Module 3

KEYS, FUNCTIONAL DEPENDENCY AND DEPENDENCY DIAGRAM

Dr. L.M. Jenila Livingston

VIT Chennai

Functional dependency

- The **functional dependency** is a relationship that exists between two attributes.
- It typically exists between the **primary key and non-key attribute** within a table. $X \rightarrow Y$.

How to locate the given value ‘X’

A	B	C
1	a	X
2	b	Y
3	b	X
4	c	y

Select A where C=‘X’;
Column name is not sufficient to identify the value ‘X’ as the row/tuple name is not given

Functional Dependency

A	B	C
1	a	X
2	b	Y
3	b	X
4	c	y

A → (B,C)

Select A where B='b' and C='X';

(B,C) → A

SuperKey

- **SuperKey**: A **key** that can be uniquely used to identify a database record, that **may contain extra attributes** that are not necessary to uniquely identify records.

In this table

{A} → {B,C}

{B,C} → {A}

{A,B} → {C}

{A,C} → {B}

Is used to identify the records.

A	B	C
1	a	X
2	b	Y
3	b	X
4	c	y

Candidate Key

- A **candidate key** is a set of attributes (or attribute) which uniquely identify the tuples in relation or table.
- **Minimal super key** form a candidate key (Subset of key is not a Super key)
- **R{A,B,C,D}**

  A→ B, C, D (Super key since all attributes involved)
  A,B→C, D (Super key)
  A,B,C→D (Super key)
  B,D → A, C (Super key)
  C→ A, D (Not a Super key)

	Super Key	Candidate Key
A→ B, C, D	✓	✓
A,B→C, D	✓	x
A,B,C→D	✓	x
B,D -> A, C	✓	✓
C→ A, D	x	x

- Student{ID, First_name, Last_name, DOB}
- Here we can see the two candidate keys {**ID**} and **{First_name, Last_name, DOB}**. So here, there are present more than one candidate keys, which can uniquely identify a tuple in a relation.

Primary Key

- Primary Key is a set of attributes (or attribute) which uniquely identify the tuples in relation or table.
- The primary key is a minimal super key (candidate key), so **there is only one primary key in any relation**.
- Primary key will not accept duplicate values
- **Student{ID, First_name, Last_name, DOB}**
- ID is a primary key

Primary Key

	Super Key	Candidate Key	Primary Key
A→ B, C, D	✓	✓	✓
A,B→C, D	✓	x	x
A,B,C→D	✓	x	x
B,D →A,B	✓	✓	x
C→ A, D	x	x	x

Unique Key

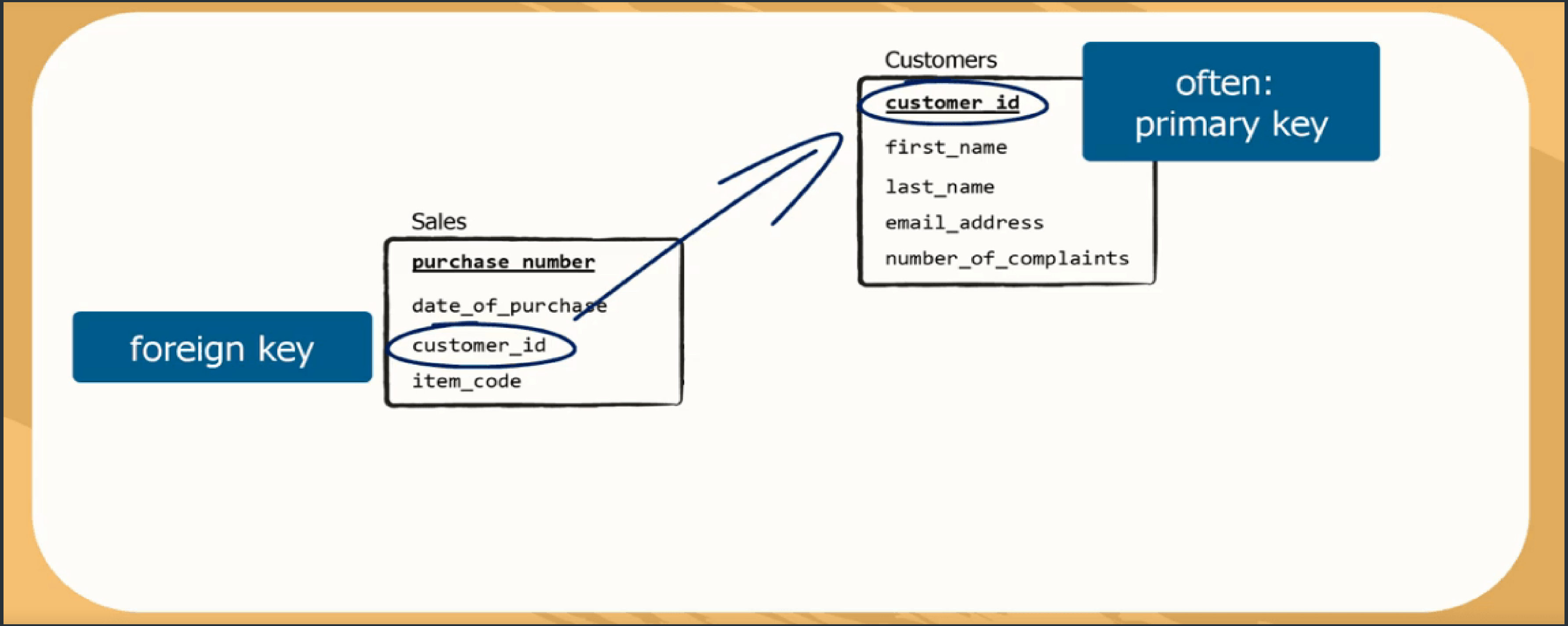
- Same concept as Primary Key
- But it will accept null values

You can say that it is little like primary key but it can accept only one null value and it cannot have duplicate values.

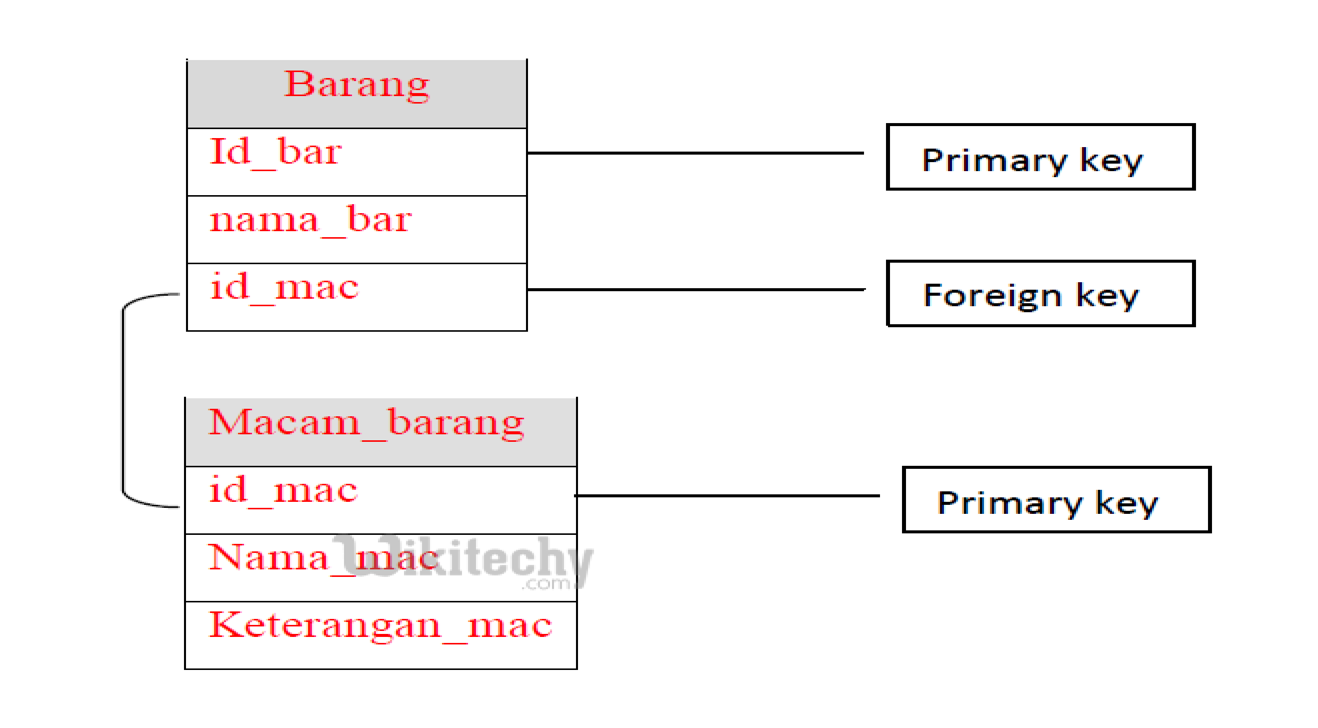
Foreign Key

- Otherwise called as reference key
- A **foreign key** is a column or group of columns in a relational database table that provides a link between data in two tables. It acts as a cross-reference between tables because it references the primary **key** of another table, thereby establishing a link between them.

Foreign Key


Customers
customer_id
first_name
last_name
email_address
number_of_complaints
often:
primary key
Sales
purchase_number
date_of_purchase
customer_id
item_code
foreign key

Foreign Key


Barang
Id_bar
nama_bar
id_mac
Primary key
Foreign key
Macam_barang
id_mac
Nama_mac
Keterangan_mac
Primary key
Wikitechy.com

Functional Dependency (FD)

- The attributes of a table is said to be dependent on each other when an attribute/ attributes of a table uniquely identifies another attribute of the same table.

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b1	C2	d2	e1
a3	b2	C1	d1	e1
a4	b2	C2	d2	e1
a5	b3	C3	d1	e1

Table R

$A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $A \rightarrow E$

Summarized as

$A \rightarrow BCDE$

From our understanding of primary keys, A is a primary key.

