	BLOCK A
7	n	1	CS	BLOCK A
8	n	1	CSE	BLOCK A
9	o	1	ECE	BLOCK A

UPDATE ANAMOLY



Redundancy

Normalization

- **Normalization** is the process of organizing the data in the database.
- **Normalization** is used to minimize the redundancy from a relation or set of relations.
- It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.

Is this a good database design?

<u>sid</u>	sname	addres s	cid	cname	grad e
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

Insertion Anomaly

- Tried to insert data in a record that makes null value in a prime attribute.

<u>sid</u>	sname	address	cid	cname	grade
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

- Let, a new course has been introduced but no student has been registered yet.
- So cid=205, cname=OS. But all the other values will be null.
- Here, sid is a primary key and can't be null.
- This type of anomaly is known as insertion anomaly.

Update Anomaly

- When we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.

<u>sid</u>	sname	address	cid	cname	grade
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

- Lets assume, Britney updated her address from USA to London.
- If it is not updated in all 3 places, database will face an inconsistent state.

Deletion Anomaly

- We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else. Or deletion make other unsaved data removal.

<u>sid</u>	sname	address	cid	cname	grade
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

- Assume, university wants to discontinue Java course with cid=203.
- The whole tuple will be deleted.
- Then we will loose all the data of Victoria.
- This type of anomaly is known as deletion anomaly.

Is this a good design?

<u>sid</u>	sname	address	cid	cname	grade
124	Britney	USA	206	Database	A++
204	Victoria	Essex	203	Java	C
124	Britney	USA	201	S/Eng I	A+
206	Emma	London	206	Database	B-
124	Britney	USA	202	Semantics	B+

- So, the design is good only for read mode.
- In write mode, it will go to inconsistent state.
- We have to decompose the table for removing all the dependency.

Functional Dependency

- Functional dependency (FD) is a set of constraints between two attributes in a relation.
- Functional dependency says that if two tuples have same values for attributes A_1, A_2, \dots, A_n , then those two tuples must have to have same values for attributes B_1, B_2, \dots, B_n .
- Functional dependency is represented by an arrow sign (\rightarrow) that is, $X \rightarrow Y$, where X functionally determines Y . (or Y is determined by X)
- The left-hand side attributes determine the values of attributes on the right-hand side.

Functional Dependency

Sid \rightarrow Sname (Sid determines Sname)

Sachin

Sachin

Is these two person same or different?

1st Possibility:

Sid \rightarrow Sname

1 Sachin

1 Sachin

2nd Possibility:

Sid \rightarrow Sname

1 Sachin

2 Sachin

$X \rightarrow Y$

Here, X is **determinant**.

Here, Y is **dependent**.

Informally we can say, if we face any difficulty in dependent side, we have to look into determinant side to solve the confusion.

1st case: redundant data of same person

2nd case: different person

Rule out the functional dependency based on the tables??

EID->ENAME

ENAME->EID

EID	ENAME
1	A
2	B
3	B

Rule out the functional dependency based on the tables??



INCORRECT

EID->ENAME

ENAME->EID

EID	ENAME
1	A
2	B
3	B

Rule out the functional dependency based on the tables??

A→B

B→A

A	B
1	1
1	2
2	2

Rule out the functional dependency based on the tables??

A→B

B→A

A	B
1	1
1	2
2	2

Rule out the functional dependency based on the tables??

A→B

B→C

B→A

C→B

C→A

A→C

A	B	C
1	1	4
1	2	4
2	1	3
2	2	3
2	4	3

Rule out the functional dependency based on the tables??

A→B

B→C

B→A

C→B

C→A

A→C

A	B	C
1	1	4
1	2	4
2	1	3
2	2	3
2	4	3

Rule out the functional dependency based on the tables??

A→B

B→C

B→A

C→B

C→A

A→C

A	B	C
1	1	4
1	2	4
2	1	3
2	2	3
2	4	3

Armstrong's Axioms

Armstrong's Axioms are a set of rules, that when applied repeatedly, generates a closure of functional dependencies.

- **Reflexive rule** – If X is a set of attributes and Y is subset of X , then X holds a value of Y . $Y \subseteq X$ then $X \rightarrow Y$
- **Augmentation rule** – If $a \rightarrow b$ holds and y is attribute set, then $ay \rightarrow by$ also holds. That is adding attributes in dependencies, does not change the basic dependencies.
- **Transitivity rule** – Same as transitive rule in algebra, if $a \rightarrow b$ holds and $b \rightarrow c$ holds, then $a \rightarrow c$ also holds. $a \rightarrow b$ is called as a functionally that determines b .

Consequences of Armstrong's axioms

- **Union:** If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow Y, Z$
- **Pseudo-transitivity:** If $X \rightarrow Y$ and $W, Y \rightarrow Z$ then $X, W \rightarrow Z$
- **Decomposition:** If $X \rightarrow Y$ and $Z \subseteq Y$ then $X \rightarrow Z$

Closure

- If F is a set of functional dependencies, then the closure of F , denoted as F^+ , is the set of all functional dependencies logically implied by F .
- It will help you to find all the candidate key in a relation.

(A minimal superkey is candidate key. A **candidate key** is a set of attributes (or attribute) which uniquely identify the tuples in relation or table .)

Example:

Question: $R(ABCD)$, $FD\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$

Solution:

$A^+ = B$ (As A can directly determine B)

$A^+ = BC$ (As B can determine C , A can also determine C , transitive)

$A^+ = BCD$ (As C can determine D , A can also determine D , transitive)

$A^+ = BCDA$ (A can determine itself, reflexive), this is the final solution

Find Candidate Key

Example:

Question: $R(ABCD)$, $FD\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$

Solution:

$A^+ = BCDA$ (A is a candidate key, as its closure holds all the attributes of R).

Now let's find the closure of B, C and D.

$B^+ = BCD$

$C^+ = CD$

$D^+ = D$

So A is the only candidate key.

Let's find the closure for $(AB)^+$

$(AB)^+ = ABCD$

AB can't be a candidate key as it should be minimal. AB can be super key.

Closure and Candidate Key

Example:

Question: $R(ABCD)$, $FD\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

Solution:

$A^+ = ABCD$

$B^+ = BCDA$

$C^+ = CDAB$

$D^+ = DABC$

Candidate key = $\{A, B, C, D\}$

Closure and Candidate Key

Example:

Question: $R(ABCDE)$, $FD\{A \rightarrow B, B \rightarrow C, E \rightarrow C, D \rightarrow A\}$

Solution:

$A^+ = AB$

$B^+ = B$

$C^+ = C$

$D^+ = ADB$

$E^+ = EC$

If you check all the dependents in the question, then you can understand that **E** can only come as a dependent (in right hand side) if we take different combinations of **E** to determine the candidate key.

