UNIT 3. NORMALIZATION

- Purpose of normalization:
- Let us consider the relation EMPLOYEE_BRANCH:

Table 6.1.1 : An example considered for data duplication

EMPLOYEE_BRANCH				
EMPNO	NAME	SALARY	BRANCH_NO	BRANCH_ADDRESS
105	MOHAN	15000	B001	Park Street, Calcutta
108	SOHAN	21000	B001	Park Street, Calcutta
109	RUCHIKA	29000	B002	Nehru Place, Delhi
115	SOURABH	18000	B001	Park Street, Calcutta
116	MITALEE	35000	B002	Nehru Place, Delhi

- There is duplication of data. Branch address is repeated for every employee working at that branch.

A normalized relation without redundancy :

The problem of data duplication can be eliminated by decomposing the relation EMPLOYEE_BRANCH into two relations :

1. EMPLOYEE
2. BRANCH

Table 6.1.2 : A normalized relation without duplicate information

EMPLOYEE			
EMPNO	**NAME**	**SALARY**	**BRANCH_NO**
105	MOHAN	15000	B001
108	SOHAN	21000	B001
109	RUCHIKA	29000	B002
115	SOURABH	18000	B001
116	MITALEE	35000	B002

BRANCH	
BRANCH_NO	**BRANCH_ADDRESS**
B001	Park street, Calcutta
B002	Nehru place, Delhi

- After decomposition, branch address appears only once. It is shown in Table. 6.1.2.
- BRANCH_NO is repeated in the relation EMPLOYEE to represent where each member of employee is located. This can not be avoided.

Anomalies in a database

6.2 Anomalies in a Database :

A relation that has redundant data may have update anomalies. These anomalies are classified as :

1. Insertion anomalies
2. Deletion anomalies
3. Modification anomalies

The concept of insertion anomaly can be understood with the help of the relation EMPLOYEE_BRANCH. There is an association between an employee and the branch he is located. A new branch without any employee in it can not be entered in the database as the primary key is EMPNO and it can not be null. But this problem will not occur after normalization.

- Un-normalized relation EMPLOYEE_BRANCH has following attributes.

  (EMPNO, NAME, SALARY, BRANCH_NO, BRANCH_ADDRESS)

- After normalization, the EMPLOYEE_BRANCH is decomposed into two relations :

  1. EMPLOYEE with the following attributes

     (EMPNO, NAME, SALARY, BRANCH_NO)

  2. BRANCH with the following attributes

     (BRANCH,_NO, BRANCH_ADDRESS)

Case I : Inserting a new branch with

BRANCH_NO = B003

BRANCH_ADDRESS = Sector 17, Chandigarh.

In un-normalized relation EMPLOYEE_BRANCH, the table after insertion is shown in Table 6.2.1.

Table 6.2.1 : Example for insertion anomaly

EMPLOYEE_BRANCH				
EMPNO	NAME	SALARY	BRANCH_NO	BRANCH_ADDRESS
105	MOHAN	15000	B001	Park street, Calcutta
108	SOHAN	21000	B001	Park street, Calcutta
109	RUCHIKA	29000	B002	Nehru place, Delhi
115	SOURABH	18000	B001	Park street, Calcutta
116	MITALEE	35000	B002	Nehru place, Delhi
Null	Null	Null	B003	Sector 17, Chandigarh

This insertion will not be allowed, as the primary key value can not be null.

- The insertion will not be allowed as the key attribute EMPNO can not have null value. Thus an un-normalized relation suffers from insertion anomaly.

Case II : Inserting a new branch with

BRANCH_NO = B003

BRANCH_ADDRESS = Sector 17, Chandigarh

in normalized relation with two relations :

1. EMPLOYEE
2. BRANCH

The tables after insertion are shown in Table 6.2.2.

Table 6.2.2 : Example for insertion without insertion anomaly

EMPLOYEE			
EMPNO	NAME	SALARY	BRANCH_NO
105	MOHAN	15000	B001
108	SOHAN	21000	B001
109	RUCHIKA	29000	B002
115	SOURABH	18000	B001
116	MITALEE	35000	B002

BRANCH	
BRANCH_NO	BRANCH_ADDRESS
B001	Park street, Calcutta
B002	Nehru place, Delhi
B003	Sector 17, Chandigarh

New entry. It is allowed after decomposition

- After normalization (decomposition), there is no insertion anomaly.
- An employee joining a branch can be inserted in the table EMPLOYEE.

