

# Database Integrity and Normalisation

---

Unit 3

Red: indicates important

# Relational Database Integrity

---

- Integrity means to maintain the consistency of data.
- Integrity constraints in a database ensures that changes made to the database by authorized users does not compromise data consistency.
- Integrity constraints do not allow damage to the database.
- There are two integrity constraints: entity constraints and referential integrity constraints.
- **Different keys:**
  1. Candidate Key:
    - In a relation  $R$ , a candidate key for  $R$  is a subset of the set of attributes of  $R$  with the following properties:
      1. Uniqueness : No two distinct tuples in  $R$  have the same value for the candidate key.
      2. Irreducible: No proper subset of the candidate key has the uniqueness property that is the candidate key.

- 
- Every relation must have atleast one candidate key.
  - A candidate key can be a composite key also.
  - Duplicate tuples are not allowed in relations.
  - Properties of a candidate key:
    - A candidate key must be unique and irreducible.
    - A candidate may involve one or more attribute. A candidate key that has more than one attribute is called a composite.
  - They are important because they provide tuple-level identification mechanism.

## 2. Primary Key :

- It's the candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set.

---

### 3. Alternate Key :

- Candidate keys which are not primary keys are called alternate keys.

### 4. Foreign Key :

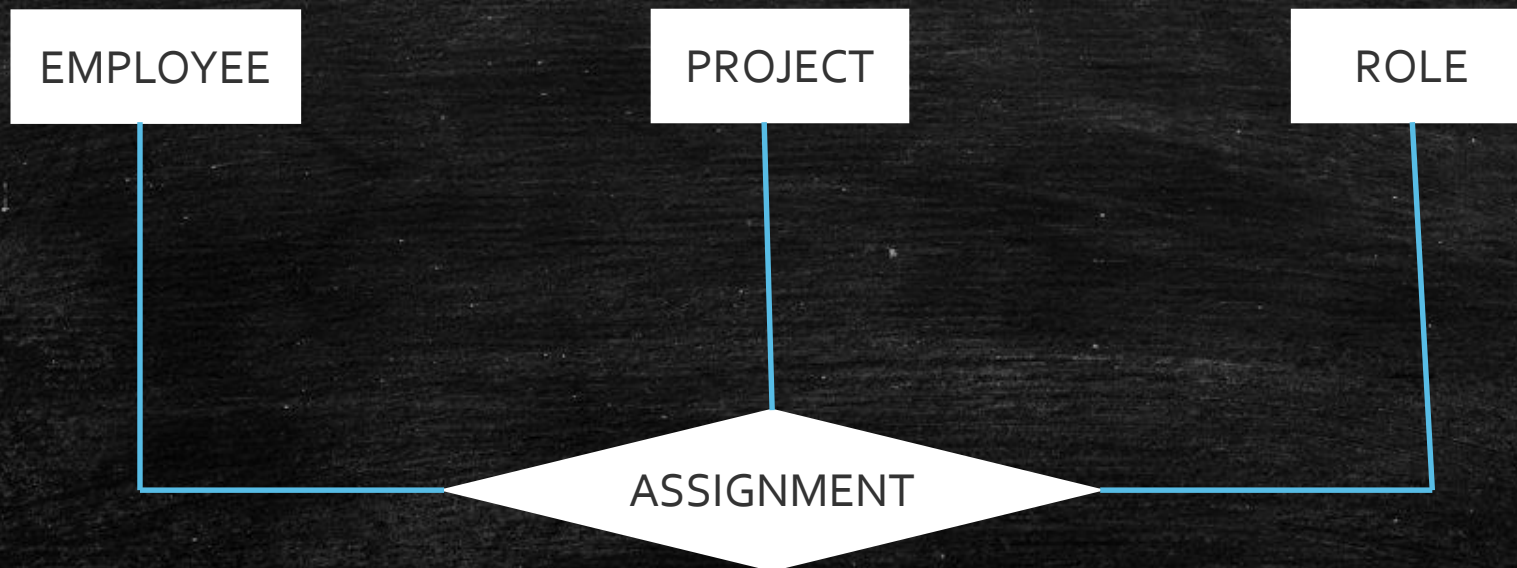
- Let  $R_2$  be relation, then a foreign key in  $R_2$  is a subset of the set of attributes of  $R_2$ , such that:
  1. There exists a relation  $R_1$  with a candidate key, and
  2. For all time, each value of a foreign key in the current state or instance of  $R_2$  is identical to the value of Candidate Key in some tuple in the current state of  $R_1$ .