Functional Dependency

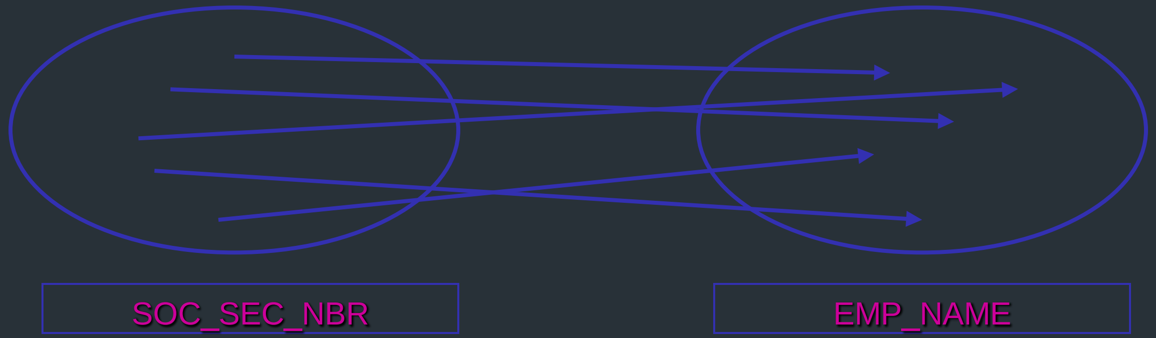
- Relationship between columns X and Y such that, given the value of X, one can *determine* the value of Y. Written as X $\longrightarrow$ Y

i.e., for a given value of X we can obtain (or look up) a specific value of X

- X is called the *determinant* of Y
- Y is said to be *functionally dependent* on X

Functional Dependency

- Example
  - SOC_SEC_NBR → EMP_NAME



-One and only one EMP_NAME for a specific SOC_SEC_NBR

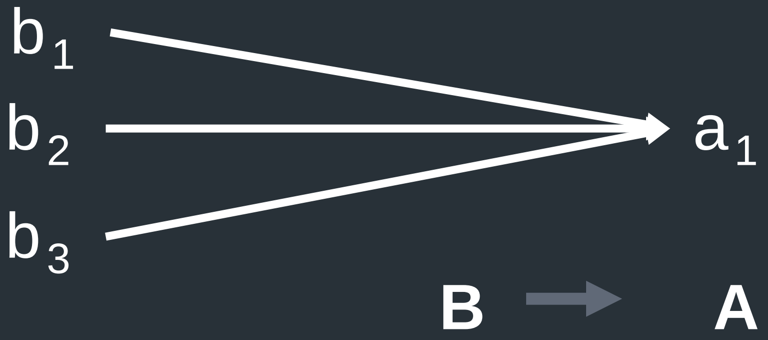
- SOC_SEC_NBR is the *determinant* of EMP_NAME

- EMP_NAME is functionally *dependent* on SOC_SEC_NBR

Functional Dependence

An attribute A is <u>functionally dependent</u> on attribute(s) B if:

given a value b for B there is ***one and only one corresponding value*** *a for A (at a time).*



STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

- STUD_NO → STUD_NAME, **FD hold** because for each STUD_NAME, there is a unique value of STUD_NO.
- STUD_NO → STUD_PHONE, **FD hold**
- STUD_NAME → STUD_NO, **FD does not hold,** because STUD-NAME ‘Ram’ is not uniquely determining STUD-ID. There are STUD-NO corresponding to Ram (1 and 2).
- STUD_NAME → STUD_STATE, **FD does not hold**

{ STUD_NO → STUD_NAME,
STUD_NO → STUD_PHONE,
STUD_NO → STUD_STATE,
STUD_NO → STUD_COUNTRY,
STUD_NO → STUD_AGE,
STUD_STATE → STUD_COUNTRY }

Properties of FD

Let *X*, *Y*, and *Z* are sets of attributes in a relation *R*. There are several properties of functional dependencies which always hold in R also known as Armstrong Axioms.

- **Reflexivity:** If *Y* is a subset of *X*, then *X* → *Y*. If $Y \subseteq X$, then $X \rightarrow Y$
- e.g.; Let X represents {E-ID, E-NAME} and Y represents {E-ID}.
  {E-ID, E-NAME} → E-ID is true for the relation.
- **Augmentation:** If *X* → *Y*, then *XZ* → *YZ*.

e.g.; Let X represents {E-ID}, Y represents {E-NAME} and Z represents {E-CITY}.
As {E-ID} →{E-NAME} is true for the relation,
so { E-ID,E-CITY} →{E-NAME,E-CITY} will also be true.

- **Transitivity:** If *X* → *Y* and *Y* → *Z*, then *X* → *Z*.
- e.g.; Let X represents {E-ID}, Y represents {E-CITY} and Z represents {E-STATE}.
  As {E-ID} →{E-CITY} and {E-CITY} →{E-STATE} is true for the relation,
  so { E-ID } →{E-STATE} will also be true.

Properties of FD

- **Union:** It states that if X determines Y and X determines Z then X must also determine Y and Z

$$\text{If } X \rightarrow Y \text{ and } X \rightarrow Z \text{ then } X \rightarrow YZ$$

- **Decomposition:** This rule states that if X determines Y and Z, then X determines Y and X determines Z separately

$$\text{If } X \rightarrow YZ \text{ then } X \rightarrow Y \text{ and } X \rightarrow Z$$

Trivial Dependency

A trivial functional dependency is the one which will always hold in a relation.

- **X → Y will always hold if X ⊇ Y** If $Y \subseteq X$, then $X \rightarrow Y$
- In the example given above, **E-ID, E-NAME → E-ID** is a trivial functional dependency
- If a functional dependency is not trivial, it is called **Non-Trivial Functional Dependency**. Non-Trivial functional dependency may or may not hold in a relation.
- e.g; **E-ID → E-NAME** is a non-trivial functional dependency which holds in the above relation.

Trivial Dependency

- Which of the following is the trivial functional dependency?
- (a) {P, R} → {S, T}
  (b) {P, R} → {R, T}
  (c) {P, S} → {S}
  (d) {P, S, U} → {Q}
- Gate 2015

Trivial Dependency

- Which of the following is the trivial functional dependency?
- (a) {P, R} → {S, T}
  (b) {P, R} → {R, T}
  (c) {P, S} → {S}
  (d) {P, S, U} → {Q}
- Ans: option (c)
  Explanation:
  A functional dependency X → Y is trivial, if Y is a subset of X.
  In the above question , {S}is a subset of {P,S}. Hence option (c) is the answer.

Transitive dependency

- Let A, B, and C designate three distinct (but not necessarily disjoint) sets of attributes of a relation. Suppose all three of the following conditions hold:
- If X → Y and Y → Z is true, then X → Y is a transitive dependency.
- X → Y
- Y does not → X
- Y → Z

{Book} → {Author}

{Author} does not → {Book}

{Author} → {Author Nationality}

Exercise

- Given the following relation instance.

X Y Z

----

1 4 2

1 5 3

1 6 3

3 2 2

Which of the following functional dependencies are satisfied by the instance?

(a) XY → Z and Z → Y
(b) YZ → X and Y → Z
(c) YZ → X and X → Z
(d) XZ → Y and Y → X

- (Gate 2000)

Exercise

- Given the following relation instance.

```
-------
X Y Z
-------
1 4 2
1 5 3
1 6 3
3 2 2
-------
```

Which of the following functional dependencies are satisfied by the instance?

(a) XY → Z and Z → Y
(b) YZ → X and Y → Z
(c) YZ → X and X → Z
(d) XZ → Y and Y → X

- (Gate 2000)

Attribute Closure (F)+

- Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.

How to find attribute closure of an attribute set?

To find attribute closure of an attribute set:

- Add elements of attribute set to the result set. (Add A to S)
- Recursively add elements to the result set which can be functionally determined from the elements of the result set.

Attribute Closure

{ STUD_NO → STUD_NAME,
STUD_NO → STUD_PHONE,
STUD_NO → STUD_STATE,
STUD_NO → STUD_COUNTRY,
STUD_NO → STUD_AGE,
STUD_STATE → STUD_COUNTRY }

- **attribute closure** can be determined as:
- (STUD_NO)+ = {STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY, STUD_AGE}
- (STUD_STATE)+ = {STUD_STATE, STUD_COUNTRY}

Exercise

- The following functional dependencies are given:

$AB \rightarrow CD$, $AF \rightarrow D$, $DE \rightarrow F$, $C \rightarrow G$, $F \rightarrow E$, $G \rightarrow A$

Which one of the following options is false?

(a) $CF^+ = \{ACDEFG\}$

(b) $BG^+ = \{ABCDG\}$

(c) $AF^+ = \{ACDEFG\}$

(d) $AB^+ = \{ABCDFG\}$

(Gate 2006, 2014)

Exercise - Solution

- Ans: option(c) and Option (d)
- Explanation:
- AF^{+} = {AFDE}
- AB^{+} = {ABCDG}.

How to check whether an FD can be derived from a given FD set?

- To check whether an FD A → B can be derived from an FD set F,
- Find (A)+ using FD set F.
- If B is subset of (A)+,
  - then A → B is true
  - else not true.

FD from FD Set

- **In a schema with attributes A, B, C, D and E following set of functional dependencies are given**
  {A → B, A → C, CD → E, B → D, E → A}
  Which of the following functional dependencies is NOT implied by the above set? (GATE IT 2005)
  A. CD → AC
  B. BD → CD
  C. BC → CD
  D. AC → BC

FD from FD Set

- Using FD set given in question,
(CD)+ = {CDEAB} which means CD → AC also holds true.
(BD)+ = {BD} which means BD → CD can't hold true. So this FD is no implied in FD set.
- Others can be checked in the same way.

- So (B) is the required option.

How to find Candidate Keys and Super Keys using Attribute Closure?

- If **attribute closure** of an attribute set contains **all attributes** of relation, the attribute set will be **super key** of the relation.

