MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

RELATIONAL DATABASE MANAGEMENT SYSTEM

Unit 3

Normalisation

Table of Contents

1. INTRODUCTION

The basic objective of relational database design is to formulate a basic set of relational schemas which helps the user to store information in the database without any redundancy and anomaly. We basically strive to develop a database from which information can be extracted effortlessly. Normalisation is one way to achieve this aim of designing relational schemas which are efficient in performance.

Relational schemas which contain the design anomalies are decomposed to convert them into various normal forms. This procedure of converting the relational databases into normal forms is referred to as normalisation which we will cover in this unit. Normalisation technique plays a vital role in designing good and efficient relational databases.

In 2nd unit, you studied various aspects of relational database systems and SQL. In this unit, you will study problems that arise due to bad designs and ways to achieve good designs using normalisation techniques.

1.1 Objectives

After studying this unit, you should be able to:

- *Identify the functional dependencies which may exist among the relations*
- *recall and describe various anomalies in a database*
- *explain the concept of schema refinement through normalisation*
- *discuss the relationship between normalisation and database design*
- *describe the need and techniques of denormalisation*

2. FUNCTIONAL DEPENDENCY

Functional dependency (FD) is the most important component of normalisation.

Two attributes A and B in any relation R are said to possess a functional dependency (FD) if for each distinct value of A, there is only one value of attribute B associated with it. FD is symbolically represented as

$$A \rightarrow B$$

This denotes that attribute A functionally determines attribute B or attribute B is functionally dependent on attribute A.

Note that it is not necessary if B is functionally dependent on A, then A is also functionally dependent on B.

Let us understand this with the help of few examples given below.

Example: Let us take an example of Relation Sample (P, Q, R) as shown in Table 3.1:

Table 3.1: Sample (P, Q, R)

P	Q	R
pl	q1	rl
p2	q2	r2
Pl	q1	r2
Pl	q1	r3

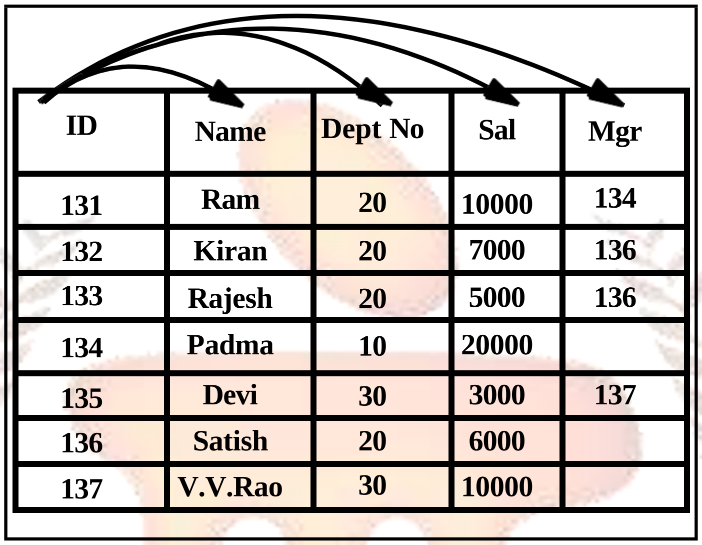
In this sample (P,Q. R), t[P] symbolises the tuple variables of the attribute P. This relation contains the functional dependency between P and Q where Q is functionally dependent on P. It can be represented as

$$P \rightarrow Q$$

Here t[P]=t[Q] which means that for every value of Q there exists a unique value of P. But there exists no functionally dependency between P and R as for each value of R there is not a unique value of P. Similarly there is no FD between Q and R in the above relation.

Example: Table 3.2 below shows functional dependency for an Employee Relation

Table 3.2: Functional Dependency



ID	Name	Dept No	Sal	Mgr
131	Ram	20	10000	134
132	Kiran	20	7000	136
133	Rajesh	20	5000	136
134	Padma	10	20000	
135	Devi	30	3000	137
136	Satish	20	6000	
137	V.V.Rao	30	10000	

Non-key attributes depends on the PK (primary key) attribute.

In this relation, Name, Dept. No., Sal, Mgr are each functionally dependent on the ID (employee Id). Thus, $\text{ID} \rightarrow \text{Name}$ $\text{ID} \rightarrow \text{Dept No}$, $\text{ID} \rightarrow \text{Sal}$ and $\text{ID} \rightarrow \text{Mgr}$.

Example: Consider another relation where every vehicle owner has a license and each license contains a unique license number. A license number uniquely determines a distinct owner. We can also say that a particular licence number is capable of uniquely determining the identity of the owner.

Thus we can say that a functional dependency exists between the two attributes licence id and license owner. This functional dependency is shown below symbolically:

$$\text{license\_id} \rightarrow \text{license\_owner}$$

This means that the License_owner is functionally dependent on license id or we can say that license_id functionally determines license_owner.

Again you must note that the converse of the above functional dependency may need not hold true necessarily. In some cases a person can possess two types of vehicles such as a two wheeler and a four wheeler then the owner will possess two licenses.

Functional dependencies are useful in refining the schema. By using Functional dependencies a relation can be replaced with smaller relations.

Decomposition of a relational schema (R) consists of replacement of the relation schema by two or more relational schemas all of which contains a subset of the attributes of R. There are two basic objectives of decomposing a particular relation. These are:

- *To reduce data redundancy in the relational schema.*
- *To retain the ability to recreate the original relation without leaving out tuples or adding new tuples.*

This method of recreating the original table from the decomposed tables is called a **join.**

Decomposition is termed as lossless join decomposition, when the parts of the table can be joined back again without adding more tuples added to the database relation. In case the re-joining of the disjoined relations results in extra tuples the join is termed as a lossy join, because it loses information with the additional tuples.

SELF-ASSESSMENT QUESTIONS – 1

1. Decomposition helps to reduce data redundancy. (True/False)
2. Functional dependencies can be used to refine the ________________.

Activity 1

Explain Armstrong's axioms for functional dependencies. You may take help of internet.

