

DATABASE MANAGEMENT SYSTEMS

– CSE/IT – II Year / IV Sem. – 22CS403



Manju S,
AP/CoE,
RTC,Coimbatore.

TOPICS

- ❑ **DATABASE DESIGN**
- ❑ Functional Dependencies – Design guidelines – Normal Forms: first, second, third – Boyce/Codd Normal Form – Fourth Normal form -Fifth Normal Form. Design of a banking database system / university database system.

Functional Dependency (FD)

- Functional Dependency is the relationship between attributes(*characteristics*) of a table related to each other.
- A functional dependency is denoted by an arrow “ \rightarrow ”.
- The functional dependency of A on B is represented by $\mathbf{A} \rightarrow \mathbf{B}$, where **A** and **B** are the attributes of the relation.
- Where attributes on left hand side is called as determinants and attributes on RHS are called as dependents.
- A Functionally determines B or B is functionally dependent on A

Functional Dependency (FD)

- Formally, a functional dependency between two sets of attributes in a relational database is expressed as $X \rightarrow Y$, where X and Y are attributes or sets of attributes in a table
- This notation means "if two tuples (rows) have the same value for X , then those two tuples must have the same value for Y ."

$X \rightarrow Y$ holds

X	Y
A	D
A	D
B	N
B	N

$X \rightarrow Y$ doesn't hold as A has 2 values

X	Y
A	D
A	S
B	N
B	N

Functional Dependency (FD)

- Consider a relation with four attributes **A**, **B**, **C** and **D**,
- **R (ABCD)**
 - **$A \rightarrow BCD$**
 - **$B \rightarrow CD$**
 - For the first functional dependency **$A \rightarrow BCD$** , attributes **B**, **C** and **D** are functionally dependent on attribute **A**.
 - Function dependency **$B \rightarrow CD$** has two attributes **C** and **D** functionally depending upon attribute **B**.

Examples of Functional Dependency:

- ❑ **Employee Database:** In an employee table, if each employee has a unique employee ID, then the employee ID can determine other attributes like Name, Department, and Salary.
 - $\text{EmployeeID} \rightarrow \text{Name, Department, Salary}$
- ❑ **Student Database:** In a university database, if each student is assigned a unique student ID, the student ID can determine other details such as Student Name, Course Enrolled.
 - $\text{StudentID} \rightarrow \text{StudentName, CourseEnrolled}$

Functional Dependency (FD) : Types

1. Trivial Functional Dependency:

- ❑ A functional dependency $X \rightarrow Y$ is called trivial if Y is a subset of X or $Y=X$.
- ❑ **Example:** In a table with attributes EmployeeID, Name, DepartmentEmployeeID, Salary
- ❑ EmployeeID, Name \rightarrow Name is trivial because Name is a subset of the left-hand side attributes

Functional Dependency (FD) : Types

2. Non-trivial Functional Dependency:

- ❑ A functional dependency is considered non-trivial if Y is not a subset of X in $X \rightarrow Y$
- ❑ **Example:** In the same employee table,
- ❑ $\text{EmployeeID} \rightarrow \text{Name, Department}$ is non-trivial because neither Name nor Department is a subset of EmployeeID

Functional Dependency (FD) : Types

3. Transitive Functional Dependency:

- A functional dependency $X \rightarrow Y$ is transitive if there is an intermediate set of attributes Z such that $X \rightarrow Z$ and $Z \rightarrow Y$
- **Example:** If in a table, $\text{EmployeeID} \rightarrow \text{DepartmentID}$ and $\text{DepartmentID} \rightarrow \text{DepartmentName}$ then $\text{EmployeeID} \rightarrow \text{DepartmentName}$ is a transitive dependency.

Functional Dependency (FD) : Types

4. Multivalued dependency

- ❑ In Multivalued functional dependency, attributes in the dependent set are not dependent on each other.
- ❑ For example, $X \rightarrow \{ Y, Z \}$, if there exists is no functional dependency between Y and Z, then it is called as *Multivalued functional dependency*.
- ❑ Consider a schema emp having attributes (Employee_id, Name, Age)
- ❑ Here, $\{ \text{Employee_Id} \} \rightarrow \{ \text{Name}, \text{Age} \}$ is a Multivalued functional dependency, since the dependent attributes **Name, Age** are not *functionally dependent* (i.e. **Name \rightarrow Age or Age \rightarrow Name doesn't exist !**).

Armstrong's axioms / Properties of FD

- William Armstrong in 1974 suggested a few rules related to functional dependency. They are called **RAT** rules.
- **1. Reflexivity**: If **A** is a set of attributes and **B** is a subset of **A**, then the functional dependency $A \rightarrow B$ holds true.
 - *For example, { Employee_Id, Name } \rightarrow Name is valid.*

Armstrong's axioms / Properties of FD

- **2. Augmentation:** If a functional dependency $A \rightarrow B$ holds true, then appending any number of the attribute to both sides of dependency doesn't affect the dependency. It remains true.
 - *For example, $X \rightarrow Y$ holds true then, $ZX \rightarrow ZY$ also holds true.*
 - *For example, if $\{ \text{Employee_Id, Name} \} \rightarrow \{ \text{Name} \}$ holds true then, $\{ \text{Employee_Id, Name, Age} \} \rightarrow \{ \text{Name, Age} \}$*

Armstrong's axioms / Properties of FD

- **3. Transitivity:** If two functional dependencies $X \rightarrow Y$ and $Y \rightarrow Z$ hold true, then $X \rightarrow Z$ also holds true by the rule of Transitivity.
 - *For example, if $\{ \text{Employee_Id} \} \rightarrow \{ \text{Name} \}$ holds true and $\{ \text{Name} \} \rightarrow \{ \text{Department} \}$ holds true, then $\{ \text{Employee_Id} \} \rightarrow \{ \text{Department} \}$ also holds true.*

Armstrong's axioms / Properties of FD

□ 4. Union

- If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow Y, Z$ *holds*. This property allows combining two functional dependencies with the same determinant into a single dependency.
- **Example:** If $\{\text{Employee_ID}\} \rightarrow \{\text{Name}\}$ and $\{\text{Employee_ID}\} \rightarrow \{\text{Department}\}$, then $\{\text{Employee_ID}\} \rightarrow \{\text{Name}, \text{Department}\}$ holds.