$(AE)^+ = ABECD$

$(BE)^+ = BECDA$

$(CE)^+ = CE$

$(DE)^+ = DEABC$

Candidate key = $\{AE, BE, DE\}$

Equivalence of Closure

- Two sets of FDs, F and G , are said to be **equivalent** if $F^+ = G^+$
- For example:
 $\{(A, B \rightarrow C), (A \rightarrow B)\}$ and
 $\{(A \rightarrow C), (A \rightarrow B)\}$
are equivalent.

Trivial Functional Dependency

- **Trivial** – If a functional dependency (FD) $X \rightarrow Y$ holds, where Y is a subset of X , then it is called a trivial FD. Trivial FDs always hold.
- **Non-trivial** – If an FD $X \rightarrow Y$ holds, where Y is not a subset of X , then it is called a non-trivial FD.

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

Find the candidate keys?

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

- $A^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

- $A^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $B^+ = \{B\}$ **(NOT A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

- $A^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $B^+ = \{B\}$ **(NOT A CANDIDATE KEY)**
- $C^+ = \{AC\}$ **(NOT A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 1 : (Find the closure)

A^+ , B^+ , C^+ , D^+

- $A^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $B^+ = \{B\}$ **(NOT A CANDIDATE KEY)**
- $C^+ = \{AC\}$ **(NOT A CANDIDATE KEY)**
- $D^+ = \{D\}$ **(NOT A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 2 : (Find the closure)

$AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+$

- $AB^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 2 : (Find the closure)

$AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+$

- $AB^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AC^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- $AB^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AC^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AD^+ = \{ABCD\}$ **(A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 2 : (Find the closure)

$AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+$

- $AB^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AC^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AD^+ = \{ABCD\}$ **(A CANDIDATE KEY)**
- $BC^+ = \{BCA\}$ **(NOT A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 2 : (Find the closure)

$AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+$

- $AB^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AC^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AD^+ = \{ABCD\}$ **(A CANDIDATE KEY)**
- $BC^+ = \{BCA\}$ **(NOT A CANDIDATE KEY)**
- $BD^+ = \{BD\}$ **(NOT A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- $AB^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AC^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AD^+ = \{ABCD\}$ **(A CANDIDATE KEY)**
- $BC^+ = \{BCA\}$ **(NOT A CANDIDATE KEY)**
- $BD^+ = \{BD\}$ **(NOT A CANDIDATE KEY)**
- $CD^+ = \{CDAB\}$ **(A CANDIDATE KEY)**

$R=\{A,B,C,D\}$

GIVEN:

$A \rightarrow BC$

$C \rightarrow A$

CANDIDATE KEY: $\{AD, CD\}$

PRIME ATTRIBUTE= $\{A,C,D\}$

NON- PRIME ATTRIBUTE= $\{B\}$

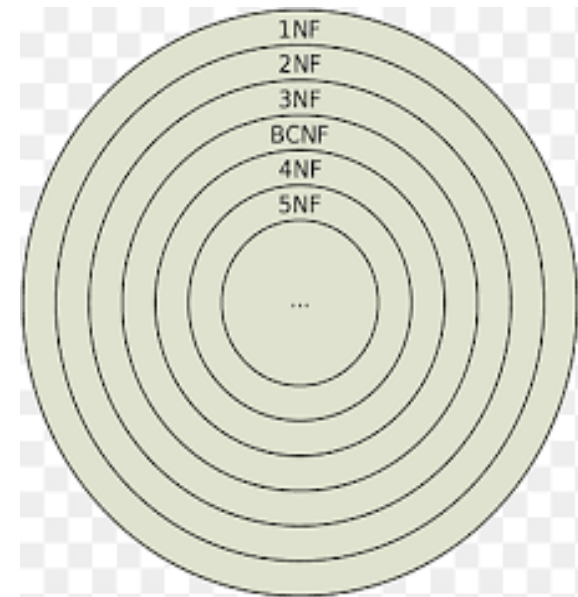
For Degree 2 : (Find the closure)

AB^+ , AC^+ , AD^+ , BC^+ , BD^+ , CD^+

- $AB^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AC^+ = \{ABC\}$ **(NOT A CANDIDATE KEY)**
- $AD^+ = \{ABCD\}$ **(A CANDIDATE KEY)**
- $BC^+ = \{BCA\}$ **(NOT A CANDIDATE KEY)**
- $BD^+ = \{BD\}$ **(NOT A CANDIDATE KEY)**
- $CD^+ = \{CDAB\}$ **(A CANDIDATE KEY)**

Normalization

- **Normalization** is a process for assigning attributes to entities. It reduces data redundancies and helps eliminate the data anomalies.
- Normalization works through a series of stages called normal forms:
 - First normal form (1NF)
 - Second normal form (2NF)
 - Third normal form (3NF)
 - Boyce–Codd normal form (or BCNF or 3.5NF)
 - Fourth normal form (4NF)
 - Fifth normal form
 -



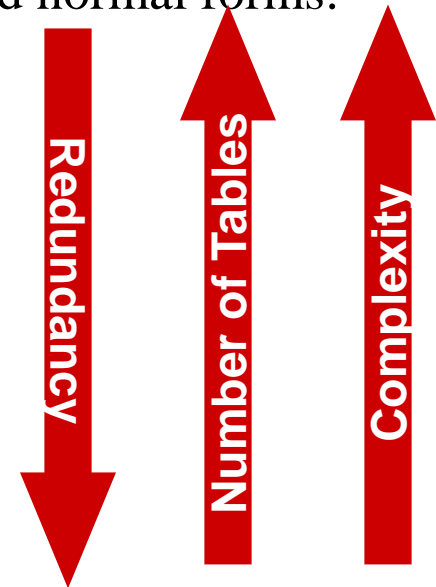
- The highest level of normalization is not always desirable.
- Which type of normalization is needed, basically depends on the given relations and specific applications

Normalization

- **Normalization** is a process for assigning attributes to entities. It reduces data redundancies and helps eliminate the data anomalies.

- Normalization works through a series of stages called normal forms:

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Boyce–Codd normal form (or BCNF or 3.5NF)
- Fourth normal form (4NF)
- Fifth normal form
-



- The highest level of normalization is not always desirable.
- Which type of normalization is needed, basically depends on the given relations and specific applications

Objective of Normalization

- Normalization presents a set of rules that tables and databases must follow to be well structured.
- It can solve different anomaly problems in write mode
- It reduces data redundancy

First Normal Form

- A table is in the first normal form iff
 - The domain of each attribute contains only ***atomic values***, and
 - The value of each attribute contains only a ***single value*** from that domain.

In layman's terms atomic value means every column of your table should only contain *single values*

Example

- For a library

Patron ID	Borrowed books
C45	B33, B44, B55
C12	B56

1-NF Solution

Patron ID	Borrowed book
C45	B33
C45	B44
C45	B55
C12	B56

Example

- For an airline

Flight	Weekdays
UA59	Mo We Fr
UA73	Mo Tu We Th Fr

1NF Solution

Flight	Weekday
UA59	Mo
UA59	We
UA59	Fr
UA73	Mo
UA73	Tu
...	...