3. ANOMALIES IN A DATABASE

Many-a-times a database is not designed appropriately which results in different types of anomalies. These database anomalies diminish the performance of the database. Therefore we need to remove these anomalies so as to design a good database. Let us first study about some of the very important anomalies, which are discussed below:

3.1 Redundancy

Redundancy is the most common type of database anomaly. If the data values stored in a relation are repeated, then the database is said to possess redundancy. Database containing multiple copies of the same data result in wastage of expensive storage space. It also leads to various other types of anomalies.

Let us take the example of relational schema Sales persons (SP). This relation contains the details of the sales person. The relational schema (SP) for the same , as an example, is designed in the following manner:

SP(SPId, SPName, SPAge, MGRId, MGRName, MGRAge, SPLocation, SPSalary) where the attributes mean as follows:

SPId:	Sales person unique identity number
SPName:	Sales person's name
SPAge:	Sales person's age
MGRId:	Manager's unique identity number
MGRName:	Manager's name
MGRAge:	Manager's Age
SPLocation:	Sales person's location
SPSalary:	Sales person's salary

One example of this relational schema is shown in Table 3.3.

Table 3.3: An Example of Relational Schema

Sales employee	Id	Name	Age	ManagerId	MName	Mage	Location	Pay
	11	Ramkumar	32	21	Shammi	35	Hisar	4000
	21	Shammi	35	44	Rajan	48	Bhiwani	6000
	13	Sam	34	21	Shammi	35	Hisar	3400
	44	Rajan	48	Null	Null	Null	Null	Null
	50	Sunny	33	44	Rajan	48	Bhiwani	3000
	26	Raj	29	13	Sam	34	Hisar	2500
	47	Harry	32	44	Rajan	48	Karnal	4000

Note that the above table contains anomaly. Observe that 'Rajan' and his age appear thrice in the Table 3.3. Imagine a large relation containing thousands of records. If in such type of table even 100 records are duplicated then this would result in the wastage of lots of storage space as well as lower the speed of query execution. This repetition of database values is termed as redundancy.

3.2 Inconsistency

A poorly designed database may also contain inconsistency. It is one of the most troublesome database anomalies. If there is any type of disagreement between data items in a database, then it is referred to as inconsistency.

Let us understand this with the help of above relation SP. Suppose we want to modify/update the age of 'Shanti' from 35 to 36. In such a situation this needs to be updated at all the places, in whichever record 'Shanti' appears in SPName column. If we do not do so then it will result in inconsistency in the database as at some she Shanti's age will be shown as 35 years old while at other places she would be 36 years old in the same database.

This data inconsistency problem is quite common in situations where there is redundancy in the database.

Therefore database designers strive to develop a database which contains very less redundancy. Their primary objective is to keep the redundancy under control rather than eliminating it completely.

3.3 Update Anomalies

When a person tries to update the redundant database, then it results in update anomalies. There are mainly three categories of update anomalies. These update anomalies are discussed below.

1. ***Insertion Anomalies:*** We will again illustrate this with the help of foregoing table SP. Suppose you want to add a new sales person 'Ajay' to the database who has recently joined the company but has not been assigned any manager yet.

In such a scenario the available information with you is SPId, SPName, SPAge, SPLocation and SPSalary. You will not have any values for attributes MGRId, MGRName and MGRAge as the sales person has not been assigned any manager. Therefore you will assign Null to these attributes.

But doing so gives rise to another problem. This results in problem of interpretation of the Null. Different people may interpret null differently. By seeing the null in the attribute a data entry operator may feel that the value was unavailable but some other person might think that the attribute is not applicable to this entry.

It is not possible to compare the Null values and therefore searching & sorting must be done. One of the design goals is therefore, to reduce the use of Null to a minimum.

Now, let us suppose 'Prakash' joins as a manager. But if there is no sale person yet associated with him then the question arises that where should this information be stored? If we insert a record with all other value as Null, then there is also be problem that any one of these attributes is the primary key for that table and it is not possible to insert null into any key attribute. Therefore this situation gives rise to an insertion anomaly in the relation.

2. ***Deletion Anomalies:*** Deletion is another type of update anomaly which occurs when inconsistencies arise due to the removal of record from a relation. Let us consider that 'Shammi' leaves the organisation. Therefore the entire records relating to 'Shammi' needs to be deleted from the database table. But accidentally the record of sales person 'Sunny' is also deleted along with the deletion of 'Shammi's" records.

All the information regarding sales person 'Sunny' is lost. Such a condition gives rise to deletion anomaly in which the deletion of one or more record unintentionally deletes some other data from the database.

3. ***Modification Anomalies:*** As was explained earlier, when an attribute's value for a particular record needs to be modified, then this change must be done in all the occurrences of the record. Otherwise, it results in modification anomaly.

Let us suppose that you want to update the age of 'Rajan' to 49 years. Therefore you need to essentially update this at all the places where 'Rajan' appears in the MName column. If by mistake even one occurrence is missed, inconsistency will arise.

Remember a good quality database design will be free from any such anomalies. Clearly, for the above mentioned causes the sales person SP table cannot be considered as a good database design.

SELF-ASSESSMENT QUESTIONS – 2

3. Which of the following is referred when there is a disagreement between data items in a database?
   a) Redundancy
   b) Inconsistency
   c) Anomaly
   d) Normalisation
4. When the data values are stored repeatedly in multiple copies in the database, it is known as ________________________.

4. THE NORMALISATION PROCESS

Normalisation comprises of various set of rules which are used to make sure that the database relations are fully normalised by listing the functional dependencies and decomposing them into smaller, efficient tables.

Normalisation primarily helps to:

1. eliminate data maintenance anomalies
2. minimise database redundancy
3. eliminates data inconsistency

Normalisation technique is established on the idea of normal forms. A table is said to be in a specific normal form if it fulfils a particular set of constraint which are defined for that normal form. These constraints are usually applicable on the attributes (column) and the relationships between them. There are various levels of normal forms (See Figure 3.1). Each normal form addresses a specific issue that could result in minimising the database anomalies.