Armstrong's axioms / Properties of FD

□ 5. Decomposition

- If $X \rightarrow Y, Z$, then $X \rightarrow Y$ and $X \rightarrow Z$ *holds individually*. This property allows splitting a functional dependency into multiple dependencies based on the components of the dependent attributes.
- **Example:** If $\{\text{Employee_ID}\} \rightarrow \{\text{Name}, \text{Department}\}$, then $\{\text{Employee_ID}\} \rightarrow \{\text{Name}\}$ and $\{\text{Employee_ID}\} \rightarrow \{\text{Department}\}$ holds

Armstrong's axioms / Properties of FD

❑ 6. Pseudo-transitivity

- ❑ If $X \rightarrow Y$ and $Y, W \rightarrow Z$, then $X, W \rightarrow Z$ *holds*. This property allows combining functional dependencies when extra attributes are involved on the right-hand side of the second dependency.
- ❑ **Example:** If $\{\text{Employee_ID}\} \rightarrow \{\text{Manager_ID}\}$ and $\{\text{Manager_ID}, \text{Department}\} \rightarrow \{\text{Department_Budget}\}$, then $\{\text{Employee_ID}, \text{Department}\} \rightarrow \{\text{Department_Budget}\}$ holds.

Normalization

- ❑ Normalization is the process of organizing the data and the attributes of a database.
- ❑ It is performed to reduce the data redundancy in a database and to ensure that data is stored logically. (same data but at multiple places).
- ❑ It is necessary to remove data redundancy because it causes anomalies in a database which makes it very hard for a database administrator to maintain it.

Normalization : Definition

- ❑ DBMS Normalization is a systematic approach to **decompose (break down) tables** to eliminate data redundancy(repetition) and undesirable characteristics like
 - Insertion anomaly in DBMS,
 - Update anomaly in DBMS, and
 - Delete anomaly in DBMS

Why normalization ?

- ❑ Eliminating redundant(useless) data, therefore handling **data integrity**, because if data is repeated it increases the chances of inconsistent data.
- ❑ Normalization helps in keeping **data consistent** by storing the data in one table and referencing it everywhere else.
- ❑ Storage optimization although that is not an issue these days because Database storage is economical.
- ❑ Breaking down large tables into smaller tables with relationships, so it makes the database structure more scalable and adaptable.
- ❑ Ensuring data dependencies make sense i.e. data is logically stored.

Why normalization ? Avoiding anomalies

- ❑ It helps with reducing the data redundancy
- ❑ A database anomaly is a flaw in the database that occurs because of poor planning and redundancy.
- ❑ **Insertion anomalies:** This occurs when we are not able to insert data into a database because some attributes may be missing at the time of insertion or inserting same data again and again.
- ❑ **Updation anomalies:** This occurs when the same data items are repeated with the same values and are not linked to each other. Multiple updates required.
- ❑ **Deletion anomalies:** This occurs when deleting one part of the data deletes the other necessary information from the database itself

Problems without Normalization : Redundancy

- Assume the following schema Student

rollno	name	branch	hod	office_tel
401	Akon	CSE	Mr. X	53337
402	Bkon	CSE	Mr. X	53337
403	Ckon	CSE	Mr. X	53337
404	Dkon	CSE	Mr. X	53337

- Data for the fields **branch**, **hod**(Head of Department), and **office_tel** are repeated for the students who are in the same branch in the college, this is **Data Redundancy**.

Problems without Normalization- Insertion Anomaly

- ❑ Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as **NULL**.
- ❑ Also, if we have to insert data for 100 students of the same branch, then the branch information will be repeated for all those 100 students.
- ❑ Solution : If you have to repeat the same data in every row of data, it's better to **keep the data separately** and **reference that data** in each row. Split tables using normalization !

Problems without Normalization -Updation anomaly

- ❑ What if Mr. X leaves the college? or Mr. X is no longer the HOD of the computer science department? In that case, all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency.
- ❑ This is an Updation anomaly because we need to update all the records in our table just because one piece of information got changed
- ❑ Solution : Split tables using Normalization !

Problems without Normalization -Deletion Anomaly

- ❑ In our **Student** table, two different pieces of information are kept together, the **Student information** and the **Branch information**.
- ❑ So if only a single student is enrolled in a branch, and that student leaves the college, or for some reason, the entry for the student is deleted, we will lose the branch information too.
- ❑ Solution : So never in DBMS, we should keep two different entities together, which in the above example is Student and branch. Split tables using normalization!

Solutions for all anomalies ?!

- ❑ Splitting tables using normalization and separating entities would help in reducing redundancy and avoiding anomalies!
- ❑ Student and branch entities are split into 2 tables respectively !
- ❑ Avoids redundancy and helps with dealing anomalies!

