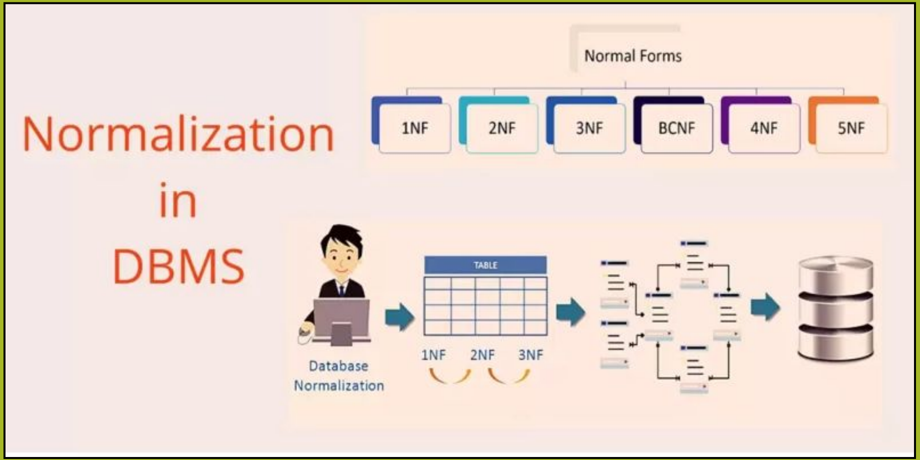
UNIT III

DATABASE DESIGN

TRANSACTION PROCESSING

CONCEPTS



CONTENT

- **Database Design:** Informal Design Guidelines for Relation Schemas; Functional Dependencies; Normal Forms Based on Primary Keys; General Definitions of Second and Third Normal Forms; Boyce-Codd Normal Form.

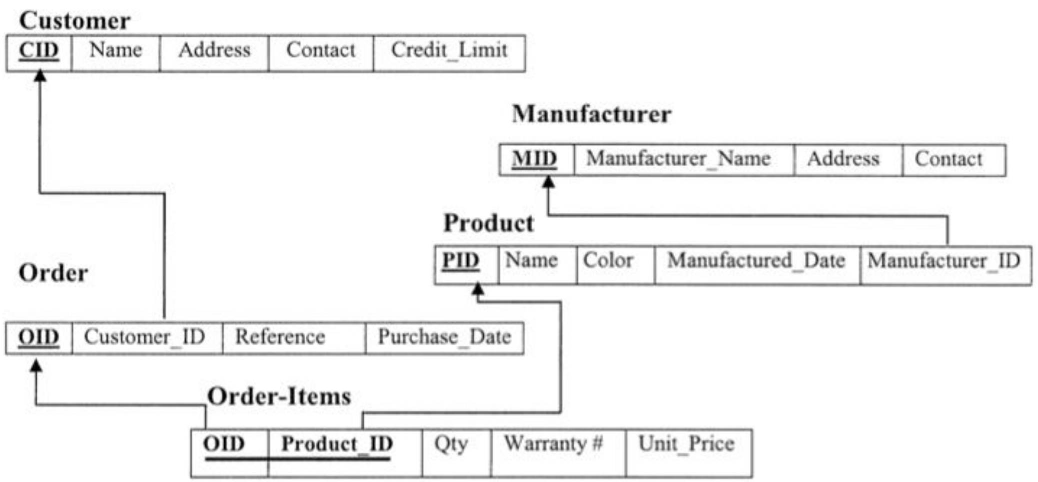
- **Transaction Processing Concepts:** Introduction to Transaction processing, Transaction and System concepts, Desirable properties of Transactions and issues with concurrent transactions. **2PL and TSO algorithms.**

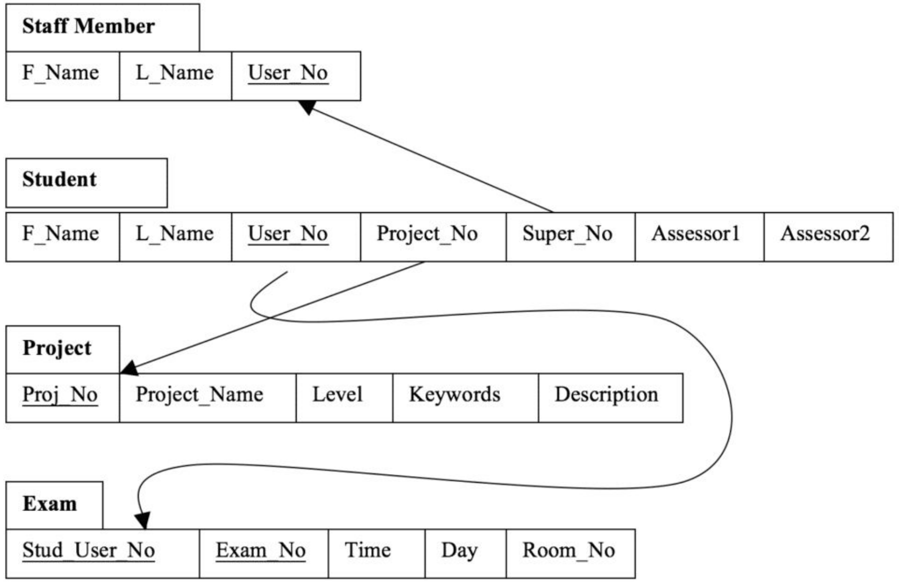
- **Course Learning Objective:**
- To introduce formal database design approach through normalization and discuss various normal forms.
- **Course Outcome (CO)**
- **CO3: Apply** the concepts of Normalization and design database which eliminates all anomalies.
- **CO5: Explain** the issues of concurrency control and algorithm to address the problems in concurrent transaction processing.