Database Normalisation uses functional dependencies present in a relation/table and the candidate key in examining the tables. In the beginning, there three normal forms were suggested; First normal Form (1NF), Second normal Form (2NF), and Third normal Form (3NF). Later on Fourth Normal Form (4NF) and Fifth Normal Form (5NF) were also introduced.

Afterwards, E.F. Codd and R. Boyce presented a more substantial definition of third Normal Form known as (Boyce-Codd Normal Form).

All the normal forms except 1NF are derived from the concept of functional dependencies among the attributes of relation.

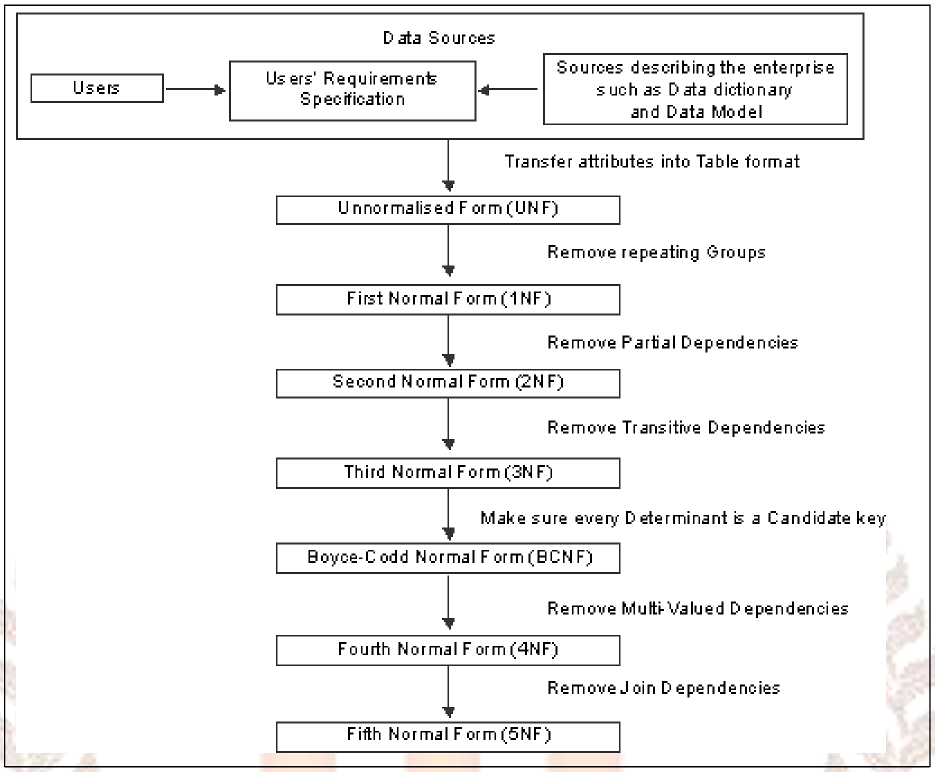

Data Sources
Users
Users' Requirements Specification
Sources describing the enterprise such as Data dictionary and Data Model
Transfer attributes into Table format
Unnormalised Form (UNF)
Remove repeating Groups
First Normal Form (1NF)
Remove Partial Dependencies
Second Normal Form (2NF)
Remove Transitive Dependencies
Third Normal Form (3NF)
Make sure every Determinant is a Candidate key
Boyce-Codd Normal Form (BCNF)
Remove Multi-Valued Dependencies
Fourth Normal Form (4NF)
Remove Join Dependencies
Fifth Normal Form (5NF)

Fig 3.1: Normalisation Process

When you initially enter the records into a database table, it is commonly in unnormalised form. Therefore you need to refine this table with the help of various types of normalisation forms which are explained below:

4.1 First Normal Form

First normal form commonly termed as 1NF is the most basic normal form. In this normal form the condition lies that there must not be any repeating groups in any column. In other words, all the columns in the table must be composed of atomic values.

Note: Atomic: A column is said to be atomic if the values are indivisible units.

The table is said to possess atomic values if there is one and only one data item for any given row & column intersection. Non-atomic values create repeating groups. A repeating group is just the repetition of a data item or cluster of data items in records. For example, consider below given Table 3.4:

Table 3.4: Employee Table with Attribute Dependents

ID	Name	DeptNo	Sal	Mgr	Dependents
131	Ram	20	10000	134	Father, Mother, Sister
132	Kiran	20	7000	136	Wife, Son
133	Rajesh	20	5000	136	Wife
134	Padma	10	20000		Son, Daughter
135	Devi	30	3000	137	Father, Mother
136	Satish	20	6000		Father, Mother
137	V.V. Rao	30	10000		Wife, First Son, Second Son

In the Table 3.4 you can see that the dependents column contains non-atomic values. Therefore to convert this table into INF, we need to modify the non-atomic values into atomic values as shown in Table 3.5.

Table 3.5: Change of Non-atomic Values into Atomic Values of Table 3.4

ID	Name	DeptNo	Sal	Mgr	Dependents
131	Ram	20	10000	134	Father
131	Ram	20	10000	134	Mother
131	Ram	20	10000	134	Sister
132	Kiran	20	7000	136	Wife
132	Kiran	20	7000	136	Son
133	Rajesh	20	5000	136	Wife
134	Padma	10	20000		Son
134	Padma	10	20000		Daughter
135	Devi	30	3000	137	Father
135	Defi	30	3000	137	Mother
136	Satish	20	6000		Father
137	V.V. Rao	30	10000		Wife
137	V.V. Rao	30	10000		First Son
137	V.V. Rao	30	10000		Second Son