- If **no subset** of this attribute set can **functionally determine all attributes** of the relation, the set will be **candidate key** as well

Finding a key – Exercise1

- **GATE Question: Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, N} and the set of functional dependencies {**

  {E, F} → {G};

  {F} → {I, J};

  {E, H} → {K, L};

  K → {M};

  L → {N}

} on R. What is the key for R? (GATE-CS-2014)

A. {E, F}

B. {E, F, H}

C. {E, F, H, K, L}

D. {E}

Finding a key

- **Answer:** Finding attribute closure of all given options, we get:
  {E,F}+ = {EFGIJ}
  {E,F,H}+ = {EFHGIJKLMN}
  {E,F,H,K,L}+ = {{EFHGIJKLMN}
  {E}+ = {E}
  {EFH}+ and {EFHKL}+ results in set of all attributes, but EFH is minimal. So it will be candidate key.
- So correct option is (B).

Finding a key – Exercise 2

- Compute the closure of the following set F of functional dependencies for relation schema R = {A, B, C, D, E}.

  $A \rightarrow BC$

  $CD \rightarrow E$

  $B \rightarrow D$

  $E \rightarrow A$

- List the candidate keys for R.

- Compute the closure of the following set F of functional dependencies for relation schema R = {A, B, C, D, E}.

  $A \rightarrow BC$

  $CD \rightarrow E$

  $B \rightarrow D$

  $E \rightarrow A$

- List the candidate keys for R.

Answer:

- $A \rightarrow BC$, $B \rightarrow D$ so $A \rightarrow D$ so $A \rightarrow DC \rightarrow E$ therefore $A \rightarrow ABCDE$
- $E \rightarrow A$, $A \rightarrow ABCDE$, so $E \rightarrow ABCDE$
- **Attribute closure:**

A+={ABCDE}
B+={BD}
C+={C}
D+={D}
E+={ABCDE}

AB+={ABCDE}
AC+={ABCDE}
AD+={ABCDE}
AE+={ABCDE}
BC+={ABCDE}
BD+={BD}
BE+={ABCDE}
CD+={ABCDE}
CE+={ABCDE}
DE+={ABCDE}

ABC+={ABCDE}
ABD+={ABCDE}
ABE+={ABCDE}
ACD+={ABCDE}
ACE+={ABCDE}
ADE+={ABCDE}
BCD+={ABCDE}
BDE+={ABCDE}
CDE+={ABCDE}

ABCD+={ABCDE}
ABCE+={ABCDE}
ABDE+={ABCDE}
ACDE+={ABCDE}
BCDE+={ABCDE}

Finding a key – Exercise 3

- Consider a relation R(A,B,C,D,E) with the following dependencies:
- $\{AB \rightarrow C, CD \rightarrow E, DE \rightarrow B\}$
- Is AB a candidate key of this relation? If not, is ABD? Explain your answer.

- A → A
- B → B
- C → C
- D → D
- E → E
- AB → ABC
- AC → AC
- AD → AD
- AE → AE
- BC → BC
- BD → BD
- BE → BE
- CD → BCDE
- CE → CE
- DE → BDE
- ABD → ABCDE
- No. The closure of AB does not give you all of the attributes of the relation.
  Yes, ABD is a candidate key. No subset of its attributes is a key.

Extraneous Attribute

- If we are able to **remove an attribute** from a functional dependency **without changing the closure** of the set of functional dependencies, that attribute is called as Extraneous Attribute.
- *Dictionary meaning of ‘Extraneous’ is ‘irrelevant’, ‘inappropriate’, or ‘unconnected’*

Extraneous Attribute

- Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.

a.) Attribute A is extraneous in α if $A \in \alpha$ and F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$

b.) Attribute A is extraneous in β if $A \in \beta$ and F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$

Extraneous Attribute

- Assume a set of functional dependencies F, and the **closure** of set of functional dependencies $\mathbf{F^+}$.
- Also, assume that we **remove an attribute (Extraneous Attribute) from** any of the FDs under **F** and find the closure of new set of functional dependencies.
- Let us mention the **new closure** of set of functional dependencies as $\mathbf{F1^+}$.
- If $\mathbf{F^+}$ **equals** the newly constituted closure (Minimal Cover) $\mathbf{F1^+}$, then the attribute which has been removed is called as Extraneous Attribute.
- In other words, that attribute does not violate any of the functional dependencies.

Extraneous Attribute

- Let us consider a relation R with schema R(A, B, C) and set of functional dependencies
- F = { AB → C, A → C }.
- The closure F^+ = {A → C, AB → C}.
- In AB → C, B is extraneous attribute. The reason is, ***there is another FD A → C, which means when A alone can determine C,*** the use of B is unnecessary (redundant).
- $F1^+$ = {A → C}.

Minimal cover

<u>Definition</u>

A set of FDs F is minimum if F has as few FDs as any equivalent set of FDs.

Simple properties/steps of minimal cover:

Steps to find the minimal cover;

1. Ensure **singleton attribute on the right hand side** of each functional dependency (**apply decomposition rule**).

2. Remove **extraneous (redundant) attribute from the left hand side** of each functional dependency.

3. Remove redundant functional dependency if any.

Canonical Cover

- Both concepts are same
- A **canonical cover** is "allowed" to have more than one attribute on the right hand side. A **minimal cover** cannot allow more than one attribute at RHS.
- As an example, the **canonical cover** is A → BC where the **minimal cover** would be A → B, A → C .

Exercise 1 – Minimal Cover

- **Consider a relation R(A,B,C,D) having some attributes and below are mentioned functional dependencies.**
- **FD1 : B → A**
- **FD2 : AD → C**
- **FD3 : C → ABD**

- **Step-1 : Decompose the functional dependencies using Decomposition rule(Armstrong's Axiom) i.e. single attribute on right hand side.**
- **FD1 : B → A**
- **FD2 : AD → C**
- **FD3 : C → A**
- **FD4 : C → B**
- **FD5 : C → D**

- Step-2 : Remove extraneous attributes from LHS of functional dependencies.
- Here, only one FD has two or more attributes of LHS i.e. AD $\rightarrow$ C.
- In this case, attribute “C” is determined by AD only.
- Hence, no extraneous attributes are present and the FD will remain the same and will not be removed.

- Step-3 : Remove FD's having transitivity.
- FD1 : B → A
- FD2 : C → A
- FD3 : C → B
- FD4 : AD → C
- FD5 : C → D

Repetition, So check B->C or C->B in the FD set. C->B exists. Use transitivity rule and remove transitivity

- Above FD1, FD2 and FD3 are forming transitive pair. Hence, using Armstrong's law of transitivity

  i.e. if X → Y, Y → X then X → Z

  should be removed.

C →B, B →A then remove C →A

Therefore we will have the following FD's left :

- FD1 : B →A
- FD2 : C →B
- FD3 : AD →C
- FD4 : C →D

Minimal Cover – Exercise 2

Let R(A, B, C) be a relation with the following set F of functional dependencies;
F = { A → B, B → A, A → C, C → A, B → C }
Find the minimal cover of F.

According to rule 1, if we have any FDs with more than one attribute on the Right Hand Side, that FD should be decomposed using decomposition rule. We don't have such FDs.

According to rule 2, if we have any FDs that have more than one attribute on the Left Hand Side (determiner), that FD must be checked for partial dependency. We don't have such FDs.
Hence, the given set satisfies both rules.

Minimal Cover

After Step 3 – Removing repetitions

F = { A → B, B → A, A → C, C → A, B → C }

- Apply transitivity rule:
- **A → C, B → C** (check A → B or B → A exists in the concrete list; **A → B** exists)

  A → B, B → C so remove A → C
- **So F = { A → B, B → A, C → A, B → C }**
- **B → A, C → A** (check B → C or C → A exists in the concrete list; **B → C** exists)

  B → C, C → A so remove B → A
- **So F_c = { A → B, C → A, B → C }**

Minimal Cover – Exercise 3

Find the minimal cover of the set of functional dependencies given;

{A → C, AB → C, C → DI, CD → I, EC → AB, EI → C}

Find the minimal cover of the set of functional dependencies given;

{A → C, AB → C, C → DI, CD → I, EC → AB, EI → C}

Solution

1). Right Hand Side (RHS) of all FDs should be single attribute. So we write F as F1, as follows;

F1 = {A → C, AB → C, C → D, C → I, CD → I, EC → A, EC → B, EI → C}

2. Remove extraneous attributes.

Extraneous attribute is a redundant attribute on the LHS of the functional dependency. In the set of FDs, AB → C, CD → I, EC → A, EC → B, and EI → C have more than one attribute in the LHS. Hence, we check one of these LHS attributes are extraneous or not.