INTRODUCTION

- Each **relation schema** consists of a number of attributes, and
- A **relational database schema** consists of a number of relation schemas.
- Attributes are grouped to form a relation schema :
  - by using the common sense of the database designer or
  - by mapping a database schema design from a conceptual data model such as the **ER or Enhanced-ER (EER)** data model.
- These models make the designer identify entity types and relationship types and their respective attributes, which leads to a natural and logical grouping of the attributes into relations when the mapping procedures are followed.
- **There is a need for some formal way of analyzing why one grouping of attributes into a relation schema may be better than another.**

Sample Relational Schema





Informal Design Guidelines for Relational Databases

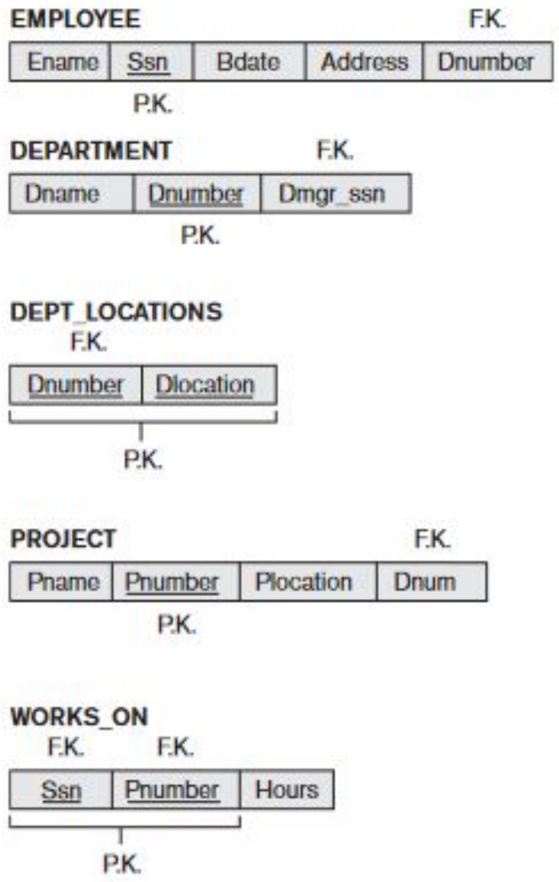
Four informal guidelines that may be used as measures to determine the quality of relation schema design:

- Making sure that the semantics of the attributes is clear in the schema
- Reducing the redundant information in tuples
- Reducing the NULL values in tuples
- Disallowing the possibility of generating spurious tuples

Imparting Clear Semantics to Attributes in Relations

- **The semantics of a relation refers to its meaning resulting from the interpretation of attribute values in a tuple -** the relational schema design should have a clear meaning.
- Whenever we group attributes to form a relation schema, we assume that:
  - attributes belonging to one relation have certain real-world meaning and
  - a proper interpretation associated with them.
- In general, the easier it is to explain the semantics of the relation, the better the relation schema design will be.

Imparting Clear Semantics to Attributes in Relations



- **The meaning of the EMPLOYEE relation schema is quite simple:** Each tuple represents an employee, with values for the employee's name (Ename), Social Security number(Ssn), birth date (Bdate), and address (Address), and the number of the department that the employee works for (Dnumber). The Dnumber attribute is a foreign key that represents an implicit relationship between EMPLOYEE and DEPARTMENT.

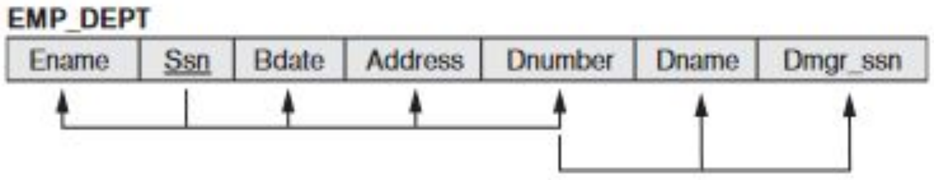
- **The semantics of the DEPARTMENT and PROJECT schemas are also straightforward:** Each DEPARTMENT tuple represents a department entity, and each PROJECT tuple represents a project entity. The attribute Dmgr_ssn of DEPARTMENT relates a department to the employee who is its manager, while Dnum of PROJECT relates a project to its controlling department; both are foreign key attributes.

The ease with which the meaning of a relation's attributes can be explained is an informal measure of how well the relation is designed.

Imparting Clear Semantics to Attributes in Relations

- Hence, all the relation schemas may be considered as easy to explain and therefore good from the standpoint of having clear semantics.
- **We can thus formulate the following informal design guideline (Guideline 1):**
- **Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship types into a single relation.**
- If a relation schema corresponds to one entity type or one relationship type, it is straightforward to interpret and to explain its meaning.
- Otherwise, if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained

Imparting Clear Semantics to Attributes in Relations



- The relation schemas also has a clear semantics.
- A tuple in the EMP_DEPT relation schema in represents a single employee but includes additional information—namely, the name (Dname) of the department for which the employee works and the SocialSecurity number (Dmgr_ssn) of the department manager.
- It violates Guideline 1 by mixing attributes from distinct real-world entities: EMP_DEPT mixes attributes of employees and departments,
- Hence, they fare poorly against the above measure of design quality.

Redundant Info. in Tuples & Update Anomalies

- Mixing attributes of multiple entities may cause problems
- Information is stored redundantly wasting storage
- Leads to an additional problem referred to as “update anomalies” , which can be classified into :
  - **Insertion anomalies**
  - **Deletion anomalies**
  - **Modification anomalies**
  - It is easy to see that these three anomalies are undesirable and cause difficulties to maintain consistency of data as well as require unnecessary updates that can be avoided; hence, we can state the next guideline as follows

EXAMPLE OF AN UPDATE ANOMALY

Consider the relation:

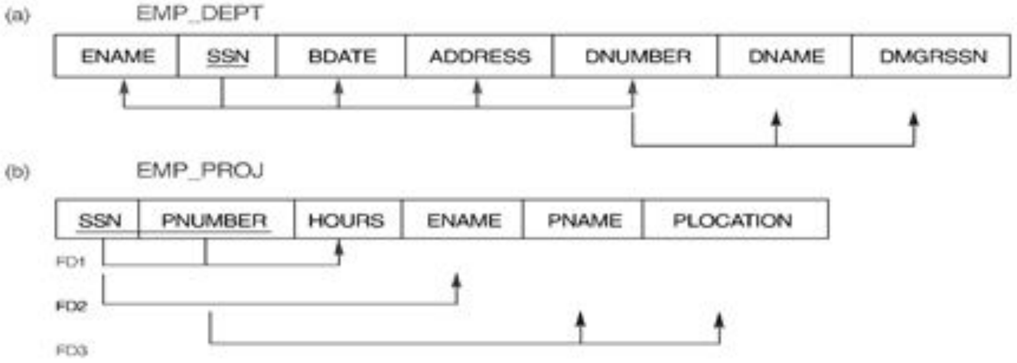
EMP_PROJ (Emp#, Proj#, Ename, Pname, No_hours)

- **Update Anomaly:** Changing the name of project number P1 from "Billing" to "Customer-Accounting" may cause this update to be made for all 100 employees working on project P1.
- **Insert Anomaly:** Cannot insert a project unless an employee is assigned to .