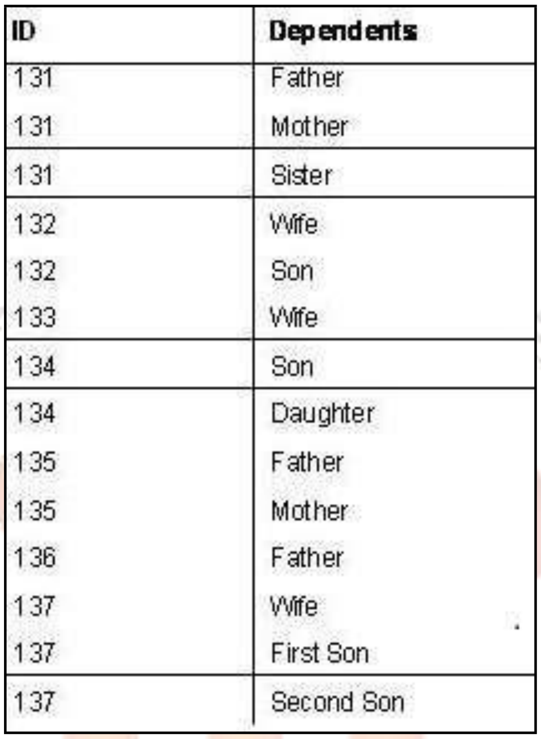
Observe in Table 3.5 the dependents column now contains atomic values. You will note that for each dependent the other employee details such as ID, Name, Dept No, Sal and Mgr are repeated which results in the creation of repeating group(data redundancy). According to first NF, the above relation employee (Table 3.5) is in 1NF. However, it is best practice to remove the groups which are being repeated in the table.

According to first normalisation rule, the table should not contain any repeating groups of column values. If there exists any such type of repeating groups then they should be decomposed and the associated columns will form their own table. Also the new resulting table must contain a link with the original table (from where it was decomposed). Thus, to remove repeating groups from the Employee relation, it can be decomposed into two relations namely Emp and Emp_Depend as shown in Table 3.6 and 3.7:

Table 3.6: Emp Relation

ID	**Name**	**DeptNo**	**Sal**	**Mgr**
131	Ram Kiran	20	10000	134
132	Rajesh	20	7000	136
	Padma			[illegible]
133	Devi	20	5000	
134	Satish	10	20000	137
135	V.V. Rao	30	3000	
136		20	6000	

Table 3.7: Emp_Depend Relation



ID	Dependents
131	Father
131	Mother
131	Sister
132	Wife
132	Son
133	Wife
134	Son
134	Daughter
135	Father
135	Mother
136	Father
137	Wife
137	First Son
137	Second Son

Here, in the above table 3.7, {ID, Dependents} combination will act as the unique key. And the tuple 'ID' is the common tuple in both the tables (table 3.6 and table 3.7) which act as a link with the original table. Now data redundancy in the columns ID, Name, Dept. No, Sal and Mgr are also eliminated and now these tables are in INF. Now let us consider another example. Suppose we have a customer table as shown in Table 3.8.

Table 3.8: Customer Table

Cust_id	Name	Address	Acc_ id	Acc_type	Min_bal	Tran_id	Tran_type	Tan_mod e	Amount	Balance
001	Ravi	Hyd	994	SB	1000	14300		B/F	1000	1000
001	Ravi	Hyd	994	SB	1000	14301	Deposit	Bycash	1000	2000
001	Ravi	Hyd	994	SB	1000	14302	Withdrawal	ATM	500	1500
110	Tim	Sec'ba d	340	CA	500	14303		B/F	3500	3500
110	Tim	Sec 'bad	340	CA	500	14304	Deposit	Payroll	3500	7000
110	Tim	Sec'ba d	340	CA	500	14305	Withdrawal	ATM	1000	6000
420	Kavi	Vizag	699	SB	1000	14306		B/F	6000	6000
420	Kavi	Vizag	699	SB	1000	14307	Credit	Bycash	2000	8000
420	Kavi	Vizag	699	SB	1000	14308	Withdrawal	ATM	6500	1500

You will notice that the Table 3.8 contains repeating group composed of Cust_id, Name and Address. Therefore to convert this table into first normal form, we need to remove this repeating group. This can be done by dividing this table into two tables: Customer and Customer Tran. (See Table 3.9 and 3.10)

(***Note:*** The primary key columns of each table are indicated in highlights in Figures).

Table 3.9: Customer Table

Cust_id	Name	Address
001	Ravi	Hyd
110	Tim	Sec'bad
420	Kavi	Vizag

Table 3.10: Customer_Tran Table

Tran_id	Cust_id	Acc_id	Acc_type	Min_bal	Tran_type	Tan_mode	Amount	Balance
14300	001	994	SB	1000		B/F	1000	1000
14301	001	994	SB	1000	Deposit	Bycash	1000	2000
14302	001	994	SB	1000	Withdrawal	ATM	500	1500
14303	110	340	CA	500		B/F	3500	3500
14304	110	340	CA	500	Deposit	Payroll	3500	7000
14305	110	340	CA	500	Withdrawal	ATM	1000	6000
14306	420	699	SB	1000		B/F	6000	6000
14307	420	699	SB	1000	Credit	Bycash	2000	8000
14308	420	699	SB	1000	Withdrawal	ATM	6500	1500

4.2 Second Normal Form

1NF table is not fully free from redundancy. It may have partial dependencies. Therefore, the Second Normal Form resolves partial dependencies.

The Second Normal Form states that

- The table must be in 1st Normal form
- All the non-key columns must be fully functional dependent on the Primary key

Any attribute (column) is said to be **partially dependent** if its value can be determined by any one or more attributes of the primary key, but not all.

Every normal form is based upon the previous normal form. Therefore the first condition for the second normal form is to have all its tables in first normal form.

The Fully Functional Dependency is for a given composite primary key (a primary key which is made of more than a single attribute), each column attribute, which is not an attribute of the Primary key, should be dependent on each and every one of the attributes.

If attributes are only partially dependent on the primary key attribute then they must be removed and placed in another table. The primary key of the new table formed must have apportion of the original key that they were dependent on.

Again consider the earlier example of Customer Relation. After converting it into1NF we have two tables: Customer and Customer_Tran. Now we need to convert it into 2NF (Second Normal Form). For doing so, the Customer Tran table is further decomposed into three tables: Customer Account, Accounts and Transaction, as shown in Table 3.11, 3.12 and 3.13.

