Normalization for Refinement of Database

By
Parteek Bhatia
Faculty, Deptt of Comp Sc & Engg
Thapar University
Patiala
parteek_bhatia@hotmail.com

Objectives of Normalization

- To create a formal framework for analyzing relation schemas based on their keys and on the functional dependencies among their attributes.
- To obtain powerful relational retrieval algorithms based on a collection of primitive relational operators.
- To free relations from undesirable insertion, update and deletion anomalies.
- To reduce the need for restructuring the relations as new data types are introduced.
- To carry out series of tests on individual relation schema so that the relational database can be normalized to some degree. When a test fails, the relation violating that test must be decomposed into relations that individually meet the normalization tests.

First N/F	Second N/F	Third N/F	Fourth N/F	Fifth N/F
Eliminate Repeating Groups	Non-key attribute Fully Functional Dependence on PK	Eliminate Transitive Dependence on PK	Remove Multi Value Dependency	Projection Join Dependency

Functional Dependence (FD)

A functional dependency is an association between two attributes of the same relational database table. One of the attributes is called the determinant and the other attribute is called the determined. For each value of the determinant there is associated one and only one value of the determined.

If A is the determinant and B is the determined then we say that *A functionally determines B* and graphically represent this as A→B. The symbols A→B can also be expressed as *B is functionally determined by* A.

Functional Dependence (FD)

- The following table illustrates A → B:

A	B
1	1
2	4
3	9
4	16
2	4
7	9

Functional Dependence (FD)

- The following table illustrates that A does not functionally determine B:

A	B
1	1
2	4
3	9
4	16
3	10

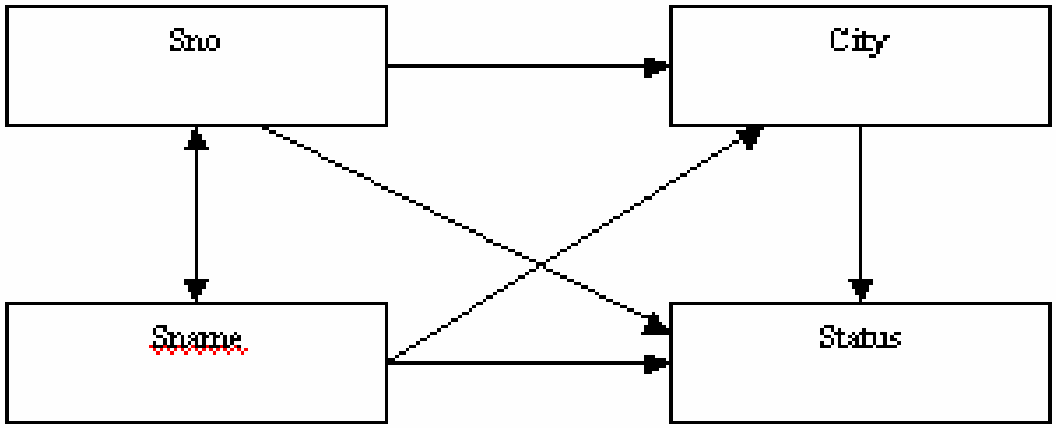
Since for A = 3 there is associated more than one value of B.

Functional dependency can also be defined as follows:

An attribute in a relational model is said to be functionally dependent on another attribute in the table if it can take only one value for a given value of the attribute upon which it is functionally dependent.

Dependency Diagrams

A dependency diagram consists of the attribute names and all functional dependencies in a given table. The dependency diagram of Supplier table is



Here following functional dependencies exist in supplier table

Sno→Sname
Sname→Sno
Sno→City
Sno→Status
Sname→City
Sname→Status
City→Status

Fully Functional Dependence (FFD)

Fully Functional Dependence(FFD) is defined as Attribute Y is FFD on attribute X , if it is FD on X and not FD on any proper subset of X. For example, in relation Supplier, different cities may have the same status. It may be possible that cities like Amritsar, Jalandhar may have the same status 10.

So, the City is not FD on Status

But, the combination of Sno,Status can give only one corresponding City because Sno is unique. Thus,

(Sno, Status) → City

It means city is FD on composite attribute (Sno,Status) however City is not fully functional dependent on this composite attribute,

Fully Functional Dependence (FFD)

Consider the another case of SP table:

Here, Qty is FD on combination of Sno, Pno.

Here, X has two proper subsets Sno and Pno

Qty is not FD on Sno, because one Sno can supply more than one quantity.

Qty is also not FD on Pno, because one Pno may be supplied many times by different suppliers with different or same quantities.

So, Qty is FFD on composite attribute of (Sno, Pno).

First Normal Form

Definition of First Normal Form

A relation is said to be in First Normal Form (1NF) if and only if every entry of the relation (the intersection of a tuple and a column) has at most a single value. In other words “a relation is in First Normal Form if and only if all underlying domains contain atomic values or single value only.”

STUDENT (Unnormalized table)

Course_ Code	Course_Name	Teacher_ Name	RollNo	Name	System_Used	Hourly_ Rate	Total _Hrs

STUDENT (Unnormalized table)

Course_ Code	Course_Name	Teacher_ Name	RollNo	Name	System_Used	Hourly_ Rate	Total _Hrs
C1	Visual Basic	ABC	100	A1	PentiumI	20	7
			101	A2	PentiumII	30	3
			102	A3	Celeron	10	6
			103	A4	PentiumIV	40	1
C2	Oracle&Dev	DEF	100	A1	P-I	20	7
			104	A5	P-III	35	3
			105	A6	P-II	30	1
			101	A2	P-II	30	2
C3	C++	KJP	106	A7	P-IV	40	3
			107	A8	P-IV	40	2
			108	A9	P-I	20	1
C4	Java	Kumar	109	A10	Cyrix	20	2

First Approach: Flattening the table

The first approach known as “flattening the table” removes repeating groups by filling in the “missing” entries of each “incomplete row” of the table with copies of their corresponding non-repeating attributes. The following example illustrates this.