Second Normal Form

- A table is in 2NF iff
 - It is in 1NF and
 - no non-prime attribute is dependent on any proper subset of any candidate key of the table (i.e. no partial dependency exists)
- A ***non-prime attribute*** of a table is an attribute that is not a part of any candidate key of the table
- A ***candidate key*** is a minimal superkey

Example

- Library allows patrons to request books that are currently out

BookNo	Patron	PhoneNo
B3	J. Fisher	555-1234
B2	J. Fisher	555-1234
B2	M. Amer	555-4321

Example

- Candidate key is {BookNo, Patron}
- We have
 - Patron \rightarrow PhoneNo (here phone no, a non-prime attribute is dependent on subset of candidate key, i.e. dependent on Patron, so partial dependency exists)
- Table is not 2NF
 - Potential for
 - Insertion anomalies
 - Update anomalies
 - Deletion anomalies

2NF Solution

- Put telephone number in separate Patron table

BookNo	Patron
B3	J. Fisher
B2	J. Fisher
B2	M. Amer

Patron	PhoneNo
J. Fisher	555-1234
M. Amer	555-4321

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)

GIVEN :

AB->C

C->D

B->E

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)

GIVEN :

AB->C

C->D

B->E

Solution:

Step 1: Find out the candidate key

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)

GIVEN :

AB->C

C->D

B->E

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+ = \{ABCDE\}$

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)

GIVEN :

AB->C

C->D

B->E

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+ = \{ABCDE\}$

Step 2: Find out the partial dependency if exist

AB->C (no partial dependency)

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)

GIVEN :

AB->C

C->D

B->E

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+ = \{ABCDE\}$

Step 2: Find out the partial dependency if exist

AB->C (no partial dependency)

C->D (no partial dependency)

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)

GIVEN :

AB->C

C->D

B->E

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+ = \{ABCDE\}$

Step 2: Find out the partial dependency if exist

AB->C (no partial dependency)

C->D (no partial dependency)

B-> E (partial dependency)

Partial dependency exist for the given relation hence it is in 1 NF but not in 2NF

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)

GIVEN :

AB->C

BC->D

Question: Find out whether the given relation is in 2nf or not?

R(ABCDE)

GIVEN :

AB->C

BC->D

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+ = \{ABCDE\}$

Step 2: Find out the partial dependency if exist

AB->C (no partial dependency)

BC->D (no partial dependency) (note: BC is not a proper subset of candidate key)

No Partial dependency exist for the given relation hence it is in 2 NF

Third Normal Form

- A table is in 3NF iff
 - it is in 2NF and
 - all its attributes are determined only by its candidate keys and not by any non-prime attributes
 - No transitive dependency

Example

- Table BorrowedBooks

BookNo	Patron	Address	Due
B1	J. Fisher	101 Main Street	3/2/15
B2	L. Perez	202 Market Street	2/28/15

- ❑ Candidate key is BookNo
- ❑ Patron → Address

3NF Solution

- Put address in separate Patron table

BookNo	Patron	Due
B1	J. Fisher	3/2/15
B2	L. Perez	2/28/15

Patron	Address
J. Fisher	101 Main Street
L. Perez	202 Market Street

Another example

- Tournament winners

Tournament	Year	Winner	DOB
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 Sept. 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975

- Candidate key is {Tournament, Year}
- Winner → DOB

3NF Solution

Tournament	Year	Winner
Indiana Invitational	1998	Al Fredrickson
Cleveland Open	1999	Bob Albertson
Des Moines Masters	1999	Al Fredrickson

Winner	DOB
Al Fredrickson	21 July 1975
Bob Albertson	28 Sept. 1968
Al Fredrickson	21 July 1975

Question: Find out whether the given relation is in 3nf or not?

R(ABCD)

GIVEN :

AB->C

BC->D

Solution:

Step 1: Find out the candidate key

AB is the only candidate key because $(AB)^+ = \{ABCDE\}$

Step 2: Find out the partial dependency if exist

AB->C (no partial dependency)

BC->D (no partial dependency) (note: BC is not a proper subset of candidate key)

No Partial dependency exist for the given relation hence it is in 2 NF but not in 3NF

Boyce-Codd Normal Form

- Stricter form of 3NF
- A table T is in BCNF iff
 - for every one of its non-trivial dependencies $X \rightarrow Y$, X is a candidate key for T
- Most tables that are in 3NF also are in BCNF

Example

Manager	Project	Branch
Alice	Alpha	Austin
Alice	Delta	Austin
Carol	Alpha	Houston
Dean	Delta	Houston

- We can assume
 - Manager \rightarrow Branch
 - {Project, Branch} \rightarrow Manager

Example

<u>Manager</u>	<u>Project</u>	Branch
Alice	Alpha	Austin
Bob	Delta	Houston
Carol	Alpha	Houston
Alice	Delta	Austin

- Not in BCNF because Manager \rightarrow Branch and Manager is not a candidate key
- Will decomposition work?

A decomposition (I)

<u>Manager</u>	Project
Alice	Alpha
Bob	Delta
Carol	Alpha
Alice	Delta

<u>Manager</u>	Branch
Alice	Austin
Bob	Houston
Carol	Houston

- Two-table solution does not preserve the dependency $\{\text{Project, Branch}\} \rightarrow \text{Manager}$

Q1 $\rightarrow R(ABCD) \{ A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A \}$

$C.K = \{ A, B, C, D \} \Rightarrow [4 C.K]$

1NF, 2NF, 3NF, BCNF

② $R(ABCDE) \{ AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow A \}$

$C.K = \{ AB, BE, BD, BC \} \Rightarrow [4 C.K]$

1NF, 2NF, 3NF, BCNF

Property of Decomposition

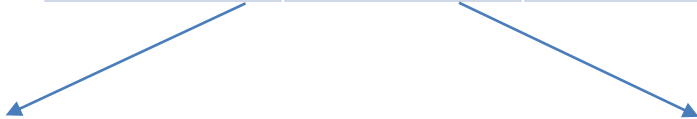
- LOSSLESS or LOSSY Decomposition
- DEPENDENCY PRESERVING

Decomposition

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c2

Decomposition

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c2



A
a1
a2


B	C
b1	c1
b2	c2

Decomposition

A
a1
a2

B	C
b1	c1
b2	c2

When we need to combine again we HAVE TO PERFORM **natural join** (CROSS PRODUCT) $R1 \times R2$