Table 3.11: Customer_Accounts Table

Cust.id	Acc_id	Balance
001	994	1500
110	340	6000
420	699	1500

Table 3.12: Accounts Table

Acc_id	Accjype	Min.bal
994	SB	1000
340	CA	500
699	SB	1000

Table 3.13: Transaction Table

Tran_id	Acc_id	Tran_type	Tan_mode	Amount
14300	994		B/F	1000
14301	994	Deposit	Bycash	1000
14302	994	Withdrawal	ATM	500
14303	340		B/F	3500
14304	340	Deposit	Payroll	3500
14305	340	Withdrawal	ATM	1000
14306	699		B/F	6000
14307	699	Credit	Bycash	2000
14308	699	Withdrawal	ATM	6500

Table 3.14: Customer Table

Cust_id	Name	Address
001	Ravi	Hyd
110	Tim	Sec'bad
420	Kavi	Vizag

As the Acc_type and Min_bal attributes of Customer_Account table (Table 3.11) are not fully functionally dependent on the primary key (dependent on acc_id), therefore a new Accounts table is formed (Table 3.12).

Similarly, the Balance is dependent on Cust_id and Acc_id, but not fully functionally dependent on the , resulting in a new Customer_Accounts table (Table 3.14).

4.3 Third Normal Form

Second normal forms are not yet completely free from redundancies. It may show some redundancies due to transitive dependencies. Thus the next higher normal form i.e. the third normal form objective is to resolve transitive dependencies. A transitive dependency arises between two attributes when any non-key attribute is functionally dependent on some other non-key column which is in turn functionally dependent on the primary key.

The essential conditions for the Third Normal Form are:

- The table must be in 2nd Normal Form
- The table must not contain any transitive dependencies

Transitive Dependencies: Columns dependent on other columns that in turn are dependent on the primary key are said to be transitively dependent.

In other words, a relation R is said to be in the third normal form (3NF) if and only if it is in 2NF and every non-key attribute must be non-transitively dependent on the Primary key.

Therefore the main objective of 3NF is to make the relation free from all transitive dependencies. Let us understand how we can do this with the help of an example.

Example: Again let us go back to our previous example. The Accounts table (Table 3.12)is in the second normal form but it has transitive dependency as follows:

In order to remove this transitive dependency, the Accounts table can be decomposed into two tables: Acc_Detail and Product as shown in Table 3.15 and 3.16:

Table 3.15: Acc_Detail Table

Acc_id	Acc_type
994	SB
340	CA
699	SB

Table 3.16: Product Table

Acc_type	Min_bal
SB	1000
CA	500

Tables after Third Normal Form are given below (Table 3.17, 3.18 and 3.19)

Table 3.17: Customer Table

Cust_id	**Name**	**Address**
001	Ravi	Hyd
110	Tim	Sec'bad
420	Kavi	Vizag

Table 3.18: Customer Accounts Table

Cust_id	Acc_id	Balance
001	994	1500
110	340	6000
420	699	1500

Table 3.19: Transaction Table

Tran_id	**Acc_id**	**Tran_type**	**Tan_mode**	**Amount**
14300	994		B/F	1000
14301	994	Deposit	Bycash	1000
14302	994	Withdrawal	ATM	500
14303	340		B/F	3500
14304	340	Deposit	Payroll	3500
14305	340	Withdrawal	ATM	1000
14306	699		B/F	6000
14307	699	Credit	Bycash	2000
14308	699	Withdrawal	ATM	6500

4.4 Boyce-Codd normal form

BCNF is the common name of Boyce-Codd normal Form. This normal form is stricter than the 3 NF. Remember that every relation which is in BCNF form is also in 3NF, but a relation, which is in 3NF may or may not be necessarily in BCNF.

The essential condition for a relational schema R to be in BCNF is that "whenever any nontrivial FD $X \rightarrow A$ holds in R, then X must be a super key of R".

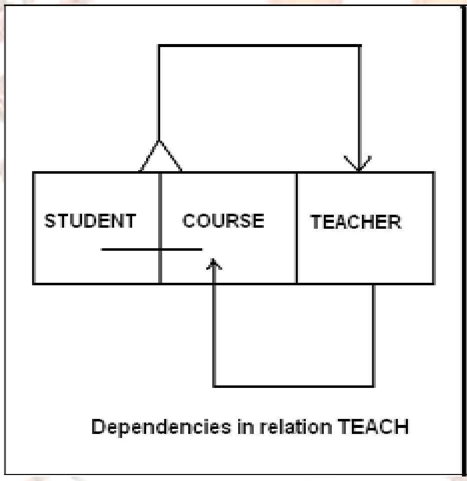
Let us understand the concept of BCNF with the help of relation schema TEACH.

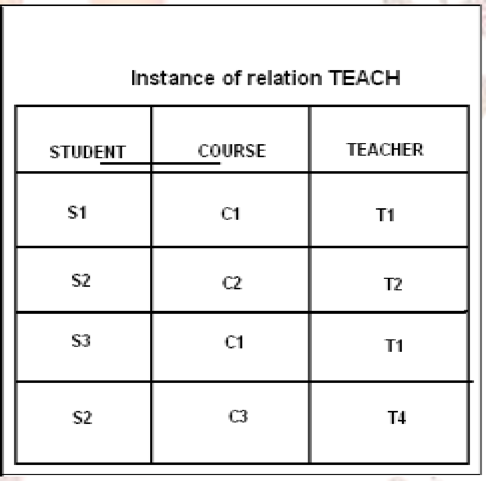
The Teach schema is composed of following attributes

student varchar (5),
Course varchar (5),
Teacher varchar (5)).

In this relation there are two dependencies. One in which (Student+Course)→Teacher, and second in which Teacher→Course.

In this example, it has been assumed that one teacher teaches only one course. (Student+Course) is the primary key in this relation.




Instance of relation TEACH

STUDENT	COURSE	TEACHER
S1	C1	T1
S2	C2	T2
S3	C1	T1
S2	C3	T4

Fig 3.2: Teach Schema

In this example we will determine whether this table Teach (Figure 3.1) is in BCNF or not. For this, we need to first check the first condition whether the relation is in 3NF.Here the relation Teach is in 3NF. Hence first condition is satisfied.

Now let us check the second condition. According to the BCNF criteria, the FD Teacher → Course which is of the form $X \rightarrow A$ must holds on this relation Teach, and Teacher should be a superkey. But, here Teacher is not a *super key*. Therefore this condition is not fulfilled. So we can say that the relation is not in BCNF.