STUDENT (Normalized Relation)

Course_ Code	Course_Name	Teacher_ Name	RollNo	Name	System_Used	Hourly _ Rate	Total _Hrs
C1	Visual Basic	ABC	100	A1	PentiumI	20	7
C1	Visual Basic	ABC	101	A2	PentiumII	30	3
C1	Visual Basic	ABC	102	A3	Celeron	10	6
C1	Visual Basic	ABC	103	A4	PentiumIV	40	1
C2	Oracle&Dev	DEF	100	A1	P-I	20	7
C2	Oracle&Dev	DEF	104	A5	P-III	35	3
C2	Oracle&Dev	DEF	105	A6	P-II	30	1
C2	Oracle&Dev	DEF	101	A2	P-II	30	2
C3	C++	KJP	106	A7	P-IV	40	3
C3	C++	KJP	107	A8	P-IV	40	2
C3	C++	KJP	108	A9	P-I	20	1
C4	Java	Kumar	109	A10	Cyrix	20	2

Second Approach: Decomposition of the table

The second approach for normalizing a table requires that the table be decomposed into two new tables that will replace the original table.

However, before decomposing the original table it is necessary to identify an attribute or a set of its attributes that can be used as table identifiers.

Rule of decomposition

- One of the two tables contains the table identifier of the original table and all the non-repeating attributes.

- The other table contains a copy of the table identifier and all the repeating attributes.

To normalize the STUDENT table we need to replace it by two new tables. The first table COURSE contains the table identifier and the non-repeating groups.These attributes are Course_Code (the table identifier), Course_Name, and Teacher_Name.

The second table contains the table identifier and all the repeating groups. Therefore, the attributes of COURSE_STUDENT table are Course_Code, Rollno, Name, System_Used, Hourly_Rate and Total_Hrs.

COURSE

Course_Code	Course_Name	Teacher_Name
C1	Visual Basic	ABC
C2	Oracle&Dev	DEF
C3	C++	KJP
C4	Java	Kumar

COURSE_STUDENT

Course_ Code	Rollno	Name	System_Used	Hourly_Rate	Total_Hrs
C1	100	A1	PentiumI	20	7
C1	101	A2	PentiumII	30	3
C1	102	A3	Celeron	10	6
C1	103	A4	PentiumIV	40	1
C2	100	A1	P-I	20	7
C2	104	A5	P-III	35	3
C2	105	A6	P-II	30	1
C2	101	A2	P-II	30	2
C3	106	A7	P-IV	40	3
C3	107	A8	P-IV	40	2
C3	108	A9	P-I	20	1
C4	109	A10	Cyrix	20	2

Anomalies in 1NF Relations (Considering STUDENT table)

STUDENT (Normalized Relation)

Course_ Code	Course_Name	Teacher_ Name	RollNo	Name	System_Used	Hourly _ Rate	Total _Hrs
C1	Visual Basic	ABC	100	A1	PentiumI	20	7
C1	Visual Basic	ABC	101	A2	PentiumII	30	3
C1	Visual Basic	ABC	102	A3	Celeron	10	6
C1	Visual Basic	ABC	103	A4	PentiumIV	40	1
C2	Oracle&Dev	DEF	100	A1	P-I	20	7
C2	Oracle&Dev	DEF	104	A7	P-III	37	3
C2	Oracle&Dev	DEF	107	A6	P-II	30	1
C2	Oracle&Dev	DEF	101	A2	P-II	30	2
C3	C++	KJP	106	A7	P-IV	40	3
C3	C++	KJP	107	A8	P-IV	40	2
C3	C++	KJP	108	A9	P-I	20	1
C4	Java	Kumar	109	A10	Cyrix	20	2

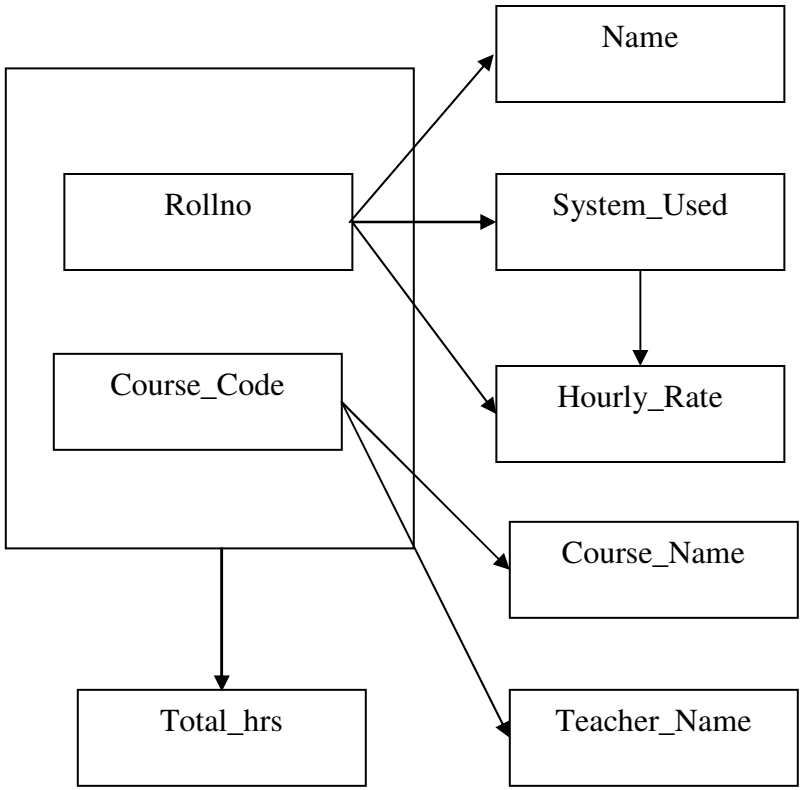
Second Normal Form

Definition

A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully functional dependent on the primary key.

