CHAPTER 4

Relational Database Design

4.0 INTRODUCTION

- A relational database is made up of a number of relations and the relational database scheme which consists of number of attributes.
- There are some features of relational databases where data is organized as logically independent tables. Because data is organized in the form of table, data retrieval, updates, processing and structural changes are more efficient.
- In this chapter, we focus on the issues involved in the design of a database scheme using a relational model.

4.1 INTRODUCTION TO RELATIONAL DATABASE DESIGN (April 17)

- The goal of a relational database design is to generate set of relation schemes that allows us to store information without unnecessary redundancy and also to retrieve the information easily.
- One approach is to design schemes with normal forms. The normal forms used to ensure that various types of anomalies and inconsistencies are not introduced into the database.
- To determine whether a relation scheme is in one of the normal forms, we actually need the additional information about real world. This additional information is given by a collection of constraints called data dependencies.

4.1.1 Undesirable Properties of a RDB Design (April 15, 16; Oct. 17)

- To understand the pitfalls in any relational database, we will see a simple example of student database.
- A student's information is maintained in the database, such as roll no, name, address, class, subject, teacher who taught this subject with teacher_id.
- This information is maintained in a single table, (Refer Table 4.1).

Table 4.1: Student

Roll no	name	address	class	subject	t_id	teacher name
1	Amita	Pune	TY	java	1	Pawar
1	Amita	Pune	TY	php	2	Kadam
2	Amey	Mumbai	FY	C	3	Patil
3	Vinita	Nashik	SY	C++	4	Rahane
3	Vinita	Nashik	SY	Mysql	1	Pawar
2	Sagar	Pune	TY	php	1	Kadam

- By observation, we can see that data is repeated again and again that is called redundancy. Redundancy consumes more space and resources that necessary.

- Redundancy may lead to inconsistency i.e. because of typing mistakes loss of information occurs. This can be removed by separating these tables which are keys.

For example:

1. Create a table student which have rollno, name and address.
2. Create a table class with class_id and class.
3. Create a table teacher with T_id and T_name.
4. Then decide the relationship between them and use foreign key concept.

- An important feature of relational database is that it cannot be spread across several table. The users can ask queries and database provides proper results. For this, the database should be designed properly.
- For designing the tables, consider following two important criterias:
  1. When a database is stored, avoid redundancy i.e. repetition of data should not be available in a database.
  2. When a database is stored, it must be in a normalize form. This will provide better efficiency and reliability. This also allows as to use storage space efficiently, eliminate data redundancy and also reduce inconsistency in database.
- Lets consider bank database from scratch. If one customer can goto the bank to take loan. So the necessary, information which we should maintain is shown in Table 4.2.

Table 4.2: Lending a loan

branch_name	branch_city	assets	cust_name	loan_no	Amount
F C Road	Pune	4,10,000	Sayali	1	2,00,000
M G Road	Pune	37,00,000	Ashmit	4	5,00,000
G P Road	Nagpur	4,50,000	Unmesh	1	2,00,000

- Suppose, if we want to add a new loan in our database i.e. lending.Assume that the loan is made by F C Road to Sayali with the amount of Rs. 50,000 and loan_no is 3. So we need to add the tuple in the lending relation.

```
(F C Road, Pune, 400000, Sayali, 3,50000).
```

- In general we were repeating assets and city data for a branch, for each loan made by that branch. So data is repeated (Redundancy occurred).
- Lets consider that by mistake the branch_city name is typed as poona in new tuple. If a query is written to find the loans done into pune city, it won't consider the newly added record. So loss of information (inconsistency) is also available.
- Similarly, if user wants to delete a loan tuple, while deleting branch_name will also get deleted. If that is the only loan made by that branch, the database will lose that information about branch_name and its assets also.
- To eliminate redundancy and inconsistency, the data should be in normalize format.

4.2 FUNCTIONAL DEPENDENCY (April 15, 16, 17, 18; Oct. 17)

- It is particular type of constraint. In a database, the value of one attribute or combination of attributes is used to look up a value of another attribute.
- Normalization techniques use functional dependencies and keys to create the most efficient database structure possible in order to obtain these values.