A	B	C
a1	b1	c1
a1	b2	c2
a2	b1	c1
a2	b2	c2


Lossy Decomposition

A
a1
a2

B	C
b1	c1
b2	c2

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c2

Original table




A	B	C
a1	b1	c1
a1	b2	c2
a2	b1	c1
a2	b2	c2

Wrong approach

Note: the right approach is there should be something in common column

Decomposition

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c2



A	B
a1	b1
a2	b1
a1	b2


A	C
a1	c1
a2	c1
a1	c2

Decomposition

A	B
a1	b1
a2	b1
a1	b2

A	C
a1	c1
a2	c1
a1	c2

Perform natural join



A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a1	b2	c1
a1	b2	c2

Lossy Decomposition

A	B
a1	b1
a2	b1
a1	b2

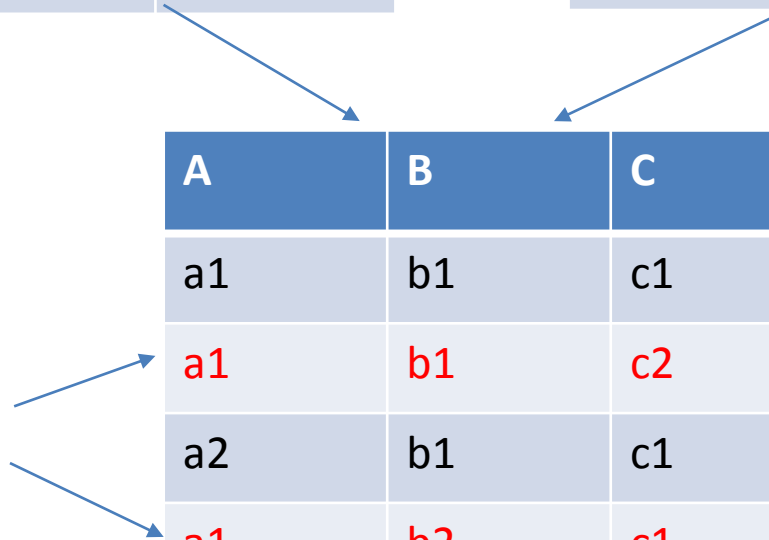
A	C
a1	c1
a2	c1
a1	c2

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c2

Original table


A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a1	b2	c1
a1	b2	c2

Extra tuple



Solution 3-Decomposition

A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c2



A	B
a1	b1
a2	b1
a1	b2

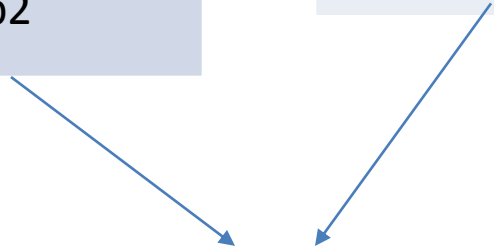
B	C
b1	c1
b2	c2

Solution 3-Decomposition

A	B
a1	b1
a2	b1
a1	b2

B	C
b1	c1
b2	c2

Perform natural join



A	B	C
a1	b1	c1
a2	b1	c1
a1	b2	c2

- No extra tuple added hence it is loss less decomposition
- If common attribute is acting as a candidate key in any one the table then it is known as lossless decomposition

Example 1:

- $R(ABC)$, FD: $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$
- $R1(AB)$, $R2(BC)$

Q. 1 LOSSLESS Decomposition?

Example 1:

- $R(ABC)$, FD: $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$
- $R1(AB)$, $R2(BC)$

Q. 1 LOSSLESS DEPENDENCY?

Ans. Yes

Functional dependency preserving

- The dependency preservation decomposition is another property of decomposed relational database schema D in which each functional dependency $X \rightarrow Y$ specified in F either appeared directly in one of the relation schemas R_i in the decomposed D or could be inferred from the dependencies that appear in some R_i .

Example: Functional dependency preserving

Let a relation $R(A,B,C,D)$ and set a FDs $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$ are given.
A relation R is decomposed into -

$R_1 = (A, B, C)$ with FDs $F_1 = \{A \rightarrow B, A \rightarrow C\}$, and

$R_2 = (C, D)$ with FDs $F_2 = \{C \rightarrow D\}$.

$F' = F_1 \cup F_2 = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$

so, $F' = F$.

And so, $F'^+ = F^+$.

Example: Functional dependency preserving

Let a relation $R(A,B,C,D)$ and set a FDs $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$ are given.
A relation R is decomposed into -

$R_1 = (A, B, C)$ with FDs $F_1 = \{A \rightarrow B, A \rightarrow C\}$, and

$R_2 = (C, D)$ with FDs $F_2 = \{C \rightarrow D\}$.

$F' = F_1 \cup F_2 = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$

so, $F' = F$.

And so, $F'^+ = F^+$.

Thus, the decomposition is dependency preserving decomposition.

Example: Functional dependency preserving

Let a relation $R(A,B,C,D)$ and set a FDs $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$ are given.
A relation R is decomposed into -

$R_1 = (A, B, C)$ with FDs $F_1 = \{A \rightarrow B, A \rightarrow C\}$, and

$R_2 = (C, D)$ with FDs $F_2 = \{C \rightarrow D\}$.

$F' = F_1 \cup F_2 = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$

so, $F' = F$.

And so, $F'^+ = F^+$.

Thus, the decomposition is dependency preserving decomposition.

Q) Let suppose, a relation R (P, Q, R, S) with a set of Functional Dependency $FD = (PQ \rightarrow R, R \rightarrow S, S \rightarrow P)$ is given. Into R1 (P, Q, R) and R2(R, S), relation R (P, Q, R, S) is decomposed. Find out whether the decomposition is dependency preserving or not.

Q) Let suppose, a relation R (P, Q, R, S) with a set of Functional Dependency FD = (PQ→R, R→S, S→P) is given. Into R1 (P, Q, R) and R2(R, S), relation R (P, Q, R, S) is decomposed. Find out whether the decomposition is dependency preserving or not.

- 1) To find the closure of FD1, we have to consider all combinations of (P, Q, R). i.e., we need to find **out** the closure of P, Q, R, PQ, QR, and RP.

closure (P) = {P} // Trivial

closure (Q) = {Q} // Trivial

closure (R) = {R, P, S} //but S can't be in closure as S is not
//present in R1 (P, Q, R).
= {R, P}

(R→ P // Removing R from right side as it is trivial attribute)

closure (PQ) = {P, Q, R, S}
= {P, Q, R}

(PQ → R // Removing PQ from right side as these are trivial attributes)

closure (QR) = {Q, R, S, P}
= {P, Q, R}

(QR → P // Removing QR from right side as these are trivial attributes)

Closure (PR) = {P, R, S}

(PR → S // Removing PR from right side as these are trivial attributes)

FD1 {R → P, PQ → R, QR → P}.

Q) Let suppose, a relation R (P, Q, R, S) with a set of Functional Dependency FD = (PQ→R, R→S, S→P) is given. Into R1 (P, Q, R) and R2(R, S), relation R (P, Q, R, S) is decomposed. Find out whether the decomposition is dependency preserving or not.