# Example of Keys

---

- Consider that in an organization an employee may perform different roles in different projects. For e.g.: John works for project A as developer and for project B as tester. The above information can be represented by 3 different relations namely EMPLOYEE, PROJECT and ROLE.
- Relational schema for the above relations:  
  
EMPLOYEE(EMPID, Name, Designation)  
  
PROJECT(PRJID, Proj\_Name, Details)  
  
ROLE(RID, Role\_description)
- In the above relations, EMPID, PRJID, RID are unique and not NULL. EMPID is the primary key of relation EMPLOYEE. PRJID and RID are the primary keys of relation PROJECT and ROLE respectively.
- Let ASSIGNMENT be a relationship between the 3 relations. It describes who is working on which project and what the role of employee is in the respective project.

- E-R Diagram:



- Instances for the above relations:

#### EMPLOYEE

EMPID	Name	Designation
1	John	Tester
2	Meera	Manager
3	Ajay	Analyst

#### PROJECT

PRJID	Proj_Name	Details
TCS	Traffic Control System	For traffic shaping
LG	Load Generator	To simulate load for inputs in TCS
B++1	B++_TREE	ISS/R turbo_sys

## ROLE

RID	Role_description
1000	Design
2000	Coding
3000	Marketing

## ASSIGNMENT

EMPID	PRJID	RID
1	TCS	1000
1	LG	2000
2	B++1	3000

- 
- Relational schema for ASSIGNMENT :

ASSIGNMENT(EMPID,PRJID,RID)

In the above relation EMPID is the foreign key in ASSIGNMENT relation, it refers to EMPLOYEE relation where EMPID is the primary key. In the same way, PRJID and RID are the foreign keys in ASSIGNMENT relation, referring to PROJECT and ROLE relation respectively.

# Referential Integrity

---

- Definition: Referential integrity states that the database must not contain any unmatched foreign key values.
- For example, in E.g.1 any value existing in the EMPID in ASSIGNMENT relation must be there in EMPID in EMPLOYEE relation. So, if we try to add a tuple with EMPID value 4 in the ASSIGNMENT relation, it will result in violation of referential integrity constraints.
- Database modification can cause violation of referential integrity. In order to preserve the referential integrity following test must be done:
  - Delete: There are two cases:

Case 1: Deletion of tuple in relation having the foreign key

- We simply delete the desired tuple.

---

### Case 2: Deletion of target of a foreign key reference:

- For example in E.g.1, if we make an attempt to delete an employee tuple in the EMPLOYEE relation whose EMPID is 1. This employee also appears in ASSIGNMENT relation. If we delete the tuple from EMPLOYEE relation then two unmatched values will be there in the ASSIGNMENT relation resulting in violation of referential integrity constraint.
- Two choices for this problem:
  1. RESTRICT: In this case, delete operation is restricted to only the case where there are no such matching tuples. For example, in E.g.1, we can delete the EMPLOYEE record EMPID 3 as there is no matching tuple in ASSIGNMENT but cannot delete record of EMPID 1.
  2. CASCADE: In this case, delete option cascades to delete all the matching tuples also. On deleting an EMPLOYEE record EMPID 1, it will also cause the deletion of the matching tuples from assignment relation.
- Insert: Insertion of tuple in the target of reference does not cause any violation. Insertion of a tuple in the relation in which, we have the foreign key, it must be ensured that all matching target candidate key exist, otherwise it will be rejected.

- 
- Update: There are 2 cases:
    - Case 1: If the operation changes the value that is the foreign key also, it must be ensured that all matching target candidate key exist , otherwise will be rejected.
    - Case 2: Modification of candidate key that is the target of a foreign key reference
    - Two choices:
      1. RESTRICT : Update operation is restricted to cases where there is no matching tuples.
      2. CASCADE : In this case, update option cascades to update all the matching tuples also.