- A functional dependency is an association between the columns of a relation. For example, consider the following example of the relation enrolled students shown in Fig. 4.1.

student_id	student_name	student_phone
101	Abhijay	3318113
102	Sagar	3388026
103	Nikita	5146048

Fig. 4.1: Enrolled Student Relation

- Here, one value of all the attribute student_id can determine a unique value from the attributes student_name and student_phone. For example, student_id 103 determines that the student name is Nikita and student phone is 5146048.
- Thus, student name and student phone are functionally dependent on student_id. Another way of saying this is that student_id functionally determines student_name and student_phone. Therefore, a given value of one attribute will always return the same value of another attribute.
- Here, student_id is a determinant and the primary key. But this is not always true. A relation can contain a determinant that does not have unique values because functional dependency actually addresses how attributes are related.

4.2.1 Basic Definition (Concepts)

- Functional dependencies are a constraint of set of legal relations. They allow us to express facts about the enterprise that we are modelling with one database.
- It is generalization of concept of superkey. Lets define the superkey. Let R be a relation schema. A subset K of R is a superkey of R if in any legal relation r (R), for all pairs of t_1 and t_2 of tuples in r such that $t_1 \neq t_2$, then $t_1[k] \neq t_2[k]$. That is no tuples in legal relation r (R) can have same values on attribute set K.
- The notion of functional dependency generalizes the notion of super key.

  Let $\alpha \subseteq R$ and $\beta \subseteq R$. The functional dependency $\alpha \rightarrow \beta$.

  Here, R is the relation, $\alpha \subseteq R$ and $\beta \subseteq R$. The β is functionally dependent on α. For any value of α there should be only one value in α which is associated with one value inβ.
- We can define superkey using functional dependency as K is superkey of R, if $K \rightarrow R$, that is, K is a superkey if whenever $t_1[k] = t_2[k]$, $t_1[R] = t_2[R]$ (i.e. $t_1 = t_2$).
- Functional dependencies allow us to express constraint that cannot be expressed using superkeys.

Example 1: In relation lending (used in Table 4.2), the attribute branch_name cannot be a superkey because we know that a branch may have many loans to many customer. But, we do expect the functional dependency:

```
branch_name → branch_city
```

- To hold, since a branch may be located in exactly one city.
- We shall use functional dependencies in following two ways:
  1. To specify constraints on set of legal relations. If F is set of functional dependencies which are satisfied on schema R, then we say that F holds on R.
  2. To test the relations to see whether the relations are legal under a given set of functional dependencies. If a relation r is legal under F, set of functional dependencies, we say that r satisfied F.

Example:

- Consider following relation r of Fig. 4.2.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_3	c_2	d_3
a_3	b_3	c_2	d_4

Fig. 4.2: Sample Relation 'r'

- The functional dependency A → C is satisfied. Since, every pair of tuple having same A-value also have same C-value. However, functional dependency C → A is not satisfied. To see this, consider the tuples $t_1 = (a_2, b_3, c_2, d_3)$ and $t_2 = (a_3, b_3, c_2, d_4)$.
- These two tuples have the same c-value c_2 but they have different A-values; a_2 and a_3 respectively. Thus, we found a pair of tuples t_1 and t_2 such that $t_1[C] = t_2[C]$ but $t_1[A] \neq t_2[A]$.

  AB → C, AB → D are some other functional dependencies. (Note: AB is {A, B}).
- In order to verify a given FD α → β is satisfied by a relation R on a relation scheme R, we find any two tuples with the same α value, if the FD α → β is satisfied in R, then the β values in these tuples must be the same. We repeat this procedure until we have examined all such pairs of tuples with the same α value.
- A simpler approach involves ordering the tuples of R on the α values so that the tuples with the same α values are together. Then it is easy to verify if the corresponding β values are also the same and verify if R satisfies the FD α → β.

Example 2: Consider the relation schema of Fig. 4.3.

branch_name	loan_no	cust_name	amount
SBI	15	Sujal	10000
SBI	15	Ashmit	10000
ICICI	20	Sunita	20000
ICICI	20	Deepali	20000
ICICI	20	Pranali	20000
CBI	45	Sonal	80500
ICICI	20	Omkar	20000

Fig. 4.3: loan_cust Relation

- The set of functional dependencies that hold on relation schema is,

  determinant → loan_no → amount

  loan_no → branch_name← dependent

  but it can not hold

  loan_no → cust_name

  The left hand side and right hand side of an FD are sometimes called the **determinant** and **dependent** respectively.