A resultant database of first normal form COURSE_CODE does not satisfy above rule, because non-key attributes Name, System_Used and Hourly_Rate are not fully dependent on the primary key (Course_Code, Rollno) because Name, System_Used and Hourly_Rate are functional dependent on Rollno and Rollno is a subset of the primary key so it does not hold the law of fully functional dependence. In order to convert COURSE_CODE database into second normal form following rule is used.

FD Diagram


Name
Rollno
System_Used
Course_Code
Hourly_Rate
Course_Name
Total_hrs
Teacher_Name

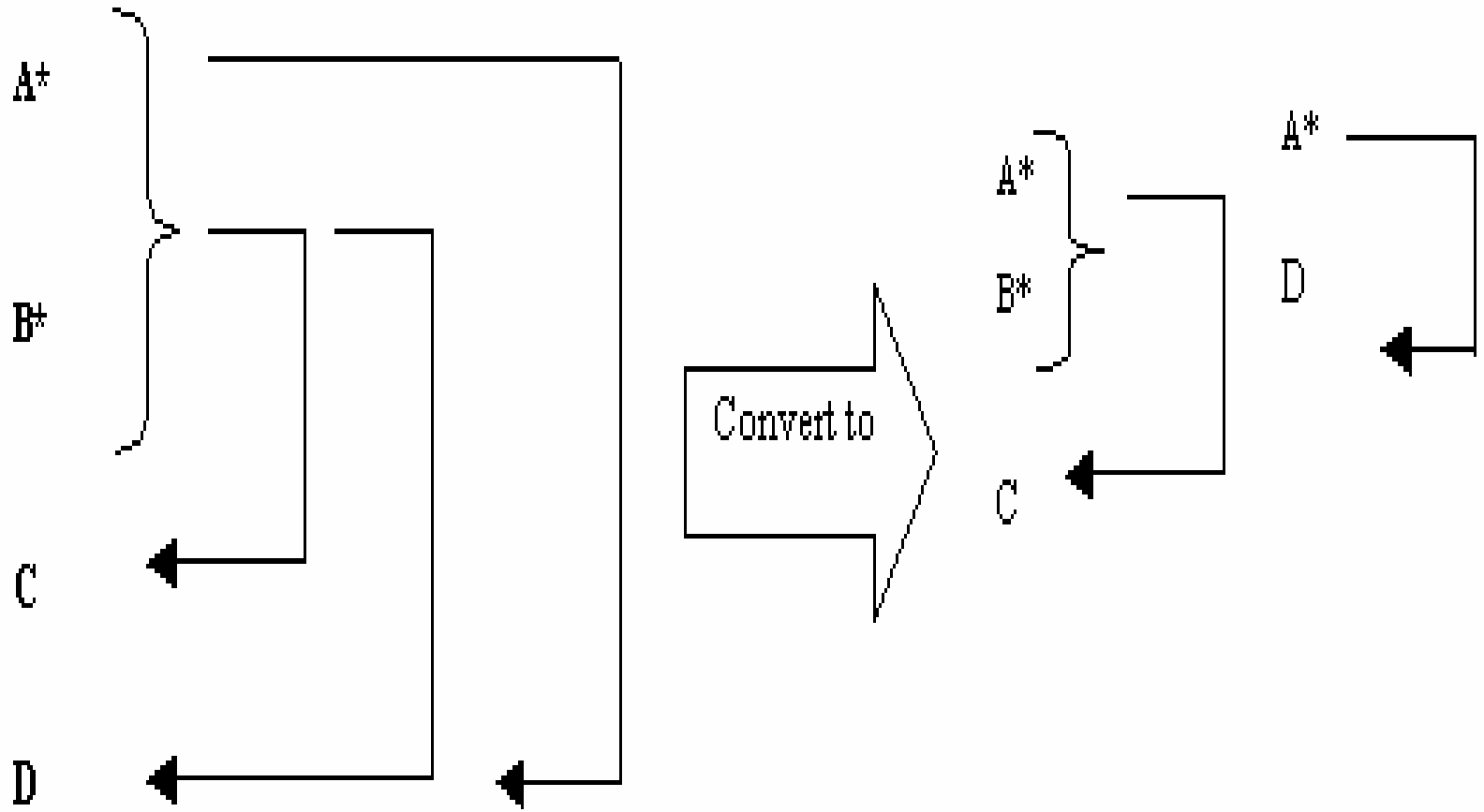
Rule to convert First Normal Form to Second Normal Form

Consider a relation where a primary key consists of attributes A and B. These two attributes determine all other attributes.

Attribute C is fully dependent on the key.

Attribute D is partially dependent on the key because we only need attribute A to functionally determine it.

Attributes C and D are nonprime or non-key attributes. Here the rule is to replace the original relation by two new relations. The first new relation has three attributes: A, B and C. The primary key of this relation is (A,B) i.e the primary key of the original relation. The second relation has A and D as its only two attributes. Observe that attribute A has been designated, as the primary key of the second relation and that attribute D is now fully dependent on the key.