We have seen the relation Teach is not in BCNF but it is in 3NF. This condition arises because, for a relation to be in 3NF, it must follow either of the two conditions which are 'either X should be a super key of R' or 'A should be a prime attribute'. In this relation as Teacher is not a superkey, first condition fails. But even then, the second condition is satisfied as course is a prime attribute of R. Therefore, the relation is in 3NF, but not in BCNF.

Comparison of BCNF with 3NF: For understanding the differences between BCNF and 3NF, we must again carefully look back to the definition of 3NF and BNF.

According to a 3NF definition "The condition for a relational schema R to be in 3NF is if whenever a nontrivial functional dependency (FD)X→A holds in R, then either of these two conditions must be fulfilled."

1. X should be a super key of R.
2. A is an prime attribute

But, according to the definition of BCNF" the condition for a relational schema R to be in BCNF is if whenever a nontrivial functional dependency (FD) X→A holds in R, then X should be a super key of R".

Thus we see that BCNF is more strict than 3NF. One can easily obtain a 3NF relational design without sacrificing the condition of lossless- join dependency preservation.

But, it is not easy to achieve BCNF design, lossless join and dependency preservation altogether. In such type of situations, in which, we cannot achieve all three objectives together; we will have to chose 3NF, lossless- join and dependency preservation.

Multi-Valued Dependencies (MVDs): MVD arises in situation where one attribute value is possibly a 'multi valued fact' about some other attribute within the same table. One special case of MVD is FD which you have studied earlier. Therefore, every FD is an MVD.

Now let us understand what MVD is with the help of few examples given below. Let us consider the relational schema CSB with following structure.

```
(Stud _ name (10),
Course char (10),
Text _ book char (10))
```

An instance of Relation CSB is shown in Table 3.20.

Table 3.20: Instance of Relation CSB

Stud_name	**Course**	**Text_book**
Brown	First_Yr_Optics	Phy - 1
Brown	First_Yr_Mech	Phy - 1
Green	First_Yr_Optics	Phy - 1
Green	First_Yr_Mech	Phy - 1
Brown	Org_Chem	Chem - 1
Brown	Inorg_Chem	Chem - 1
Jones	French_litter	French - 1
Jones	French_grmr	French - 1

In this relation the two attributes 'Stud_name' and 'Text_book' are independent multi-valued facts about the attribute 'course'. Therefore, we can simply say that this relation contains multi-valued dependency. Here Stud_name and Text_book are independent multi-valued facts about course because the student has no control over the textbooks which are used for a particular course.

Let us take one more example of a relation schema Emp_Profile with following three attributes:

(Emp _ name char (15),
Equipment char (15),
Languagechar (15)).

An instance of the relation Emp_Profile is shown in Table 3.21.

Table 3.21: Instance of Relation Emp_Profile

Equipment	Emp_name	Language
PC	Smith	French
PC	Smith	German
Workstation	Smith	German
Workstation	Smith	French
Workstation	Jones	French
Workstation	Jones	German
Workstation	Jones	German

In this relation, Equipment and language are the two independent multi- valued facts about employee_name. Therefore we can say that the relation also contains MVD as shown in Figure 3.3 below.

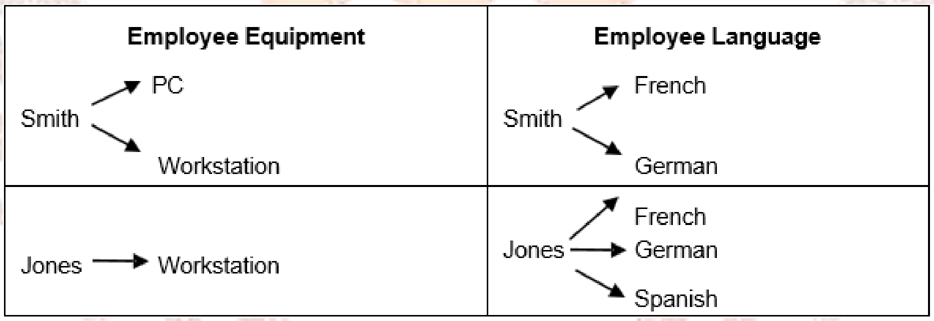


Fig 3.3: Equipment and Language are Independent Multivalued Facts about Employee

Note that this relation Emp_Profile is in BCNF. Here all the attributes are required for the uniquely identifying the records, hence *Emp_name + Equipment +Language* is the primary key. But, this relation still contains redundancy problem.

Therefore we can further decompose it to higher normal form i.e. 4NF to resolve the problem of redundancy. In the next section we will see the definition of 4NF and how MVDs are associated with this.

4.5 Fourth Normal Form

The Fourth Normal form is the next higher normal form after 3NF/BCNF. It is based on the concept of multi-valued dependency. A Multivalued dependency arise in a condition where a relation contains at least ways three columns, one column has several rows whose values are similar to the values of a single row of one of the other columns (See Table 3.22)

A more formal definition of MVD states that: "A multi valued dependency exists if, for each value of an attribute A, there exists a finite set of values of attribute B that are associated with A and a finite set of values of attribute C that are also associated with A. Attributes B and C are independent of each other."

4NF - Addressing Multi-Valued Dependencies

Let us take the example of a relation Branch_Staff_Client (Table 3.22) which contains information about the various clients for a bank branch, the various staff who addresses the client's needs and the various requirements of each client.

Table 3.22: Branch_Staff_Client

BranchNumber	**StaffName**	**ClientName**	**ClientRequirement**
B-41	Radha	Surya	A
B-41	Radha	Ravi	B
B-41	Smitha	Surya	B
B-41	Smitha	Ravi	A

The above relation contains MVD. In this relation, the Client name determines the Staff name that serves the client and the Client name also determines the client require¬ments. But Staff_name and Client_requirement are not dependent on each other i.e. they both are independent facts about Client_Name. Hence, there exists MVD.