6.2.2 Deletion Anomaly :

If the only employee located in branch discontinues, information about the branch will be lost as this is the only row showing the association between the employee and the branch he is located.

Case I : Deletion of a record of the employee with EMPNO = 118 from un-normalized relation EMPLOYEE_BRANCH is shown in Table 6.2.3.

Table 6.2.3 : Example for deletion anomaly

EMPLOYEE_BRANCH				
EMPNO	**NAME**	**SALARY**	**BRANCH_NO**	**BRANCH_ADDRESS**
105	MOHAN	15000	B001	Park street, Calcutta
108	SOHAN	21000	B001	Park street, Calcutta
109	RUCHIKA	29000	B002	Nehru place, Delhi
115	SOURABH	18000	B001	Park street, Calcutta
116	MITALEE	35000	B002	Nehru place, Delhi
118	UMANG	26000	B003	Sector 17, Chandigarh

After deletion of the record, information about the branch B003 will be lost.

Case II : Deletion of a record of the employee with EMPNO = 118 from normalized relation EMPLOYEE is shown in Table 6.2.4.

Table 6.2.4 : Example for deletion without anomaly

EMPLOYEE			
EMP NO	**NAME**	**SALARY**	**BRANCH_NO**
105	MOHAN	15000	B001
108	SOHAN	21000	B001
109	RUCHIKA	29000	B002
115	SOURABH	18000	B001
116	MITALEE	35000	B002
118	UMANG	26000	B003

BRANCH	
BRANCH_NO	**BRANCH_ADDRESS**
B001	Park street, Calcutta
B002	Nehru place, Delhi
B003	Sector 17, Chandigarh

Deletion of the record will have no effect on information about the branch B003.

... deletion anomaly.

6.2.3 Modification Anomaly :

In the EMPLOYEE_BRANCH, if we change the address of a branch of a particular employee, we must update the tuples of all employees who work in that branch. Otherwise, the database will become inconsistent. If we fail to do this, some branch will have more than one addresses.

- A database should be designed so that it has no insertion, deletion or modification anomalies. A database without any anomaly will work correctly and it will always be consistent.

6.3 Functional Dependency :

PU - Dec. 2005

Functional dependency describes the relationship between attributes. Functional dependency arises naturally in many ways. If R represents a relation whose attributes are A_1 , A_2 , ... A_n and X is a set of attributes that uniquely determines the value of the attributes in Y then we say :

X functionally determines Y and it is denoted by $X \rightarrow Y$. X is called the **determinant** of the functional dependency (FD) and the FD is denoted by $X \rightarrow Y$.

Example : Let us consider the relation EMPLOYEE_BRANCH as given in Table 6.3.1.

Table 6.3.1 : Example for functional dependency

EMPLOYEE_BRANCH				
EMPNO	**NAME**	**SALARY**	**BRANCH_NO**	**BRANCH_ADDRESS**
105	MITALEE	15000	B001	Park street, Calcutta
108	SOURABH	21000	B001	Park street, Calcutta
109	RUCHIKA	29000	B002	Nehru place, Delhi
115	SOURABH	18000	B001	Park street, Calcutta
116	MITALEE	35000	B002	Nehru place, Delhi
118	UMANG	18000	B003	Sector 17, Chandigarh

Relation scheme EMPLOYEE_BRANCH is given by :

EMPLOYEE_BRANCH (EMPNO, NAME, SALARY, BRANCH_NO, BRANCH_ADDRESS)

In the given Table 6.3.1, we can uniquely determine the details of an employee with EMPNO = 109 (say). This will be done by locating the only row containing

EMPNO = 109. Thus, if the EMPNO of an employee is known then the following details about an employee can be uniquely obtained.

1. NAME
2. SALARY
3. BRANCH_NO
4. BRANCH_ADDRESS

In terms of functional dependencies we can say :

EMPNO → NAME

EMPNO → SALARY

EMPNO → BRANCH_NO

EMPNO → BRANCH_ADDRESS

These functional dependencies can be combined together :

EMPNO → NAME, SALARY, BRANCH_NO, BRANCH_ADDRESS.