- 1) To find the closure of FD1, we have to consider all combinations of (P, Q, R). i.e., we need to find **out** the closure of P, Q, R, PQ, QR, and RP.

closure (P) = {P} // Trivial

closure (Q) = {Q} // Trivial

closure (R) = {R, P, S} //but S can't be in closure as S is not
//present in R1 (P, Q, R).
= {R, P}

(R→ P // Removing R from right side as it is trivial attribute)

closure (PQ) = {P, Q, R, S}
= {P, Q, R}

(PQ → R // Removing PQ from right side as these are trivial attributes)

closure (QR) = {Q, R, S, P}
= {P, Q, R}

(QR → P // Removing QR from right side as these are trivial attributes)

Closure (PR) = {P, R, S}

(PR → S // Removing PR from right side as these are trivial attributes)

FD1 {R → P, PQ → R, QR → P}.

Q) Let suppose, a relation R (P, Q, R, S) with a set of Functional Dependency $FD = (PQ \rightarrow R, R \rightarrow S, S \rightarrow P)$ is given. Into R1 (P, Q, R) and R2(R, S), relation R (P, Q, R, S) is decomposed. Find out whether the decomposition is dependency preserving or not.

- **2)** Similarly $FD2 \{R \twoheadrightarrow S\}$

-

- In the original Relation Dependency $FD = \{PQ \rightarrow R, R \rightarrow S, S \rightarrow P\}$.

- $PQ \twoheadrightarrow R$ is present in FD1.
- $R \twoheadrightarrow S$ is present in FD2.
- $S \twoheadrightarrow P$ is not preserved.

Q) Let suppose, a relation R (P, Q, R, S) with a set of Functional Dependency $FD = (PQ \rightarrow R, R \rightarrow S, S \rightarrow P)$ is given. Into R1 (P, Q, R) and R2(R, S), relation R (P, Q, R, S) is decomposed. Find out whether the decomposition is dependency preserving or not.

- **2) Similarly $FD2 \{R \twoheadrightarrow S\}$**
- In the original Relation Dependency $FD = \{PQ \rightarrow R, R \rightarrow S, S \rightarrow P\}$.
- $PQ \twoheadrightarrow R$ is present in $FD1$.
- $R \twoheadrightarrow S$ is present in $FD2$.
- $S \twoheadrightarrow P$ is not preserved.
- From the given result, in $FD1$, PQ holds R ($PQ \twoheadrightarrow R$) and in $FD2$, R holds S ($R \twoheadrightarrow S$). But, there is no follow up in Functional Dependency S holds P ($S \twoheadrightarrow P$).
- $FD1 \cup FD2$ is a subset of FD .
- So as a consequence, given decomposition is not dependency preserving.

Example 1:

- $R(ABC)$, FD: $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$
- $R1(AB), R2(BC)$

R1(AB)	R2(BC)
A \rightarrow B B \rightarrow A	B \rightarrow C C \rightarrow A

Q. 1 LOSSLESS DEPENDENCY?

Ans. Yes

Q. 2 Dependency preserving?

Example 1:

- $R(ABC)$, FD: $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$
- $R1(AB)$, $R2(BC)$

Q. 1 LOSSLESS DEPENDENCY?

Ans. Yes

Q. 2 Dependency preserving?

Ans. Yes

Example 2:

- $R(ABCD)$, FD: $\{AB \rightarrow CD, D \rightarrow A\}$
- $R1(AD)$, $R2(BCD)$

Q. 1 LOSSLESS DEPENDENCY?

Q. 2 Dependency preserving?

Example 2:

- $R(ABCD)$, FD: $\{AB \rightarrow CD, D \rightarrow A\}$
- $R1(AD)$, $R2(BCD)$

Q. 1 LOSSLESS DEPENDENCY?

ANS. YES

Q. 2 Dependency preserving?

Example 2:

- $R(ABCD)$, FD: $\{AB \rightarrow CD, D \rightarrow A\}$
- $R1(AD)$, $R2(BCD)$

Q. 1 LOSSLESS DEPENDENCY?

ANS. YES

Q. 2 Dependency preserving?

R1(AD)	R2(BCD)
$A \rightarrow D$ (NOT HOLD) $D \rightarrow A$	$B \rightarrow B$ $C \rightarrow C$ $D \rightarrow D$ $BC \rightarrow BC$ $CD \rightarrow CD$ $BD \rightarrow BDC$

Example 2:

ONLY NON TRIVIAL DEPENDENCY

- $R(ABCD)$, FD: $\{AB \rightarrow CD, D \rightarrow A\}$
- $R1(AD)$, $R2(BCD)$

R1(AD)	R2(BCD)
D \rightarrow A	BD \rightarrow C

Q. 1 LOSSLESS DEPENDENCY?

ANS. YES

Q. 2 Dependency preserving?

ANS NO

Degree of Redundancy

1NF>2NF>3NF>BCNF



0 % redundancy for FD
Not 0 % redundancy for MVD

DB Design Goals	1NF	2NF	3NF	BCNF
1. 0% REDUNDANCY	NO	NO	NO	YES(FD) NO(MVD)
2. LOSSLESS JOIN DECOMPOSITION	YES (ALWAYS)	YES (ALWAYS)	YES (ALWAYS)	YES (ALWAYS)
3. DEPENDENCY PRESERVING PROPERTY	YES (ALWAYS)	YES (ALWAYS)	YES (ALWAYS)	NOT (ALWAYS)

How good is BCNF?

<i>course</i>	<i>teacher</i>	<i>book</i>
database	Avi	DB Concepts
database	Avi	Ullman
database	Hank	DB Concepts
database	Hank	Ullman
database	Sudarshan	DB Concepts
database	Sudarshan	Ullman
operating systems	Avi	OS Concepts
operating systems	Avi	Stallings
operating systems	Pete	OS Concepts
operating systems	Pete	Stallings

classes

- There are no non-trivial functional dependencies and therefore the relation is in BCNF
- Insertion anomalies – i.e., if Marilyn is a new teacher that can teach database, two tuples need to be inserted
(database, Marilyn, DB Concepts)
(database, Marilyn, Ullman)

How good is BCNF? (Cont.)

- Therefore, it is better to decompose *classes* into:

<i>course</i>	<i>teacher</i>
database	Avi
database	Hank
database	Sudarshan
operating systems	Avi
operating systems	Jim

teaches

<i>course</i>	<i>book</i>
database	DB Concepts
database	Ullman
operating systems	OS Concepts
operating systems	Shaw

text

This suggests the need for higher normal forms, such as Fourth Normal Form (4NF), which we shall see later.