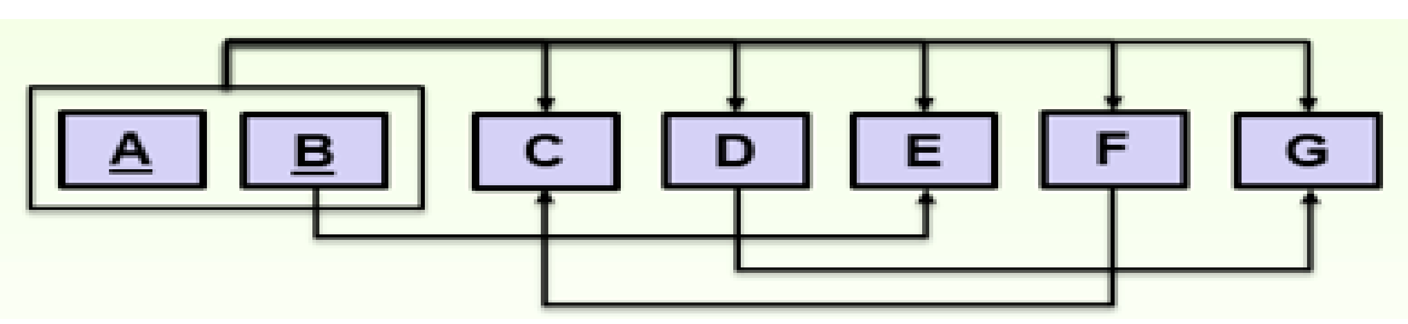
F2 = {A → C, C → D, C → I, EC → A, EC → B}

3. Eliminate redundant functional dependency.

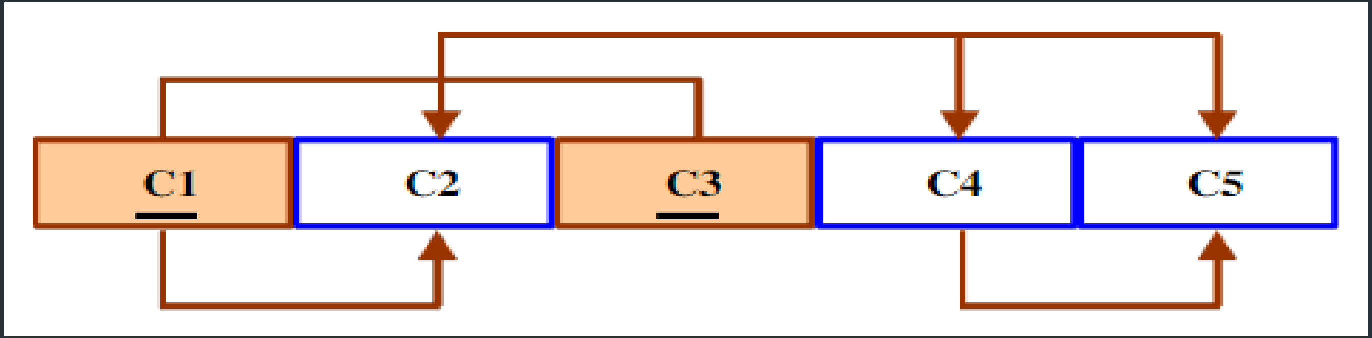
- None of the FDs in F2 is redundant. Hence, F2 is minimal cover.
- Hence, **set of functional dependencies F2 is the minimal cover for the set F.**

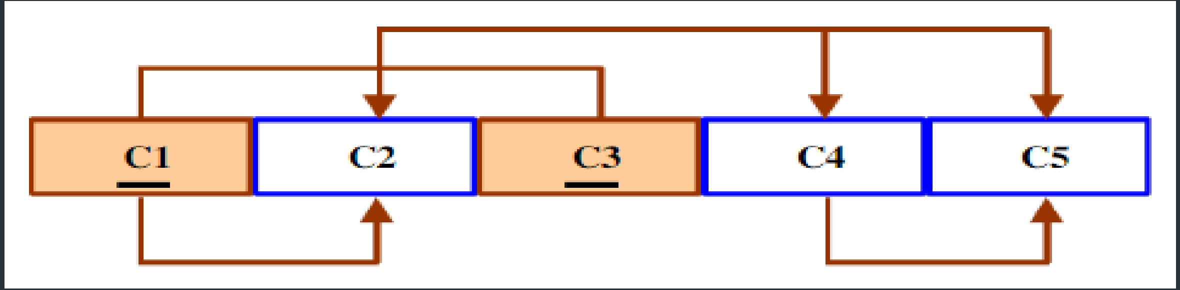
F2 = {A → C, C → D, C → I, EC → A, EC → B}

Dependency diagram



Dependency diagram





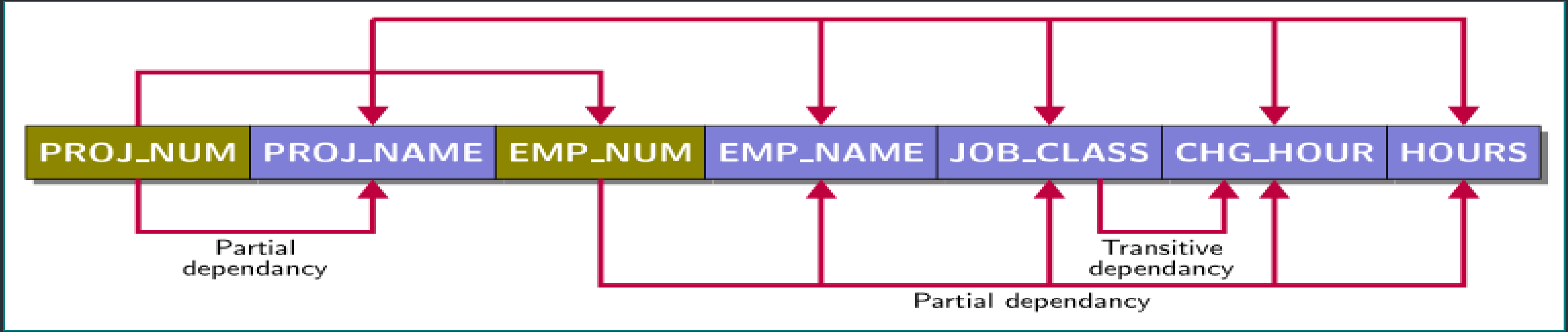
- The following dependencies are identified:
- C1 and C3, are the Primary Key.
- Partial Dependencies:

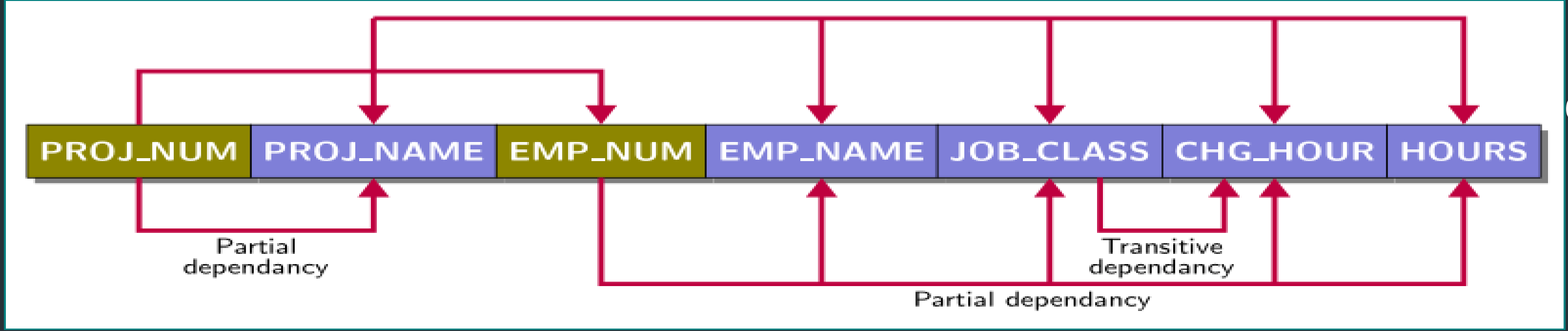
  C1—> C2
- Transitive Dependency:

  C4 —> C5

Dependency diagram

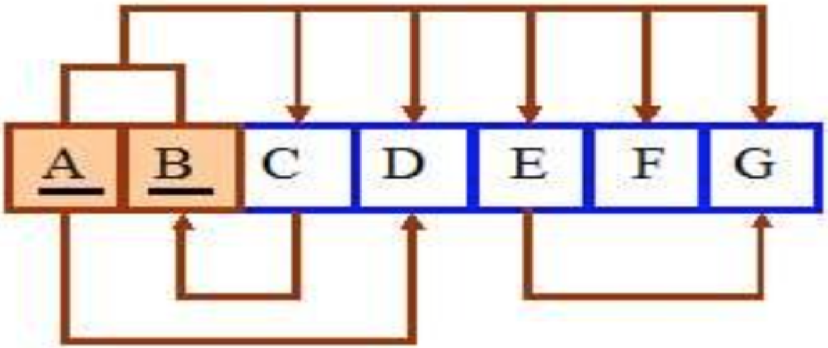
- A dependency diagram, shown in Figure, illustrates the various dependencies that might exist in a *non-normalized table*. A non-normalized table is one that has data redundancy in it.





- The following dependencies are identified:
- Proj_Num and Emp_Num, combined, are the PK.
- Partial Dependencies:
  - Proj_Num —> Proj_Name
  - Emp_Num —> Emp_Name, Job_Class, Chg_Hour, Hours
- Transitive Dependency:
  - Job_Class —> Chg_Hour

a) Given the following dependency diagram, label all the dependencies.



b) Redesign the database to 2NF. Show all the steps.
c) Redesign the database to 3NF. Show all the steps.


Thank
You!

Database Design: Normalization

Dr. Jenila Livingston

Data Normalization

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that ***avoid unnecessary duplication of data***
- The process of decomposing relations with anomalies to produce smaller, ***well-structured*** relations

Results of Normalization

- Removes the following modification *anomalies* (integrity errors) with the database

  - Insertion
  - Deletion
  - Update

A Typical Spreadsheet File

Emp No	Employee Name	Time Card No	Time Card Date	Dept No	Dept Name
10	Thomas Arquette	106	11/02/2002	20	Marketing
10	Thomas Arquette	106	11/02/2002	20	Marketing
10	Thomas Arquette	106	11/02/2002	20	Marketing
10	Thomas Arquette	115	11/09/2002	20	Marketing
99	Janice Smitty			10	Accounting
500	Alan Cook	107	11/02/2002	50	Shipping
500	Alan Cook	107	11/02/2002	50	Shipping
700	Ernest Gold	108	11/02/2002	50	Shipping
700	Ernest Gold	116	11/09/2002	50	Shipping
700	Ernest Gold	116	11/09/2002	50	Shipping

Employee, Department, and Time Card Data in Three Tables

Table: Employees

EmpNo	EmpFirstName	EmpLastName	DeptNo
10	Thomas	Arquette	20
500	Alan	Cook	50
700	Ernest	Gold	50
99	Janice	Smitty	10