# Entity Integrity

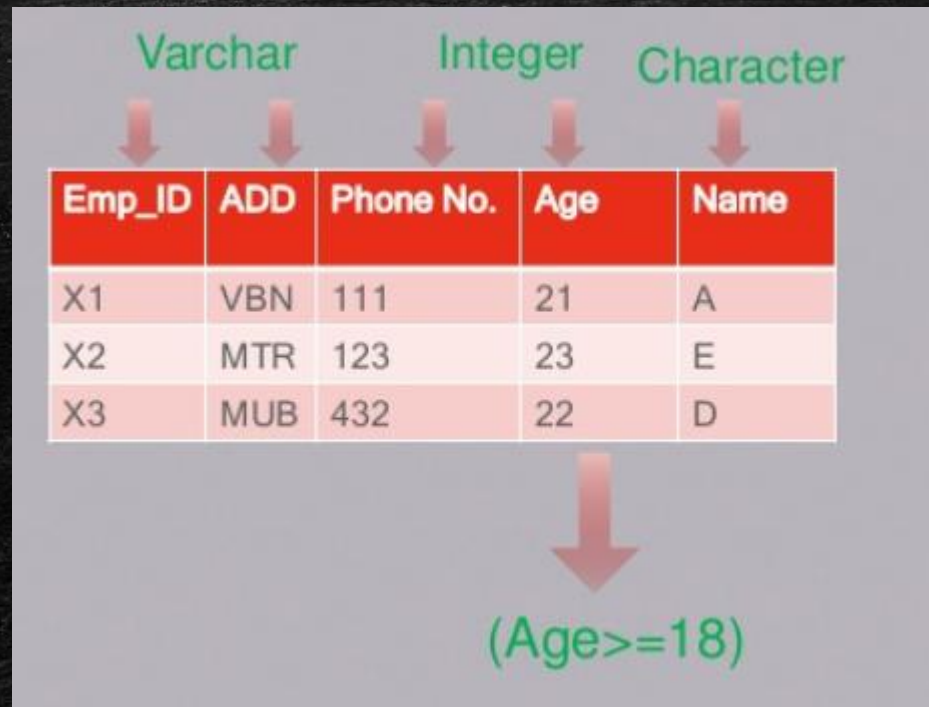
- It states that no primary key or components of a primary key of a relation is allowed to accept NULL .
- Example:

EMPID	Name	Designation
1	John	Tester
NULL	Meera	Manager
3	Ajay	Analyst

The above table results in entity integrity constraints. As EMPID is the primary key and it has NULL value.

Domain constraints :

- It means the definition of a valid set of values for an attribute.



Emp_ID	ADD	Phone No.	Age	Name
X1	VBN	111	21	A
X2	MTR	123	23	E
X3	MUB	432	22	D

(Age >= 18)

# Redundancy and Associated Problems

- Duplication of data in a database is called data redundancy.
- It results in wastage of space and other problems called database anomalies. There are three anomalies in database systems.
  1. Insertion Anomaly
  2. Update Anomaly
  3. Deletion Anomaly

Relation STUDENT: Example 1

Rno	Sname	Address	Cno	Cname	Tutor	Office
14	Rahul	Delhi	MCA-011	DAA	Karan	100
14	Rahul	Delhi	MCA-012	DBMS	Akash	105
17	Anjana	Delhi	MCA-014	Math	Danya	104

---

## I. Insertion Anomaly:

- An insertion anomaly is the inability to add data to the database due to absence of other data.
- The above relation has Rno and Cno as primary key.  
Any new tuple to be inserted in the relation must have a value for the primary key, if not it results in violation of entity integrity constraints.
- If we want to insert number and name of new course, it would not be possible until a student enrolls in that course.
- Information about a new student cannot be inserted in the database until the student enrolls in a course.

---

## II. Update Anomaly:

- Caused due to data redundancy.
- Updating becomes more difficult due to redundant data.
- For example, in the above relation STUDENT if we want to change the tutor of MCA-014, this would require updating all the tuples having MCA-014 enrolment data. Due to some reason, if the tuples are not updated this will result in inconsistent data. This is called update anomaly.
- Update anomaly results in data inconsistency.