Student

rollno	name	branch
401	Akon	CSE
402	Bkon	CSE
403	Ckon	CSE
404	Dkon	CSE

Branch

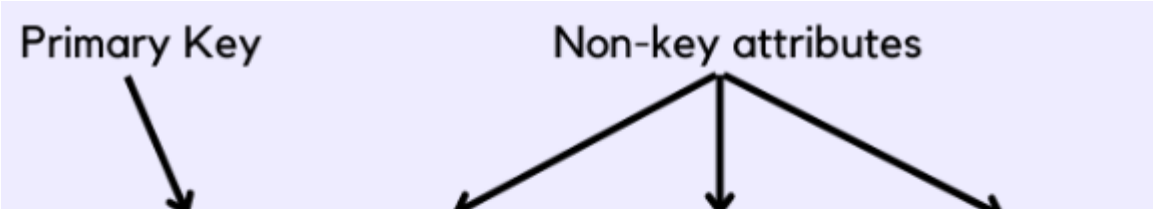
branch	hod	office_tel
CSE	Mr. X	53337

Can you think of even better design ?

Shall have a Branch id and Link tables using PK AND FK ?

Just to refresh !

- ❑ What is a Primary key?
- ❑ What is meant by prime attributes?
- ❑ What are non prime attributes or non key attributes?



student_id	student_name	mobile	gender
1	John	9797979797	Male
2	Ron	7878787878	Male
3	Pom	8282828282	Female

Normal Forms : Types

- ❑ Normalization rules are divided into the following normal forms:
 - First Normal Form : No multivalued attributes
 - Second Normal Form : No Partial Dependencies
 - Third Normal Form : No transitive dependencies
 - BCNF : functional dependency ($X \rightarrow Y$), X should be a Super Key.
 - Fourth Normal Form : No multivalued dependencies
 - Fifth Normal Form : No join dependencies

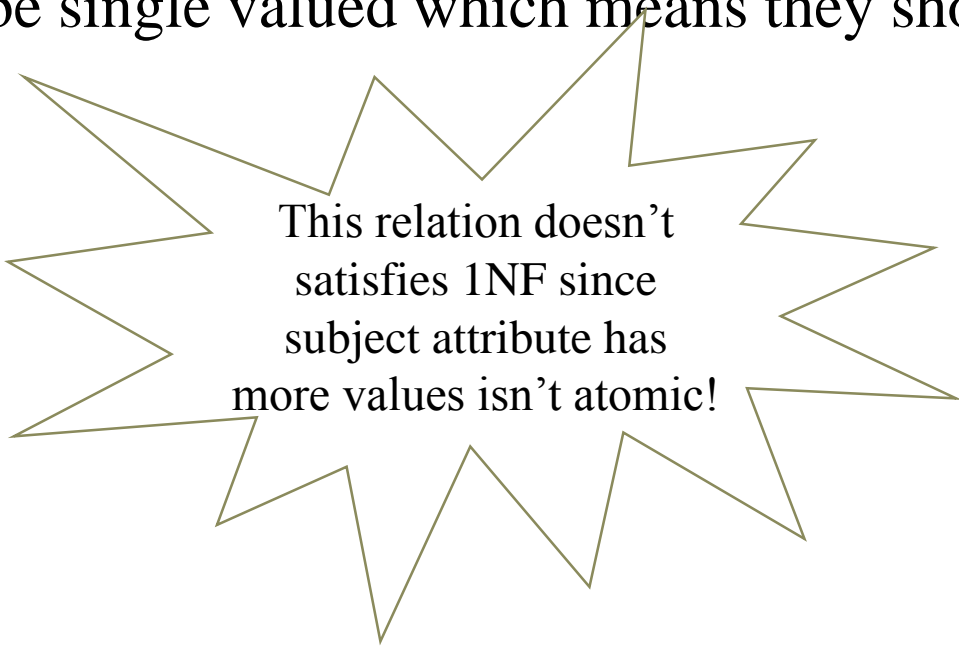
First Normal Form : 1NF

□ Rules for a relation to be in 1NF:

1. Single Valued Attributes / No multivalued Attributes / Atomic Values to be present!

- Each column of our table should be single valued which means they should not contain multiple values

roll_no	name	subject
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++



This relation doesn't satisfy 1NF since subject attribute has more values isn't atomic!

First Normal Form

2. Attribute Domain/Values should not change

- E.g. : DOB column if defined as Date, then all rows should have only Date values!

3. Unique name for Attributes/Columns

- No 2 attributes can have same name which avoids confusion in fetching data

4. Order doesn't matters

- Data will be fetched properly when condition is specified properly, order doesn't matter!

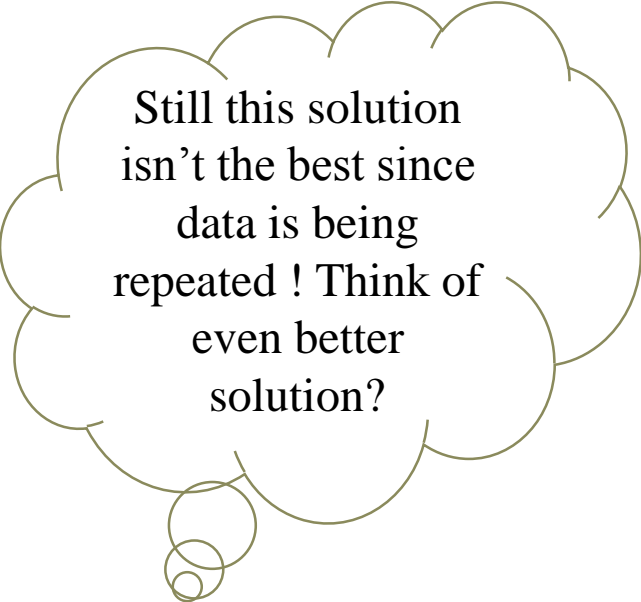
roll_no	name	subject	dob	name
101	Akon	OS, CN	March 2,1990	CSE
103	Ckon	Java	8/4/1993	CSE
102	Bkon	C, C++	9 th April 2000	CSE

Usually 2,3,and 4th rules of 1NF are ensured while creating schema itself which DBMS verifies upon create commands! Rule 1 is to be ensured .

