Infosys®
POWERED BY INTELLECT
DRIVEN BY VALUES

RDBMS-Day2

Logical Database design
Normalization

Education and Research

Recap of Day1 session

- RDBMS handles data in the form of relations, tuples and fields
- Keys identify tuples uniquely
- ER modeling is a diagrammatic representation of the conceptual design of a database
- ER diagrams consist of entity types, relationship types and attributes

Since day 2 is a continuation of day1 content, this recap is done here to maintain the continuity

Logical database design

Process of converting the conceptual model into an equivalent representation in the implementation model (relational/hierarchical/network etc.)

We will focus on the relational model

Relational database design

Convert ER model into relational schema (a specification of the table definitions and their foreign key links)

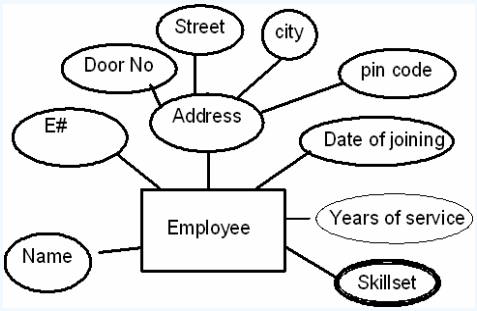
There are well defined rules for this conversion

Converting Strong entity types

- Each **entity type** becomes a **table**
- Each **single-valued attribute** becomes a **column**
- **Derived attributes** are **ignored**
- **Composite attributes** are represented by **components**
- **Multi-valued attributes** are represented by a **separate table**
- The **key attribute** of the entiry type becomes the **primary key** of the table

Entity example

- Here address is a composite attribute
- Years of service is a derived attribute (can be calculated from date of joining and current date)
- Skill set is a multi-valued attribute



- **The relational Schema**

Employee (E#, Name, Door_No, Street, City, Pincode, Date_Of_Joining)

Emp_Skillset(E#, Skillset)

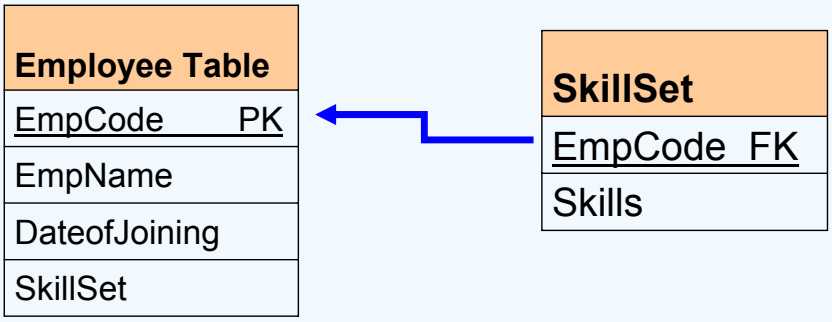
As per the rules:

Derived attributes are **ignored**

Composite attributes are represented by **components**

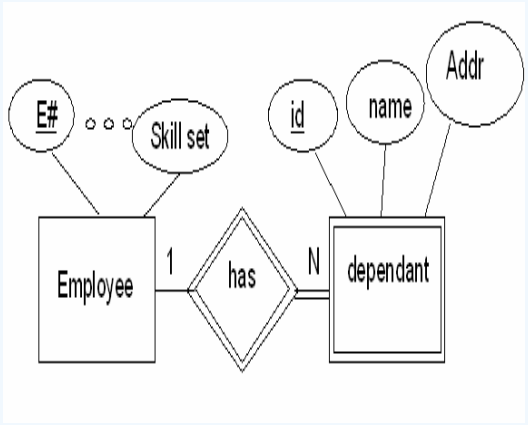
Multi-valued attributes are represented by a **separate table**

Entity Example (Contd…)



Converting weak entity types

- Weak entity types are converted into a table of their own, with the primary key of the strong entity acting as a foreign key in the table
- This foreign key along with the key of the weak entity form the composite primary key of this table
- **The Relational Schema**

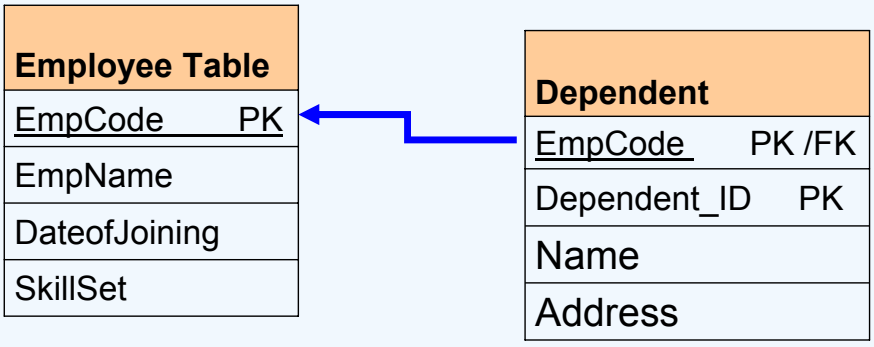


Employee (E# ,…….)

Dependant (Employee, Dependant_ID, Name, Address)

Here dependant is a weak entity. Dependant doesn't mean anything to the problem without the information on for which employee the person is a dependant.

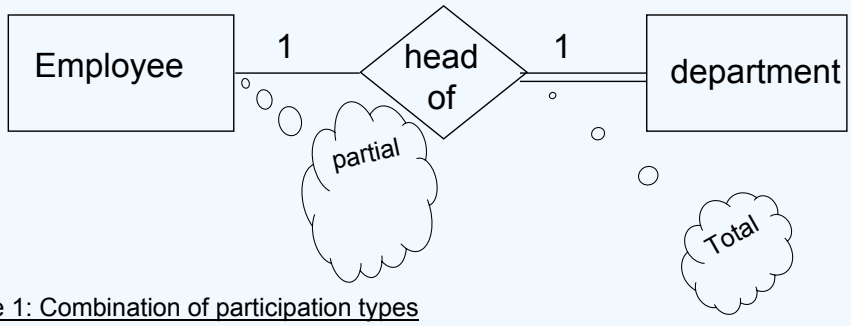
Converting weak entity types (Contd…)



Converting relationships

- The way relationships are represented depends on the cardinality and the degree of the relationship
- The possible cardinalities are:
  1:1, 1:M, N:M
- The degrees are:
  Unary
  Binary
  Ternary …