---

### III. Deletion anomaly

- Loss of useful information is called deletion anomaly. I.e., useful information may be lost when a tuple is deleted.
- For example: 1, if we delete tuple corresponding to student 17 enrolled for MCA-014, we will lose relevant information about the student(name, address, rno).
- For example: 2, if we delete a tuple having Sname "Rahul" and Cno MCA-012, we will lose relevant information about the course MCA-012(Cname, tutor, office).

# Single-Valued Dependencies

---

Functional dependency:

- Consider a relation  $R$  that has two attributes  $A$  and  $B$ . The attribute  $B$  of the relation is functionally dependent on the attribute  $A$  if and only if for each value of  $A$ , no more than one value of  $B$  is associated.
- Attributes of a table is said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table.
- The FD between the attributes can be written as:  $R.A \rightarrow R.B$  or simply  $A \rightarrow B$ ,  $B$  is functionally dependent on  $A$  or  $A$  functionally determines  $B$ .

- 
- Dependencies of STUDENT relation is written as:

Rno- $\rightarrow$  Sname

Rno- $\rightarrow$  Address

Cno- $\rightarrow$  Cname

Cno- $\rightarrow$  Tutor

Tutor- $\rightarrow$  Office

- It's a many-to-one relationship between two sets of attributes of a given relation.
- Given a relation R, the FD A- $\rightarrow$  B is said to hold on R if and only if, whenever two tuples of R have the same value for A, and they also have the same for B.

# Single valued Normalisation and Normal Forms

---

- Codd in 1972 presented three normal forms (1NF, 2NF, 3NF). This is based on functional dependencies between attributes of a relation.
- Boyce and Codd proposed another normal form called Boyce-Codd Normal form.
- Fourth and fifth normal forms were proposed later. Its based on multivalued and joined dependencies.

## The First Normal Form(1NF):

- A relation is said to be in 1NF if:
  1. There are no duplicate rows or tuples in the relation.
  2. Each data value stored in the relation is single-valued.
  3. Entries in a column(attribute) are of the same kind(type).

- 
- Order of tuples and attributes does not matter.
  - A relation in 1 NF must have a key, key can be single attribute or composite keys .
  - The STUDENT relation is in 1 NF. Primary key of the relation is Rno and Cno.
  - 1 NF only defines the structure of the relation and does not resolve the database anomalies.

### **The Second Normal Form(2NF)**

- A relation is said to be in 2 NF if it is in 1 NF and every non-key attribute is fully dependent on each candidate key of the relation.
- A relation having single attribute key has to be in 2 NF.
- In the case of composite key, partial dependency is not allowed.
- 2NF it ensures that all information in a relation is about one thing.
- Non-key attributes are those that are not part of candidate key.

---

- Example: consider relation STUDENT

FDs are: Rno  $\rightarrow$  Sname, Address (1)

Cno  $\rightarrow$  Cname, Tutor (2)

Tutor  $\rightarrow$  Office (3)

Key attributes are: Rno+Cno

Non-key attributes are: rest of the attributes

FDs (1) and (2) relate to partial dependency on the key (Rno+Cno)

To convert the relation into 2NF we decompose the relation STUDENT.

On consider FD (1) Rno is unique and determines student name and address. First decomposed relation:

STUDENT<sub>1</sub>(Rno, Sname, Address)

---

- On considering FD (2) Cno uniquely identifies Cname and tutor. As per FD 3 tutor uniquely identifies office,

i.e. Cno  $\rightarrow$  Tutor

Tutor  $\rightarrow$  Office

so, Cno  $\rightarrow$  Office (transitive dependency)

So, FD 2 becomes Cno  $\rightarrow$  Cname, Tutor, Office

The second decomposed relation is:

COURSE(Cno, Cname, Tutor, Office)

---

Relation to join the decomposed relations:

STUDENT\_COURSE(Rno, Cno)----- In 2NF

So, the STUDENT relation in 2NF is: STUDENT<sub>1</sub>(Rno, Sname, Address)      2NF(a)

COURSE(Cno, Cname, Tutor, Office) 2NF(b)

STUDENT\_COURSE(Rno, Cno)    2NF(c)

---

### The Third Normal Form (3NF)

- A relation is in third normal form, if it is in 2NF and every non-key attribute of the relation is non-transitively dependent on each candidate key of the relation.