Following associations can be additionally derived from Table 6.3.1.

1. BRANCH_NO can be used to find the BRANCH_ADDRESS.

   BRANCH_NO → BRANCH_ADDRESS
2. NAME can not be used to find the other attributes of an employee. Two employees with the same NAME can have different EMPNO.

Following dependencies hold for the relation EMPLOYEE_BRANCH :

1. EMPNO → NAME, SALARY, BRANCH_NO, BRANCH_ADDRESS
2. BRANCH_NO → BRANCH_ADDRESS.

These functional dependencies can also be shown as given in Fig. 6.3.1.

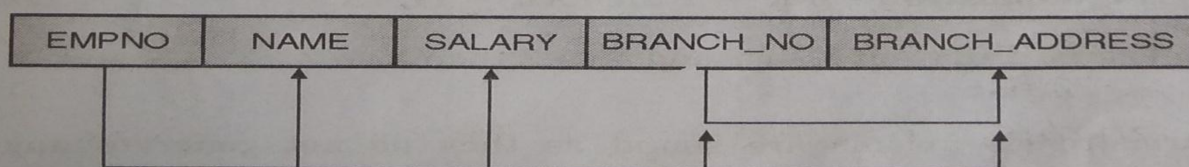


Fig. 6.3.1 : Pictorial representation of functional dependencies

Example :

Let us consider a relation scheme R(CITY, STREET, PINCODE).

- CITY and STREET determines the PINCODE.
- PINCODE determines the city but not the STREET.

Thus, there are following dependencies :

(CITY, STREET) → PINCODE

PINCODE → CITY

6.7 Decomposition :

PU - Dec. 2010, May 2011, Dec. 2011, May 2012

Decomposition on a relation scheme R is carried out to eliminate anomalies contained in R. Decomposition of a relation scheme involves splitting R into a collection of relation schemes.

- Decomposition should preserve the original information contained in R.
- Join of the decomposed relation should give the same set of tuples as the original relation.
- Dependencies of original relation should be preserved in decomposed relations.

A careless decomposition may have the following problems :

1. Lossy (loss of tuples)
2. Loss of dependencies.

Example :

Let us consider a relation R (Name, Deptt, Advisor)

With following functional dependencies :

Name → Deptt

Name → Advisor

Advisor → Deptt

Now, the relation R is decomposed as given below :

R1 (Name, Deptt)

R2 (Deptt, Advisor)

Tables associated with R, R1 and R2 are shown in Table 6.7.1 (a to c)

Table 6.7.1 : Example of lossy decomposition

R		
Name	**Deptt.**	**Advisor**
Vivek	Comp	James
Negi	ECE	Clark
Deepak	Civil	Smith
Pradeep	Comp	Brown
Puja	ECE	Black
Snehal	Comp	James

(a)

R1	
Name	**Deptt.**
Vivek	Comp
Negi	ECE
Deepak	Civil
Pradeep	Comp
Puja	ECE
Snehal	Comp

(b)

R2	
Deptt.	**Advisor**
Comp	James
ECE	Clark
Civil	Smith
Comp	Brown
ECE	Black

(c)

Let us see if the three dependencies are preserved in decomposition shown in
Table 6.7.1 (a to c).

1. Name → Deptt. – It is preserved as the relation **R1** contains both **Name**, **Deptt**
2. Name → Advisor – It is lost as neither R1 nor R2 contain both Name and **Advisor**.
3. Advisor → Deptt. – It is preserved as the relation **R2** contains both **Deptt.** and **Advisor**.

Thus the decomposition shown in Table 6.7.1, does not preserve dependency.

To check, whether the decomposition is lossy, **R1** and **R2** are combined into a new table **R3**.

- If **R** and **R3** are identical then the decomposition is **lossless**.

- If R and R3 are not identical then the decomposition is lossy.

The table **R3** is shown in Table 6.7.2.