Solution to First Normal Form!

- ❑ Column subject is a multivalued attribute! Its values aren't atomic!
- ❑ Solution shall be,

roll_no	name	subject	Dob	branchname
101	Akon	OS	2/3/1990	CSE
101	Akon	CN	2/3/1990	CSE
103	Ckon	Java	8/4/1993	CSE
102	Bkon	C	9/4/2000	CSE
102	Bkon	C++	9/4/2000	CSE



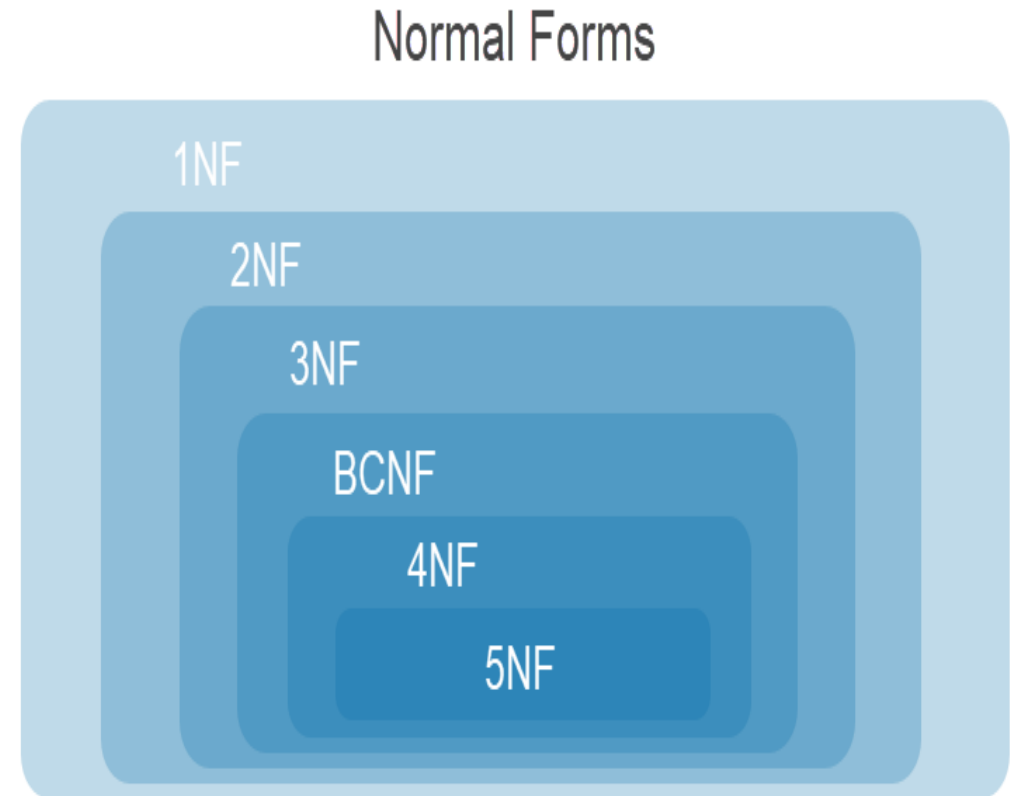
Still this solution isn't the best since data is being repeated ! Think of even better solution?

Splitting tables shall be the best solution ! Remember splitting phone number attribute to a new table using FK reference?! Different entities different tables!

Second Normal Form : 2NF

- Rules for a relation to be in 2NF:
 - It should be in 1NF
 - It shouldn't have partial functional dependencies

For a table or relation to be in higher normal forms, it should have satisfied lower normal form rules as well!



FFD Vs. PFD

- FFD (Fully Functionally Dependant) : if the dependent attribute is functionally dependent on the entire set of attributes in the determinant and not on any proper subset of it.
 - $\{A,B\} \rightarrow \{X\}$: X should depend on both A and B i.e. to identify X either A or B shouldn't be enough. Both A and B should be needed
- PFD (Partially Functionally Dependant) : partial dependency occurs when a non-primary-key attribute is functionally dependent on part of a composite primary key (a primary key consisting of more than one attribute), but not on the whole Primary key.
 - $\{A,B\} \rightarrow \{X\}$: If X can be found with either A or B then the FD is partial. X doesn't depend on the entire PK set.

Schema for understanding partial dependency!

Student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat

subject_id	subject_name
1	Java
2	C++
3	Php

score_id	student_id	subject_id	marks	teacher
1	10	1	70	Java Teacher
2	10	2	75	C++ Teacher
3	11	1	80	Java Teacher

- **Student** has student_id as Primary Key
- **Subject** has subject_id and subject_name fields and subject_id will be the primary key
- **Score**, to store the marks obtained by students in the respective subjects has teacher column who teaches that subject along with marks . Student id and subject id is the primary key / candidate key which finds a students' mark in a subject!

What's partial dependency?

- ❑ `student_id` + `subject_id` forms a Candidate Key in score table
- ❑ Score table, has column name `teacher` which is only dependent on the subject, for Java it's Java Teacher and for C++ it's C++ Teacher & so on.
- ❑ Teacher's name only depends on subject, hence the `subject_id`, and has nothing to do with `student_id`.
- ❑ $\{\text{subject_id}\} \rightarrow \{\text{teacher}\}$: Partial dependency as it doesn't depend on other part of PK i.e. `student_id`.
- ❑ By the rule of 2NF, All attributes should be fully functionally dependent on Primary key/ Composite Key which is not satisfied!

Removing Partial Dependencies to be in 2NF

- ❑ Solution is to remove teacher column and to add in subject table!