- Example:

Consider the relation 2NF(b)

COURSE(Cno, Cname, Tutor, Office)

Let us assume that Cname is not unique and Cno is the only candidate key.

---

FDs are: Cno->Tutor

Tutor->Office

Cno-> Office(transitive dependency)

Therefore, the above relation is not in 3NF. So this relation is further decomposed to resolve database anomalies.

COU(Cno, Cname, Tutor)

TUT(Tutor, Office)

So the relation in 3NF becomes:

STUDENT<sub>1</sub>(Rno, Sname, Address)

COU(Cno, Cname, Tutor)

TUT(Tutor, Office)

STUDENT\_COURSE(Rno, Cno)

© Indira Gandhi National Open University

---

### Boyce-Codd Normal Form(BCNF)

- A relation is in BCNF if, it is in 3NF and if every determinant is a candidate key.
- A determinant is left side of FD.
- Most relations that are in 3NF are also in BCNF. A 3NF relation is not in BCNF if all the following applies:
  - a) Candidate keys in the relation are composite keys.
  - b) There is more than one overlapping candidate keys in the relation, and some attributes in the keys are overlapping and some are not overlapping.
  - c) There is a FD from the non-overlapping attributes of one candidate key to non-overlapping attribute(s) of other candidate key.

- 
- Consider a relation NEWSTUDENT: NEWSTUDENT(Rno, Sno, Sname, Cno, Cname)

FDs:

Rno  $\rightarrow$  Sname (1)

Sname  $\rightarrow$  Rno (2)

Cno  $\rightarrow$  Cname (3)

Cname  $\rightarrow$  Cno (4)

The above relation is in 3NF, but not in BCNF. So, any one of the FDs (1) and (2); and any of the FDs (3) and (4) are decomposed.

STUD<sub>1</sub>(Rno, Sname)

COUR<sub>1</sub>(Cno, Cname)

Third relation that joins the two relations are: ST\_CO(Rno, Cno)

# Desirable Properties Of Decomposition

---

- Decomposition is a process of splitting a relation into its projection that will not be disjoint.
- Desirable properties of decomposition are :
  - Attribute preservation
  - Lossless-join decomposition
  - Dependency preservation
  - Lack of redundancy

Attribute preservation :

- It's a simple and obvious requirement. It involves preserving all the attributes that were there in the relation that is being decomposed.

### Lossless-Join Decomposition :

- A careless decomposition may lead to loss of information.
- A lossless or nonloss decomposition is that which guarantees that the join will result in exactly the same relation as was decomposed.
- Example :

Consider a relation : ENROL(Stno, cno, Date-enrolled, Room-no, Instructor)

St no	cno	Date-enrolled	Room-no	Instructor
1123	MCS-011	20-06-2004	1	Navyug
1123	MCS-012	26-09-2004	2	Anurag Sharma
1259	MCS-011	26-09-2003	1	Preeti Anand
1134	MCS-015	30-10-2005	5	Preeti Anand
2223	MCS-016	05-02-2004	6	Shashi Bhushan

- Suppose we decompose the above relation into two relations ENROL<sub>1</sub> and ENROL<sub>2</sub> :

ENROL<sub>1</sub>

St no	Cno	Date-enrolled
1123	MCS-011	20-06-2004
1123	MCS-012	26-09-2004
1259	MCS-011	26-09-2003
1134	MCS-015	30-10-2005
2223	MCS-016	05-02-2004

ENROL<sub>2</sub>

Date-enrolled	Room-no	Instructor
20-06-2004	1	Navyug
26-09-2004	2	Anurag Sharma
26-09-2003	1	Preeti Anand
30-10-2005	5	Preeti Anand
05-02-2004	6	Shashi Bhushan

- 
- All the relation that was in ENROL are still there in ENROL<sub>1</sub> and ENROL<sub>2</sub> but this is not so.
  - Suppose we want to retrieve the student numbers of all the students studying under Preeti Anand, we need to join the relations ENROL<sub>1</sub> and ENROL<sub>2</sub>.
  - The common attribute for joining is date-enrolled. So the resulting relation we get will not be the same as the original relation before decomposition. The resulting relation will contain a number of spurious tuples that were not in the original relation. Because of these additional tuples we will lose the right information(students studying under Preeti Anand ). Such decompositions are called lossy decompositions.
  - If the common attribute is the primary key of at least one of the decomposed relations, the problem of losing information will not exist. In the above example, date-enrolled is not a primary key. This arises because several enrolments happen on the same date.