Table 6.7.2 : The combined table does not result in original table R

R3		
Name	**Deptt.**	**Advisor**
Vivek	Comp	James
Vivek	Comp	Brown
Negi	ECE	Clark
Negi	ECE	Black
Deepak	Civil	Smith
Pradeep	Comp	James
Pradeep	Comp	Brown
Puja	ECE	Clark
Puja	ECE	Black
Snehal	Comp	James
Snehal	Comp	Brown

These tuples are not in R

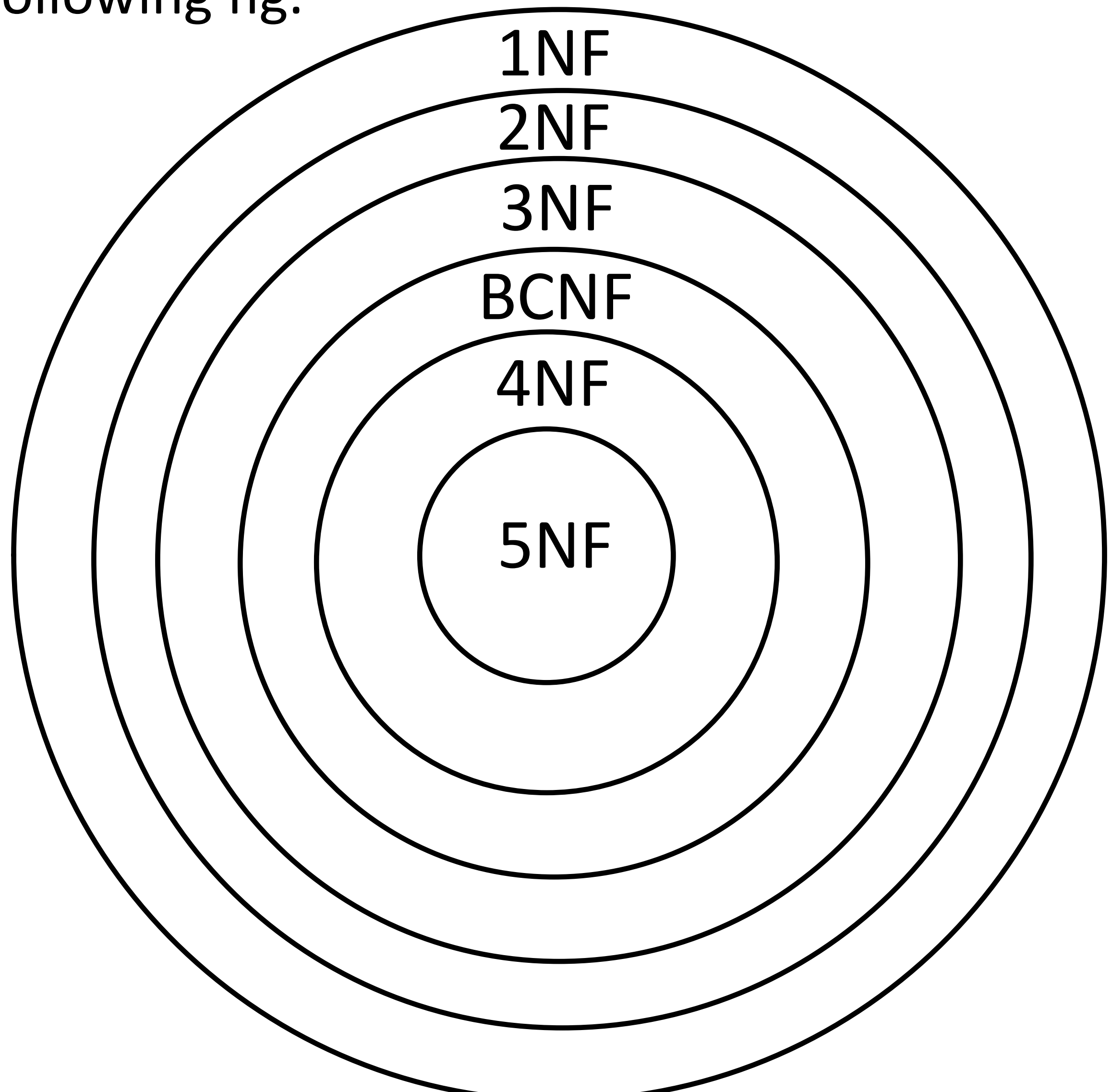
The Join of **R1** and **R2** contains tuples that did not exist in the original relation **R**. Thus the decomposition is **lossy**.