A*
B*
C
D
Convert to
A*
B*
C
A*
D

HOURS_ASSIGNED

Course_Code	RollNo	Total__Hrs
C1	100	7
C1	101	3
C1	102	6
C1	103	1
C2	100	7
C2	104	3
C2	105	1
C2	101	2
C3	106	3
C3	107	2
C3	108	1
C4	109	2

STUDENT_SYSTEM_CHARGE

Rollno	Name	System_Used	Hourly_Rate
100	A1	PentiumI	20
101	A2	PentiumII	30
102	A3	Celeron	10
103	A4	PentiumIV	40
104	A5	P-III	35
105	A6	P-II	30
106	A7	P-IV	40
107	A8	P-IV	40
108	A9	P-I	20
109	A10	Cyrix	20

COURSE

Course_Code	Course_Name	Teacher_Name
C1	Visual Basic	ABC
C2	Oracle&Dev	DEF
C3	C++	KJP
C3	C++	KJP
C4	Java	Kumar

Data Anomalies in 2NF Relations

Relations in 2NF are still subject to data anomalies. For sake of explanation, let us assume that the system on which a student works functionally determines the hourly rate charged from the student. That is, System_Used → Hourly_Rate. This fact was not considered in the explanation of the previous normal form but it is not an unrealistic situation.

Insertion anomalies

Insertion anomalies occur in the STUDENT_SYSTEM_CHARGE relation. For example consider a situation where we would like to set in advance the rate to be charged from the students for a particular system. We cannot insert this information until there is a student assigned to that type of system. Notice that the rate that is charged from student for a particular system is independent of whether or not any student uses that system or not.

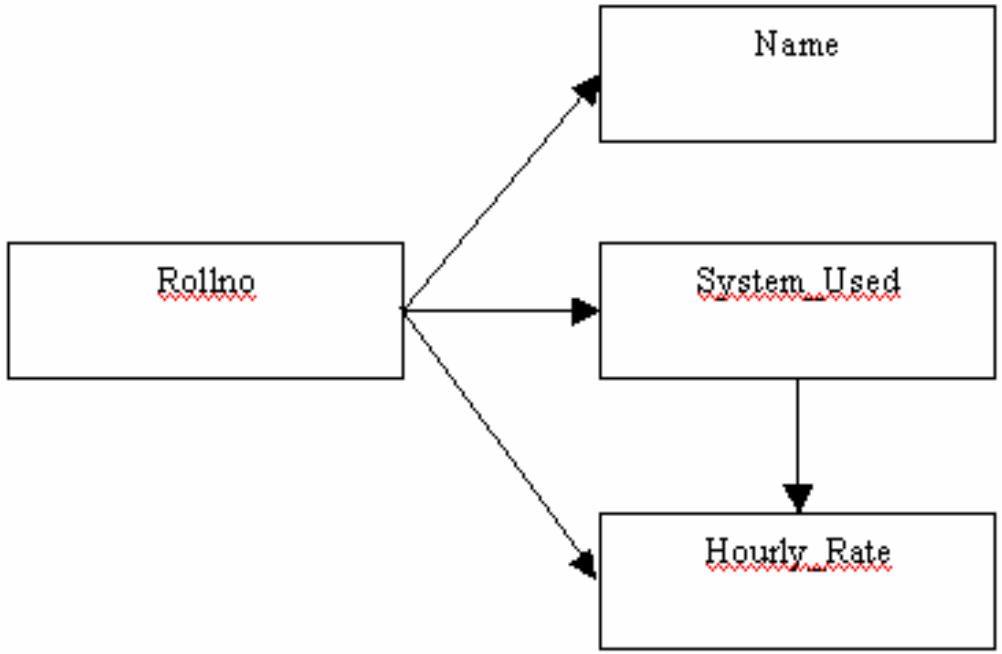
Update anomalies

Update anomalies will also occur in the STUDENT_SYSTEM_CHARGE relation because there may be several students which are working on the same type of the system. If the Hourly_Rate for that particular system changes, we need to make sure that the corresponding rate is changed for all students that work on that type of system. Otherwise the database may end up in an inconsistent state.

Delete anomalies

The STUDENT_SYSTEM_CHARGE relation is also susceptible to deletion anomalies. This type of anomaly occurs whenever we delete the tuple of a student who happens to be the only student left which is working on a particular system. In this case we will also lose the information about the rate that we charge for that particular system.

The anomalies discussed above occurs due to transitive dependence of Hourly_Rate on the primary key (Rollno) of STUDENT_SYSTEM_CHARGE database as shown in Functional dependence diagram below:



The solution of all above anomalies is provided by the third normal form, which deals with the problem of transitive dependence.

Third Normal Form

A relation R is in Third Normal Form (3NF) if and only if the following conditions are satisfied simultaneously:

(1) R is already in 2NF

(2) No nonprime attribute is transitively dependent on the key.

Another way of expressing the conditions for Third Normal Form is as follows:

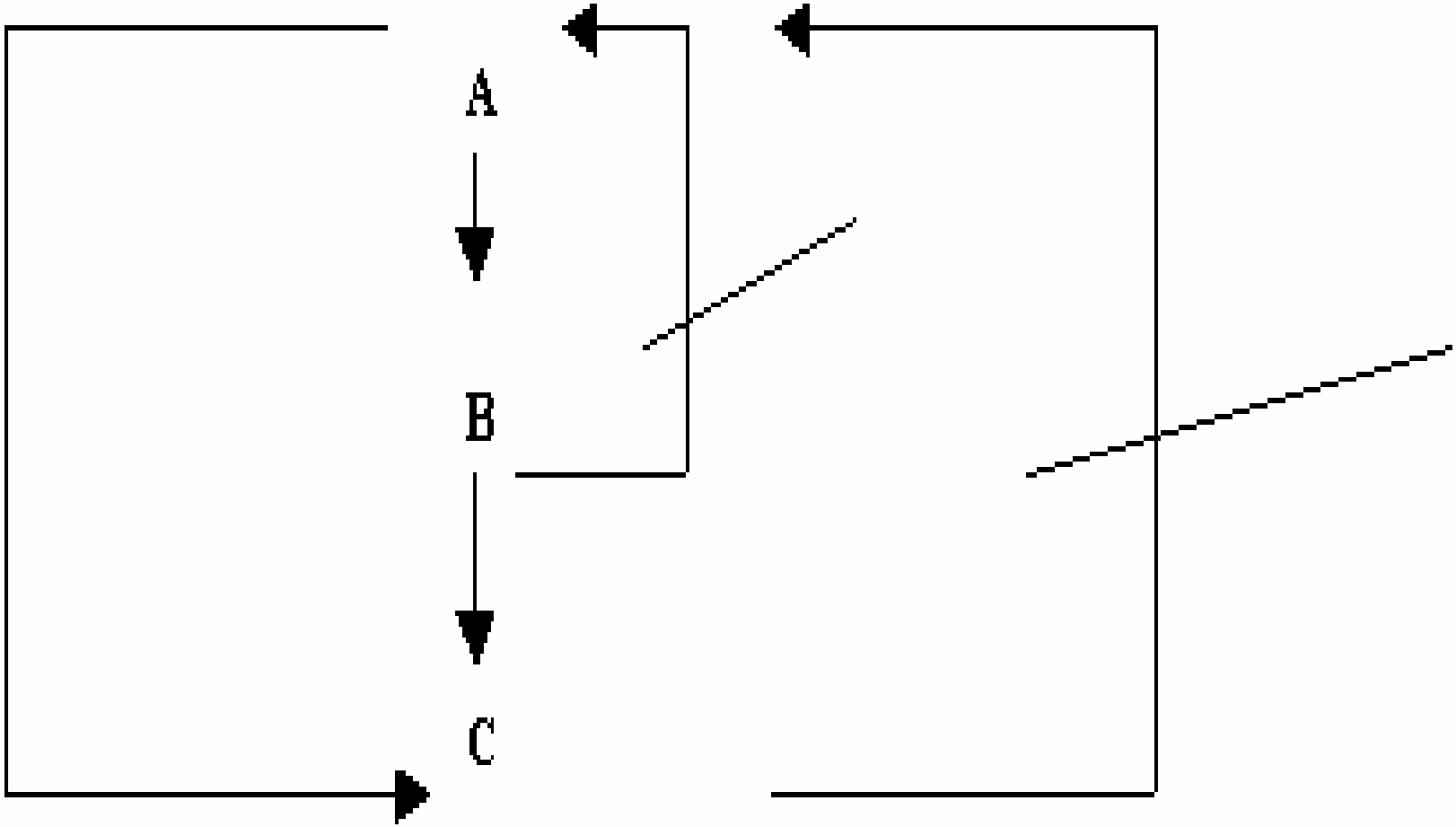
(1) R is already in 2NF

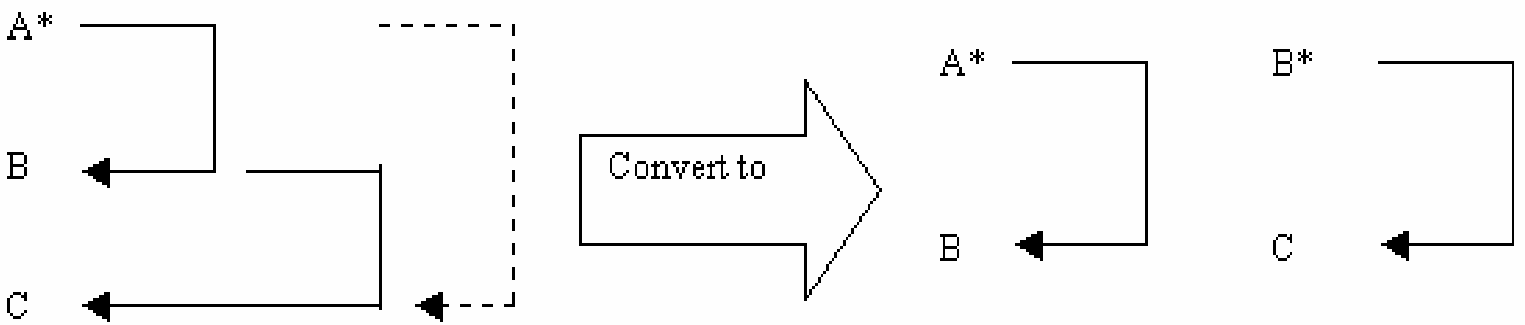
(2) No nonprime attribute functionally determines any other nonprime attribute.

These two sets of conditions are equivalent.

Transitive Dependencies

Assume that A,B and C are the set of attributes of a relation R. Further assume that the following functional dependencies are satisfied simultaneously: A→B, B↛A (B not functionally depends A), B→C, A→C and C↛A (C not functionally depends A). Observe that C→ B is neither prohibited nor required. If all these conditions are true, we will say that attribute C is transitively dependent on attribute A. It should be clear that these functional depend


A
B
C


A*
B
C
Convert to
A*
B
B*
C

Conversion of STUDENT_SYSTEM_CHARGE

(Rollno, Name, System_Used, Hourly_Rate) to Third Normal Form

The scheme of the first relation is

STUDENT_SYSTEM (Rollno,Name,System_Used).

The scheme of the second relation is

CHARGES (System_Used,Hourly_Rate).