---

### Dependency preservation :

- Dependency preservation is important requirement because dependency is a constraint on the database.
- If all the attributes appearing on the left and the right side of a dependency appears in the same relation, then dependency is said to be preserved.
- Dependency preservation can be checked easily.
- If a constraint is split over more than one relation, the constraint would be difficult to meet.
- A decomposition into 3NF is lossless and dependency preserving whereas a decomposition into BCNF is lossless but may or may not be dependency preserving.

### Lack of Redundancy :

- Repetition of data in a database is called data redundancy.
- Data redundancy may lead to data inconsistency. Controlled redundancy is sometimes important for the recovery in database system.

# Normalization

---

- Normalization is decomposition of a relation into smaller relations based on the concept of functional dependence to overcome the undesirable anomalies.
- Different normal forms :
  - 1NF
  - 2 NF
  - 3NF
  - BCNF
- The purpose of normalization is to avoid redundancy and to avoid undesirable anomalies.

- 
- There are several rules for Normalisation process:
    1. Eliminate Repeating Groups
    2. Eliminate Redundant Data
    3. Eliminate columns not dependent on key
    4. Isolate independent multiple relationships
    5. Isolate Semantically Related Multiple Relationships

---

Eliminate repeating groups :

Make a separate relation for each set of related attributes, and give each relation a primary key.

Eliminate redundant data :

If an attribute depends on only a part of attribute key, remove it to a separate relation.

Eliminate columns not dependent on key :

If attributes do not contribute to a description of the key, remove them to a separate relation.

Isolate independent multiple relationships :

No relation may contain two or more 1:n or n:m relationships that are not directly related.

Isolate semantically related multiple relationships :

There may be practical constraints on information that justify separating logically related many-to-many relationships.

- **Steps of normalization :**
- Consider a list of employees in the company. For time being we do not consider Department and Department-Loc till step 3.

Emp-ID	Emp-Name	Database-Known	Department	Department-Loc
1	Gurpreet Malhotra	Oracle,	A	N-Delhi
2	Faisal Khan	Access	A	N-Delhi
3	Manisha Kukreja	FoxPro	B	Agra
4	Sameer Singh	DB2, Oracle	C	Mumbai

## 1. Eliminate Repeating groups :

- Consider the query : Find out the list of employees who knows DB2.
- For finding this we need to make an awkward scan of the list looking for references to DB2. This is an inefficient and untidy way to retrieve information.
- We convert the relation into 1NF.

Employee Relation		Database Relation		
Emp-ID	Emp-Name	D-ID	Emp-ID	Database-name
1	Gurpreet Malhotra	1	2	Access
2	Faisal Khan	2	4	DB2
3	Manisha Kukreja	3	3	FoxPro
4	Sameer Singh	4	1	Oracle
		4	4	Oracle

- 
- In the above conversion Department and department-Loc will be in employee relation. Emp-ID is the primary key of Employee relation providing foreign key for relating the two relations with a join operation.

## 2. Eliminate Redundant Data :

- In the database relation, the primary key is made up of Emp-ID and D-ID. The database-name depends only on D-ID. The same database name will appear redundantly every time its associated D-ID appears in the database relation.
- The database relation has redundancy, for example D-ID value 4 is oracle is represented twice. Another problem is that we cannot enter Sybase in the relation as no employee has that skill. One another problem is deletion anomaly, if we remove employee with Emp-ID 3; no employee with FoxPro knowledge remains and the information that D-ID 3 is the code for FoxPro will be deleted.

- To avoid the above problems, we convert the relation to 2NF. This is done by isolating the attributes depending on both parts of the key from those depending only on the D-ID. This results in two relations :
  - ❑ Database : it gives the name for each D-ID.
  - ❑ Emp-database : it lists the databases for each employee.