Binary 1:1



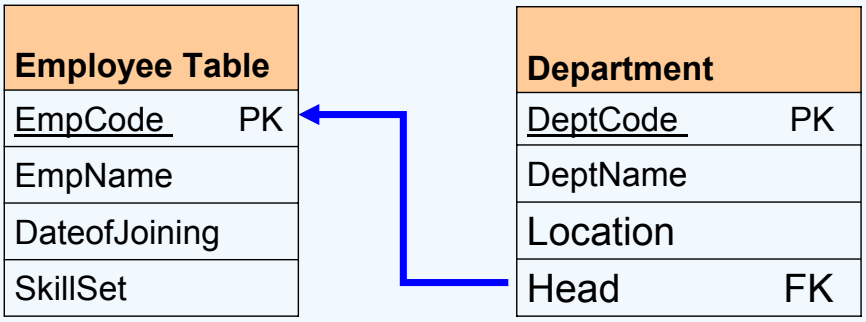
- Case 1: Combination of participation types

  The primary key of the partial participant will become the foreign key of the total participant

  Employee(E#, Name,…)

  Department (Dept#, Name…,Head)

Binary 1 : 1



Binary 1:1

- Case 2: Uniform participation types

The primary key of either of the participants can become a foreign key in the other

Employee (E#,name…)

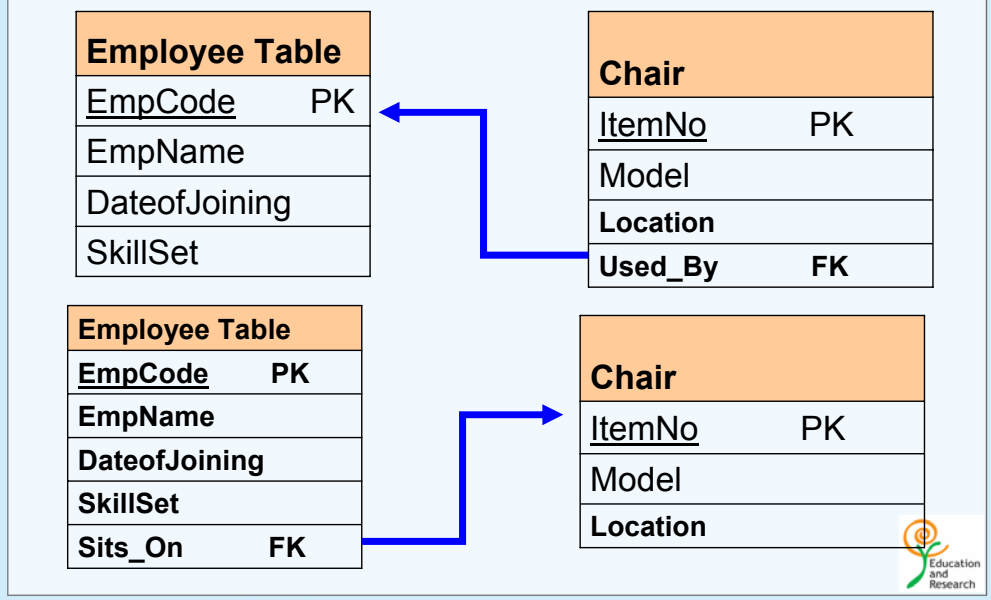
Chair(item#, model, location, used_by)

(or)

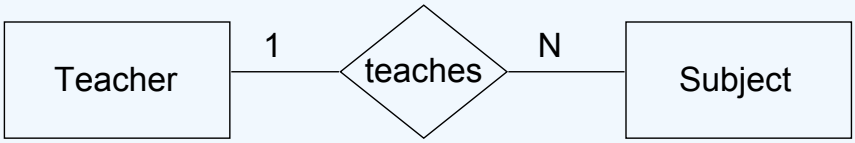
Employee (E#, Name….Sits_on)

Chair (item#,….)

Binary 1 : 1


Employee Table
EmpCode PK
EmpName
DateofJoining
SkillSet
Chair
ItemNo PK
Model
Location
Used_By FK
Employee Table
EmpCode PK
EmpName
DateofJoining
SkillSet
Sits_On FK
Chair
ItemNo PK
Model
Location

Binary 1:N

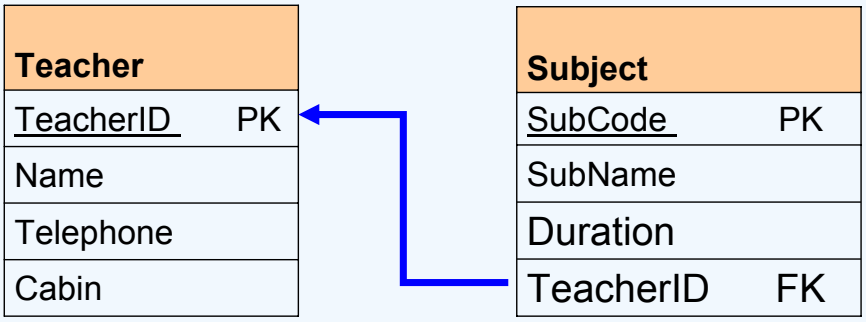


The primary key of the relation on the “1” side of the relationship becomes a foreign key in the relation on the “N” side

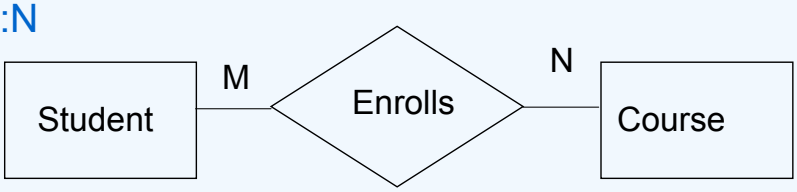
Teacher (ID, Name, Telephone, ...)