Example 3: Consider the following relation schema of Fig. 4.4.

Schedule = (Prof, Course, Room, Max_enrol, Day, Time)

Prof	Course	Room	Max_Enroll	Day	Time
Rahul	353	A 532	40	Mon	1145
Anagha	351	C 320	60	Wed	115
Juhi	354	C 320	60	Tue	115
Juhi	354	H 940	300	Thu	845
Juhi	355	B 278	45	Tue	1015
Ajay	321	H 940	200	Thu	845
Rahul	353	A 532	40	Mon	1145

Fig. 4.4: Schedule Relation

The FD course → Prof is satisfied by this relation. But Prof → course is not satisfied.

4.2.2 Closure of Attribute Set

- We have not yet given an effective algorithm for computing the closure F^+ of F.
- To test whether α is superkey or not, we must have some method to computing the set of attributes functionally determined by α.
- Let α be the set of attributes. We call the set of all attributes functionally determined by α under a set of F of FD's as the closure of α under F; denote it as α^+.
- The algorithm for this is as follows:

Algorithm:

```
Result:=α;
while (changes to result) do
for each FD β⇒γ in F do
begin
    if β⊆ result then
    result:= result ∪γ
end
```

4.2.2.1 Algorithm to Identify Closure Set of Attributes

- The above algorithm is explained in simple manner.
- Here, x is a result.
  1. Equate an attribute or attributes to x for which closure need to be identified.
  2. Repeatedly take functional dependencies one by one and check whether left hand side attribute is available in x or not. If available, then add right hand side attribute of functional dependency to x.
  3. Repeat step 2 as many times as possible to cover all functional dependencies.
  4. Stop the process if no more attributes can be added to x.

4.2.2.2 Applications of Closure Set of Attributes

- The applications of closure set of attributes includes:
  1. To identify additional functional dependencies.
  2. To identify equivalences.
  3. To identify candidate keys.
  4. To identify irrelational set of functional dependencies or canonical form of functional dependencies.

Example 4: To illustrate how the algorithm works, we shall use the algorithm to compute $(AG)^+$ with the FDs. **Oct. 17**

$A \rightarrow B$

$A \rightarrow C$

$CG \rightarrow H$

$CG \rightarrow I$

$B \rightarrow H$

The input to the algorithm will be,

$F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$ and $\alpha = (AG)$

We start with result := AG.

The first time we execute while loop we find that $A \rightarrow B$ causes us to include B in result since $A \rightarrow B$ is in F, $A \subseteq$ result, so result := result $\cup$ B and result = AGB.

To test the second FD,

$A \rightarrow C$ causes us to include C in result and result becomes AGBC.

Similarly,

$CG \rightarrow H$, causes us to include H in result and result becomes AGBCH.

$CG \rightarrow I$, causes to include I in result and result becomes AGBCHI.

$B \rightarrow H$, causes to include H which is already included in result.

So, new value of result is (AGBCHI).

During next iteration of while, no new attribute gets added to result so algorithm terminates. Since algorithm terminates, if the result contains all the attributes, then α is superkey as it determines all the attributes of relation otherwise α is not superkey.

Example 5: $\alpha = (CG)$ find $(CG)^+$

$A \rightarrow B$ and $A \rightarrow C$ will not add anything to result.

$CG \rightarrow H$ will add H to result and result = CGH

$CG \rightarrow I$ will add I and result = CGHI

$B \rightarrow H$ will not add anything

So new value of result after first iteration is (CGHI).

In 2nd iteration, nothing will be added to result. So algorithm terminates, but result does not contain all attributes of relation, so α i.e. (CG) is not a superkey.

Example 6: Compute the closure for relational scheme. **April 16**

$R = \{A, B, C, D, E\}$ where the functional dependencies are:

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

List candidate keys of R.

Solution: $R = \{A, B, C, D, E\}$

F is the set of functional dependencies = $\{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$

Compute the closure for each β in $\beta \rightarrow \gamma$ in F.

We will find the closure of A, B, C, D, E separately.