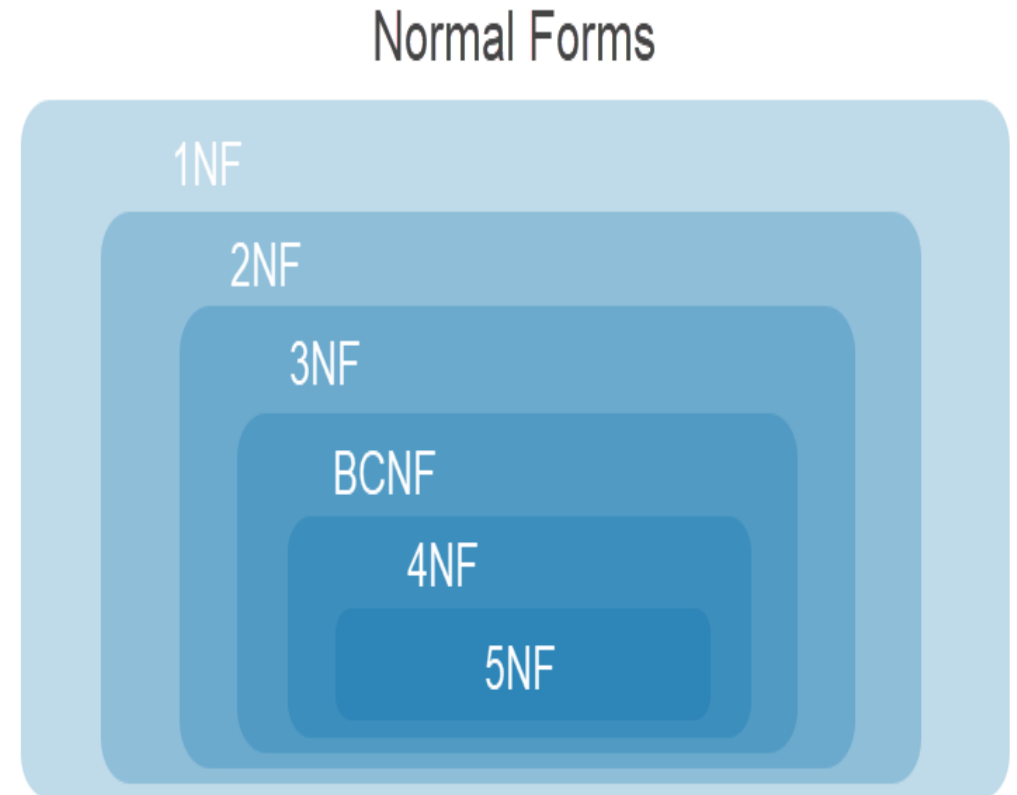
subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	75
3	11	1	80

- ❑ In few other examples, tables shall also be split to make relations in 2NF!

Third Normal Form : 3NF

- Rules for a relation to be in 3NF:
 - It should be in 2NF
 - (Its clear that for a relation to be in 2NF it should be in 1NF)
 - It shouldn't have transitive functional dependencies



Third Normal Form : 3NF

- ❑ Add 2 more columns exam type and total marks in score table
- ❑ Exam_type signifies if the exam is theory or workshop or mains or practical or preliminary etc.,

score_id	student_id	subject_id	marks	teacher	Exam_type	Total_marks
1	10	1	70	Java Teacher		
2	10	2	75	C++ Teacher		
3	11	1	80	Java Teacher		

- ❑ Exam_type depends on the primary key (StudentId +SubjectID)
- ❑ But total_marks depends on exam_type which is a non prime attribute here i.e whats the total marks of a student in a subject in his theory type? Exam_type is also needed right?

Transitive dependency ?!

- $\{\text{Student_id}, \text{Subject_id}\} \rightarrow \text{exam_type}$
- But $\{\text{exam_type}\} \rightarrow \text{totalmarks}$ also exists which is observed to be transitive.
- i.e. $X \rightarrow Y$ and $Y \rightarrow Z$ holds then $X \rightarrow Z$ is transitive
- Transitive Dependency is when a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

Removing Transitive Dependency

- Solution : Split the columns exam_type and total_marks into a new table exam and link it to the score table with some Fk 'exam_id'

score_id	student_id	subject_id	marks	exam_id
1	10	1	70	1
2	10	2	75	2
3	11	1	55	3
4	11	2	57	3

exam_id	exam_name	total_marks
1	Workshop	200
2	Mains	70
3	Practicals	30

To do !

- ❑ Assume the following schema and make sure the relations is in 3NF.

OrderID	CustomerName	CustomerEmail	ProductID	ProductName	ProductPrice	Quantity	OrderDate
1	John Doe	johndoe@example.com	P001	T-shirt	20	2	2023-04-10
2	Jane Smith	janesmith@example.com	P002	Jeans	50	1	2023-04-11
1	John Doe	johndoe@example.com	P003	Cap	15	1	2023-04-10
3	Alice Johnson	alicej@example.com	P001	T-shirt	20	3	2023-04-12

Solution !

- ❑ PK : (OrderID and ProductID)
- ❑ It is in 1NF since all are atomic values
- ❑ It is not in 2NF since both Product details and Customer details are not fully dependent on the composite key (OrderID, ProductID) .
- ❑ So split it to 4 tables
 - Orders Table, OrderDetails Table, Products Table, Customers Table

Normalized Tables!