Multi-valued dependencies in Branch_Staff_Client relation can be symbolically represented as:

Clientname $\rightarrow$ StaffName

Clientname $\rightarrow$ ClientRequirements

The necessary conditions for the Fourth Normal form are as follows:

1. The table should be in Boyce-Codd normal form
2. There should be no multi-valued dependencies.

Thus the 4NF basic objective is to eliminate multi-valued dependencies from the relation. In order to remove multi-valued dependencies from a table, we need to decompose the table and shift the related columns into separate tables along with a copy of the determinant. This copy will serve as a foreign key to the original table.

Table 3.23: Branch_Staff Table before Fourth Normal Form

BranchNumber	StaffName	ClientName
B-41	Radha	Surya
B-41	Radha	Ravi
B-41	Smitha	Surya
B-41	Smitha	Ravi

Table 3.24: Branch_staff Table after Fourth Normal Form

BranchNumber	**ClientName**	**BranchNumber**	**StaffName**
B-41	Surya	B-41	Radha
B-41	Ravi	B-41	Smitha

4.6 Fifth Normal Form

Fifth normal form is the highest normal form used in relational database designing. It is mostly used when there is a large relational database.

The Fifth Normal form was developed by an IBM researcher, Ronald Fagin. According to the Fagin's theorem, "The original table must be reconstructed from the tables into which it has been decomposed." 5NF allows decomposing a relation into three or more relations.

Fifth normal form is based on the concept of join dependency. Join dependency means that a relation, after being broken down into 3 or more smaller relations, should be capable of being combined all over again on similar keys to result in the creation of the original table. The join dependency is more general form of multi-valued dependency.

A relation (R) meets the condition of fifth normal form $(R_1, R_2 \ldots R_n)$ if R is equal to the join of $R_1, R_2 \ldots R_n$
(Here, Ri are subsets of the set of attributes of R)

Any relation R is said to be in 5NF or PJNF) (project - join normal form) if for all join dependencies in any case one of the following holds.

(a) (R1, R2…. Rn) is trivial join-dependency (that is, one of Rt is R)
(b) Every Ri is an candidate key for relation R.

Definition of Fifth Normal Form: A relation should be in fifth normal form (5NF) if and only if all join dependency in the table is connoted by candidate keys of the relation.

Table before Fifth Normal Form

Table 3.25: Dept-Subject

Dept.	**Subject**	**Student**
Comp. Sc.	CP1000	John Smith
Mathematics	MA1000	John Smith
Comp. Sc.	CP2000	Arun Kumar
Comp. Sc.	CP3000	Reena Rani
Physics	PHI 000	Raymond Chew
Chemistry	CH2000	Albert Garcia

Table after Fifth Normal Form

Table 3.26, 3.27 and 3.28 are formed after converting Table 3.25 into Fifth Normal Form.

Table 3.26: Dept-Subject

Dept.	Subject	Student
Comp. Sc.	CP1000	John Smith
Mathematics	MA1000	John Smith
Comp. Sc.	CP2000	Arun Kumar
Comp. Sc.	CP3000	Reena Rani
Physics	PHI 000	Raymond Chew
Chemistry	CH2000	Albert Garcia

Table 3.27: Subject-Student

Subject	**Student**
CP1000	John Smith
MA 1000	John Smith
CP2000	Arun Kumar
CP3000	Reena Rani
PHI 000	Raymond Chew
CH2000	Albert Garcia

Table 3.28: Dept-Student

Dept.	Student
Comp. Sc.	John Smith
Mathematics	John Smith
Comp. Sc.	Arun Kumar
Comp. Sc.	Reena Rani
Physics	Raymond Chew
Chemistry	Albert Garcia

SELF-ASSESSMENT QUESTIONS – 3

5. How does Normalisation help?
   a) By eliminating various database anomalies
   b) By minimising redundancy
   c) By eliminating data inconsistency
   d) All of the above
6. An attribute (column) is said to be if its value can be determined by any one or more attributes of the primary key, but not all.
7. A table which is in ________________ normal form may contain redundancies due to transitive dependencies.
8. The Fifth Normal form is usually useful when we have large relational data models. (True/False)
9. The join dependency is more generalised form of __________________ dependency.
10. An FD is a special case of an MVD and every FD is an MVD. (True/False)
11. The fifth normal form is also called ___________________.

5. NORMALISATION AND DATABASE DESIGN

Normalisation and database design are two closely integrated terms. In this section, we will study about the relationship between the two. A database design refers to the process of moving from the real-life business models to the database model, which meets those requirements. Normalisation is one such technique.

You have already studied in detail about normalisation. Normalisation as you have learnt earlier is a technique that is used for designing relations in which data redundancies are minimised.

By using the normalisation technique we want to design for our relational database that has following set of properties:

1) It holds all the data required for the purposes that the database is to serve.
2) It must have as less redundancy as possible,
3) It must hold manifold values for types of data that requires them,
4) It must allow efficient updates of the data in the database, and
5) It must avoid the risk of accidental data loss.

You have studied that there are mainly five normal forms. However, of these, there are three forms that are most commonly used practically. These three forms are the: first normal, second normal, and third normal. When you convert an ER (Entity-Relationship) model in Third Normal Form (3NF) to a relational model:

- Relations are referred as tables.
- Attributes are referred as columns.
- Relationships are referred as data references (primary and foreign key references).

Third Normal Form is considered as the standard normal form the viewpoint of the relational database model. Normalised database tables are easy to maintain and also easily understood by the developers. However it is not necessary that a fully normalised database is the best database design. In most of the cases, it is suggested that database must be optimised up to third normal form. Therefore we often require to denormalise our database (you will study in detail about denormalisation in the next section 3.6) relations so as to meet the optimum performance level. Therefore we can say that an efficiently normalised database has the following advantages:

1. Simplified and easy data maintenance
2. Enhanced speed of data processing
3. Enhanced design quality

SELF-ASSESSMENT QUESTIONS – 4

12. From a ________________ point of view, it is standard to have tables that are in Third Normal Form.
13. According to relational database rules, a completely normalised database always has the best performance. (True/False).
14. Denormalisation is done to increase the performance of the database. (True/False).