1. Closure for A:

Iteration	Result	Using
1	A	
2	ABC	$A \rightarrow BC$
3	ABCD	$B \rightarrow D$
4	ABCDE	$CD \rightarrow E$
5	ABCDE	

$A^+ = ABCDE$. Hence, A is a super key.

2. Closure for CD:

Iteration	Result	Using
1	CD	
2	CDE	$CD \rightarrow E$
3	ACDE	$E \rightarrow A$
4	ABCDE	$A \rightarrow BC$
5	ABCDE	

$CD^+ = ABCDE$. Hence, CD is a super key

3. Closure for B:

Iteration	Result	Using
1	B	
2	BD	$B \rightarrow D$
3	BD	

$B^+ = BD$. Hence, B is NOT a super key

Try applying Armstrong axioms, to find alternate keys.

$B \rightarrow D$

$BC \rightarrow CD$ (by Armstrong's augmentation rule).

4. Closure for BC:

Iteration	Result	Using
1	BC	
2	BCD	$BC \rightarrow CD$
3	BCDE	$CD \rightarrow E$
4	ABCDE	$E \rightarrow A$

$BC^+ = ABCDE$. Hence BC is a super key

5. Closure for E:

Iteration	Result	Using
1	E	
2	AE	$E \rightarrow A$
3	ABCE	$A \rightarrow BC$
4	ABCDE	$B \rightarrow D$
5	ABCDE	

$E^+ = ABCDE$

A and E are minimal super keys.

To see whether CD is a minimal super key, check whether its subsets are super keys.

$C^+ = C$

$D^+ = D$

Since C and D are not super keys, CD is a minimal super key.

To see whether BC is a minimal super key, check whether its subsets are super keys.

$B^+ = CD$

$C^+ = C$

Since B and C are not super keys, BC is a minimal super key.

Conclusion on solution:

Since A, BC, CD, E are minimal super keys, they are the candidate keys.

A, BC, CD, E

4.2.3 Armstrong's Axioms

(Oct. 16, April 17, 19)

- It is not sufficient to consider the given set of functional dependencies. Rather, we need to consider all functional dependencies that hold.
- We shall see that, given a set F of FDs, we can prove that certain other FD's hold. We say that F logically implies such FDs.

Example 7: Suppose relation scheme R = (A, B, C, G, H, I) and set of FDs is

$\{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

The FD $A \rightarrow H$ is logically implied. This is because suppose we have two tuples t_1 and t_2 such that $t_1[A] = t_2[A]$. As there is FD $A \rightarrow B$ we have $t_1[B] = t_2[B]$. Also we have FD', $B \rightarrow H$, so we have $t_1[H] = t_2[H]$. Therefore, we have shown that whenever t_1 and t_2 are tuples such that $t_1[A] = t_2[A]$, it must be that $t_1[H] = t_2[H]$.

But that is exactly the definition of $A \rightarrow H$.

- There are some techniques for finding closure of FD.

Example: loan (br_name, loan_no, cust_name, amount)

and

loan_no → amount

loan_no → cust_name

∴ loan_no → br_name is logically implies.

Given F, we can compute F^+ directly from the formal definition of functional dependency. But, if F is very large, this is a lengthy process, so we use certain simple techniques.

One technique is based on there axioms or rules of inference, are called as Armstrong's axioms, used to compute F^+.

- The Armstrong's axioms are as follows:

1. **Reflexivity:** If α is a set of attributes and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ holds.

   Example 8:

α	β
rollno.name	name

   ∴ rollno.name → name

2. **Augmentation:** If $\alpha \rightarrow \beta$ holds and γ is a set of attributes then $\gamma\alpha \rightarrow \gamma\beta$ holds.

   Example 9:

   rollno → name

   ∴ city.rollno → city.name

3. **Transitivity:** If $\alpha \rightarrow \beta$ holds and $\beta \rightarrow \gamma$ holds then $\alpha \rightarrow \gamma$ holds.

Example 10:

rollno → city

city → state

∴ rollno → state

- These rules do not generate any incorrect FD's. Also they are complete, as they give us F^+ from F. Though, these rules are sound and complete, it's still lengthy to compute entire F^+ using these 3 rules. To simplify this some additional rules are listed below. These rules are proved using the above Armstrong's axioms.

1. **Union Rule:** If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds

   Example 11:

   rollno → name

   rollno → city

   ∴ rollno → name city

2. **Decomposition Rule:** If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds.