Subject (Code, Name, ..., Teacher)

Binary 1 : N



Binary M:N

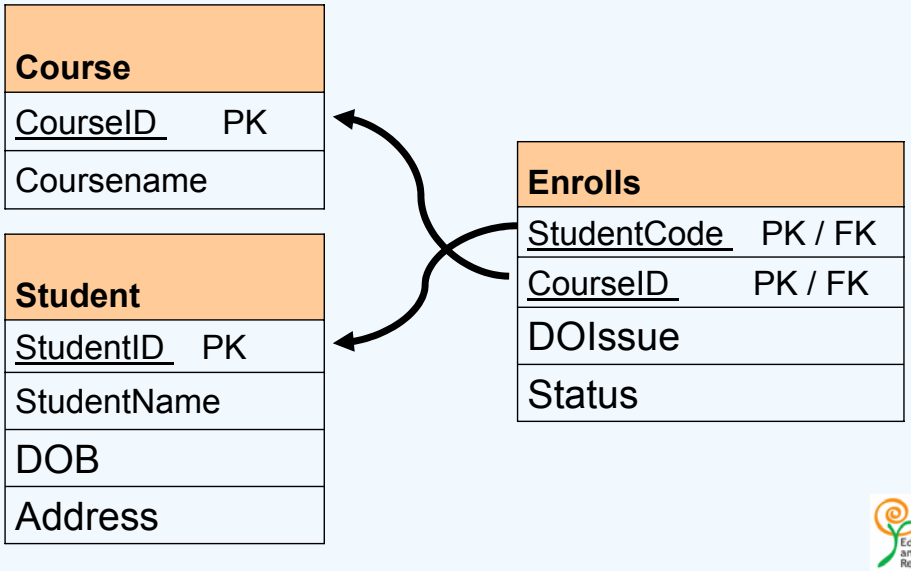


- A new table is created to represent the relationship
- Contains two foreign keys - one from each of the participants in the relationship
- The primary key of the new table is the combination of the two foreign keys

Student (Sid#, Title…) Course(C#, CName,…)

Enrolls (Sid#, C#)

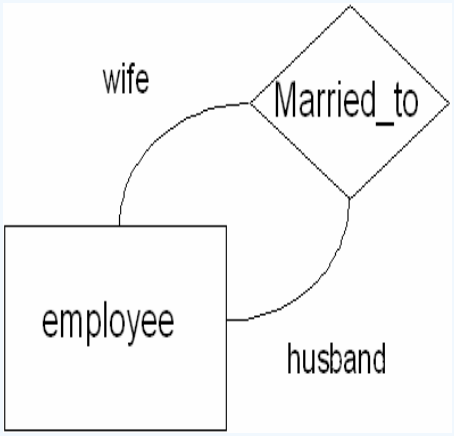
Binary M : N



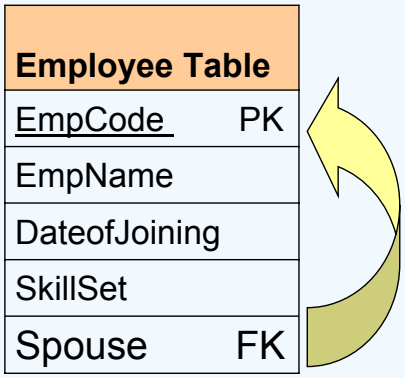
Unary 1:1

- Consider employees who are also a couple
- The primary key field itself will become foreign key in the same table

Employee(E#, Name,... **Spouse**)

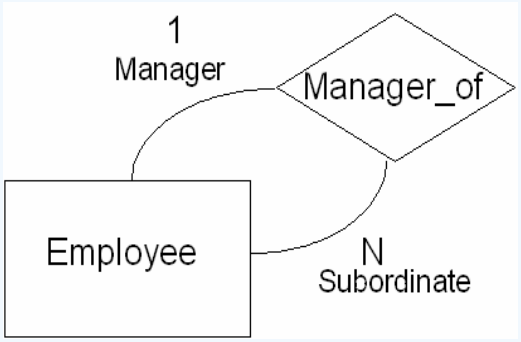


Unary 1 : 1



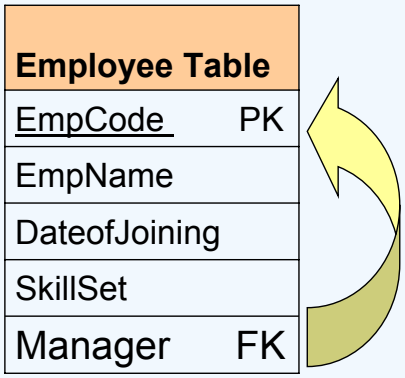
Unary 1:N

- The primary key field itself will become foreign key in the same table
- Same as unary 1:1

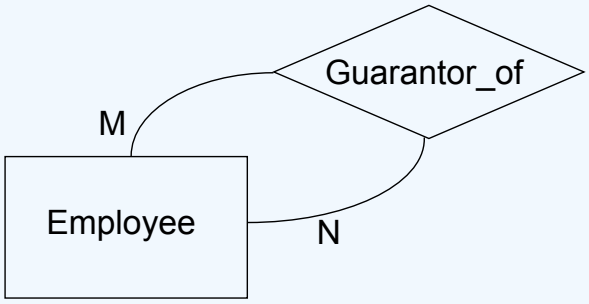


Employee(E#, Name,…,**Manager**)

Unary 1 : N



Unary M:N

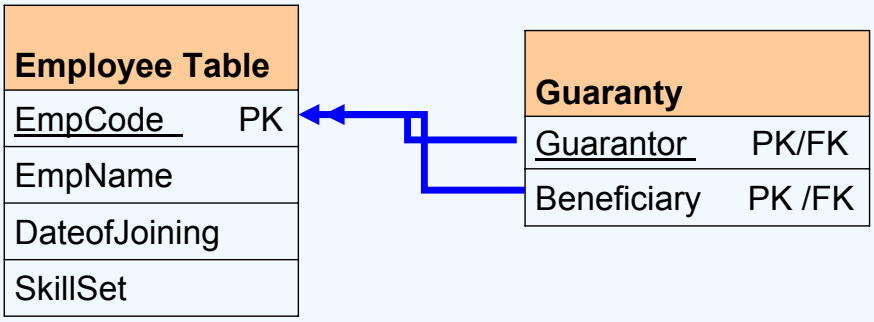


- There will be two resulting tables. One to represent the entity and another to represent the M:N relationship as follows

Employee(E#, Name,…)

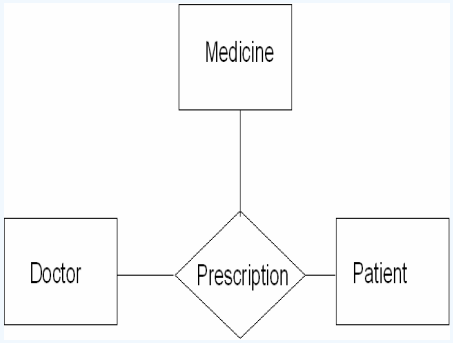
Guaranty(Guarantor, beneficiary)