Multivalued dependencies

- Assume the column headings in a table are divided into three disjoint groupings X , Y , and Z
- For a particular row, we can refer to the data beneath each group of headings as x , y , and z respectively

Multivalued dependencies

- A ***multivalued dependency*** $X \twoheadrightarrow Y$ occurs if
 - For any x_c actually occurring in the table and the list of all the $x_c y z$ combinations that occur in the table, we will find that x_c is associated with the same y entries regardless of z .
- A ***trivial multivalued dependency*** $X \twoheadrightarrow Y$ is one where either
 - Y is a subset of X , or
 - Z is empty ($X \cup Y$ has all column headings)

Fourth Normal Form

- A table is in 4NF iff
 - For every one of its non-trivial multivalued dependencies $X \twoheadrightarrow Y$, X is either:
 - A candidate key or
 - A superset of a candidate key

Example

Restaurant	Pizza	DeliveryArea
Pizza Milano	Thin crust	SW Houston
Pizza Milano	Thick crust	SW Houston
Pizza Firenze	Thin crust	NW Houston
Pizza Firenze	Thick crust	NW Houston
Pizza Milano	Thin crust	NW Houston
Pizza Milano	Thick crust	NW Houston

Discussion

- The table has no non-key attributes
 - Key is { Restaurant, Pizza, DeliveryArea }
- Two non-trivial multivalued dependencies
 - Restaurant \Rightarrow Pizza
 - Restaurant \Rightarrow DeliveryArea

since each restaurant delivers the same pizzas to all its delivery areas

4NF Solution

Restaurant	DeliveryArea
Pizza Milano	SW Houston
Pizza Firenze	NW Houston
Pizza Milano	NW Houston

- Two separate tables

Restaurant	Pizza
Pizza Milano	Thin crust
Pizza Milano	Thick crust
Pizza Firenze	Thin crust
Pizza Firenze	Thick crust

Join dependency

- A table T is subject to a ***join dependency*** if it can always be recreated by ***joining*** multiple tables each having a subset of the attributes of T
- The join dependency is said to be ***trivial*** if one of the tables in the join has all the attributes of the table T
- *Notation:* $\{ A, B, \dots \}$ on T

Fifth normal form

- A table T is said to be 5NF iff
 - Every non-trivial join dependency in it is implied by its candidate keys
- A join dependency $*\{A, B, \dots Z\}$ on T is implied by the candidate key(s) of T if and only if each of A, B, \dots, Z is a superkey for T

An example

<i>Store</i>	<i>Brand</i>	<i>Product</i>
Circuit City	Apple	Tablets
Circuit City	Apple	Phones
Circuit City	Toshiba	Laptops
CompUSA	Apple	Laptops

- Note that Circuit City sells Apple tablets and phones but only Toshiba laptops

A bad decomposition

<i>Store</i>	<i>Product</i>
Circuit City	Tablets
Circuit City	Phones
Circuit City	Laptops
CompUSA	Laptops

<i>Brand</i>	<i>Product</i>
Apple	Tablets
Apple	Phones
Apple	Laptops
Toshiba	Laptops

- Let see what happens when we do a natural join

The result of the join

<i>Store</i>	<i>Brand</i>	<i>Product</i>
Circuit City	Apple	Tablets
Circuit City	Apple	Phones
Circuit City	Apple	Laptops
Circuit City	Toshiba	Laptops
CompUSA	Apple	Laptops
CompUSA	Toshiba	Laptops

- Introduces two spurious tuples

A different table

<i>Store</i>	<i>Brand</i>	<i>Product</i>
Circuit City	Apple	Tablets
Circuit City	Apple	Phones
Circuit City	Apple	Laptops
Circuit City	Toshiba	Laptops
CompUSA	Apple	Laptops

- Assume now that any store carrying a given brand and selling a product that is made by that brand will ***always*** carry that product

The same decomposition

<i>Store</i>	<i>Product</i>
Circuit City	Tablets
Circuit City	Phones
Circuit City	Laptops
CompUSA	Laptops

<i>Brand</i>	<i>Product</i>
Apple	Tablets
Apple	Phones
Apple	Laptops
Toshiba	Laptops

- Let see what happens when we do a natural join

The result of the join

<i>Store</i>	<i>Brand</i>	<i>Product</i>
Circuit City	Apple	Tablets
Circuit City	Apple	Phones
Circuit City	Apple	Laptops
Circuit City	Toshiba	Laptops
CompUSA	Apple	Laptops
CompUSA	Toshiba	Laptops

- Still one spurious tuple

The right decomposition

<i>Store</i>	<i>Product</i>
Circuit City	Tablets
Circuit City	Phones
Circuit City	Laptops
CompUSA	Laptops

<i>Brand</i>	<i>Product</i>
Apple	Tablets
Apple	Phones
Apple	Laptops
Toshiba	Laptops

<i>Store</i>	<i>Brand</i>
Circuit City	Apple
Circuit City	Toshiba
CompUSA	Apple

Lossless Decomposition: General Concept

- If $R(A, B, C)$ satisfies $A \rightarrow B$
 - We can project it on A, B and A, C
without losing information
 - Lossless decomposition
- $R = \pi_{AB}(R) \bowtie \pi_{AC}(R)$
 - $\pi_{AB}(R)$ is the projection of R on AB
 - \bowtie is the natural join operator

Example

R

<i>Course</i>	<i>Instructor</i>	<i>Text</i>
4330	Paris	none
4330	Cheng	none
3330	Hillford	Patterson & Hennessy

- Observe that Course \rightarrow Text

A lossless decomposition

$\pi_{\text{Course, Text}}(R)$

<i>Course</i>	<i>Text</i>
4330	none
3330	Patterson & Hennessy

$\pi_{\text{Course, Instructor}}(R)$

<i>Course</i>	<i>Instructor</i>
4330	Paris
4330	Cheng
3330	Hillford

A different case

R

<u>Course</u>	<u>Instructor</u>	<i>Text</i>
4330	Paris	Silberschatz and Peterson
4330	Cheng	none
3330	Hillford	Patterson & Hennessy

- Now Course \nrightarrow Text
- R cannot be decomposed

A lossy decomposition

$\pi_{\text{Course, Text}}(R)$

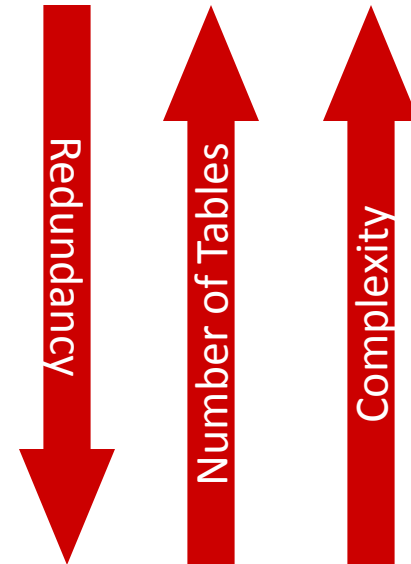
<i>Course</i>	<i>Text</i>
4330	none
4330	Silberschatz & Peterson
3330	Patterson & Hennessy

$\pi_{\text{Course, Instructor}}(R)$

<i>Course</i>	<i>Instructor</i>
4330	Paris
4330	Cheng
3330	Hillford

Levels of Normalization

- Levels of normalization based on the amount of redundancy in the database.
- Various levels of normalization are:
 - First Normal Form (1NF)
 - Second Normal Form (2NF)
 - Third Normal Form (3NF)
 - Boyce-Codd Normal Form (BCNF)
 - Fourth Normal Form (4NF)
 - Fifth Normal Form (5NF)
 - Domain Key Normal Form (DKNF)



Most databases should be 3NF or BCNF in order to avoid the database anomalies.