STUDENT_STSTEM

Rollno	Name	System_Used
100	A1	PentiumI
101	A2	PentiumII
102	A3	Celeron
103	A4	PentiumIV
104	A5	P-III
105	A6	P-II
106	A7	P-IV
107	A8	P-IV
108	A9	P-I
109	A10	Cyrix

CHARGES

System_Used	Hourly_ Rate
PentiumI	20
PentiumII	30
Celeron	10
PentiumIV	40
Pentium-III	35
Cyrix	20

COURSE

Course_Code	Course_Name	Teacher_Name
C1	Visual Basic	ABC
C2	Oracle&Dev	DEF
C3	C++	KJP
C4	Java	Kumar

HOURS_ASSIGNED

Course_Code	RollNo	Total__Hrs
C1	100	7
C1	101	3
C1	102	6
C1	103	1
C2	100	7
C2	104	3
C2	105	1
C2	101	2
C3	106	3
C3	107	2
C3	108	1
C4	109	2

7.10 Case Study

In order to understand the normalization further, consider the following case of Supplier Part database, which is represented, in following database.

SUPPLIER (Sno, City, Status, Pno, Qty)

Sno	City	Status	Pno	Qty
S1	AMRITSAR	20	P1	300
			P2	200
			P3	400
			P4	200
			P5	100
			P6	100
S2	MUMBAI	10	P1	300
			P2	400
S3	MUMBAI	10	P2	200
		10	P2	200
S4	DELHI	20	P4	300
		20	P5	400

BCNF

Manufacturer

Id_No	Name	Item_No	Quantity
M101	Electronics USA	H3772	1000
M101	Electronics USA	J08732	700
M101	Electronics USA	Y23490	200
M322	Electronics-R-Us	H3772	900

For example, consider a relation

SSP (Sno, Sname, Pno, Qty)

BCNF is simply a stronger definition of 3NF. BCNF makes no explicit reference to first and second normal form as such, nor the concept of full and transitive dependence.

BCNF states that

A relation R is in Boyce/Codd N/F (BCNF) if and only if every determinant is a candidate key. Here determinant is a simple attribute or composite attribute on which some other attribute is fully functionally dependent.

For example Qty is FFD on (Sno, Pno)

(Sno, Pno)→ Qty, here

(Sno, Pno) is a composite determinant.

Sno → Sname

Here Sno is simple attribute determinat.

Similarities between 3NF and BCNF

Consider a relation COURSE_STUDENT and STUDENT_SYSTEM_CHARGE which are not in 3NF are also not in BCNF.

COURSE_STUDENT (Course_Code,Rollno,Name,System_Used,Hourly_Rate,Total_Hours)

Here, (Course_Code, Rollno) → Total_Hours

Rollno→ Name | System_Used | Hourly_Rate

Here, Rollno is a determinant but not candidate key so relation COURSE_STUDENT is not in BCNF.

In relation STUDENT_SYSTEM_CHARGE (Rollno,Name,System_Used,Hourly_Rate)

Rollno → Name | System_Used | Hourly_Rate

System_Used → Hourly_Rate

Here, System_Used is also a determinant but it is not unique, so relation STUDENT_SYSTEM_CHARGE is not in BCNF.

COURSE (Course_Code,Course_Name,Teacher_Name)

COURSE_ASSIGNED (Course_Code,Rollno,Total_Hours)

STUDENT_SYSTEM (Rollno,Name,System_Used)

CHARGE(System_Used,Hourly_Rate)

In conclusion, we can say that in these relations which have only single candidate key can be normalized both with 3NF and BCNF without any problem.

Overlapping of Candidate keys:

Two candidate keys overlap if they involve two or more attributes each and have an attribute in common.