6. DENORMALISATION

Normalisation is implemented to preserve data integrity. Nevertheless, in a real world project, you need some level of data redundancy for reasons relating to performance or maintaining history.

During the normalisation process, you need to decompose database tables into smaller tables. However if you create more tables, the database needs to execute more joins while solving queries. But remember joins has a poor effect on performance. Hence, denormalisation is done to enhance the performance.

Denormalisation is the process of converting higher normal forms to lower normal forms with the objective of getting faster access to database.

Keep in mind that denormalisation is a common and essential element of database design process, but it must follow appropriate normalisation.

Techniques used for Denormalisation: There are mainly four techniques used for denormalisation. Given below is a brief summary of the techniques:

1. ***Duplicate Data:*** The most easy technique is the method of adding the duplicate data into the relational table. Doing this will help to minimise the number of joins which are required to execute a given query. It also minimises the CPU and I/O resources being utilised as well as boosts up the performance.
2. ***Summary data:*** Summarising the data stored in the relational database table is another useful technique used for denormalising the database. In this technique the records are summarised in some summary columns thereby reducing the number of records stored in a table. This technique enhances the database performance as now the database server needs to process lesser records for a given query execution.
3. ***Horizontal partitioning:*** Horizontal Fragmentation is another denormalisation technique in which the database table is split by rows. This reduces the number of records per table and hence drives the performance.
4. ***Vertical fragmentation:*** Vertical fragmentation breaks tables/relations by columns. The method makes 2 or more than 2 tables by allocating the original key to all and allocating a few of the non-key columns to every newly made identical keyed table.

SELF-ASSESSMENT QUESTIONS – 5

15. Denormalisation is a technique to move from higher to lesser normal forms of database modelling in order to get faster access to database.(True/ False)
16. ____________________ splits tables by rows, thus reducing the number of records per table.

Activity 2

What problems can you encounter when you decide to introduce some denormalisation into your model?

7. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- Normalisation is a technique used to design tables in which data redundancies are minimised.
- Normalisation is based on the concept of normal forms i.e. 1NF, 2NF , 3NF, BCNF , 4NF and 5NF.
- A partial dependency refers to a condition in which any attribute is functionally dependent upon only a part of a multi-attribute Primary key.
- A transitive dependency is a condition where attribute is functionally dependent on another non-key attribute.
- A table is in 1NF when every key attributes is defined and when all the remaining attributes are dependent upon Primary key .
- A table is in 2NF if it is in 1NF and contains no partial dependencies.
- Boyce-Codd (BCNF) is a special case of 3NF in which all the determinant keys are also candidate keys.
- Denormalisation is done to enhance the performance of the database.

8. GLOSSARY

- ***Boyce-Codd normal form (BCNF):*** When each determinant in a relation is a candidate key, a relation is said to be in Boyce-Codd Normal Form (BCNF).
- ***First normal form:*** A database is said to be in if each of the values of all the attributes in a relation are atomic in nature.
- ***Functional dependency:*** Two attributes A and B in any relation R are said to possess a functional dependency (FD) if for each distinct value of A, there is only one value of attribute B associated with it.
- ***Normalisation:*** It is the process of obtaining good database design by decomposing the relations into normal forms based on functional dependencies.
- ***Second normal form:*** If every non-key attribute of a relation schema is fully FD (functionally dependent) on the key then the relation is said to be in 2NF.
- ***Third normal form:*** A relation is said to be 3NF if it is in 2NF and no and does not contain transitive dependencies.

9. TERMINAL QUESTIONS

1. Explain the various types of database anomalies.
2. Define functional dependency. Give examples.
3. What is normalisation? Explain why normalisation is required in database design.
4. Explain 1NF with a suitable example.
5. Explain second normal form with example.
6. Explain transitive dependencies with examples. Show how these are significant in designing databases.
7. What is a third normal form? Give example.
8. Explain how BCNF and 3 NF differ.
9. What is fourth normal form and fifth normal form? Explain with an example.
10. Write a short note on denormalisation.

10. ANSWERS

Self-Assessment Questions

1. True
2. Schema
3. (b)Inconsistency
4. Redundancy
5. (d) All of the above
6. Partially dependent
7. Second
8. True
9. Multi-valued
10. True
11. Project-Join Normal Form (PJNF).
12. Rational model
13. False
14. True
15. True
16. Horizontal Fragmentation

Terminal Questions

1. There are mainly three types of anomalies in database: first is redundancy, second is inconsistency and the third is update. Refer Section 3 for more details.
2. Functional dependency is a type of constraint in which attribute is dependent upon another attribute. . Refer Section 2 for more details.
3. Normalisation is the process of designing a good database by converting it into various normal forms by eliminating all the database anomalies. Refer Section 4 for more details.
4. In, 1NF, all attribute values of a relation are atomic in nature. Refer Section 4 for more details.

5. When all the non-key attributes of a relational schema are fully functionally dependent on the primary key then that relation is said to be in 2NF. Refer Section 4 for more details.
6. A transitive dependency is a condition where one attribute is functionally dependent on another non-key attribute. Refer Section 4 for more details.
7. A table is said to be in 3NF if it is in 2NF and also it does not contains any transitive dependencies. Refer Section 4 for more details.
8. Boyce-Codd (BCNF) is a strict case of 3NFwhere all the determinant keys are also candidate keys. Refer Section 4 for more details.
9. A 4NF table necessarily has two conditions i.e. firstly it must be in Boyce-Codd normal form and secondly it must be free from any multi- valued dependencies. Refer Section 4 for more details.
10. Denormalisation is done to enhance the performance of a normalised database. Refer Section 6 for more details.

11. REFERENCES

- Peter Rob, Carlos Coronel, *"Database Systems: Design, Implementation, and Management",* (7thEd.), Thomson Learning
- Silberschatz, Korth, Sudarshan, *"Database System Concepts",* (4th Ed.), McGraw-Hill
- Elmasari Navathe, *"Fundamentals of Database Systems",* (3rd Ed.), Pearson Education Asia

E-references

- http://www.techbaba.com/q/2494-denormalization+database.aspx
- http://www.gantthead.com/process/popup.cfm?ID=23451