  Inversely - Cannot insert an employee unless an he/she is assigned to a project.
- **Delete Anomaly:** When a project is deleted, it will result in deleting all the employees who work on that project. Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.
-

- **GUIDELINE 2:**
- Design a schema that does not suffer from the insertion, deletion and update anomalies. If there are any present, then note them so that applications can be made to take them into account

Relation schemas suffering from update anomalies



EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith,John B.	ProductX	Bellaire
123456789	2	7.5	Smith,John B.	ProductY	Sugarland
666884444	3	40.0	Narayan,Ramesh K.	ProductZ	Houston
453453453	1	20.0	English,Joyce A.	ProductX	Bellaire
453453453	2	20.0	English,Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong,Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong,Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong,Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong,Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya,Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya,Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar,Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar,Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace,Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace,Jennifer S.	Reorganization	Houston
888665555	20	null	Borg,James E.	Reorganization	Houston

Null Values in Tuples

- In some schema designs we may group many attributes together into a “fat” relation.
- If many of the attributes do not apply to all tuples in the relation, we end up with many NULLs in those tuples.
    - This can waste space at the storage level
    - lead to problems with understanding the meaning of the attributes
    - specifying JOIN operations at the logical level.
        - SELECT and JOIN operations involve comparisons; if NULL values are present, the results may become unpredictable.
    - Another problem with NULLs is how to account for them when aggregate operations such as COUNT or SUM are applied .
    - Moreover, NULLs can have multiple interpretations:

Null Values in Tuples

- The attribute does not apply to this tuple.

  For example,Visa_status may not apply to U.S. students.

- The attribute value for this tuple is unknown.

  For example, the Date_of_birthmay be unknown for an employee.

- The value is known but absent; that is, it has not been recorded yet.

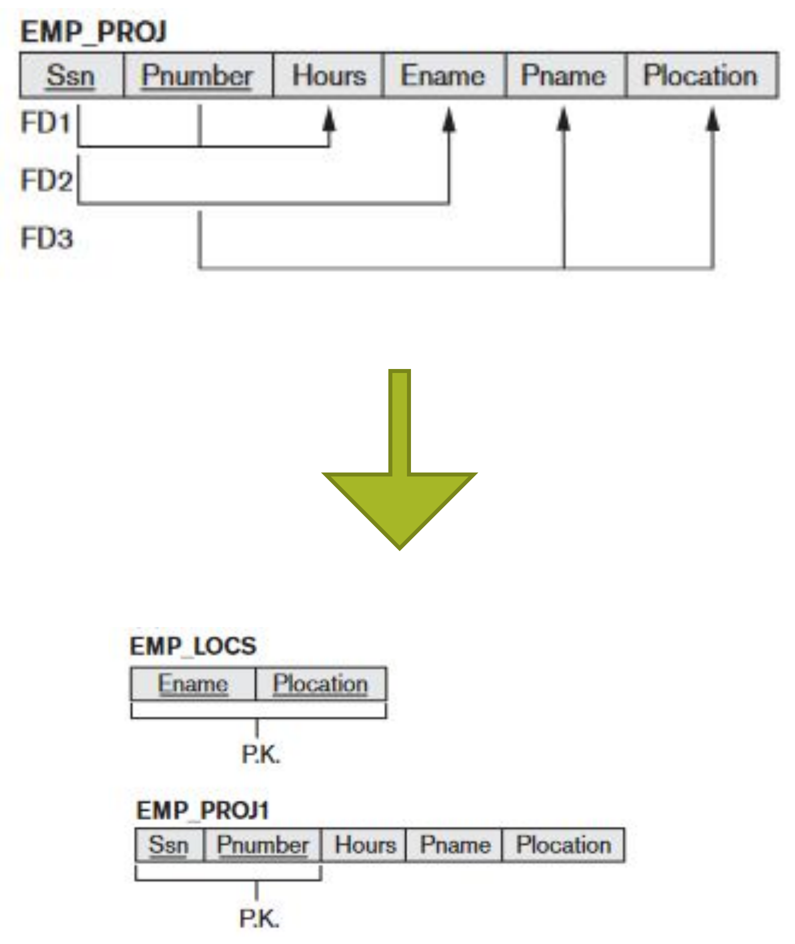
  For example, the Home_Phone_Numberfor an employee may exist, but may not be available and recorded yet.

Having the same representation for all NULLs compromises the different meanings they may have.

Therefore, we may state another guideline.

- **GUIDELINE 3:**
- As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation

Generation of Spurious Tuples



EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

- Suppose that we wish to recover the information that was originally in EMP_PROJ from EMP_PROJ1 and EMP_LOCS by using EMP_PROJ1 and EMP_LOCS as the base relations.
    - If we attempt a NATURAL JOIN operation on EMP_PROJ1 and EMP_LOCS, the result produces many more tuples than the original set of tuples in

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
	333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
	333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
*	333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
	333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

- Result of applying NATURAL JOIN to the tuples above the dashed lines in EMP_PROJ1 and EMP_LOCS of
- Generated spurious tuples are marked by asterisks.

Generation of Spurious Tuples

GUIDELINE 4:

Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated. Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

DATABASE DESIGN

Functional Dependencies

- Formal tool for analysis of relational schemas.
- **Functional dependencies (FDs) are used to specify *formal measures* of the "goodness" of relational designs**
- **FDs and keys are used to define normal forms for relations**.
- **A FD is a constraint between two sets of attributes from the database.**
  - The FD is a relationship that exists between two attributes.
  - It typically exists between the primary key and non-key attribute within a table.
- **FD = a constraint between two attributes or two sets of attributes.**

Functional Dependencies

- FDs are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes.