Unary M : N

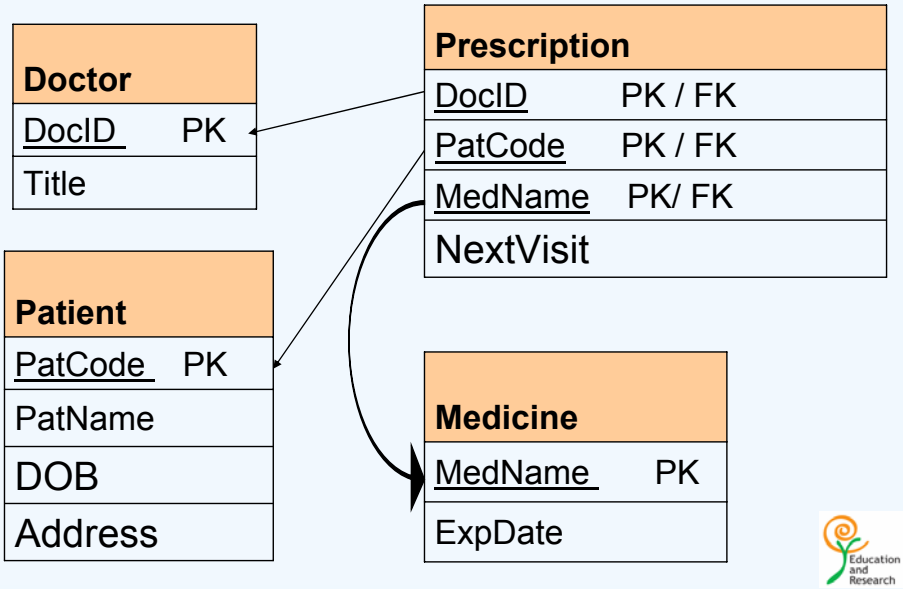


Ternary relationship

- Represented by a new table
- The new table contains three foreign keys - one from each of the participating Entities
- The primary key of the new table is the combination of all three foreign keys
- Prescription (Doctor#, Patient #, Medicine_Name)



Ternary


Doctor
DocID PK
Title
Prescription
DocID PK / FK
PatCode PK / FK
MedName PK/ FK
NextVisit
Patient
PatCode PK
PatName
DOB
Address
Medicine
MedName PK
ExpDate

Deriving Logical Schema for Banking Application

- Each Entity represented in the E-R model can be defined as a table in the relational scheme. All the attributes of the Entity will become columns of the table.

  Example: *Let us consider the CUSTOMER Entity of the banking database scenario. We can translate this Entity to a "CUSTOMER" table with the following columns.*

- **CUSTOMER**
  - (Customer#,Name,Telephone#,Address)

  Example: *Similarly a* ***"Bank"*** *table can be created with Bank Bankcode, Name and Address columns*

- **BANK**
  - (BankCode, Name, Address)

Deriving Logical Schema for Banking Application

- Weak Entity types are converted into a table of their own, with the primary key of the strong Entity acting as a foreign key in the table. This Foreign key along with the key of the Weak Entity form the composite primary key of this table.

  Example: *As per this guideline, a "Branch" table can be created with the following structure.*

- **BRANCH**
  - (BankCode, Branch#, Name, Address)

Deriving Logical Schema for Banking Application

- Each relationship can be defined as separate table in relational schema. Key attributes of *participating entities* will become key attribute of the Relationship.

  Example: *We can define Loan_Detail table with Loan# and Customer# together as primary key with other relevant attributes like DateOfSanction, InterestRate, LoanAmount, Duration etc.*

- **LOAN_DETAILS**
  - (Loan#, Customer#, DateofSanction, InterestRate, LoanAmount, Duration)

Participating entities: The entities which are joined by the relation.

Deriving Logical Schema for Banking Application

- In a Many to Many relationship, it is necessary to create separate tables for participating entities and relationships. In the banking application we have Customer and Loan Entities have a Many to Many relationship. Hence one should create separate tables for CUSTOMER, LOANS and LOAN_DETAILS. Here LOAN_DETAILS refers to relationship table.

Break

Infosys®

POWERED BY INTELLECT
DRIVEN BY VALUES

Normalization

Education and Research

What is Normalization?

- Database designed based on the E-R model may have some amount of
  - Inconsistency
  - Uncertainty
  - Redundancy

To eliminate these draw backs some **refinement** has to be done on the database.

- **Refinement** process is called **Normalization**
- Defined as a step-by-step process of decomposing a complex relation into a simple and stable data structure.
- The formal process that can be followed to achieve a good database design
- Also used to check that an existing design is of good quality
- The different stages of normalization are known as “normal forms”
- To accomplish normalization we need to understand the concept of Functional Dependencies.

Need for Normalization

Student_Course_Result Table

Student_Details			Course_Details				Result_Details		
101	Davis	11/4/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	82	A
102	Daniel	11/6/1987	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	62	C
101	Davis	11/4/1986	H6	American History		4	11/22/2004	79	B
103	Sandra	10/2/1988	C3	Bio Chemistry	Basic Chemistry	11	11/16/2004	65	B
104	Evelyn	2/22/1986	B3	Botany		8	11/26/2004	77	B
102	Daniel	11/6/1987	P3	Nuclear Physics	Basic Physics	13	11/12/2004	68	B
105	Susan	8/31/1985	P3	Nuclear Physics	Basic Physics	13	11/12/2004	89	A
103	Sandra	10/2/1988	B4	Zoology		5	11/27/2004	54	D
105	Susan	8/31/1985	H6	American History		4	11/22/2004	87	A
104	Evelyn	2/22/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	65	B

- Insert Anomaly
- Delete Anomaly
- Update Anomaly
- Data Duplication

Functional dependency

- In a given relation R, X and Y are attributes. Attribute Y is **functionally dependent** on attribute X if each value of X determines **EXACTLY ONE** value of Y, which is represented as X -> Y (X can be composite in nature).
- We say here “x determines y” *or* “y is functionally dependent on x”
  X→Y does not imply Y→X
- If the value of an attribute “Marks” is known then the value of an attribute “Grade” is determined since Marks→ Grade
- Types of functional dependencies:
  - Full Functional dependency
  - Partial Functional dependency
  - Transitive dependency

Functional Dependencies

Consider the following Relation

REPORT (STUDENT#,COURSE#, CourseName, IName, Room#, Marks, Grade)

- **STUDENT#** - Student Number
- **COURSE#** - Course Number
- **CourseName** - Course Name
- **IName** - Name of the Instructor who delivered the course
- **Room#** - Room number which is assigned to respective Instructor
- **Marks** - Scored in Course COURSE# by Student STUDENT#
- **Grade** - obtained by Student STUDENT# in Course COURSE#