Customers Table

CustomerID	CustomerName	CustomerEmail
C001	John Doe	johndoe@example.com
C002	Jane Smith	janesmith@example.com
C003	Alice Johnson	alicej@example.com

Products Table

ProductID	ProductName	ProductPrice
P001	T-shirt	20
P002	Jeans	50
P003	Cap	15

Orders Table

OrderID	CustomerID	OrderDate
1	C001	2023-04-10
2	C002	2023-04-11
3	C003	2023-04-12

OrderDetails Table

OrderID	ProductID	Quantity
1	P001	2
1	P003	1
2	P002	1
3	P001	3

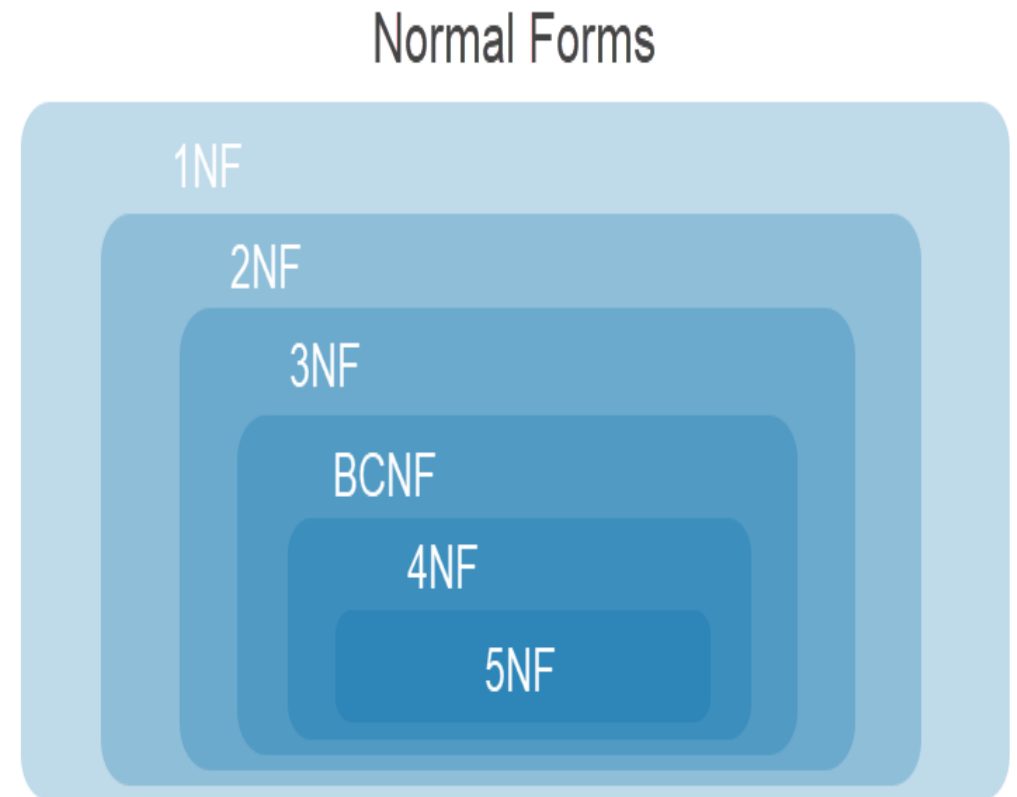
Try it yourself!

- Assume the following Schema and make the relation fall until 3NF.

DeptID	DeptName	Projects	Employees
D01	Engineering	[Alpha, Beta, Gamma]	[Alice, Bob, Charlie]
D02	Marketing	[Delta, Epsilon]	[David, Eve]
D03	HR	[Zeta]	[Frank]

BCNF : Boyce Codd Normal Form

- ❑ Rule for relation to be in Boyce-Codd Normal Form:
 - It should be in the **Third Normal Form**.
 - And, for any dependency $A \rightarrow B$, A should be a **super key**.
- ❑ It is a stricter version of 3NF



BCNF : Boyce Codd Normal Form

- ❑ Consider a student enrollment table
- ❑ Which can act as a primary key and identify the FD?

Student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- ❑ One student can enroll for multiple subjects. For example, student with student_id 101, has opted for subjects - Java & C++
- ❑ For each subject, a professor is assigned to the student.
- ❑ And, there can be multiple professors teaching one subject like we have for Java.
- ❑ student_id, subject together form the primary key, because using student_id and subject, we can find all the columns of the table.
- ❑ one professor teaches only one subject, but one subject may have two different professors.
- ❑ there is a dependency between subject and professor here, where subject depends on the professor name.
- ❑ Professor \rightarrow subject

BCNF : Boyce Codd Normal Form : FD's

- ❑ Studentid,subject→professor
- ❑ Professor→subject
- ❑ The table is in 1NF,2NF,3NF!
- ❑ What about BCNF?
- ❑ It has a FD professor→subject where professor is not a super key i.e. a non prime attribute which is not allowed by BCNF!
- ❑ To remove it, lets split tables!

BCNF : Satisfying BCNF

- ❑ Splitting the student enrollment table into 2 tables
student_details and professor_details