- A FD is a property of the meaning or semantics of the attributes, I.e., a property of the relation schema.
  - They must hold on all relation states (extensions) of R.
  - Relation extensions r(R) that satisfy the FD are called **legal extensions**.

- If X and Y are two sets of attributes of R (X and Y are subsets of R) , FD is denoted by:

  X → Y

  - If X → Y, we **say X functionally determines Y or Y is functionally dependent on X.**
  - The left side of the above FD diagram is called the ***determinant***, and the right side is the ***dependent*.**
  - An attribute may be functionally dependent on two or more attributes

Functional Dependencies

Functional Dependency

A -> B

B - functionally dependent on **A**
A - determinant set
B - dependent attribute

- A set of attributes A *functionally determines* a set of attributes B if the value of A determines a unique value for Y - **if for any two tuples t_1 and t_2 in r that have $t_1[A]= t_2[A]$, they must also have $t_1[B]= t_2[B]$.**

Functional Dependencies

Examples:

ISBN → Title

- indicates that ISBN determines Title.

SIN, Course → DateCompleted

- SIN and Course determine the date completed

Emp_Id → Emp_Name

- Emp_Name is functionally dependent on Emp_Id.

DeptId → DeptName

- DeptId is functionally determines DeptName

Functional Dependencies

- Given a populated relation
  - Cannot determine which FDs hold and which do not.
  - Unless meaning of and relationships among attributes is known
  - **Can state that FD does not hold if there are tuples that show violation of such an FD.**
- The database designers will use their understanding of the semantics of the attributes of R—that is, how they relate to one another—to specify the functional dependencies that should hold on all relation states (extensions) r of R.
- **FDs are determined by semantics &** are derived from the real-world constraints on the attributes.
  - **You can't say that a FD exists just by looking at data.**
  - **But can say whether it does not exist by looking at data.**

Functional Dependencies

- **Example:**

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

- In this example do the following FD's hold ?
- 1) Employee number -> Employee Name ✔
- 2) Employee number -> city ✔
- 3) Employee number -> salary ✔
- We can say that the city, Employee Name, and salary are functionally depended on Employee number i.e., **Employee number -> {Employee Name , city, salary}**

- **There is a FD from the primary key to each of the other attributes in a table.**

Functional Dependencies

- **Example:**

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo
4	Andrew	30000	London
5	Francis	30000	NYC

- 4) Employee Name -> salary X
- 5) Employee Name -> City X
- 6) Salary -> City X

Functional Dependencies

- **Example:**

roll_no	name	dept_name	dept_building
42	abc	CO	A4
43	pqr	IT	A3
44	xyz	CO	A4
45	xyz	IT	A3
46	mno	EC	B2
47	jkl	ME	B2

- Identify some valid FD's .

- **Some valid FD's:**
- roll_no → name
- roll_no → dept_name
- roll_no → dept_building
- **roll_no → { name, dept_name, dept_building }**
- dept_name → dept_building
- {roll_no, name} → {dept_name, dept_building} etc
- **FD's that do not hold:**
- name → dept_name
- name → dept_building
- dept_building → dept_name
- dept_name, dept_building → name
- name → roll_no,
- {name, dept_name} → roll_no,
- dept_building → roll_no etc.

Functional Dependencies

- Consider the table with attributes A, B and C

A	B	C
1	33	100
2	33	200
3	22	200
1	33	101
2	33	350
4	67	350
5	67	101

- Suppose there are many more rows that are not shown.
- 1) Is there a functional dependency from B to A? Explain your answer.
- For row 1 & 2 : t1 [B] = 33 = t2[B]
- but t1[A] =1 & t2[A] = 2 => t1[A] ≠ t2[A]
- Therefore FD B-> A does not hold.
- 2) The rows that are shown suggest there could be a functional dependency A → B ?

Functional Dependencies

- Consider the following Department and Course tables :

DEPARTMENT

<u>deptCode</u>	deptName	deptLocn	deptPhone	chairName

COURSE

<u>deptCode</u>	<u>courseNo</u>	title	description	creditHours

- deptCode → deptName
- deptCode → deptLocn
- deptCode → deptPhone
- deptCode → chairName

deptCode, courseNo →title

deptCode, courseNo →description

deptCode, courseNo → credit hours

Functional Dependencies

- Consider the following relation:

A	B	C	TUPLE#
10	b1	c1	1
10	b2	c2	2
11	b4	c1	3
12	b3	c4	4
13	b1	c1	5
14	b3	c4	6

- Given the previous extension (state), which of the following dependencies may hold in the above relation? If the dependency cannot hold, explain why by specifying the tuples that cause the violation.

- I .A→B,
- ii. B→C,
- Iii .C→B,
- iv. B→A,
- v. C→A

Functional Dependencies

- All addresses in the same town have the same zip code

SSN	Name	Town	Zip
1234	Joe	Huntingdon	16652
2345	Mary	Huntingdon	16652
3456	Tom	Huntingdon	16652
5948	Harry	Alexandria	16603