Table: Departments

DeptNo	DeptName
10	Accounting
20	Marketing
50	Shipping

Table: Time Card Data

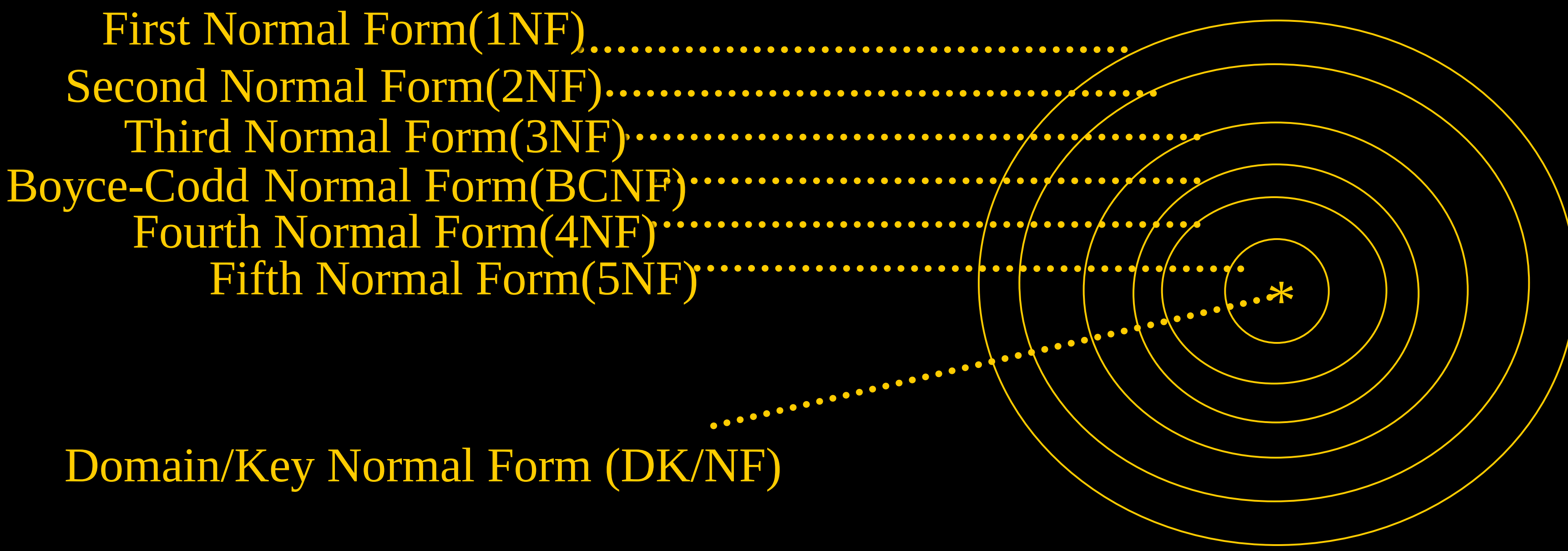
TimeCardNo	EmpNo	TimeCardDate
106	10	11/02/2002
107	500	11/02/2002
108	700	11/02/2002
115	10	11/09/2002
116	700	11/09/2002

Primary Key

NORMAL FORMS

- ✓ 1 NF
- ✓ 2NF
- ✓ 3NF
- BCNF (Boyce-Codd Normal Form)
- 4NF
- 5NF
- *DK (Domain-Key) NF*

Relationships of Normal Forms



Normal Forms

- First Normal Form
  - No repeating groups in tables
- Second Normal Form
  - Table is 1st normal form and no partial key dependencies
- Third Normal Form
  - Table is in 2nd normal form and has no transitive dependencies

Normal Forms

- Boyce-Codd Normal Form
  - Every determinant of a non-key attribute is a candidate key
- Fourth Normal Form
  - A table has no multi-valued dependencies
- Fifth Normal Form
  - There are no lossey joins between two or more tables

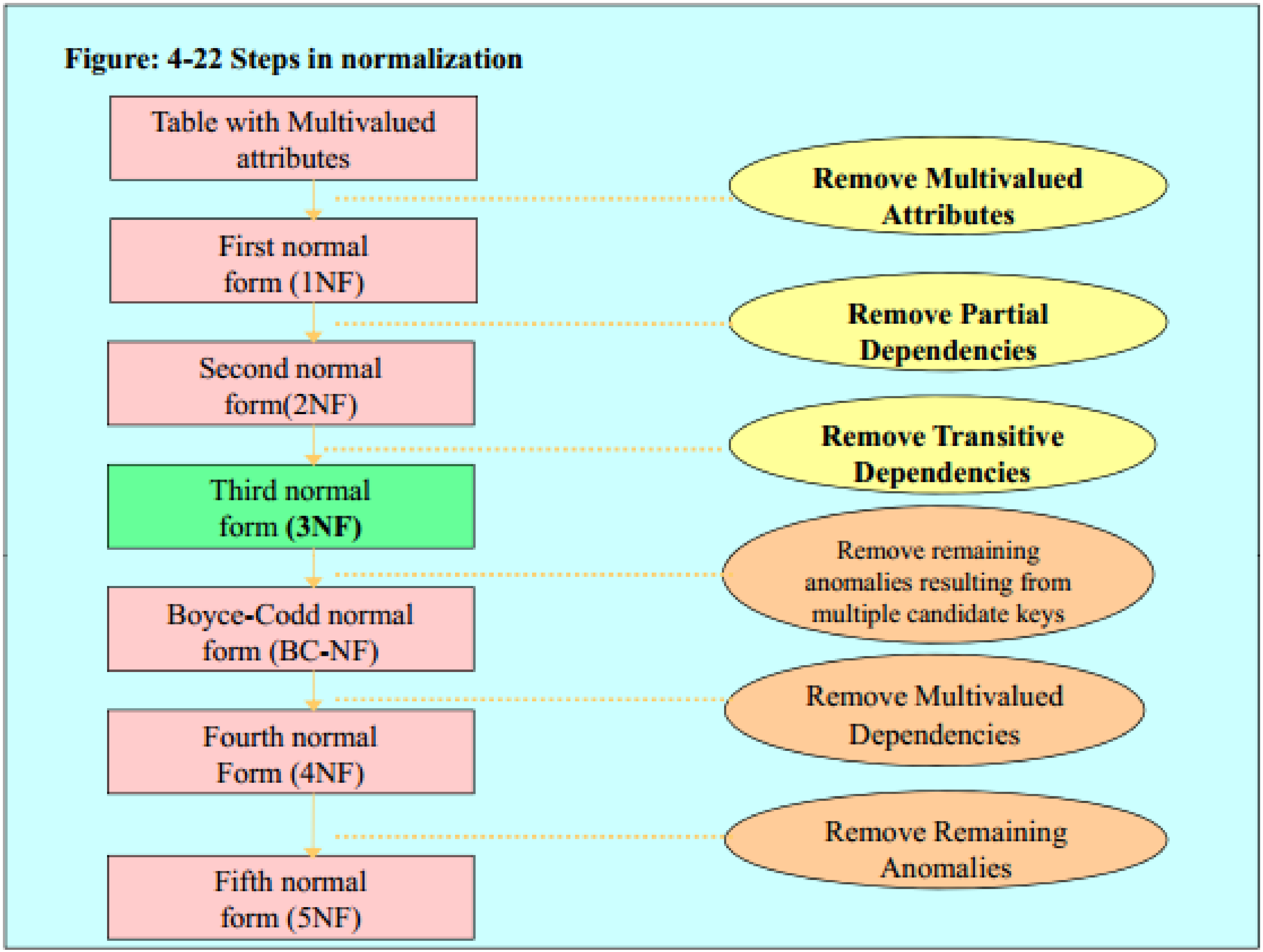


Figure: 4-22 Steps in normalization
Table with Multivalued attributes
Remove Multivalued Attributes
First normal form (1NF)
Remove Partial Dependencies
Second normal form(2NF)
Remove Transitive Dependencies
Third normal form (3NF)
Remove remaining anomalies resulting from multiple candidate keys
Boyce-Codd normal form (BC-NF)
Remove Multivalued Dependencies
Fourth normal Form (4NF)
Remove Remaining Anomalies
Fifth normal form (5NF)

- **Based on dependency the Normalization forms are classified as follows:**

1. First Normal Form (1NF)

Normalization using Functional dependency

2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce Codd Normal Form (BCNF)

Normalization using Multi-valued dependency

5. Fourth Normal Form (4NF)

Normalization using Join dependency

6. Fifth Normal Form (5NF) **(or)**
   Project-Join Normal Form (PJNF)
7. Domain Key Normal Form (DKNF)

1NF

- Rule

A table is said to be in First Normal Form, if each cell of the table contains only one value.

- A university uses the following relation:

After 1NF

Students

FirstName	LastName	Knowledge
Thomas	Mueller	Java, C++, PHP
Ursula	Meier	PHP, Java
Igor	Mueller	C++, Java

Startsituation

Result after Normalisation

Students

FirstName	LastName	Knowledge
Thomas	Mueller	C++
Thomas	Mueller	PHP
Thomas	Mueller	Java
Ursula	Meier	Java
Ursula	Meier	PHP
Igor	Mueller	Java
Igor	Mueller	C++

Take the following table.

StudentID is the primary key.

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English Maths Info Tech	$50 $50 $100	B A B+

Is it 1NF?

No. There are repeating groups (subject, subjectcost, grade)

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English Maths Info Tech	$50 $50 $100	B A B+

How can you make it 1NF?

Create new rows so each cell contains only one value

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English Maths Info Tech	$50 $50 $100	B A B+

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

2NF

- Rule

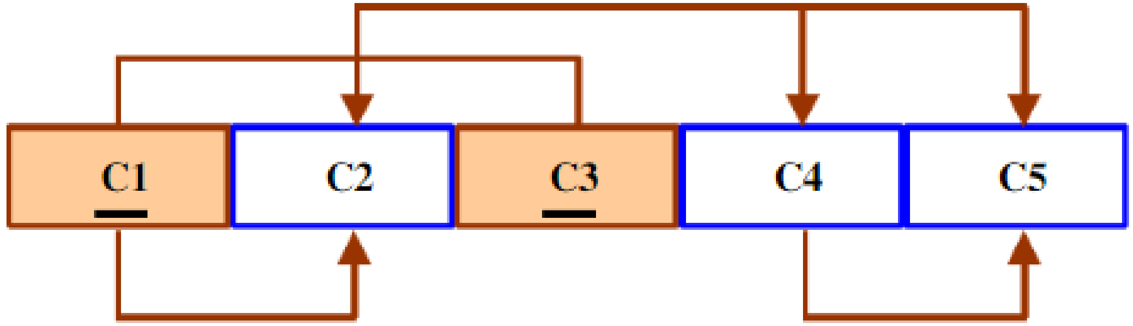
A table is said to be in Second Normal Form (2NF), when it's in 1NF and every attribute in the row is functionally dependent on whole key (fully), not just part of the key.