- Normalization is a process of decomposing a relation into smaller relations to achieve:
  - 1. Minimizing Redundancy
  - 2.Minimizing the insertion, updation and deletion anomalies.
- The main purpose of normalization is to reduce redundancy.
- Updates to the database with redundancies may become inconsistent.
- Information in a table should be stored only once.
- Duplication leads to waste of storage space.

- Normalization of a relation schema is based upo:
    1. Functional dependencies
    2. Keys
- Normalization is carried out in stages. There are successive higher normal forms.
- Each normal form is an improvement over the earlier NF.
- These are:
    1. First Normal Form (1NF)
    2. Second Normal Form (2NF)
    3. Third Normal Form (3NF)
    4. Boyce-Codd Normal Form (BCNF)
    5. Fourth Normal Form (4NF)
    6. Fifth Normal Form (5NF)

- First Normal Form (1NF) is for tabular representation of a relation. Each attribute of a tuple must be atomic.
- Second Normal Form (2NF), Third Normal Form (3NF) & Boyce- Codd Normal Form (BCNF) are based on dependency among attributes in a relation.
- Fourth Normal Form (4NF) & Fifth Normal Form (5NF) are for multi-valued dependency among attributes.

- A higher normal form relation is a subset of lower form as shown in following fig.

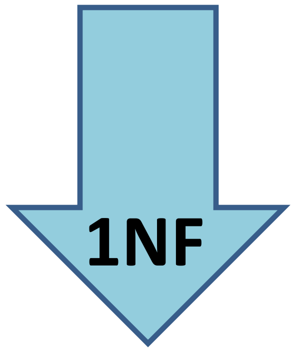


First Normal Form (1NF)

- A domain is atomic if elements of the domain are considered to be indivisible units.
- We say that a relation schema R is in First Normal Form if the domains of all attributes of R are atomic.
- A relation schema is said to be in 1NF if the value of each attribute in each tuple is atomic.
- In other words, only one value is associated with each attribute and the value is not a set of values or a list of values.

Emp-Code	Name	Dependants
105	Ramesh	Deepak,Amit
107	Akbar	Javed,Salim
108	Mohan	Deepika

The given relation can be brought to 1NF by introducing a separate tuple for each dependent.

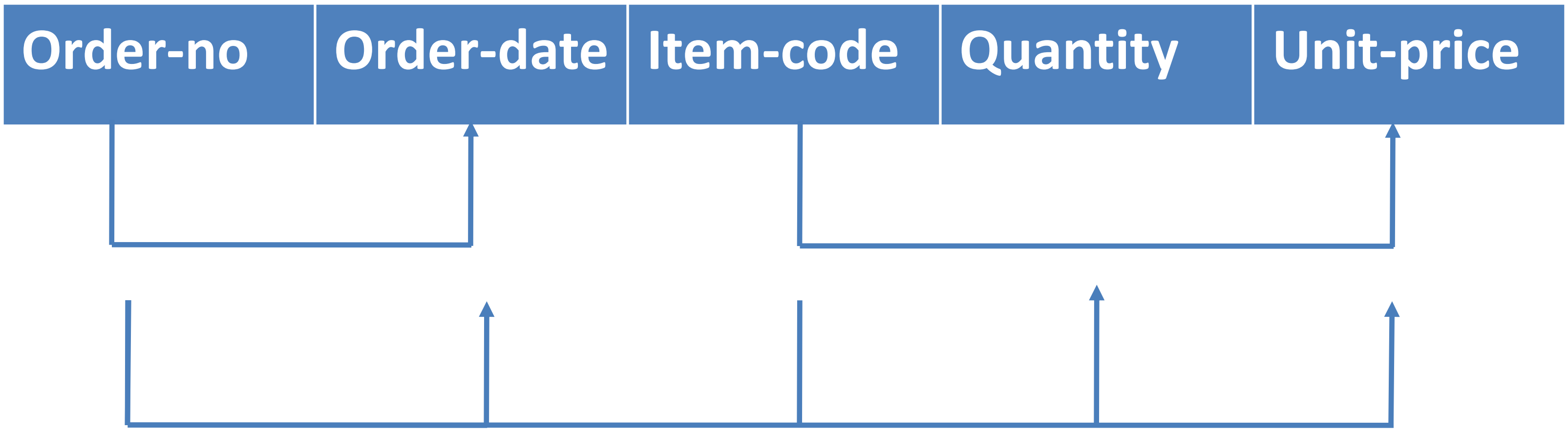


Emp-Code	Name	Dependants
105	Ramesh	Deepak
105	Ramesh	Amit
107	Akbar	Javed
107	Akbar	Salim
108	Mohan	Deepika

Second Normal Form (2NF)

- A relation is said to be in Second normal form if:
  - It is in 1NF.
  - Non-key attributes should be fully dependent on the key attributes.
  - In case of composite key (key containing multiple attributes) , no non-key attribute should be functionally dependent on a part of the key attribute.