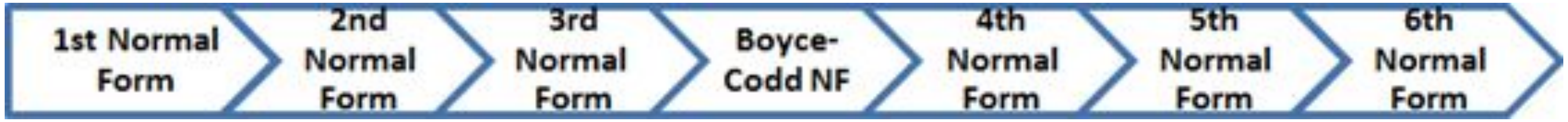
NORMAL FORMS

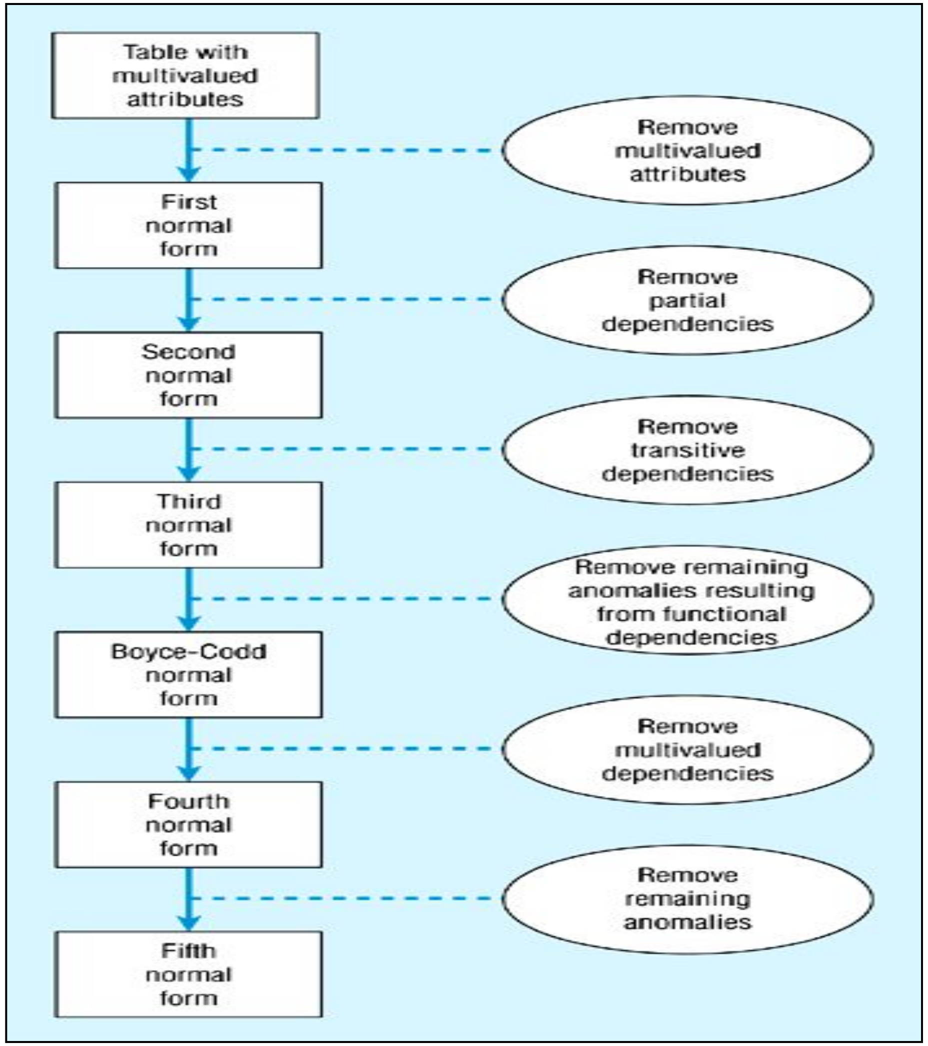
WHY NORMALIZATION?

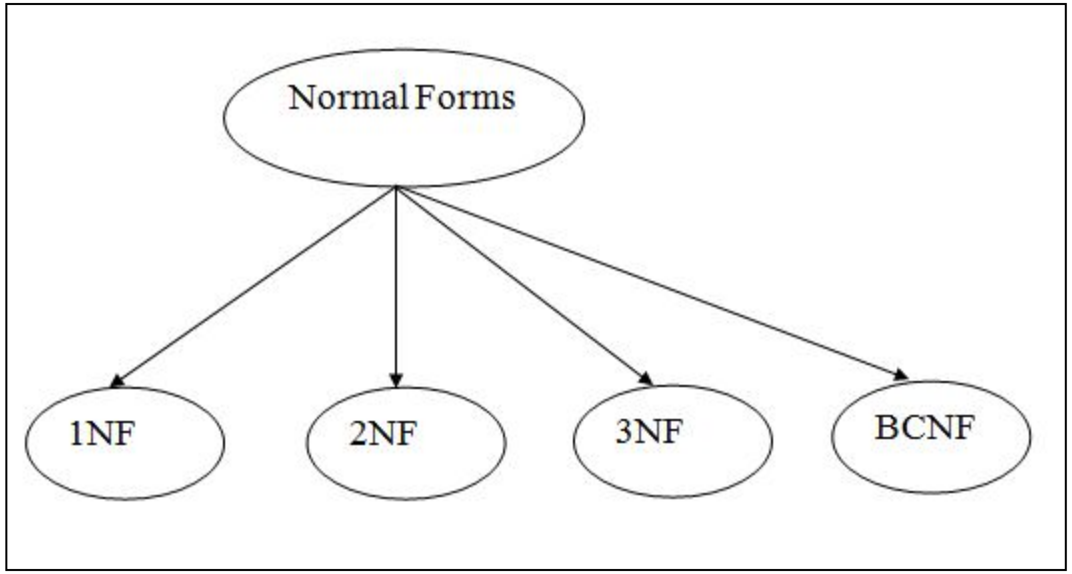
- Database is a storage for the structured collection of data.
- These data can be stored, accessed and manipulated across various locations within the database.
- The wide range of storage leads to the problem of data redundancy - wherein a piece of the same data information is stored or replicated in multiple locations within the database.
- Database with redundancy is at a disadvantage w.r.t to its storage, consistency and data manipulation.
- **Redundancy in relation may cause insertion, deletion and updation anomalies.**
- **So, it helps to minimize the redundancy in relations.**
- Normalization in DBMS is the process of effectively organizing data into multiple relational tables **to minimize data redundancy** and ensure data is stored logically.

WHY NORMALIZATION?

- The inventor of the relational model E.F. Codd proposed the theory of normalization of data with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form.
- Later he joined Raymond F. Boyce to develop the theory of Boyce-Codd Normal Form.
- The Theory of Data Normalization is still being developed further. For example, there are discussions even on 6th Normal Form.
- **However, in most practical applications, normalization achieves its best in 3rd Normal Form**.
- The evolution of Normalization in SQL theories is illustrated below-







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 all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
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.