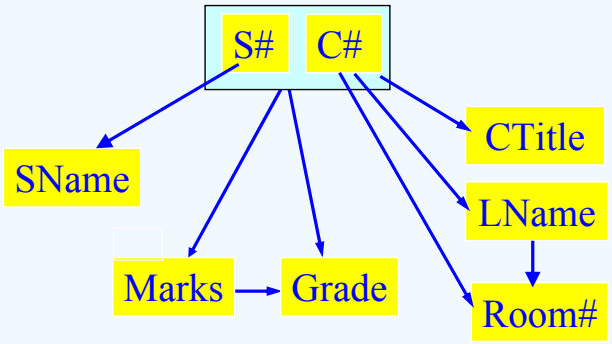
Functional Dependencies- From the previous example

- **STUDENT# COURSE# → Marks**
- **COURSE# → CourseName,**
- **COURSE# → IName (Assuming one course is taught by one and only one Instructor)**
- **IName → Room# (Assuming each Instructor has his/her own and non-shared room)**
- **Marks → Grade**

Dependency diagram

Report(S#,C#,SName,CTitle,LName,Room#,Marks,Grade)

- **S# → SName**
- **C# → CTitle,**
- **C# → LName**
- **LName → Room#**
- **C# → Room#**
- **S# C# → Marks**
- **Marks → Grade**
- **S# C# → Grade**

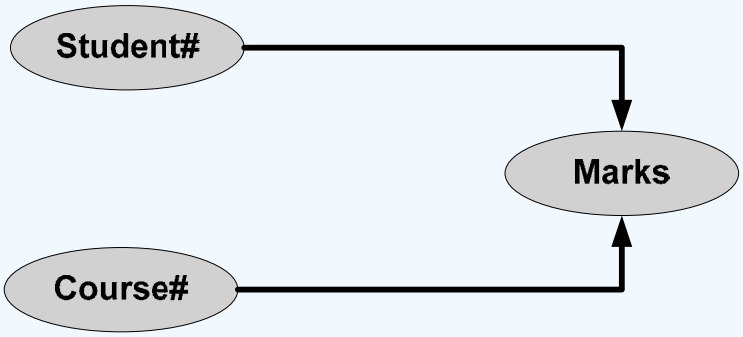


Assumptions:

- Each course has only one lecturer and each lecturer has a room.
- Grade is determined from Marks.

Full dependencies

X and Y are attributes.
X Functionally determines Y
Note: Subset of X should not functionally determine Y

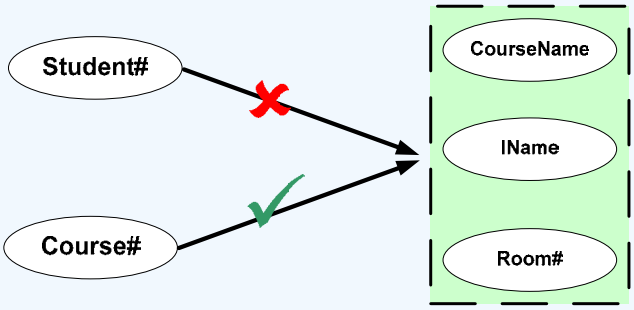


In above example Marks is fully functionally dependent on STUDENT# COURSE# and **not on sub set of** STUDENT# COURSE#. This means Marks can not be determined either by STUDENT# **OR** COURSE# alone. It can be determined only using STUDENT# **AND** COURSE# together. Hence Marks is fully functionally dependent on STUDENT# COURSE#.

CourseName is not fully functionally dependent on STUDENT# COURSE# because subset of STUDENT# COURSE# i.e only COURSE# determines the CourseName and STUDENT# does not have any role in deciding CourseName. Hence CourseName is not fully functionally dependent on STUDENT# COURSE#.

Partial dependencies

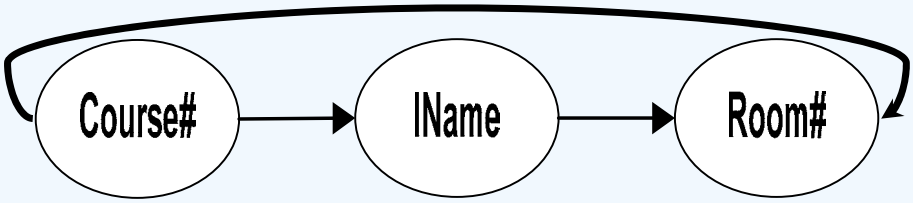
X and Y are attributes.
Attribute Y is partially dependent on the attribute X only if it is dependent on a sub-set of attribute X.



In the above relationship CourseName, IName, Room# are partially dependent on composite attributes STUDENT# COURSE# because COURSE# alone defines the CourseName, IName, Room#.

Transitive dependencies

X Y and Z are three attributes.
X -> Y
Y-> Z
=> X -> Z



In above example, Room# depends on IName and in turn IName depends on COURSE#. Hence Room# transitively depends on COURSE#.

Similarly Grade depends on Marks, in turn Marks depends on STUDENT# COURSE# hence Grade depends Fully *transitively* on STUDENT# COURSE#.

Transitive: Indirect

First normal form: 1NF

- **A relation schema is in 1NF :**
  - if and only if all the attributes of the relation R are atomic in nature.
  - **Atomic:** the smallest level to which data may be broken down and remain meaningful

In relational database design it is not practically possible to have a table which is not in 1NF.

Student_Course_Result Table

Student_Details			Course_Details				Results		
101	Davis	11/4/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	82	A
102	Daniel	11/6/1987	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	62	C
101	Davis	11/4/1986	H6	American History		4	11/22/2004	79	B
103	Sandra	10/2/1988	C3	Bio Chemistry	Basic Chemistry	11	11/16/2004	65	B
104	Evelyn	2/22/1986	B3	Botany		8	11/26/2004	77	B
102	Daniel	11/6/1987	P3	Nuclear Physics	Basic Physics	13	11/12/2004	68	B
105	Susan	8/31/1985	P3	Nuclear Physics	Basic Physics	13	11/12/2004	89	A
103	Sandra	10/2/1988	B4	Zoology		5	11/27/2004	54	D
105	Susan	8/31/1985	H6	American History		4	11/22/2004	87	A
104	Evelyn	2/22/1986	M4	Applied Mathematics	Basic Mathematics	7	11/11/2004	65	B