- An example that is not in 2NF:
- Let us consider a relation R with following attributes:
- R = (order-no, order-date, item-code, quantity, unit-price)set
- The set of functional dependencies for R is shown in following Fig:



- These functional dependencies can also be written as:

  order-no→ order-date

  item-code→ unit-price

  (order-no, item-code)→ order-date, quantity, unit-price

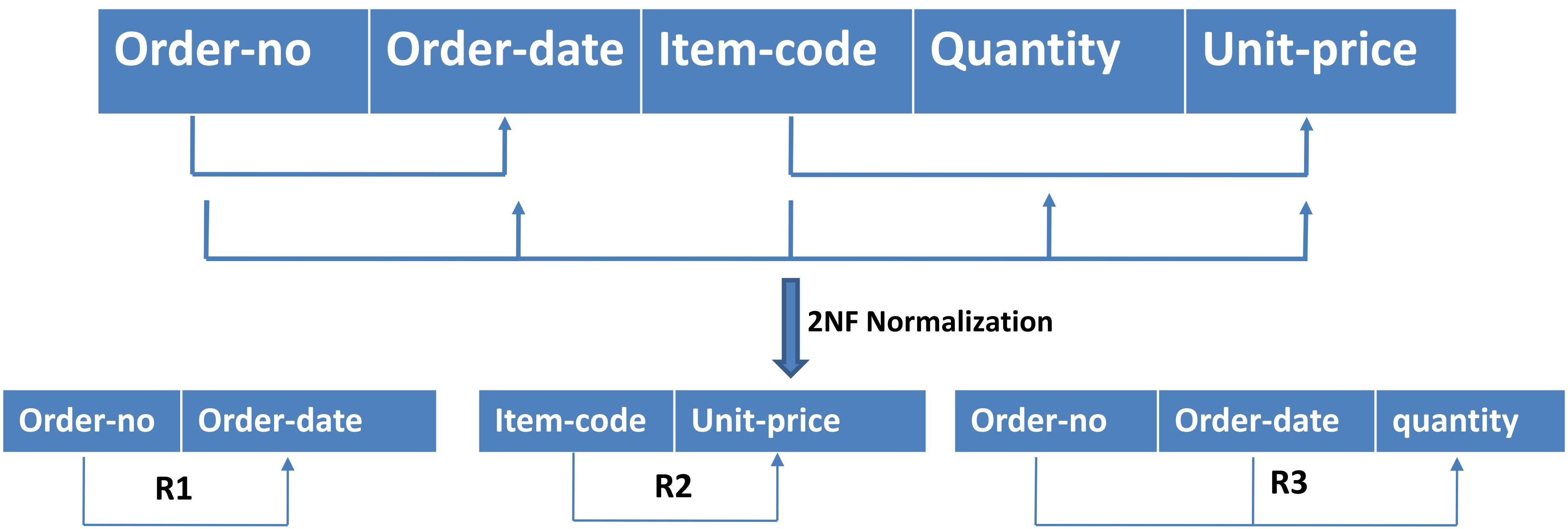
- The relation R has the composite key (order-no, item-code)

- The non-key attribute unit-price is functionally dependent on item-code. Item-code is part of the composite key (order-no, item-code).

- The non-key attribute order-date is functionally dependent on order-no. order-no is part of the composite key (order-no, item-code)

Bringing the relation R in to 2NF

- If a relation schema is not in 2NF, then it can be decomposed into a number of 2NF relations in which non-key attributes are fully functionally dependent on the key and not its part.
- A 2NF decomposition of relation R is shown in following fig:



2NF normalization of relation R, by splitting it into R1, R2 & R3

Third Normal Form (2NF)

- A relation is said to be in Third normal form (3NF) if:
  - It is in 2NF.
  - No non-key attributes is functionally dependent on another non-key attribute (or a set of non-key attributes).
  - This also implies that there should be no transitive dependency of a non-key attribute on the primary key.