FIRST NORMAL FORM -1NF

1NF states that the:

- **domain of an attribute must include only atomic(simple, indivisible) values** and that the value of any attribute in a tuple must be a **single value from the domain of that attribute**.
- Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a single tuple.
- In other words, 1NF disallows relations within relations or relations as attribute values within tuples.
- First normal form - allows only atomic (or indivisible) values

  - disallows the multi-valued attribute, composite attribute, and their combinations.

FIRST NORMAL FORM (1NF)

1NF states that the:

- **domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute.**
- Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a single tuple.
- In other words, 1NF disallows relations within relations or relations as attribute values within tuples.
- **First normal form - allows only atomic (or indivisible) values**

  - disallows the multi-valued attribute, composite attribute, and their combinations.

FIRST NORMAL FORM (1NF)

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

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

Full Names	Physical Address	Movies rented	Salutation
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

TECHNIQUES TO ACHIEVE 1NF

There are three main techniques to achieve first normal form for such a relation

1.Remove the attribute that violates 1NF and place it in a separate relation along with the primary key of the original relation.

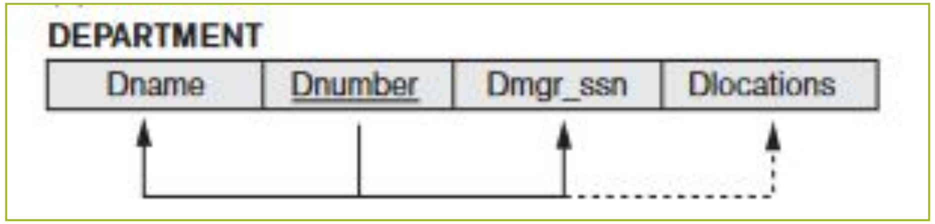
2. Expand the key so that there will be a separate tuple in the original. In this case, the primary key becomes a composite key.

This solution has the disadvantage of introducing redundancy in the relation.

3.I f a maximum number of values is known for the attribute—for example, if it is known that at most three locations can exist for a department—replace the Dlocations attribute by three atomic attributes:Dlocation1,Dlocation2, and Dlocation3.

This solution has the disadvantage of introducing NULL values if most departments have fewer than three locations.

EXAMPLE



A relation schema that is not in 1NF

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

San

EXAMPLE

Technique # 1

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Technique #2

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Technique # 3

Dname	Dnumber	Dmgr_ssn	DLocation1	DLocation2	DLocation3
Research	5	333445555	Bellaire	Sugarland	Houston
Administration	4	987654321	Stafford		
Headquarters	1	899665555	Houston		

FIRST NORMAL FORM (1NF)

Full Names	Physical Address	Movies rented	Salutation
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Full Names	Physical Address	Movies rented	Salutation
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

FIRST NORMAL FORM (1NF)

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

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

FIRST NORMAL FORM (1NF)

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

FIRST NORMAL FORM (1NF)

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	INDIA
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

SECOND NORMAL FORM - 2NF

Definition:

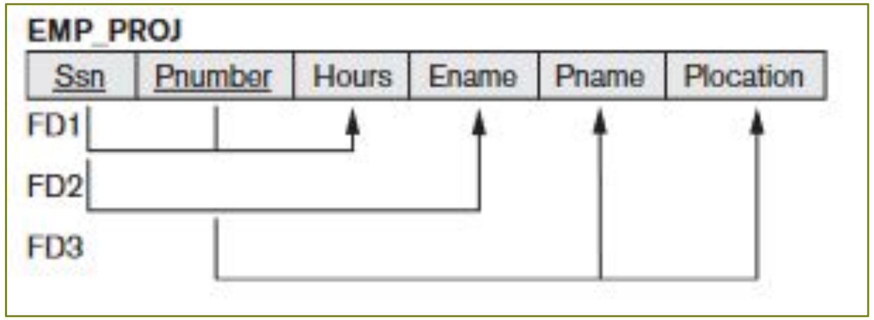
- Relational must be in 1NF.
- All non-key attributes are fully functional dependent on the primary key
- The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key. **If the primary key contains a single attribute, the test need not be applied at all.**

Technique:

- If a relation schema is not in 2NF, it can be second normalized or 2NF normalized into a number of 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent.

SECOND NORMAL FORM - 2NF

Example :