Table in 1NF

Student_Course_Result Table

Student#	Student Name	Dateof Birth	Course#	CourseName	Pre Requisite	Duration InDays	DateOf Exam	Marks	Grade
101	Davis	04-Nov-1986	M4	Applied Mathematics	Basic Mathematics	7	11-Nov-2004	82	A
102	Daniel	06-Nov-1986	M4	Applied Mathematics	Basic Mathematics	7	11-Nov-2004	62	C
101	Davis	04-Nov-1986	H6	American History		4	22-Nov-2004	79	B
103	Sandra	02-Oct-1988	C3	Bio Chemistry	Basic Chemistry	11	16-Nov-2004	65	B
104	Evelyn	22-Feb-1986	B3	Botany		8	26-Nov-2004	77	B
102	Daniel	06-Nov-1986	P3	Nuclear Physics	Basic Physics	13	12-Nov-2004	68	B
105	Susan	31-Aug-1985	P3	Nuclear Physics	Basic Physics	13	12-Nov-2004	89	A
103	Sandra	02-Oct-1988	B4	Zoology		5	27-Nov-2004	54	D
105	Susan	31-Aug-1985	H6	American History		4	22-Nov-2004	87	A
104	Evelyn	22-Feb-1986	M4	Applied Mathematics	Basic Mathematics	7	11-Nov-2004	65	B

Second normal form: 2NF

- ***A Relation is said to be in Second Normal Form if and only if :***
  - ***It is in the First normal form, and***
  - ***No partial dependency exists between non-key attributes and key attributes.***

- An attribute of a relation R that belongs to any key of R is said to be a prime attribute and that which doesn't is a **non-prime attribute**

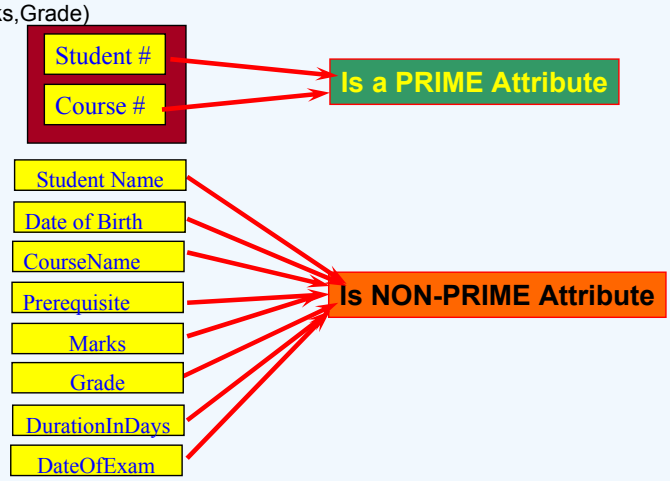
To make a table 2NF compliant, we have to remove all the partial dependencies

Note : - All partial dependencies are eliminated

Prime Vs Non-Prime Attributes

- An attribute of a relation R that belongs to any key of R is said to be a **prime** attribute and that which doesn't is a **non-prime attribute**

Report(**S#,C#,**StudentName,DateOfBirth,CourseName,PreRequisite,DurationInDays,DateOfExam,Marks,Grade)



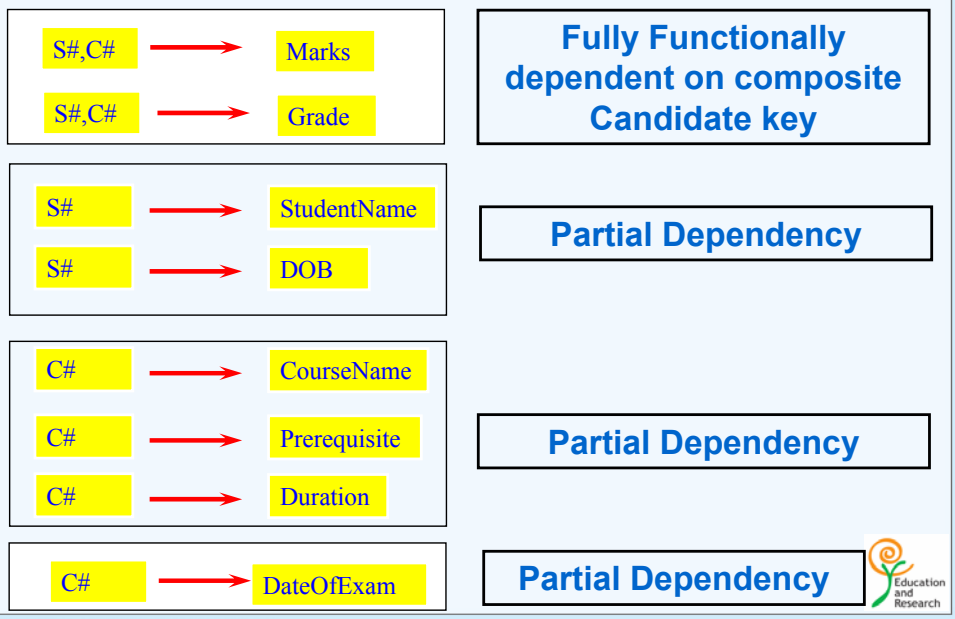
Second Normal Form

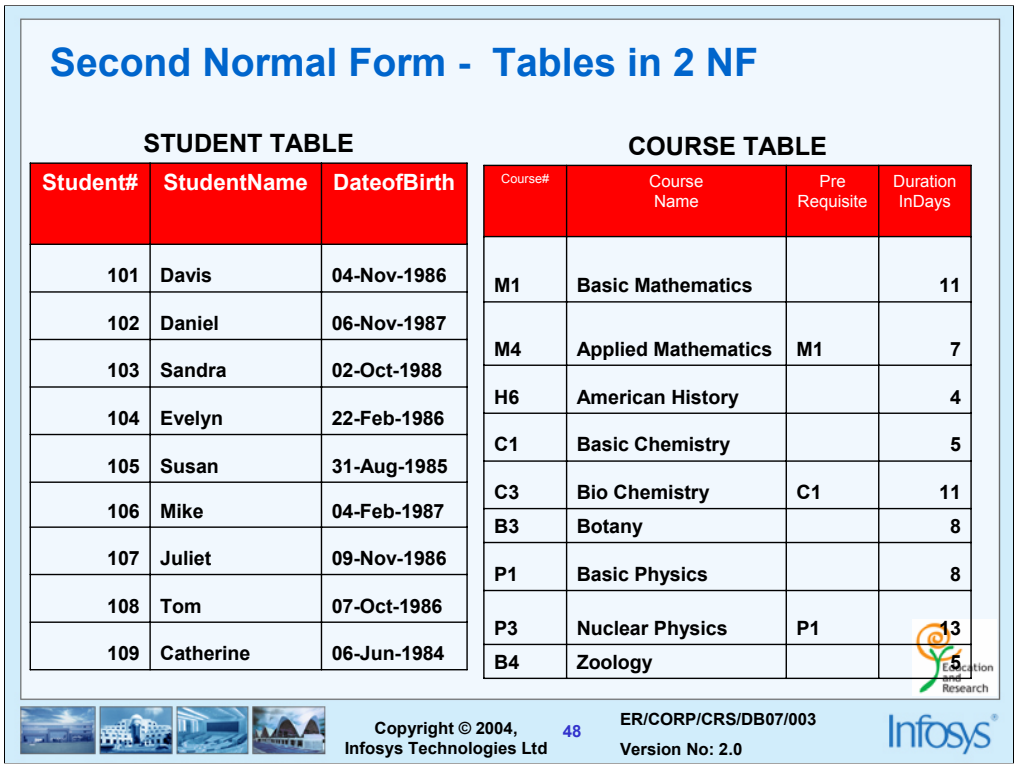
- STUDENT# is key attribute for Student,
- COURSE# is key attribute for Course
- STUDENT# COURSE# together form the composite key attributes for Results relationship.
- Other attributes like StudentName (Student Name), DateofBirth, CourseName, PreRequisite, DurationInDays, DateofExam, Marks and Grade are non-key attributes.