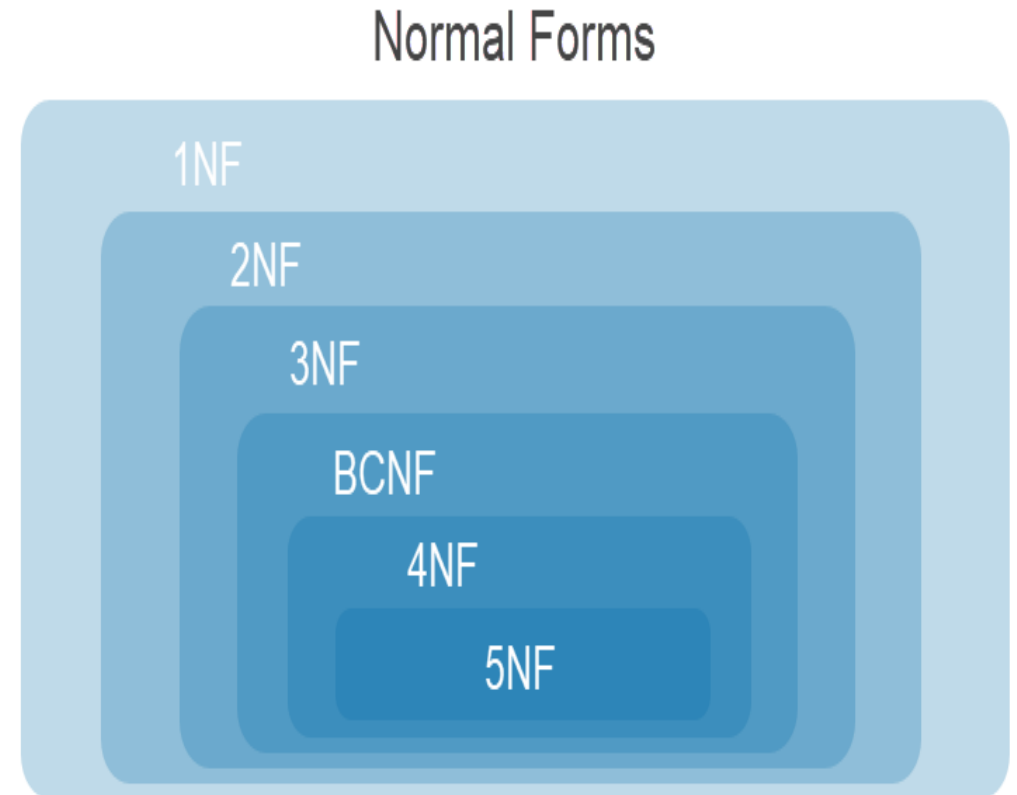
student_id	p_id
101	1
101	2

p_id	professor	subject
1	P.Java	Java
2	P.Cpp	C++

- ❑ The decomposed tables satisfy BCNF

Fourth Normal Form : 4NF

- Rules for a relation to be in 4NF:
 - It should be in BCNF
 - the table should not have any **Multi-valued Dependency**.



4NF: Multi Valued Dependency

- A table is said to have multi-valued dependency, if the following conditions are true,
 - For a dependency $A \twoheadrightarrow B$, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.
 - Also, a table should have at-least 3 columns for it to have a multi-valued dependency.
 - And, for a relation $R(A,B,C)$, if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.
- If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

4NF : Example : MVD

- ❑ College enrolment table with columns s_id, course and hobby.
- ❑ Student with s_id **1** has opted for two courses, **Science** and **Maths**, and has two hobbies, **Cricket** and **Hockey**.

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey

- ❑ What problem this can lead to?
- ❑ Well the two records for student with s_id 1, will give rise to two more records, as shown below, because for one student, two hobbies exists, hence along with both the courses, these hobbies should be specified. When join is performed 4 rows will get generated for a single student!

S_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

4NF : Removing Multi Valued Dependency

- ❑ Decompose the table into 2 tables student_course and student_hobby

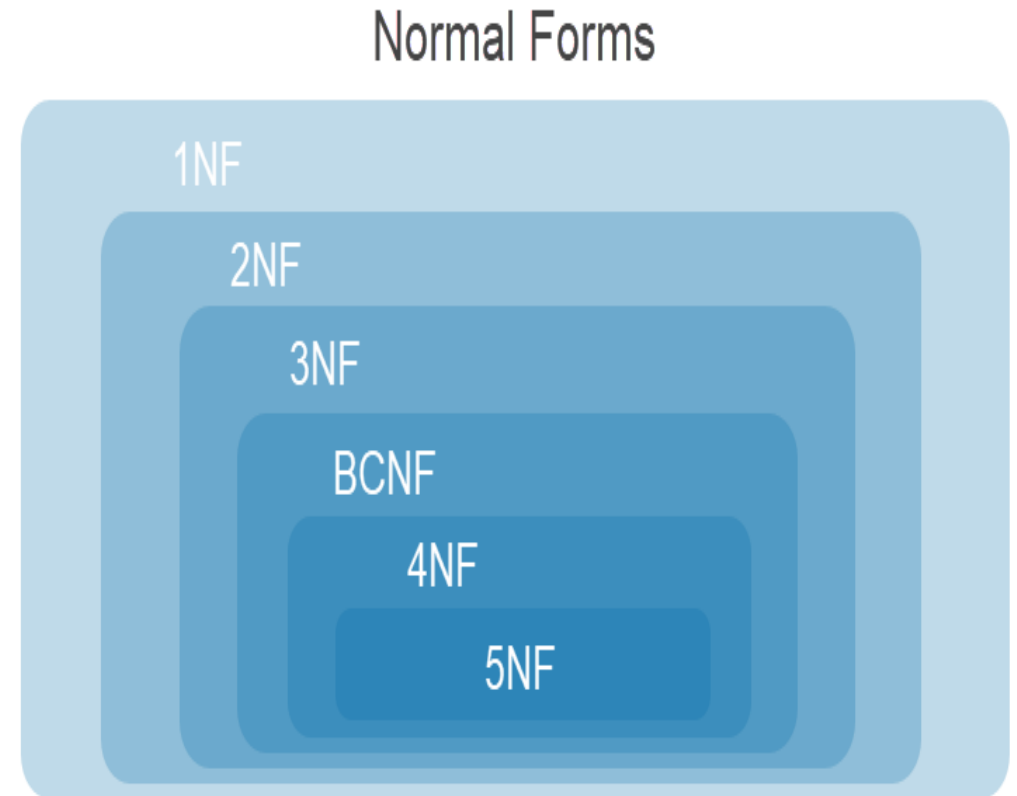
s_id	course
1	Science
1	Maths
2	C#
2	Php

s_id	hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey

- ❑ Now the tables satisfy 4NF

Fifth Normal form : 5NF

- ❑ Also called as Projection Join NF
- ❑ Rules for a relation to be in 5NF:
 - It should be in 4NF
 - The table shouldn't not contain any join dependencies
 - Or further lossless decomposition of tables couldn't be achieved



5NF : Join Dependency

- A relation is said to have join dependency if it can be recreated by joining multiple sub relations and each of these sub relations has a subset of the attributes of the original relation
- Let R be a relation schema and R_1, R_2, \dots, R_n be the decomposition of R , R is said to satisfy the join dependency (R_1, R_2, \dots, R_n) , if and only if every legal instance $r(R)$ is equal to join of its projections on R_1, R_2, \dots, R_n .