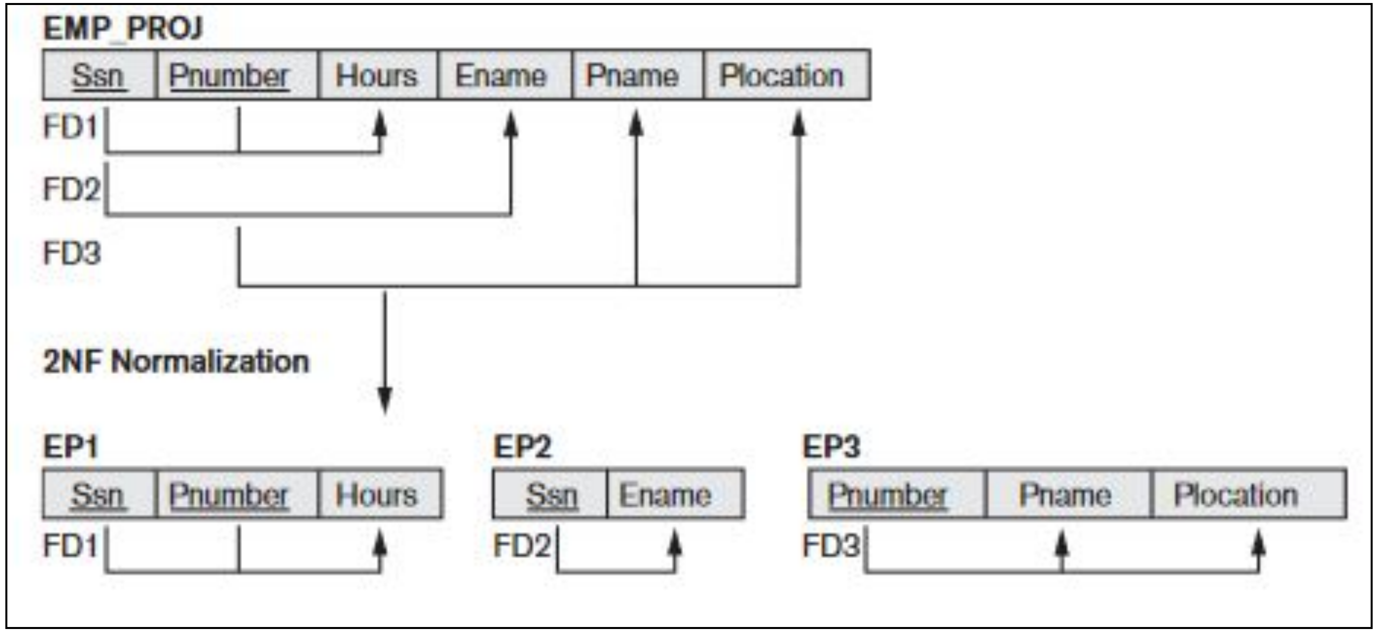


FD1 : Ssn, Pnumber -> Hours **FD2: Ssn -> Ename** **FD3: Pnumber -> Pname, Plocation**

- The EMP_PROJ relation is in 1NF but is not in 2NF.
- The nonprime attribute Ename violates 2NF because of FD2,as do the nonprime attributes Pname and Plocation because ofFD3. The functional dependencies FD2 and FD3 make Ename, Pname, and Plocation partially dependent on the primary key {Ssn,Pnumber} of EMP_PROJ, thus violating the 2NF test.

SECOND NORMAL FORM - 2NF

The functional dependencies FD1,FD2, and FD3 lead to the decomposition of EMP_PROJ into the three relation schemas EP1,EP2, and EP3 shown in Figure below , each of which is in 2NF.



SECOND NORMAL FORM - 2NF

Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this:

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

Candidate Keys: {teacher_id, subject}

Non prime attribute: teacher_age

SECOND NORMAL FORM - 2NF

- The table is in 1 NF because each attribute has atomic values.
- However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".
- To make the table complies with 2NF we can break it in two tables like this:

teacher_details table:

teacher_id	teacher_age
111	38
222	38
333	40

teacher_subject table:

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

SECOND NORMAL FORM - 2NF

Consider the following **Score** table:

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

Candidate Key: {student_id + subject_id}

Non prime attribute: marks, teacher_age

THIRD NORMAL FORM -3NF

- Although 2NF relations have less redundancy than those in 1NF, they may still suffer from update anomalies - If we update only one tuple and not the other, the database would be in an inconsistent state.
- This update anomaly is caused by a **transitive dependency**.
- We need to remove such dependencies by progressing to Third Normal Form (3NF).
- When a table is in the Second Normal Form and has no transitive dependency, then it is in the Third Normal Form.
- For a table to be in the third normal form,
  - **It should be in the Second Normal form.**
  - **And it should not have Transitive Dependency.**

THIRD NORMAL FORM -3NF

- **Note –**

  If A->B and

  B->C are two FDs

  then A->C is called transitive dependency.

TECHNIQUE TO ACHIEVE 3NF

- The normalization of 2NF relations to 3NF involves the **removal of transitive dependencies.**
- If a transitive dependency exists, we remove the transitively dependent attribute(s) from the relation by placing the attribute(s) in a new relation along with a copy of the determinant.

NOTE

- Third Normal Form (3NF) is considered *adequate* for normal relational database design because most of the 3NF tables are free of insertion, update, and deletion anomalies.

- Moreover, 3NF *always ensures functional dependency preserving and lossless*.

EXAMPLE

STUD_NO	STUD_NAME	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	HARYANA	INDIA	20
2	RAM	PUNJAB	INDIA	19
3	SURESH	PUNJAB	INDIA	21

FD set:

{STUD_NO -> STUD_NAME,

STUD_NO -> STUD_STATE,

STUD_STATE -> STUD_COUNTRY,

STUD_NO -> STUD_AGE }

Candidate Key:

{STUD_NO}

EXAMPLE

STUD_NO	STUD_NAME	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	HARYANA	INDIA	20
2	RAM	PUNJAB	INDIA	19
3	SURESH	PUNJAB	INDIA	21

- For this relation :
  - **STUD_NO -> STUD_STATE and**
  - **STUD_STATE -> STUD_COUNTRY are true.**
- **So STUD_COUNTRY is transitively dependent on STUD_NO – it violates the third normal form.**
- To convert it in third normal form, **we will decompose the relation** STUDENT (STUD_NO, STUD_NAME, STUD_STATE, STUD_COUNTRY_STUD_AGE) as:
  - STUDENT (STUD_NO, STUD_NAME, STUD_STATE, STUD_AGE)
  - STATE_COUNTRY (STATE, COUNTRY)

EXAMPLE

- Consider the following example:

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

- Is the relation in 3NF?

EXAMPLE

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

- FD's:
  - Book ID → {Genre ID, Genre Type, Price } and
  - Genre ID → Genre Type.
- Therefore, Book ID determines Genre Type via Genre ID and **we have transitive functional dependency.**
- **Therefore this relation does not satisfy third normal form.**

EXAMPLE

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

- To bring this table to third normal form, we split the table into two as follows:

TABLE_BOOK

Book ID	Genre ID	Price
1	1	25.99
2	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99

TABLE_GENRE

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

EXAMPLE

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

- **Is the relation in 3NF?**
- **Candidate key: {EMP_ID}**
- **Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.**
- **FD's:**
  - **EMP_ID** → { EMP_NAME, EMP_ZIP, EMP_STATE, EMP_CITY}
  - **EMP_ZIP** → {EMP_STATE, EMP_CITY}

EXAMPLE

- Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID.
- **The non-prime attributes (EMP_STATE, EMP_CITY) are transitively dependent on key(EMP_ID). It violates the rule of third normal form.** That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

EXAMPLE

A bank uses the following relation:

Vendor

ID	Name	Account_No	Bank_Code_No	Bank

Startsituation

Is the relation in 3NF?

SOLUTION :

1NF - All attributes are single valued (1NF). Therefore the relation is in 1NF.