   Example 12:

   rollno → city name

   ∴ rollno → city

   ∴ rollno → name

3. **Psuedo Transitivity Rule:** If $\alpha \rightarrow \beta$ holds, and $\gamma\beta \rightarrow \delta$ holds then $\alpha\gamma \rightarrow \delta$ holds.

4.2.4 Closure of Relation (F^+)

(April 16, 17; Oct. 16, 17)

- The closure of functional Dependency means the complete set of all possible attributes that can be functionally derived from given functional dependency using the inference rules known as Armstrong's Rules.
- If "F" is a functional dependency then closure of functional dependency can be denoted using "{F}+".

Example 13: By applying these rules on above relation R with attributes A, B, C, G, H, I we can find some other FD's.

$\{A \rightarrow B, A \rightarrow C, B \rightarrow H, CG \rightarrow H, CG \rightarrow I, AG \rightarrow CG\}$

(i) $A \rightarrow H$ Using transitivity rule ($A \rightarrow B$ and $B \rightarrow H$)

(ii) $CG \rightarrow HI$ Using union rule ($CG \rightarrow H$ and $CG \rightarrow I$)

(iii) $AG \rightarrow CG$ Using augmentation rule ($A \rightarrow C$)

(iv) $AG \rightarrow I$ Using pseudo transitivity rule ($A \rightarrow C$, $CG \rightarrow I$)

or transitivity rule ($AG \rightarrow CG$ and $CG \rightarrow I$)

Example 14: Consider the relation R = (A, B, C, D, E) and set of FDs defined on R, F as,

$\{A \rightarrow B, CD \rightarrow E, A \rightarrow C, B \rightarrow D, E \rightarrow A\}$.

Using the above inference rules, we compute F^+ as follows:

(i) $A \rightarrow D$: Using transitivity rule ($A \rightarrow B, B \rightarrow D \Rightarrow A \rightarrow D$).

(ii) $A \rightarrow E$: Using pseudo transitivity rule ($A \rightarrow C, CD \rightarrow E \Rightarrow A \rightarrow E$)

(iii) $CD \rightarrow A$: Using transitivity rule ($CD \rightarrow E, E \rightarrow A \Rightarrow CD \rightarrow A$)

(iv) $BC \rightarrow E$: Using pseudo transitivity rule ($B \rightarrow D, CD \rightarrow E \Rightarrow BC \rightarrow E$)

(v) $A \rightarrow BC$: Using union rule ($A \rightarrow B, A \rightarrow C \Rightarrow A \rightarrow BC$).

4.3 CONCEPT OF DECOMPOSITION (Oct. 18, April 17)

- Decomposition means breaking up a relation schema into smaller relation schemas. In order to have a good database design it is necessary to decompose big relations into smaller ones.

Goals of Decomposition:

1. To eliminate redundancy by decomposing a relation into multiple relations.
2. It is also important to check that decomposition of relation does not lead in to a bad design.

- In decomposition, a relation is replaced with a collection of smaller relations with specific relationship between them.
- Formally, the decomposition of a relation schema R is defined as its replacement by a set of relation sehemas such that each relation schema contains a subset of the attributes of R.
- For example, the relation schema BOOK_PUBLISHER has two redundancy and null values thus, the schema of this relation requires to be modified so that it has less redundancy of information and fewer null values.
- The undesirable features form the relation schema BOOK_PUBLISHER can be eliminated by decomposing it into two relation schemas as given below:

  `BOOK_DETAIL (ISBN, Book_Title, P_ID) and`

  `PUBLISHER_DETAIL(P_Id, P_Name, Phone)`

- Fig. 4.5 shows the decomposition of the BOOK_PUBLISHER relation and shows the instances corresponding to decomposed relation schemas BOOK_DETAIL and PUBLISHER_DETAIL.