5NF Example 1

- Lets assume this relation/table R1 has Agent, company, Product attributes together forms PK

Agent	Company	Product
smith	Ford	car
smith	Ford	truck
Smith	GM	car
smith	GM	Truck
Jones	Ford	car

- The table is in 4NF and all are key columns.
- The table has more redundancy that smith is agent for ford being repeated twice! And hence we move to 5NF
 - So split into 3 tables

5NF Example 1

Agent	Company	Product
smith	Ford	car
smith	Ford	truck
Smith	GM	car
smith	GM	Truck
Jones	Ford	car

Relation R

Relation R1

Agent	Company
smith	Ford
smith	GM
Jones	Ford

Relation R2

Company	Product
Ford	car
Ford	truck
GM	Car
GM	truck

Relation R3

Agent	Product
smith	car
smith	Truck
jones	car

5NF Example : Loseless Decomposition

- Now, let's join all the 3 tables/ relations R1,R2,R3.
 - If we could get the original table R, then the decomposition is lossless and there is a join dependency and hence decomposition can be allowed. After which, say the tables are in 5NF after decomposition. Further more decomposition is not possible.
 - If the decomposition will be lossy then the tables should be decomposed further! Meaning it doesn't have any join dependency and need not be decomposed further!

5NF : Natural Join of R1 and R2

□ Join of R1 and R2 yields R12

Relation
R1

Agent	Company
smith	Ford
smith	GM
Jones	Ford

Relation
R2

Company	Product
Ford	car
Ford	truck
GM	Car
GM	truck

Agent	Company	Product
Smith	ford	car
smith	ford	truck
smith	GM	car
smith	GM	Truck
Jones	ford	Car
Jones	ford	truck

Relation R12

5NF : Natural join of R12 and R3

□ Join of R12 and R3

Agent	Company	Product
Smith	ford	car
smith	ford	truck
smith	GM	car
smith	GM	Truck
Jones	ford	Car
Jones	ford	truck

Relation R12

Agent	Company	Product
smith	Ford	car
smith	Ford	truck
Smith	GM	car
smith	GM	Truck
Jones	Ford	car

Relation R

Agent	Product
smith	car
smith	Truck
jones	car

Relation R3

5NF : Join Dependency

- It is observed that natural join of R1 , R2 and R3 yields the same relation R concluding the relation R has join dependency and hence not in 5NF and so decomposition of R into R1,R2 and R3 is recommended to avoid redundancy
 - Join dependency comes out to be: {(Agent,Product), (Company, Product), (Company, Agent)}
- Incase if natural join of R1 , R2 and R3 was different from relation R, then R is said to be already in 5NF with absence of join dependency! So no further decomposition is required!

Note!

- ❑ Decomposing tables helps in reducing redundancy & improves data integrity!
- ❑ But, managing data across multiple tables requires establishing relationships (foreign keys) and potentially writing more complex queries to retrieve all the information at once.
- ❑ Think wise before decomposing!
- ❑ While higher normal forms can improve data integrity, they might lead to performance degradation due to an increased number of joins required to gather data spread across more tables. Therefore, practical considerations sometimes limit the extent of normalization.

Try it yourself!

- Assume the schema and check if it is in 5NF

Agent	Company	Product
smith	Ford	car
smith	Ford	truck
Jones	GM	car
Jones	Ford	truck

Splitting this table into 3 tables and joining them will generate some spurious records which aren't present in original table and hence the decomposition is lossy. So the table is already in 5NF and doesn't need any further decomposition



Normalize the unnormalized tables!

Think before decomposing!

To do ! Perform NF checks !

Custo merID (PK)	Name	Address	Ph.No.	Email	AccountNumb er	AccountT ype	Balance	LoanID	LoanAm ount	LoanBala nce
1001	A	USA	555	A@abc.com	123456789, 987654321	Savings, Checking	1000.00, 250.00	NULL	NULL	NULL
1002	B	UK	3224	bb@abc.com	234567890	Checking	500	10001	10000	5000
1003	C	India	544	cc@c.com, cc@abc.com				NULL	NULL	NULL

To do ! Food delivery System !

OrderID	Customer Name	CustomerEmail	OrderDate	ItemID	ItemName	ItemCategory	Quantity	Price	Payment Method	Delivery Address	Chef
001	John Doe	john@example.com	2024-04-20	F001	Pizza	Main Course	2	20	Credit Card	123 Elm St	Chef A
001	John Doe	john@example.com	2024-04-20	F002	Cake	Dessert	1	10	Credit Card	123 Elm St	Chef B
002	Jane Smith	jane@example.com	2024-04-21	F003	Salad	Appetizer	3	15	PayPal	456 Oak St	Chef A
003	Emily Ray	emily@example.com	2024-04-22	F001	Pizza	Main Course	1	20	Cash	789 Pine St	Chef A
003	Emily Ray	emily@example.com	2024-04-22	F002	Cake	Dessert	2	10	Cash	789 Pine St	Chef B



Thank you!