An Example

Normalisation Example

- We have a table representing orders in an online store
- Each row represents an item on a particular order
- Primary key is {Order, Product}
- Columns
 - Order
 - Product
 - Quantity
 - UnitPrice
 - Customer
 - Address

Functional Dependencies

- Each order is for a single customer:
 - Order \rightarrow Customer
- Each customer has a single address
 - Customer \rightarrow Address
- Each product has a single price
 - Product \rightarrow UnitPrice
- As Order \rightarrow Customer and Customer \rightarrow Address
 - Order \rightarrow Address

2NF Solution (I)

- ***First decomposition***

- First table

<u>Order</u>	<u>Product</u>	Quantity	UnitPrice
--------------	----------------	----------	-----------

- Second table

<u>Order</u>	Customer	Address
--------------	----------	---------

2NF Solution (II)

- ***Second decomposition***

- First table

<u>Order</u>	<u>Product</u>	Quantity
--------------	----------------	----------

- Second table

<u>Order</u>	Customer	Address
--------------	----------	---------

- Third table

<u>Product</u>	UnitPrice
----------------	-----------

3NF

- In second table

<u>Order</u>	Customer	Address
--------------	----------	---------

– Customer → Address

- Split second table into

<u>Order</u>	Customer
--------------	----------

<u>Customer</u>	Address
-----------------	---------

Normalisation to 2NF

- Second normal form means no partial dependencies on candidate keys
 - $\{\text{Order}\} \rightarrow \{\text{Customer}, \text{Address}\}$
 - $\{\text{Product}\} \rightarrow \{\text{UnitPrice}\}$
- To remove the first FD we project over
 - $\{\text{Order}, \text{Customer}, \text{Address}\}$
(R1)
 - and
 - $\{\text{Order}, \text{Product}, \text{Quantity}, \text{UnitPrice}\}$
(R2)

Normalisation to 2NF

- R1 is now in 2NF, but there is still a partial FD in R2
 $\{\text{Product}\} \rightarrow \{\text{UnitPrice}\}$
- To remove this we project over $\{\text{Product}, \text{UnitPrice}\}$ (R3)
and
 $\{\text{Order}, \text{Product}, \text{Quantity}\}$ (R4)

Normalisation to 3NF

- R has now been split into 3 relations - R1, R3, and R4
 - R3 and R4 are in 3NF
 - R1 has a transitive FD on its key
- To remove $\{\text{Order}\} \rightarrow \{\text{Customer}\} \rightarrow \{\text{Address}\}$
 - we project R1 over
 - $\{\text{Order}, \text{Customer}\}$
 - $\{\text{Customer}, \text{Address}\}$

Normalisation

- 1NF:
 - {Order, Product, Customer, Address, Quantity, UnitPrice}
- 2NF:
 - {Order, Customer, Address}, {Product, UnitPrice}, and {Order, Product, Quantity}
- 3NF:
 - {Product, UnitPrice}, {Order, Product, Quantity}, {Order, Customer}, and {Customer, Address}

Another Example

First Normal Form (1NF)

Example (Not 1NF)

ISBN	Title	AuName	AuPhone	PubName	PubPhone	Price
0-321-32132-1	Balloon	Sleepy, Snoopy, Grumpy	321-321-1111, 232-234-1234, 665-235-6532	Small House	714-000-0000	\$34.00
0-55-123456-9	Main Street	Jones, Smith	123-333-3333, 654-223-3455	Small House	714-000-0000	\$22.95
0-123-45678-0	Ulysses	Joyce	666-666-6666	Alpha Press	999-999-9999	\$34.00
1-22-233700-0	Visual Basic	Roman	444-444-4444	Big House	123-456-7890	\$25.00

AuName and AuPhone columns are not scalar

1NF - Decomposition

1. Place all items that appear in the repeating group in a new table
2. Designate a primary key for each new table produced.
3. Duplicate in the new table the primary key of the table from which the repeating group was extracted or vice versa.

Example (1NF)

ISBN	Title	PubName	PubPhone	Price
0-321-32132-1	Balloon	Small House	714-000-0000	\$34.00
0-55-123456-9	Main Street	Small House	714-000-0000	\$22.95
0-123-45678-0	Ulysses	Alpha Press	999-999-9999	\$34.00
1-22-233700-0	Visual Basic	Big House	123-456-7890	\$25.00

ISBN	AuName	AuPhone
0-321-32132-1	Sleepy	321-321-1111
0-321-32132-1	Snoopy	232-234-1234
0-321-32132-1	Grumpy	665-235-6532
0-55-123456-9	Jones	123-333-3333
0-55-123456-9	Smith	654-223-3455
0-123-45678-0	Joyce	666-666-6666
1-22-233700-0	Roman	444-444-4444

Second Normal Form (2NF)

Example 1 (Not 2NF)

Scheme \rightarrow {City, Street, HouseNumber, HouseColor, CityPopulation}

1. key \rightarrow {City, Street, HouseNumber}
2. {City, Street, HouseNumber} \rightarrow {HouseColor}
3. {City} \rightarrow {CityPopulation}
4. CityPopulation does not belong to any key.
5. CityPopulation is functionally dependent on the City which is a proper subset of the key

Example 2 (Not 2NF)

Scheme \rightarrow {studio, movie, budget, studio_city}

1. Key \rightarrow {studio, movie}
2. {studio, movie} \rightarrow {budget}
3. {studio} \rightarrow {studio_city}
4. studio_city is not a part of a key
5. studio_city functionally depends on studio which is a proper subset of the key

2NF - Decomposition

Example 1 (Convert to 2NF)

Old Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}

New Scheme → {City, Street, HouseNumber, HouseColor}

New Scheme → {City, CityPopulation}

Example 2 (Convert to 2NF)

Old Scheme → {Studio, Movie, Budget, StudioCity}

New Scheme → {Movie, Studio, Budget}

New Scheme → {Studio, City}

Third Normal Form (3NF)

Example 1 (Not in 3NF)

Scheme → {Studio, StudioCity, CityTemp}

1. Primary Key → {Studio}
2. {Studio} → {StudioCity}
3. {StudioCity} → {CityTemp}
4. {Studio} → {CityTemp}
5. Both StudioCity and CityTemp depend on the entire key hence 2NF
6. CityTemp transitively depends on Studio hence violates 3NF

Example 2 (Not in 3NF)

Scheme → {BuildingID, Contractor, Fee}

1. Primary Key → {BuildingID}
2. {BuildingID} → {Contractor}
3. {Contractor} → {Fee}
4. {BuildingID} → {Fee}
5. Fee transitively depends on the BuildingID
6. Both Contractor and Fee depend on the entire key hence 2NF

BuildingID	Contractor	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

3NF - Decomposition

Example 1 (Convert to 3NF)