Make new tables

- Make a **new table for each primary key** field
- Move columns from the original table to the new table that matches their primary key
  - Table with keys

    (OR)
  - Table with keys + other common attributes

2NF



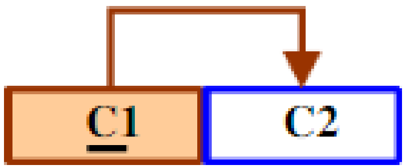


Table 1

Primary key: C1
Foreign key: None
Normal form: 3NF

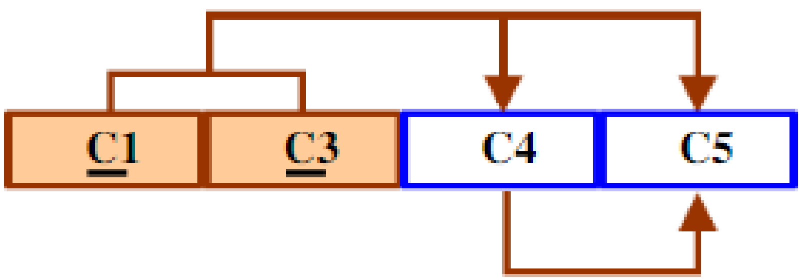


Table 2

Primary key: C1 + C3
Foreign key: C1 (to Table 1)
Normal form: 2NF, because the table exhibits the transitive dependencies C4 $\longrightarrow$ C5

Bringing a Relation to 2NF – Example 2

- Goal: Remove Partial Dependencies

Composite Primary Key

Partial Dependencies

STUDENT

<u>Stud_ID</u>	Name	<u>Course_ID</u>	Units
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

Bringing a Relation to 2NF

CUSTOMER

Stud_ID	Name	Course_ID	Units
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

STUDENT_COURSE

Stud_ID	Course_ID
101	MSI 250
101	MSI 415
125	MSI 331

STUDENT

Stud_ID	Name
101	Lennon
101	Lennon
125	Johnson

COURSE

Course_ID	Units
MSI 250	3.00
MSI 415	3.00
MSI 331	3.00

2NF – Example 3

StudentId	UnitCode	UnitName
0023765	UG45783	Advance Database
0023765	UG45832	Network Systems
0023765	UG45734	Multi-User Operating Systems
0035643	UG45832	Network Systems
0035643	UG45951	Project
0061234	UG45783	Advance Database

2NF

StudentId	UnitCode
0023765	UG45783
0023765	UG45832
0023765	UG45734
0035643	UG45832
0035643	UG45951
0061234	UG45783

Tables in Second Normal Form

UnitCode	UnitName
UG45783	Advance Database
UG45832	Network Systems
UG45734	Multi-User Operating Systems
UG45951	Project

2NF – Example 4

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

Step1

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

Step 1

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

Step 2

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

SUBJECTS TABLE (key = Subject)

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

RESULTS TABLE (key = StudentID+Subject)

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

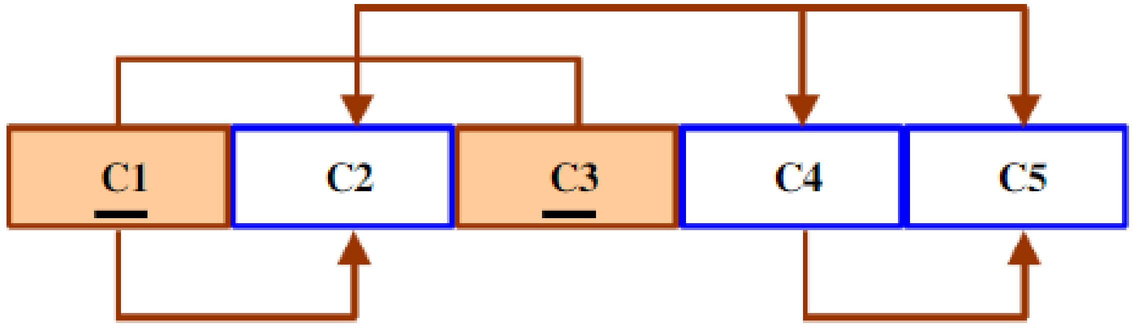
3NF

- Rule

A table is said to be in Third Normal Form (3NF), when it's in 2NF and primary key is functionally dependent on every non-key attribute.

A **transitive dependency** is a type of functional dependency in which the value in a non-key field is determined by the value in another non-key field and that field is not a candidate key.

3NF



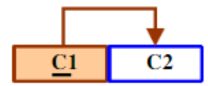


Table 1

Primary key: C1
Foreign key: None
Normal form: 3NF

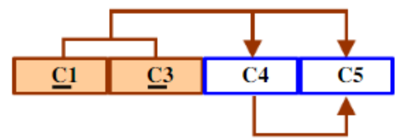


Table 2

Primary key: C1 + C3
Foreign key: C1 (to Table 1)
Normal form: 2NF, because the table exhibits the transitive dependencies C4 ⟶ C5

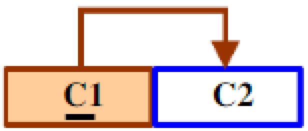


Table 1
Primary key: C1
Foreign key: None
Normal form: 3NF

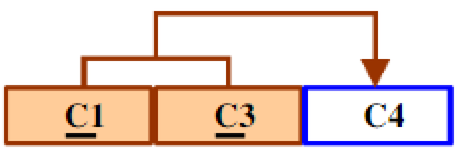


Table 2
Primary key: C1 + C3
Foreign key: C1 (to Table 1)
C4 (to Table 3)
Normal form: 3NF

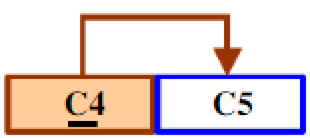


Table 3
Primary key: C4
Foreign key: None
Normal form: 3NF

3NF – Example 2

STUDENT TABLE (key = StudentID)

StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

And 3NF says that non-key fields must depend on ***nothing but the key***

SUBJECTS TABLE (key = Su

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Again, carve off the offending fields

StudentTable

StudentID	StudentName	Address
19594332X	Mary Watson	10 Charles Street

Primary key: StudentID

HouseTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

SUBJECTS TABLE (key = Su

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

After 3NF

StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID

∞

1

1

HouseTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

SUBJECTS TABLE (key = Su

Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

∞

∞

1

StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

The Reveal

Before...

StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English Maths Info Tech	$50 $50 $100	B A B+

After...

HouseTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID

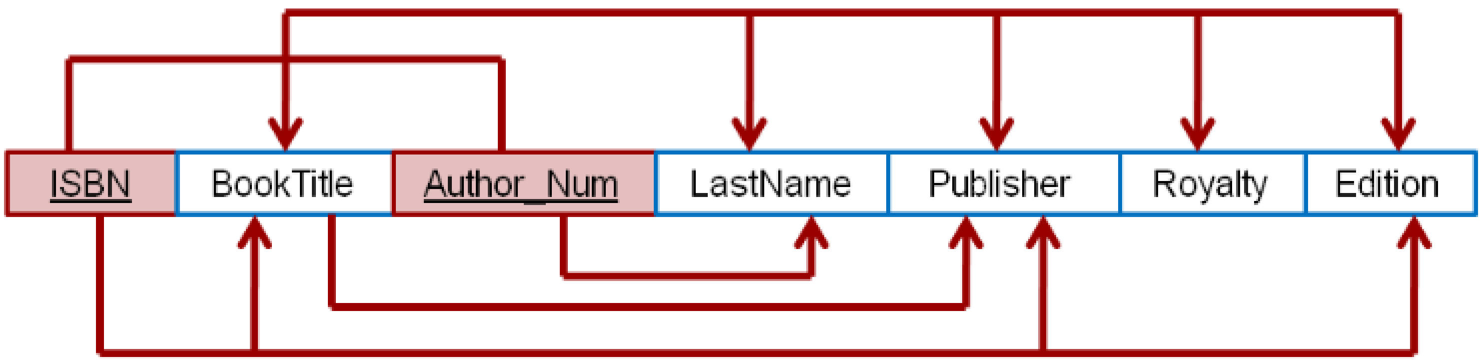
Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

SUBJECTS TABLE (key = Subject)