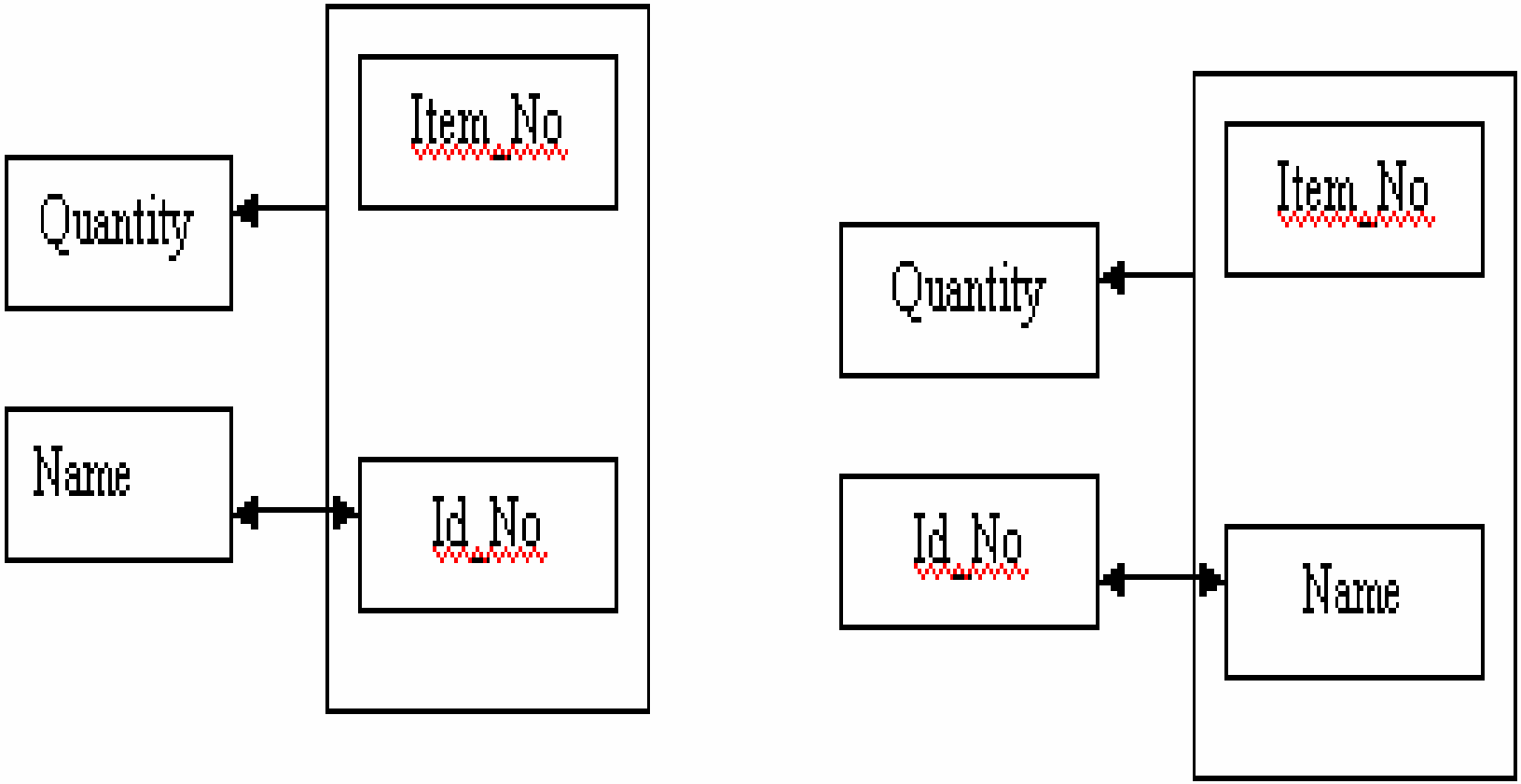
(Id_no, Item_No) → Quantity

(Name,Item_No) → Quantity

Item_No→Name

Name→Item_No

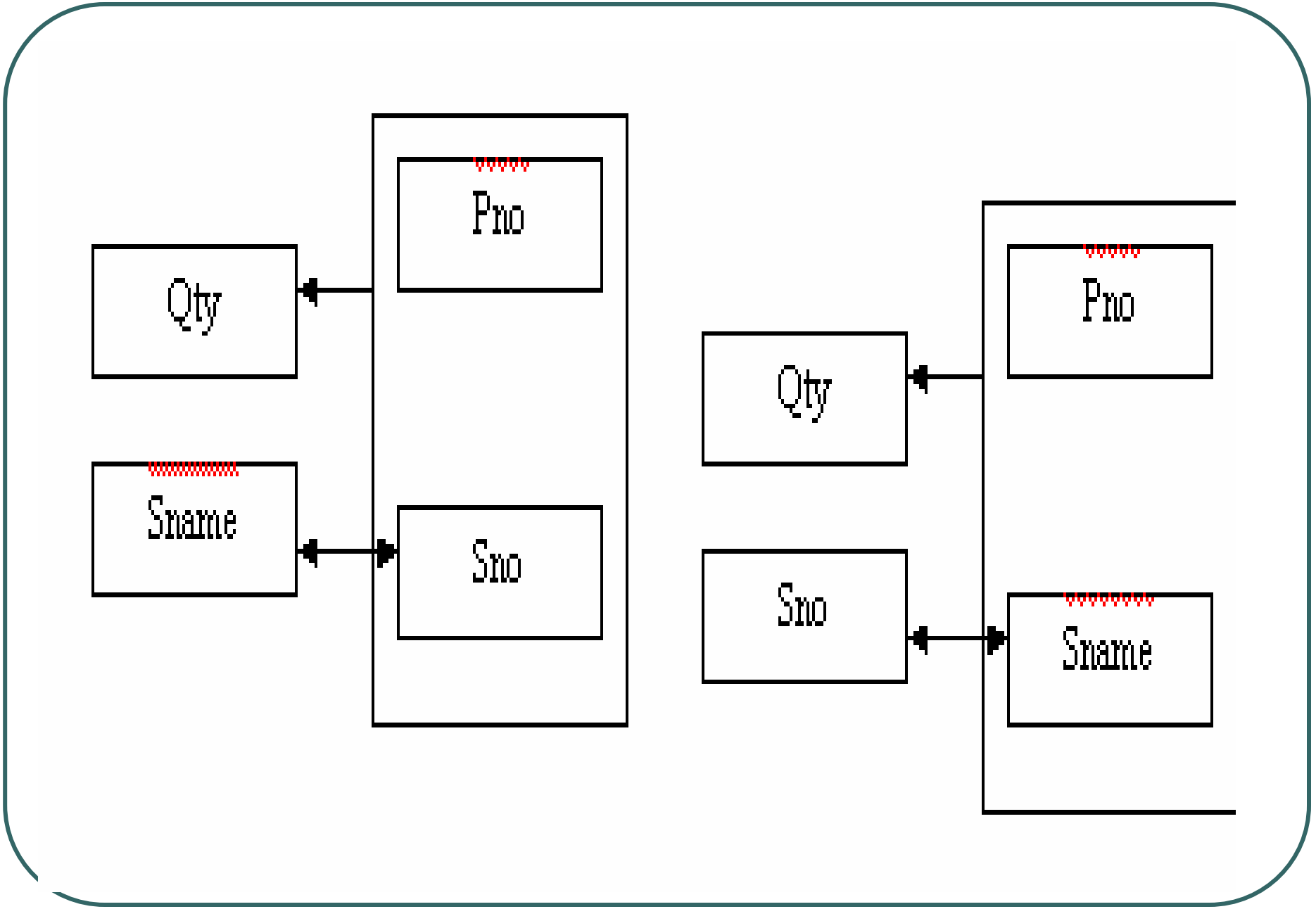
Possible FD diagram of this case is:



Another Case

- Sno, Sname, Pno, Qty)
- Here, let us suppose that Sname (supplier name) is unique for each Sno (supplier number) as shown below:

Sno	Sname	Pno	Qty
S1	Rahat	P1	300
S2	Raju	P2	200
S1	Rahat	P3	100
S2	Raju	P1	200


Pno
Qty
Sname
Sno
Pno
Qty
Sno
Sname

Fourth Normal Form (4NF)

A relation R is in Fourth Normal Form (4NF) if and only if the following conditions are satisfied simultaneously:

(1) R is already in 3NF or BCNF.