Employee Relation	
Emp-ID	Emp-Name
1	Gurpreet Malhotra
2	Faisal Khan
3	Manisha Kukreja
4	Sameer Singh

Emp-database Relation	
Emp-ID	D-ID
2	1
4	2
3	3
1	4
4	4

Database Relation	
D-ID	Database
1	Access
2	DB2
3	FoxPro
4	Oracle

### 3. Eliminate Columns Not Dependent On Key :

- The employee relation satisfies :
  - ☐ First Normal Form : As it contains no repeating groups
  - ☐ Second Normal Form : As it doesn't have a multi-attribute key.
- The employee relation is in 2NF but not in 3NF. The final employee relation is :

Employee Relation			
Emp-ID	Emp-Name	Department	Department-Loc
1	Gurpreet Malhotra	A	N-Delhi
2	Faisal Khan	A	N-Delhi
3	Manisha Kukreja	B	Agra
4	Sameer Singh	C	Mumbai

- The key is Emp-ID and the Dept-Name and location describes only about department, not an employee.
- To achieve third normal form must be moved into a separate relation "Department" .
- This is done to avoid update, insertion and deletion anomalies.

Employee-List	
Emp-ID	Emp-Name
1	Gurpreet Malhotra
2	Faisal Khan
3	Manisha Kukreja
4	Sameer Singh

Department-Relation		
Dept-ID	Department	Department Loc
1	A	N-Delhi
2	B	Agra
3	C	Mumbai

- The last two steps : Isolate Independent Multiple Relationships and Isolate Semantically Related Multiple Relationships converts the relations to higher normal forms.
- **All BCNF is in 3NF, but all 3NF is not in BCNF. Justify.**

A schema  $R$  is in BCNF with respect to a set  $F$  of functional dependencies, if for all functional dependencies in  $F^+$  of the form  $\alpha \rightarrow \beta$ , where  $\alpha \subseteq R$  and  $\beta \subseteq R$ , at least one of the following holds:

- $\alpha \rightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$ ).
- $\alpha$  is a superkey for  $R$ .

A relational schema  $R$  is in 3NF if for every functional dependency  $\alpha \rightarrow \beta$  (where  $\alpha$  is a subset of the attributes and  $\beta$  is a single attribute) either

- $\alpha \rightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$ )
- $\alpha$  is a superkey
- $\beta$  is part of some key for  $R$ .

Thus the definition of 3NF permits a few additional functional dependencies involving key attributes that are prohibited by BCNF. O, we can say that *All BCNF is in 3NF, but all 3NF is not in BCNF.*

A 3NF relation may be in BCNF only if

- (a) the candidate keys in the relation are composite keys (that is, they are not single attributes),
- (b) there is more than one candidate key in the relation, and
- (c) the keys are not disjoint, that is, some attributes in the keys are common.

### Example

Patient No	Patient Name	Appointment Id	Time	Doctor
1	John	0	09:00	Zorro
2	Kerr	0	09:00	Killer
3	Adam	1	10:00	Zorro
4	Robert	0	13:00	Killer
5	Zane	1	14:00	Zorro

Lets consider the database extract shown above. From this scenario we can extract the following determinants:

DB(Patno,PatName,appNo,time,doctor)      and      Patno  $\rightarrow$  PatName  
Patno,appNo  $\rightarrow$  Time,doctor  
Time  $\rightarrow$  appNo

Now we have to decide what the primary key of DB is going to be. From the information we have, we could chose:

DB(Patno,PatName,appNo,time,doctor)

**DB(Patno,PatName,appNo,time,doctor)**

a) 1NF Eliminate repeating groups.

None: so just as before

b) 2NF Eliminate partial key dependencies

DB(Patno,appNo,time,doctor)  
R1(Patno,PatName)

c) 3NF Eliminate transitive dependencies

None: so just as 2NF

d) BCNF: Every determinant is a candidate key

DB(Patno,appNo,time,doctor)

R1(Patno,PatName)

BCNF: rewrite to  
DB(Patno,time,doctor)  
R1(Patno,PatName)  
R2(time,appNo)

This example has demonstrated three things:

- BCNF is stronger than 3NF, relations that are in 3NF are not necessarily in BCNF
- BCNF is needed in certain situations to obtain full understanding of the data model
- there are several routes to take to arrive at the same set of relations in BCNF.
- Unfortunately there are no rules as to which route will be the easiest one to take.