- An example that is not in 3NF:
- Let us consider a relation R with following attributes:
- R = (rollno, name, department, year, hostel) with following dependencies:
- Rollno $\rightarrow$ (name, department, year)
- Year $\rightarrow$ hostel
- A table for R is given

Rollno	Name	Department	Year	Hostel
101	Abhishek	Electronics	1	Kaveri
105	Arvind	Computer	2	Godavari
107	Maya	Electrical	1	Kaveri
109	Santosh	Mechanical	3	Krishna
125	Singh	Computer	2	Godavari

- rollno is the key and all the other attributes are functionally dependent on it.
- The relation R is in 2NF.
- Non-key attribute hostel is functionally dependent on the non-key attribute year.
- The relation R is not in 3NF as the non-key attribute hostel is functionally dependent on the non-key attribute year.

- As the relation R is not in 3NF, there is data redundancy (duplication) for the attribute hostel.
- To transform the relation R into 3NF, we should split the relation R into R1 and R2 such that the derived relations R1 and R2 should not have functionally related non-key attributes.

R1=(rollno, name, department, year)

R2-=(year,hostel)

R1

Rollno	Name	Department	Year
101	Abhishek	Electronics	1
105	Arvind	Computer	2
107	Maya	Electrical	1
109	Santosh	Mechanical	3
125	Singh	Computer	2

R2

Year	Hostel
1	Kaveri
2	Godavari
3	Krishna

Boyce-codd Normal Form (BCNF)

- A relation is said to be in BCNF if, the left side of every functional dependency is a candidate key.
- Left side of a functional dependency is also known as the determinant.
- **BCNF is stronger than 3NF:**
  - If there is a functional dependency X→ Y, where Y is a prime attribute and X is not a candidate key:
    1. it is allowed in 3NF
    2. it is not allowed in BCNF.
- Therefore BCNF is stronger form of 3NF.
- Every relation in BCNF is also in 3NF, however a relation in 3NF is not necessarily in BCNF.

- Let us consider a relation R = (student, course, teacher) with following functional dependencies.

  (student, course) → teacher

  Teacher→ course

- (student, course) is a candidate key for this relation.
- The relation is in 3NF as there is no association among non-key attributes.
- The relation is not in BCNF as the prime attribute course is functionally dependent on teacher.

- Decomposition of this relation so that it satisfies BCNF is not straight forward.
- A BCNF decomposition may have the following problems:
    1. Lossy
    2. Loss of functional dependencies
- In general, a relation not in BCNF should be decomposed to meet the following properties.
    1. decomposition should be lossless.
    2. Decomposition should be dependency preserving.
- In some cases, it may not be possible to preserve dependencies, but the decomposition must be lossless.

- Consider a relation R = (student, course, teacher) with following functional dependencies.

  (student, course) → teacher

  Teacher→ course

- The relation is in 2NF. Only non-key attribute teacher is functionally dependent on the key {student, course}.

- The relation is in 3NF. No non-key attribute is functionally dependent on another non-key attribute.

- The relation is not in BCNF. The key attribute course is functionally dependent on teacher.

- The relation can be decomposed in one of the following ways:

    1. {teacher , course} and {teacher, student}

    2. {student, teacher} and {student, course}

    3. {course, teacher} and {course, student}

- Out of these 3 the first is lossless as the common attribute teacher is a key for {teacher, course}.

- All the 3 decompositions, loose the functional dependency {student, course → teacher}.

- Hence the first decomposition should be accepted as it is lossless.

Already in 2NF & 3NF

{ student, course, teacher }

R1 {teacher , course}

R2 {teacher, student}

BCNF Decomposition

Fourth Normal Form (4NF)

- Normalizations as per 2NF, 3NF and BCNF are based on functional dependencies.
- Another type of dependency called as multi-valued dependency (MVD) can also cause insertion, deletion and update anomalies.
- 4NF has been defined for relation schemas that have both FDs and MVDs.
- 4NF imposes constraints on the MVDs allowed in the relation scheme.
- 4NF is stronger than BCNF.

❖Multi-valued dependency:

✓For example, an employee E1 works on two projects P1 & P2.