Old Scheme → {Studio, StudioCity, CityTemp}

New Scheme → {Studio, StudioCity}

New Scheme → {StudioCity, CityTemp}

Example 2 (Convert to 3NF)

Old Scheme → {BuildingID, Contractor, Fee}

New Scheme → {BuildingID, Contractor}

New Scheme → {Contractor, Fee}

BuildingID	Contractor
100	Randolph
150	Ingersoll
200	Randolph
250	Pitkin
300	Randolph

Contractor	Fee
Randolph	1200
Ingersoll	1100
Pitkin	1100

Boyce Codd Normal Form (BCNF)

Example 2 - Movie (Not in BCNF)

Scheme \rightarrow {MovieTitle, MovieID, PersonName, Role, Payment }

1. **Key1 \rightarrow {MovieTitle, PersonName}**
2. **Key2 \rightarrow {MovieID, PersonName}**
3. **Both role and payment functionally depend on both candidate keys thus 3NF**
4. **{MovieID} \rightarrow {MovieTitle}**
5. **Dependency between MovieID & MovieTitle Violates BCNF**

Example 3 - Consulting (Not in BCNF)

Scheme \rightarrow {Client, Problem, Consultant}

1. **Key1 \rightarrow {Client, Problem}**
2. **Key2 \rightarrow {Client, Consultant}**
3. **No non-key attribute hence 3NF**
4. **{Client, Problem} \rightarrow {Consultant}**
5. **{Client, Consultant} \rightarrow {Problem}**
6. **Dependency between attributes belonging to keys violates BCNF**

BCNF - Decomposition

Example 2 (Convert to BCNF)

Old Scheme \rightarrow {MovieTitle, MovieID, PersonName, Role, Payment }

New Scheme \rightarrow {MovieID, PersonName, Role, Payment}

New Scheme \rightarrow {MovieTitle, PersonName}

- Loss of relation {MovieID} \rightarrow {MovieTitle}

New Scheme \rightarrow {MovieID, PersonName, Role, Payment}

New Scheme \rightarrow {MovieID, MovieTitle}

- We got the {MovieID} \rightarrow {MovieTitle} relationship back

Example 3 (Convert to BCNF)

Old Scheme \rightarrow {Client, Problem, Consultant}

New Scheme \rightarrow {Client, Consultant}

New Scheme \rightarrow {Client, Problem}

Fourth Normal Form (4NF)

- Fourth normal form eliminates independent many-to-one relationships between columns.
- To be in Fourth Normal Form,
 - a relation must first be in Boyce-Codd Normal Form.
 - a given relation may not contain more than one multi-valued attribute.

Example (Not in 4NF)

Scheme → {MovieName, ScreeningCity, Genre}

Primary Key: {MovieName, ScreeningCity, Genre}

1. All columns are a part of the only candidate key, hence BCNF
2. Many Movies can have the same Genre
3. Many Cities can have the same movie
4. Violates 4NF

Movie	ScreeningCity	Genre
Hard Code	Los Angeles	Comedy
Hard Code	New York	Comedy
Bill Durham	Santa Cruz	Drama
Bill Durham	Durham	Drama
The Code Warrior	New York	Horror

Fourth Normal Form (4NF)

Example 2 (Not in 4NF)

Scheme → {Manager, Child, Employee}

1. Primary Key → {Manager, Child, Employee}
2. Each manager can have more than one child
3. Each manager can supervise more than one employee
4. 4NF Violated

Manager	Child	Employee
Jim	Beth	Alice
Mary	Bob	Jane
Mary	NULL	Adam

Example 3 (Not in 4NF)

Scheme → {Employee, Skill, ForeignLanguage}

1. Primary Key → {Employee, Skill, Language }
2. Each employee can speak multiple languages
3. Each employee can have multiple skills
4. Thus violates 4NF

Employee	Skill	Language
1234	Cooking	French
1234	Cooking	German
1453	Carpentry	Spanish
1453	Cooking	Spanish
2345	Cooking	Spanish

4NF - Decomposition

1. Move the two multi-valued relations to separate tables
2. Identify a primary key for each of the new entity.

Example 1 (Convert to 3NF)

Old Scheme → {MovieName, ScreeningCity, Genre}

New Scheme → {MovieName, ScreeningCity}

New Scheme → {MovieName, Genre}

Movie	Genre
Hard Code	Comedy
Bill Durham	Drama
The Code Warrior	Horror

Movie	ScreeningCity
Hard Code	Los Angeles
Hard Code	New York
Bill Durham	Santa Cruz
Bill Durham	Durham
The Code Warrior	New York

4NF - Decomposition

Example 2 (Convert to 4NF)

Old Scheme → {Manager, Child, Employee}

New Scheme → {Manager, Child}

New Scheme → {Manager, Employee}

Manager	Child
Jim	Beth
Mary	Bob

Manager	Employee
Jim	Alice
Mary	Jane
Mary	Adam

Example 3 (Convert to 4NF)

Old Scheme → {Employee, Skill, ForeignLanguage}

New Scheme → {Employee, Skill}

New Scheme → {Employee, ForeignLanguage}

Employee	Skill
1234	Cooking
1453	Carpentry
1453	Cooking
2345	Cooking

Employee	Language
1234	French
1234	German
1453	Spanish
2345	Spanish

Domain Key Normal Form (DKNF)

- **The relation is in DKNF when there can be no insertion or deletion anomalies in the database.**

MCQ

A functional dependency is a relationship between or among

- A. Entities
- B. Rows
- C. Attributes
- D. Tables

MCQ

A functional dependency is a relationship between or among

- A. Entities
- B. Rows
- C. Attributes
- D. Tables

MCQ

If one attribute is determinant of second, which in turn is determinant of third, then the relation cannot be:

- A. Well-structured
- B. 1NF
- C. 2NF
- D. 3NF

MCQ

If one attribute is determinant of second, which in turn is determinant of third, then the relation cannot be:

- A. Well-structured
- B. 1NF
- C. 2NF
- D. 3NF

MCQ

A relation is in this form if it is in BCNF and has no multivalued dependencies

- A. 2NF
- B. 3NF
- C. 4NF
- D. DKNF

MCQ

A relation is in this form if it is in BCNF and has no multivalued dependencies

- A. 2NF
- B. 3NF
- C. 4NF
- D. DKNF

MCQ

In the _____ normal form, a composite attribute is converted to individual attributes.

- a) First
- b) Second
- c) Third
- d) Fourth

MCQ

In the _____ normal form, a composite attribute is converted to individual attributes.

- a) First
- b) Second
- c) Third
- d) Fourth

MCQ

Which is a bottom-up approach to database design that design by examining the relationship between attributes:

- a) Functional dependency
- b) Database modeling
- c) Normalization
- d) Decomposition

MCQ

Which is a bottom-up approach to database design that design by examining the relationship between attributes:

- a) Functional dependency
- b) Database modeling
- c) **Normalization**
- d) Decomposition