(2) If it contains no multi-valued dependencies.

Multi-Valued Dependency (MVD)

MVD is the dependency where one attribute value is potentially a ‘multi-valued fact’ about another. Consider the table

Table CUSTOMER_ADDRESS

Customer_Name	Address
Raj	New Delhi
Raj	Amritsar
Suneet	Amritsar
Suneet	Batala
Ankit	Qadian

MVD can be defined informally as follows:

MVDs occur when two or more independent multi valued facts about the same attribute occur within the same table. It means that if in a relation R having A, B and C as attributes, B and C are muti-value facts about A, which is represented as A→→B and A→→C ,then muti value dependency exist only if B and C are independent of each other.

Table COURSE_STUDENT_BOOK

Course	Student_name	Text_book
Physics	Rahat	Mechanics
Physics	Ankit	Mechanics
Physics	Rahat	Optics
Physics	Ankit	Optics
Chemistry	Ankit	Organic_chemistry
Chemistry	Ankit	Inorganic_chemistry
English	Raj	English_literature
English	Raj	English_grammar

Here in above database following MVDs exists:

Course→→Student_name

Course→→Text_book

Rule to transform a relation into Fourth Normal Form

A relation R having A,B, and C, as attributes can be non loss-decomposed into two projections R1(A,B) and R2(A,C) if and only if the MVD A→→B|C hold in R.

Course→→Student_name

Course→→Text_book

To put it into 4NF, two separate tables are formed as shown below:

COURSE_STUDENT (Course,Student_name)

COURSE_BOOK (Course,text_book)

COURSE_STUDENT

Course	Student_name
Physics	Rahat
Physics	Ankit
Chemistry	Ankit
English	Raj

COURSE_BOOK

Course	Text_book
Physics	Mechanics
Physics	Optics
Chemistry	Organic_chemistry
Chemistry	Inorganic_ chemistry
English	English_literature
English	English_grammar

Fifth Normal Form (5NF)

A relation R is in Fifth Normal Form (5NF) if and only if the following conditions are satisfied simultaneously:

(1) R is already in 4NF.

(2) It cannot be further non-loss decomposed.

5NF is of little practical use to the database designer, but it is of interest from a theoretical point of view and a discussion of it is included here to complete the picture of the further normal forms.

In all of the further normal forms discussed so far, no loss decomposition was achieved by the decomposing of a single table into two separate tables. No loss decomposition is possible because of the availability of the join operator as part of the relational model. In considering 5NF, consideration must be given to tables where this non-loss decomposition can only be achieved by decomposition into three or more separate tables.

Agent	Company	Product_Name
Suneet	ABC	Nut
Suneet	ABC	Screw
Suneet	CDE	Bolt
Raj	ABC	Bolt

P1

Agent	Company
Suneet	ABC
Suneet	CDE
Raj	ABC

P2

Agent	Product_Name
Suneet	Nut
Suneet	Bolt
Suneet	Screw
Raj	Bolt

The redundancy has been eliminated, but the information about which companies make which products and which of these products they supply to which agents has been lost. The natural join of these projections over the 'agent' columns is:

Agent	Company	Product_Name
Suneet	ABC	Nut
Suneet	ABC	Bolt*
Suneet	ABC	Screw
Suneet	CDE	Nut*
Suneet	CDE	Bolt
Suneet	CDE	Screw*
Raj	ABC	Bolt

P3

Company	Product_Name
ABC	Nut
ABC	Bolt
ABC	Screw
CDE	Bolt

Agent	Company	Product_Name
Suneet	ABC	Nut
Suneet	ABC	Bolt*
Suneet	ABC	Screw
Suneet	CDE	Nut*
Suneet	CDE	Bolt
Suneet	CDE	Screw*
Raj	ABC	Bolt

Agent	Company	Product_Name
Suneet	ABC	Nut
Suneet	ABC	Bolt*
Suneet	ABC	Screw
Raj	ABC	Bolt
Suneet	CDE	Bolt

Agent	Company	Product_Name
Suneet	ABC	Nut
Raj	ABC	Bolt
Raj	ABC	Nut
Suneet	CDE	Nut
Suneet	ABC	Bolt

P1

Agent	Company
Suneet	ABC
Suneet	CDE
Raj	ABC

P2

Agent	Product_Name
Suneet	Nut
Suneet	Bolt
Raj	Bolt
Raj	Nut

P3

Company	Product_Name
ABC	Nut
ABC	Bolt
CDE	Nut

All redundancy has been removed, if the natural join of P1 and P2 is taken, the result is:

Agent	Company	Product_Name
Suneet	ABC	Nut
Suneet	ABC	Bolt
Suneet	CDE	Nut *
Suneet	CDE	Bolt
Raj	ABC	Bolt
Raj	ABC	Nut

References

Simplified Approach to DBMS
Kalyani Publishers
By
Parteek Bhatia