✓Employee E1 has two dependents D1 & D2.

✓The relation is as shown:

Employee	Project	Dependent
E1	P1,P2	D1,D2

✓The relation shown in table has two multi-valued attributes.

✓The relation can be converted to 1NF as:

Employee	Project	Dependent
E1	P1	D1
E1	P1	D2
E1	P2	D1
E1	P2	D2

✓ The constraint that we have to repeat each value of one of the attributes with every value of other attribute is due to multi-valued dependency.

✓Multi-valued dependency results in data redundancy.

✓Suppose the employee is assigned one more project P3, though there is no any direct connection between the project and the dependents we must create tuple for every combination of E1and its dependents D1 and D2 to maintain consistency.

Employee	**Project**	**Dependent**
E1	P1	D1
E1	P1	D2
E1	P2	D1
E1	P2	D2
E1	P3	D1
E2	P3	D2

✓ There is multi-valued dependency on project and employee.

✓ MVD is represented using following notation:

Employee→→Project

✓ There is a multi-valued dependency of Dependent on Employee

Employee→→Dependent

✓ A multi-valued dependency can be further defined as:

1. Trivial MVD

2. Non-trivial MVD.

- ✓ A multi-valued dependency X→Y in relation R is defined as trivial if:

1. Y is a subset of X

Or

2. X U Y = R

- ✓ A multi-valued dependency that is not trivial is called as non-trivial MVD.

- ✓ **Fourth Normal Form (4NF):** A relation that is in BCNF and contains no non-trivial multi-valued dependencies is in 4NF.

✓Example: Assume that a sports club keeps records of its members.

✓For each member it keeps a list of sports that the member is most interested in and a list of most striking characteristics of each member which are inherent in the members and are in no way dependent on the sport.

MEMBER-INTEREST-CHARACTERISTIC

Member	Interest	Characteristic
Rao	Hockey	High Stamina
Rao	Hockey	Poor Speed
Rao	Soccer	High Stamina
Rao	Soccer	Poor Speed
Das	Soccer	High Strength
Das	Soccer	High Speed
Das	Cricket	High Strength
Das	Cricket	High Speed
Das	WaterPolo	High Strength
Das	WaterPolo	High Speed

✓ According to the above structure, if another interest e.g. Cycling is added for Mr. Rao, then two more additional rows will be added – one for 'High Stamina' and one for 'Poor Speed'.

✓The above relation can be normalized further to get two more relations to reduce the BCNF relation to 4NF as:

MEMBER-INTEREST

Member	Interest
Rao	Hockey
Rao	Soccer
Das	Soccer
Das	Cricket
Das	WaterPolo

MEMBER-CHARACTERISTIC

Member	Characteristic
Rao	High Stamina
Rao	Poor Speed
Das	High Strength
Das	High Speed

Fifth Normal Form (5NF)

- It Is also called as “Project Join Normal Form” (PJNF).
- Decomposition of a relation into two relations should have the lossless join.
- In some cases it becomes a requirement to decompose a relation into more than two relations.
- This is required to avoid spurious tuples that are generated when relations are remitted through a natural join operation.
- Let us consider a vendor-items-project relation as shown.

Vendor	Items	Project
S1	I1	P1
S1	I2	P2
S2	I1	P2
S3	I2	P3
S2	I3	P1
S2	I1	P1
S1	I1	P2

✓ The relation shown in table has no MVDs.

✓ The vendor S1 supplies two items I1 and I2 to two projects P1 & P2 For an MVD this will create four tuples.

✓Suppose there is an additional constraint that whenever a vendor S supplies Item I and a project P uses item I, and the vendor S supplies at least one part to project P, then vendor S will also be supplying item I to project P.

✓ To bring the given relation into 5NF, it must be broken down into 3 relations :

1. R1 (Vendor, Items)
2. R2 (Vendor, Project)
3. R3 (Items, Project)

Vendor	Items
S1	I1
S1	I2
S2	I1
S3	I2
S2	I3

Vendor	Project
S1	P1
S1	P2
S2	P2
S3	P3
S2	P1

Item	Project
I1	P1
I2	P2
I1	P2
I2	P3
I3	P1

✓ The natural join of R1 and R2 will provide many spurious tuples which can be removed with the help of the third relation R3.