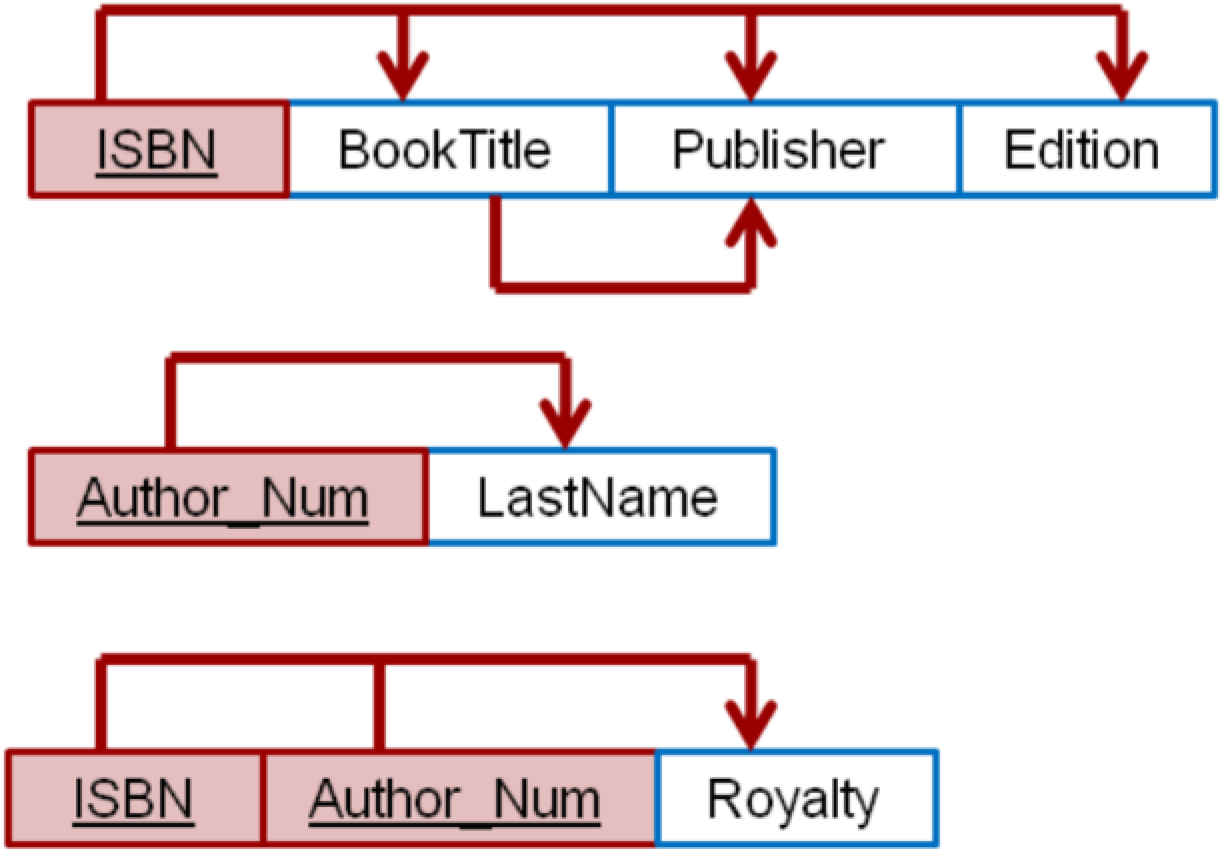
StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+

RESULTS TABLE (key = StudentID+Subject)

Exercise -1

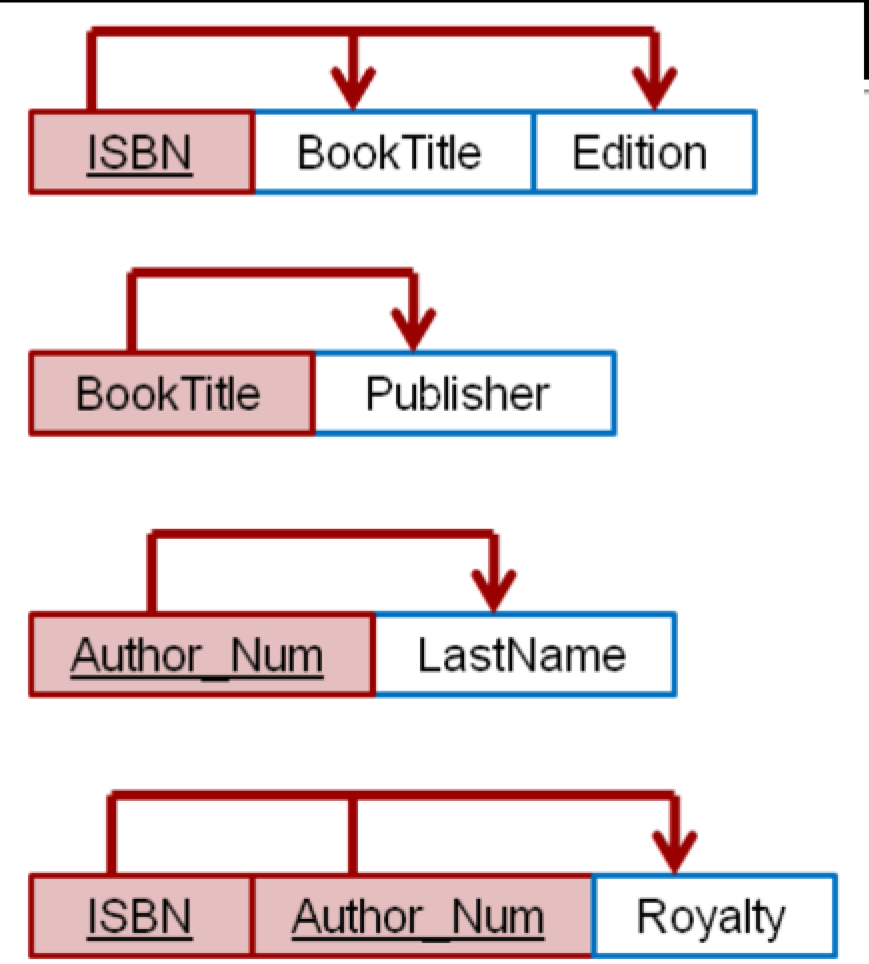

ISBN
BookTitle
Author_Num
LastName
Publisher
Royalty
Edition

2NF



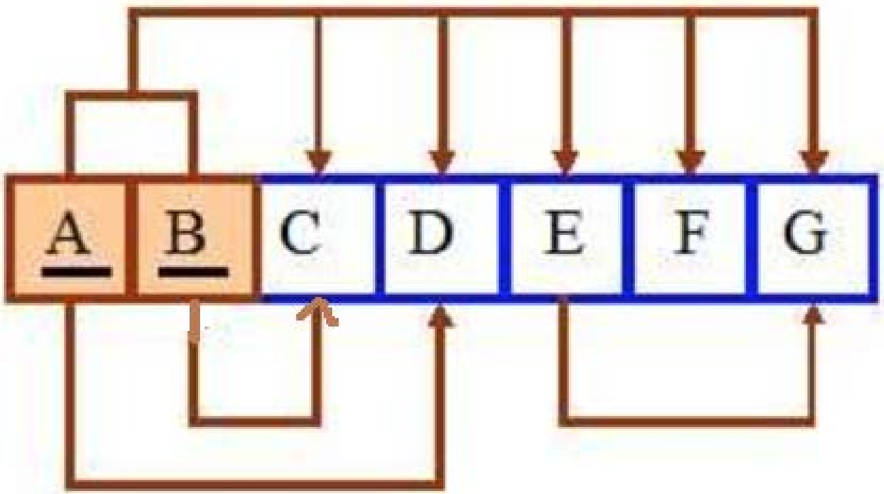
ISBN
BookTitle
Publisher
Edition
Author_Num
LastName
ISBN
Author_Num
Royalty

3NF


ISBN
BookTitle
Edition
BookTitle
Publisher
Author_Num
LastName
ISBN
Author_Num
Royalty

Exercise 2

a) Given the following dependency diagram, label all the dependencies.



b) Redesign the database to 2NF. Show all the steps.
c) Redesign the database to 3NF. Show all the steps.

BCNF

- **Boyce Codd normal Form** (or **BCNF** or **3.5NF**)
- BCNF was developed in 1974 by Raymond F. Boyce and Edgar F. Codd to address certain types of anomalies not dealt with by 3NF as originally defined.[
- Rule

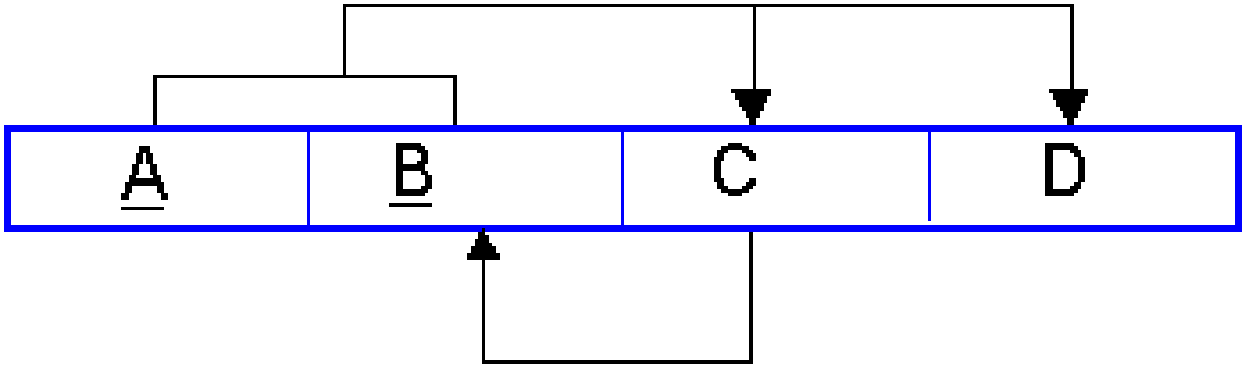
A table is said to be in Boyce Codd Normal Form (BCNF), if and only if every determinant (attribute) is a candidate key.

CANDIDATE KEY:

An attribute (or) set of attributes that uniquely identifies each row is called " Candidate key ".

BCNF

This occurs when a non key attribute is a determinant of a key attribute.

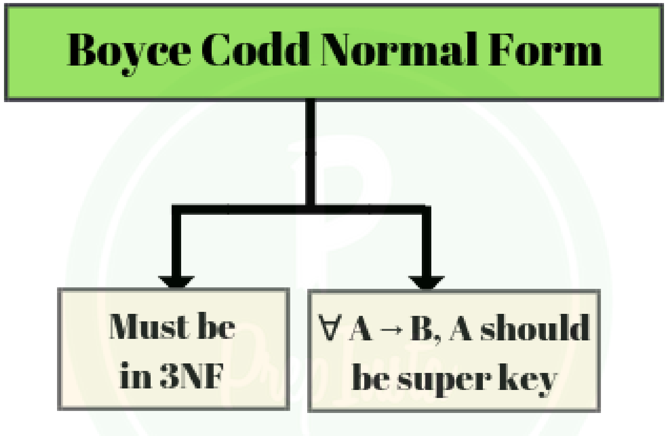


- C->B

BCNF

Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one Candidate Key.