To make this table 2NF compliant, we have to remove all the partial dependencies.

Student #, Course# -> Marks, Grade
Student# -> StudentName, DOB,
Course# -> CourseName, Prerequiste, DurationInDays
Course# -> Date of Exam

Second Normal Form





Second Normal Form - Tables in 2 NF

STUDENT TABLE

Student#	StudentName	DateofBirth
101	Davis	04-Nov-1986
102	Daniel	06-Nov-1987
103	Sandra	02-Oct-1988
104	Evelyn	22-Feb-1986
105	Susan	31-Aug-1985
106	Mike	04-Feb-1987
107	Juliet	09-Nov-1986
108	Tom	07-Oct-1986
109	Catherine	06-Jun-1984

COURSE TABLE

Course#	Course Name	Pre Requisite	Duration InDays
M1	Basic Mathematics		11
M4	Applied Mathematics	M1	7
H6	American History		4
C1	Basic Chemistry		5
C3	Bio Chemistry	C1	11
B3	Botany		8
P1	Basic Physics		8
P3	Nuclear Physics	P1	13
B4	Zoology		5

Let us re-visit our 1NF table structure.

STUDENT# is key attribute for Student,

COURSE# is key attribute for Course

STUDENT# COURSE# together form the composite key attributes for Results relationship.

Other attributes like StudentName (Student Name), DateofBirth, CourseName, PreRequisite, DurationInDays, DateofExam, Marks and Grade are non-key attributes.

To make this table 2NF compliant, we have to remove all the partial dependencies.

StudentName, DateofBirth, Address depends only on STUDENT#

CourseName, PreRequisite, DurationInDays depends only on COURSE#

DateofExam depends only on COURSE#

Marks and Grade depends on STUDENT# COURSE#

To remove this partial dependency we can create four separate tables, Student, Course and Result Exam_Date tables as shown below.

In the first table (STUDENT), the key attribute is STUDENT# and all other non-key attributes are fully functionally dependant on the key attributes.

In the second table (COURSE), COURSE# is the key attribute and all the non-key attributes are fully functionally dependant on the key attributes.

In third table (Result) STUDENT# COURSE# together are key attributes and all other non key attributes Marks and Grade fully functionally dependant on the key attributes.

In the fourth table (Exam_Date), DateOfExam depends only on Course#.

These four tables also are compliant with **First Normal Form** definition. Hence these four tables are in **Second Normal Form (2NF)**.

Second Normal form – Tables in 2 NF

Student#	Course#	Marks	Grade
101	M4	82	A
102	M4	62	C
101	H6	79	B
103	C3	65	B
104	B3	77	B
102	P3	68	B
105	P3	89	A
103	B4	54	D
105	H6	87	A
104	M4	65	B

Second Normal form – Tables in 2 NF

Exam_Date Table

Course#	DateOfExam
M4	11-Nov-04
H6	22-Nov-04
C3	16-Nov-04
B3	26-Nov-04
P3	12-Nov-04
B4	27-Nov-04

Third normal form: 3 NF

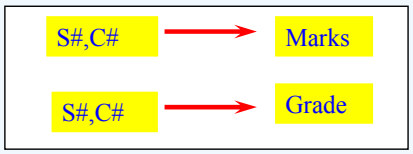
A relation R is said to be in the Third Normal Form (3NF) if and only if

- ***It is in 2NF and***
- ***No transitive dependency exists between non-key attributes and key attributes.***

• STUDENT# and COURSE# are the key attributes.

• All other attributes, except grade are non-partially, non-transitively

dependent on key attributes.



- **Student#, Course# - >** Marks
- **Marks ->** Grade

Note : - All transitive dependencies are eliminated

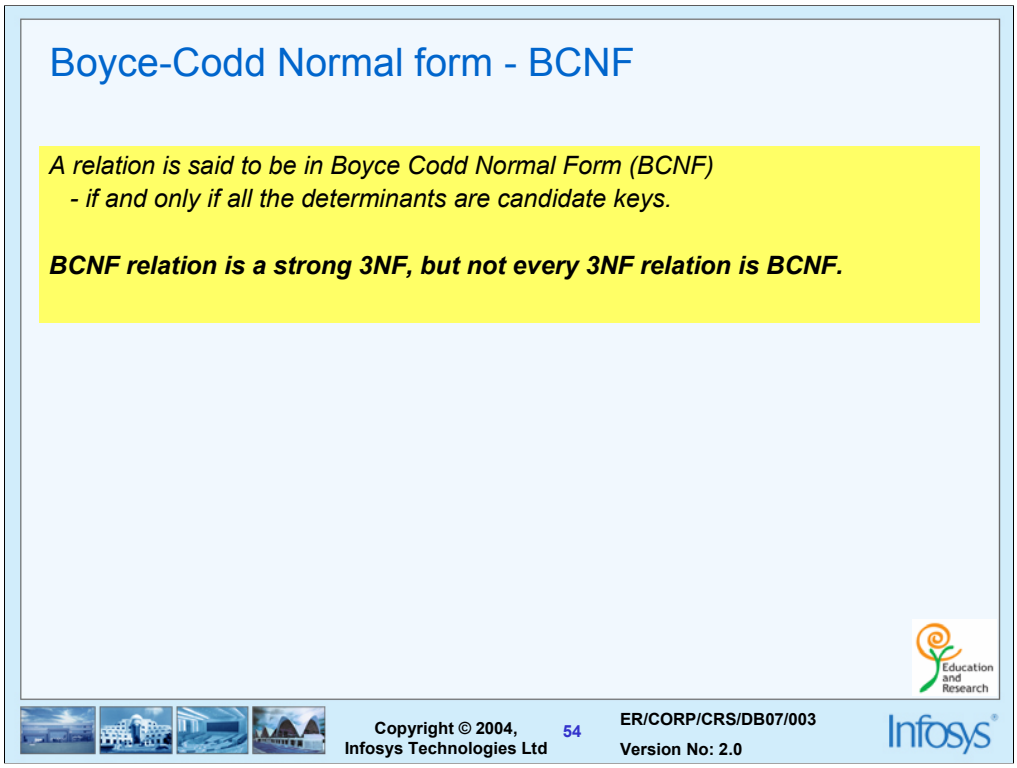
3NF Tables

Student#	Course#	Marks
101	M4	82
102	M4	62
101	H6	79
103	C3	65
104	B3	77
102	P3	68
105	P3	89
103	B4	54
105	H6	87
104	M4	65