Key – {ID}

EXAMPLE

Vendor

ID	Name	Account_No	Bank_Code_No	Bank

Startsituation

The following FD's exist:

ID --> Name, Account_No, Bank_Code_No,Bank

Bank_Code_No --> Bank (Bank is functionally dependent on)

2NF - No partial dependencies exist. Therefore the relation is in 2NF.

3NF - there is a transitive dependency. Non prime attribute Bank **is transitively dependent on key**.

Therefore the relation is NOT in 3NF.

EXAMPLE

Vendor

ID	Name	Account_No	Bank_Code_No	Bank

Startsituation

Result after normalisation

Vendor

ID	Name	Account_No	Bank_Code_No

Bank

Bank_Code_No	Bank

BCNF – Boyce Codd Normal Form

- Boyce-Codd Normal Form or BCNF is an extension to the third normal form, and is also known as 3.5 Normal Form.
- BCNF is stricter than 3NF.
- For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:
  - **It should be in the Third Normal Form.**
  - **And, for any dependency A → B, A should be a super key.**
- **For BCNF, the table should be in 3NF, and for every FD, LHS is super key.**

BCNF – Boyce Codd Normal Form

Decomposition into BCNF

- When a table is in 3NF, it may or may not be in the Boyce Codd Normal Form. Each table/relation will have a set of functional dependency. If the FD does not satisfy the second condition of BCNF, the table is decomposed (breaking into smaller tables) recursively until all the functional dependency meets the super key criteria.
- The algorithm to be followed for decomposition is:
  - Determine the functional dependency that violates the BCNF.
  - For every functional dependency X->Y which violates, decompose the relation into R-Y and XY. Here R is a relation.
  - Repeat until all the relations satisfy BCNF.

BCNF – Boyce Codd Normal Form

Decomposition into BCNF

- When a table is in 3NF, it may or may not be in the Boyce Codd Normal Form. Each table/relation will have a set of functional dependency. If the FD does not satisfy the second condition of BCNF, the table is decomposed (breaking into smaller tables) recursively until all the functional dependency meets the super key criteria.
- The algorithm to be followed for decomposition is:
  - Determine the functional dependency that violates the BCNF.
  - For every functional dependency X->Y which violates, decompose the relation into R-Y and XY. Here R is a relation.
  - Repeat until all the relations satisfy BCNF.

BCNF – Example

Consider the following relationship : **R (A,B,C,D)**

and following dependencies :

A -> BCD
BC -> AD
D -> B

Above relationship is already in 3rd NF. Keys are **A** and **BC.**

Hence, in the functional dependency, **A -> BCD**, A is the super key.
in second relation, **BC -> AD**, BC is also a key.
but in, **D -> B**, D is not a key.

Hence we can break our relationship R into two relationships **R1** and **R2**.

R(A, B, C, D)

R1 (A, D, C)   R2(D, B)

Breaking, table into two tables, one with A, D and C while the other with D and B.

BCNF – Example

Example #1

- Let’s consider a Relation R with five attributes

  R=ABCDE

- **The functional dependencies are**

  FD = {A -> BC, C -> DE)

- **Candidate keys are {A}**

Is the relation n BCNF ?

BCNF – Example

- Inspect each of the FD to check whether it satisfies the second condition of BCNF as it is in 3NF.
  - **The first FD A -> BC does not violate BCNF --------- since A is a key for R.**
  - **Second FD C -> DE violates BCNF ----------- since C is not a key of R**
- We decompose R into **R1 {A,B,C} and R2 {C,D,E}.**
- The two schemas are created with the FD attributes which violates and the other with original attributes minus the right-hand side of the violating FD.
- Now we will check both the newly created relations to check whether they are in BCNF or not. A is the key in (ABC), and C is the key (CDE) they do not violate BCNF. Thus the relation is in BCNF.

BCNF – Example

- Consider the following table (St_Maj_Adv).

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

- The functional dependencies for this table are listed below
  - Student_id, Major ——> Advisor
  - Advisor ——> Major
- PK can be Student_id, Major or Student_id, Advisor.

BCNF – Example

- Consider the following table (St_Maj_Adv).

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

- The functional dependencies for this table are listed below
  - Student_id, Major ——> Advisor
  - Advisor ——> Major

BCNF – Example

- The functional dependencies for this table are listed below
  - Student_id, Major ——> Advisor – satisfies BCNF as {Student_id, Major} is a superkey
  - Advisor ——> Major violates BCNF as {Advisor} is not a superkey
- To reduce the St_Maj_Adv relation to BCNF, you create two new tables:
  - St_Adv (Student_id, Advisor)
  - Adv_Maj (Advisor, Major)

St_Adv table

Student_id	**Advisor**
111	Smith
111	Chan
320	Dobbs
671	White
803	Smith

Adv_Maj table

Advisor	**Major**
Smith	Physics
Chan	Music
Dobbs	Math
White	Physics

BCNF – Example

Consider the following table (Client_Interview):

ClientNo	InterviewDate	InterviewTime	StaffNo	RoomNo
CR76	13-May-02	10.30	SG5	G101
CR56	13-May-02	12.00	SG5	G101
CR74	13-May-02	12.00	SG37	G102
CR56	1-July-02	10.30	SG5	G102

FD1 – ClientNo, InterviewDate –> InterviewTime, StaffNo, RoomNo

FD2 – staffNo, interviewDate, interviewTime –> clientNO

FD3 – roomNo, interviewDate, interviewTime –> staffNo, clientNo

FD4 – staffNo, interviewDate –> roomNo

Is the table in BCNF?

BCNF – Example

A relation is in BCNF if, and only if, every determinant is a superkey. We need to create a table that incorporates the first three FDs (**Client_Interview2** table) and another table (**StaffRoom** table) for the fourth FD.

ClientNo	InterviewDate	InterViewTime	StaffNo
CR76	13-May-02	10.30	SG5
CR56	13-May-02	12.00	SG5
CR74	13-May-02	12.00	SG37
CR56	1-July-02	10.30	SG5

StaffNo	InterviewDate	RoomNo
SG5	13-May-02	G101
SG37	13-May-02	G102
SG5	1-July-02	G102