A table is in BCNF if every functional dependency A → B, A is the super key of the table.



BCNF

- Example of a table not in BCNF:

Student	Course	Teacher
Sok	DB	John
Sao	DB	William
Chan	E-Commerce	Todd
Sok	E-Commerce	Todd
Chan	DB	William

- Key: {Student, Course}
- Functional Dependency:
  - {Student, Course}→ Teacher
  - Teacher → Course
- Problem: ***Teacher*** is not a superkey but determines ***Course***.

After BCNF

Table1

Course	Teacher
DB	John
DB	William
E-Commerce	Todd

Table2

Student	Course
Sok	DB
Sao	DB
Chan	E-Commerce
Sok	E-Commerce
Chan	DB

After BCNF

S_Num	T_Code	Offering#	Review Date
123599	FIT104	01764	2nd March
123599	PIT305	01765	12th April
123599	PIT107	01789	2nd May
346700	FIT104	01764	3rd March
346700	PIT305	01765	7th May

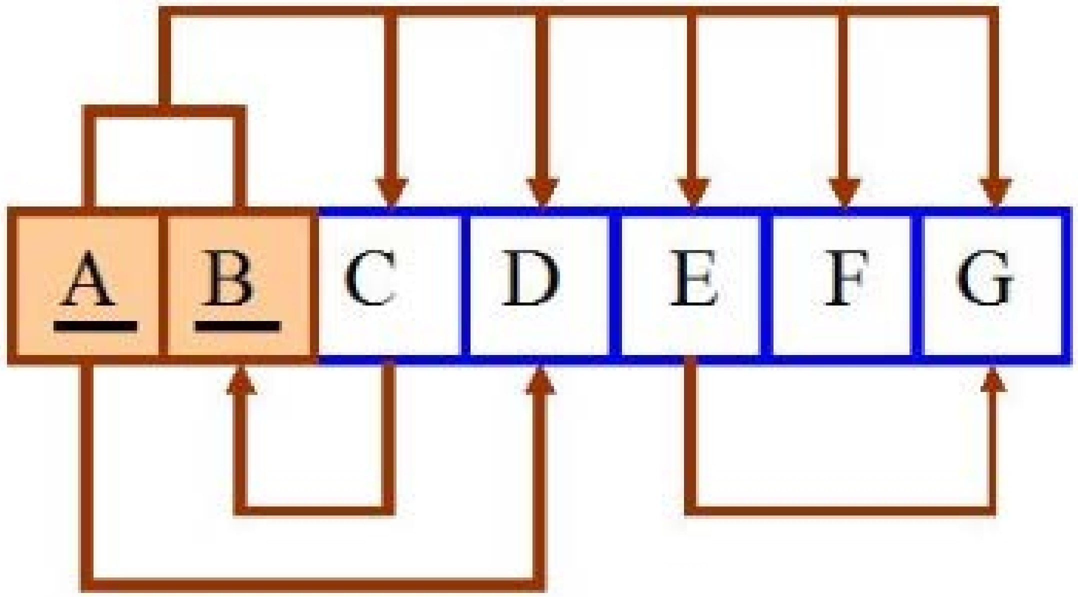
S_Num	Offering#	Review Date
123599	01764	2nd March
123599	01765	12th April
123599	01789	2nd May
346700	01764	3rd March
346700	01765	7th May

OfferingTeacher

Offering#	T_Code
01764	FIT104
01765	PIT305
01789	PIT107

Exercise 1

Initial Dependency Diagram for Problem



a) Break up the dependency diagram shown in Figure to create two new dependency diagrams: in 2NF.
b) Modify the dependency diagrams you created in part a to produce a set of dependency diagrams that are in 3NF. (*Hint:*One of your dependency diagrams should be in 3NF but not in BCNF.)
c) Modify the dependency diagrams you created in Part b to produce a collection of dependency diagrams that are in 3NF and BCNF.

Exercise 2

- Find the highest normal form of a relation R(A, B, C, D, E) with FD set as:
- { C->D, AC->BE, B->E }

Solution

- $(AC)^+ = \{A, C, B, E, D\}$
- So there will be only 1 candidate key {AC}

Normalization using Multi-valued dependency

Fourth Normal Form (4NF)

Multi-Valued Dependency

- **Multi-valued dependency (or) Tuple-generating dependency**
  - It is a type of Functional dependency, where the determinant (attribute) can determine more than one value.

4NF

- Rule

A table is said to be in Fourth Normal Form (4NF), when it is in BCNF and it has one independent Multi-valued dependency (or) one independent multi-valued dependency with a functional dependency.

(or)

Reduce BCNF entities to 4NF by removing any independent multi-valued components of the primary key to two new parent entities.

Example

- Let us consider the following table "sports_event"

Stud_id	Major	Activities
C100	MCA	Baseball
C100	MCA	Volleyball
M100	MBA	Cricket
M100	MBA	Volleyball
T200	ITC	Swimming

- To convert the sport_event relation to 4NF :

Reduce BCNF entities to 4NF by removing any independent multi-valued components of the primary key to two new parent entities.

4NF

stud_major

Stud_id	Major
C100	MCA
C100	MCA
M100	MBA
M100	MBA
T200	ITC

(or)

Stud_id	Major
C100	MCA
M100	MBA
T200	ITC

Stud_activity

Stud_id	Activities
C100	Baseball
C100	Volleyball
M100	Cricket
M100	Volleyball
T200	Swimming

4NF – Example 2

Pizza Delivery Permutations

Restaurant	Pizza Variety	Delivery Area
A1 Pizza	Thick Crust	Springfield
A1 Pizza	Thick Crust	Shelbyville
A1 Pizza	Thick Crust	Capital City
A1 Pizza	Stuffed Crust	Springfield
A1 Pizza	Stuffed Crust	Shelbyville
A1 Pizza	Stuffed Crust	Capital City
Elite Pizza	Thin Crust	Capital City
Elite Pizza	Stuffed Crust	Capital City
Vincenzo's Pizza	Thick Crust	Springfield
Vincenzo's Pizza	Thick Crust	Shelbyville
Vincenzo's Pizza	Thin Crust	Springfield
Vincenzo's Pizza	Thin Crust	Shelbyville

4NF

Varieties By Restaurant

Restaurant	Pizza Variety
A1 Pizza	Thick Crust
A1 Pizza	Stuffed Crust
Elite Pizza	Thin Crust
Elite Pizza	Stuffed Crust
Vincenzo's Pizza	Thick Crust
Vincenzo's Pizza	Thin Crust

Delivery Areas By Restaurant

Restaurant	Delivery Area
A1 Pizza	Springfield
A1 Pizza	Shelbyville
A1 Pizza	Capital City
Elite Pizza	Capital City
Vincenzo's Pizza	Springfield
Vincenzo's Pizza	Shelbyville

Normalization using Join dependency

6. Fifth Normal Form (5NF)

(or)

Project-Join Normal Form (PJNF)

Example of Lossless-Join Decomposition

- **Lossless join decomposition**
- Decomposition of $R = (A, B, C)$

$$R_1 = (A, B) \qquad R_2 = (B, C)$$

A	B	C
α	1	A
β	2	B

r

A	B
α	1
β	2

$\Pi_{A,B}(r)$

B	C
1	A
2	B

$\Pi_{B,C}(r)$

$\Pi_A (r) \bowtie \Pi_B (r)$

A	B	C
α	1	A
β	2	B

5NF or PJNF

- Rule

A table is in fifth normal form (5NF) or Project-Join Normal Form (PJNF) if it is in 4NF and it cannot have a lossless decomposition into any number of smaller tables.

5NF or PJNF

- Fifth normal form is satisfied when all tables are broken into as many tables as possible in order to avoid redundancy. Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.

Example

Agent	Company	Product Name
Suneet	ABC	Nut
Suneet	ABC	Screw
Suneet	CDE	Bolt
Raj	ABC	Bolt

After 5NF

P 1

Agent	Company
Suneet	ABC
Suneet	CDE
Raj	ABC

P 2

Agent	Product_name
Suneet	Nut
Suneet	Screw
Suneet	Bolt
Raj	Bolt

Results After Join

Agent	Company	Product_Name
Suneet	ABC	Nut
Suneet	ABC	Screw
Suneet	ABC	Bolt*
Suneet	CDE	Nut*
Suneet	CDE	Screw*
Suneet	CDE	Bolt
Raj	ABC	Bolt

After 5NF

P1

Agent	Company
Suneet	ABC
Suneet	CDE
Raj	ABC

P2

Agent	Product_Name
Suneet	Nut
Suneet	Bolt
Raj	Bolt
Raj	Nut

P3

Company	Product_Name
ABC	Nut
ABC	Bolt
CDE	Bolt

After Performing Join Operation

Agent	Company	Product Name
Suneet	ABC	Nut
Suneet	ABC	Screw
Suneet	CDE	Bolt
Raj	ABC	Bolt

DKNF

- Constraint
- An rule governing static values of an attribute such that we can determine if this constraint is True or False. Example
  - Functional Dependencies
  - Multi-valued Dependencies
  - Inter-relation rules
  - Intra-relation rules

DKNF

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

DKNF

Wealthy Person

Wealthy Person	Wealthy Person Type	Net Worth in Dollars
Steve	Eccentric	124,543,621
Roderick	Evil	6,553,228,893
Katrina	Eccentric	8,829,462,998
Gary	Evil	495,565,211

Wealthiness Status

Status	Minimum	Maximum
Millionaire	1,000,000	999,999,999
Billionaire	1,000,000,000	999,999,999,999

Normal Form - Summary

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and no partial key dependency exists.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A relation is in BCNF, if it is in 3NF & if and only if, every determinant is a candidate key.
4NF	A relation will be in 4NF if it is in Boyce Codd Normal Form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

Thank you!