Third Normal Form – Tables in 3 NF

MARKSGRADE TABLE

UpperBound	LowerBound	Grade
100	95	A+
94	85	A
84	70	B
69	65	B-
64	55	C
54	45	D
44	0	E



A relation is said to be in Boyce Codd Normal Form (BCNF) if and only if all the determinants are candidate keys. BCNF relation is a strong 3NF, but not every 3NF relation is BCNF.

Let us understand this concept using slightly different RESULT table structure. In the above table, we have two candidate keys namely

STUDENT# COURSE# and **COURSE# EmailId**.

COURSE# is overlapping among those candidate keys.

Hence these candidate keys are called as

“Overlapping Candidate Keys”.

The non-key attributes Marks is non-transitively and fully functionally dependant on key attributes. Hence this is in 3NF. But this is not in BCNF because there are four determinants in this relation namely:

STUDENT# (STUDENT# decides EmailiD)

EMailID (EmailID decides STUDENT#)

STUDENT# COURSE# (decides Marks)

COURSE# EMailID (decides Marks).

All of them are not candidate keys. Only combination of STUDENT# COURSE# and COURSE# EMailID are candidate keys.

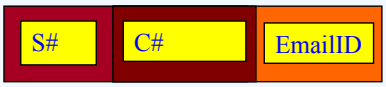
Consider this Result Table

Student#	EmailID	Course#	Marks
101	Davis@myuni.edu	M4	82
102	Daniel@myuni.edu	M4	62
101	Davis@myuni.edu	H6	79
103	Sandra@myuni.edu	C3	65
104	Evelyn@myuni.edu	B3	77
102	Daniel@myuni.edu	P3	68
105	Susan@myuni.edu	P3	89
103	Sandra@myuni.edu	B4	54
105	Susan@myuni.edu	H6	87
104	Evelyn@myuni.edu	M4	65

BCNF

Candidate Keys for the relation are

Since **Course #** is overlapping, it is referred as Overlapping Candidate Key.



Valid Functional Dependendencies are

S#	→	EMailID	**(Non Key Determinant)**
EMailID	→	S#	**(Non Key Determinant)**
S#,C#	→	Marks	
C#,EMailID	→	Marks	

BCNF

STUDENT TABLE

Student#	EmailID
101	Davis@myuni.edu
102	Daniel@myuni.edu
103	Sandra@myuni.edu
104	Evelyn@myuni.edu
105	Susan@myuni.edu

Now both the tables are not only in 3NF, but also in BCNF because all the determinants are Candidate keys. In the first table, STUDENT# decides EMailID and EMailID decides STUDENT# and both are candidate keys. In second table, STUDENT# COURSE# decides all other non-key attributes and they are composite candidate key as well as determinants.

Note: If the table has a single attribute as candidate key or no overlapping candidate keys and if it is in 3NF, then definitely the table will also be in BCNF.

Basically BCNF takes away the redundancy, anomalies which exist among the key attributes. At Infosys, we rarely (around 1% of database design) normalize the databases to BCNF.

BCNF Tables

Student#	Course#	Marks
101	M4	82
102	M4	62
101	H6	79
103	C3	65
104	B3	77
102	P3	68
105	P3	89
103	B4	54
105	H6	87
104	M4	65

Merits of Normalization

- Normalization is based on a mathematical foundation.
- Removes the redundancy to a greater extent. After 3NF, data redundancy is minimized to the extent of foreign keys.
- Removes the anomalies present in INSERTs, UPDATEs and DELETEs.

Demerits of Normalization

- Data retrieval or SELECT operation performance will be severely affected.
- Normalization might not always represent real world scenarios.

Summary of Normal Forms

Input	Operation	Output
Un-normalized Table	Create separate rows or columns for every combination of multivalued columns	Table in 1 NF
Table in 1 NF	Eliminate Partial dependencies	Tables in 2NF
Tables in 2 NF	Eliminate Transitive dependencies	Tables in 3 NF
Tables in 3 NF	Eliminate Overlapping candidate key columns	Tables in BCNF

Points to Remember:

Normal Form	Test	Remedy (Normalization)
1NF	Relation should have atomic attributes. The domain of an attribute must include only atomic (simple, indivisible) values.	Form new relations for each non-atomic attribute
2NF	For relations where primary key contains multiple attributes (composite primary key), non-key attribute should not be functionally dependent on a part of the primary key.	Decompose and form a new relation for each partial key with its dependent attribute(s). Retain the relation with the original primary key and any attributes that are fully functionally dependent on it.
3NF	Relation should not have a non-key attribute functionally determined by another non-key attribute (or by a set of non-key attributes). In other words there should be no transitive dependency of a non-key attribute on the primary key.	Decompose and form a relation that includes the non-key attribute(s) that functionally determine(s) other non-key attribute(s).

Summary

- Normalization is a refinement process. It helps in removing anomalies present in INSERTs/UPDATEs/DELETEs
- Normalization is also called **"Bottom-up approach**", because this technique requires very minute details like every participating attribute and how it is dependant on the key attributes, is crucial. If you add new attributes after normalization, it may change the normal form itself.
- There are four normal forms that were defined being commonly used.
- 1NF makes sure that all the attributes are atomic in nature.
- 2NF removes the partial dependency.

Summary – contd.

- 3NF removes the transitive dependency.
- BCNF removes dependency among key attributes.
- Too much of normalization adversely affects SELECT or RETRIEVAL operations.
- It is always better to normalize to 3NF for INSERT, UPDATE and DELETE intensive (On-line transaction) systems.
- It is always better to restrict to 2NF for SELECT intensive (Reporting) systems.
- While normalizing, use common sense and don't use the normal forms as absolute measures.

Thank You!