BOOK_PUBLISHER

ISBN	Book_Title	P_ID	P_Name	Phone
001-982-760-9	Java	P001	Pragati Publication	713401900
001-353-921-1	C++	P001	Pragati Publication	713401910
001-986-650-5	C	P001	Pragati Publication	713401910
002-679-980-4	RDBMS	P002	Sunshine Publisher	654890915

BOOK_DETAIL

ISBN	Book_Title	P_ID
001-982-760-9	Java	P001
001-353-921-1	C++	P001
001-986-650-5	C	P001
002-679-980-4	RDBMS	P002
002-765-880-2	LINUX	P002
004-765-409-7	UNIX	P003

PUBLISHER_DETAIL

P_ID	P_Name	Phone
P001	Pragati Publication	713401910
P002	Sunshine Publisher	654890915
P003	Success Publication	767898520

Fig. 4.5

- Now, a tuple for a new publisher can be inserted easily in relation PUBLISHER_DETAIL, even if no book is associated with that particular P_ID in relation BOOK_DETAIL. Similarly, a tuple for a particular P_ID in relation PUBLISHER_DETAIL can be deleted without losing any information (taking foreign key constraints into account).
- Moreover, name or phone for a particular P_ID can be changed by updating a single tuple in relation PUBLISHER_DETAIL. This is more efficient than updating several tuples and the scope for inconsistency is also eliminated.

Definition of Decomposition:

- Let R be a relation schema then a set of relation schemas $\{R_1, R_2, ..., R_n\}$ is a decomposition of R if,
  1. $R = R_1 \cup R_2 \cup .... \cup R_n$
  2. Each R_i is a subset of R (for i = 1, 2, ..., n)
- This means that relation R contains attributes $A_1 ... A_n$. A decomposition of R consists of replacing R by two or more relations such that; Each new relation scheme contains a subset of the attributes of R (and no attributes that do not appear in R), and every attribute of R appears as an attribute of one of the new relations.

  For example: A relation R(x, y, z) this relation can be decomposed into 2 subsets i.e., R_1(x, z) and R_2 (y, z). If we union R_1 and R_2, we get $R = R_1 \cup R_2$.
- The decomposition of a relation scheme $R = \{A_1, A_2, A_3, ..., A_n\}$ is it's replacement by a set of relation schemas $\{R_1, R_2, R_3, ..., R_m\}$ such that, OR

  $R = R_1 \bowtie R_2 \bowtie R_3 \bowtie ..... \bowtie R_m$.

- For Example: Consider the relation schema for relation lending and it is shown in Table 4.3.

Table 4.3: Lending a Laon

branch_name	branch_city	assets	cust_name	loan_no	amount
F C Road	Pune	4,10,000	Sayali	1	200000
M G Road	Pune	37,00,000	Ashmit	4	500000
G P Road	Nagpur	4,50,000	Unmesh	1	200000

- The lending schema is decomposed into following two schemas:

  `branch_cust (branch_name, branch_city, assets, cust_name)`

  `cust_loan (cust_name, loan_no, amount)`

- These two relations are actually constructed by following relational algebra expression from Table 4.1.

  `branch_cust = π branch_name, branch_city, assets, cust_name(lending)`

  `cust_loan = π cust_name, loan_no, amount (lending).`

- Let's add certain records to these two newly created relations. This is shown in Fig. 4.6 and Fig. 4.7.

branch_name	branch_city	assets	cust_name
F C Road	Pune	400000	Sayali
M G Road	Pune	3700000	Ashmit
G P Road	Nagpur	450000	Unmesh

Fig. 4.6: branch_cust Relation

cust_name	loan_no	amount
Sayali	1	200000
Ashmit	4	500000
Unmesh	1	200000
Sayali	2	100000

Fig. 4.7: cust_loan Relation

- We can reconstruct the original lending relation as follows:

  `branch_cust ⋈ cust_loan`

- The result of this natural join is shown in Fig. 4.9 and the intermediate result is shown in Fig. 4.8.

branch_name	branch_city	assets	cust_name	cust_name	loan_no	amount
F C Road	Pune	400000	Sayali	Sayali	1	200000
F C Road	Pune	400000	Sayali	Ashmit	4	500000
F C Road	Pune	400000	Sayali	Unmesh	1	200000
F C Road	Pune	400000	Sayali	Sayali	2	100000
M G Road	Pune	3700000	Ashmit	Sayali	1	200000
M G Road	Pune	3700000	Ashmit	Ashmit	4	500000
M G Road	Pune	3700000	Ashmit	Unmesh	1	200000
M G Road	Pune	3700000	Ashmit	Sayali	2	100000
G P Road	Nagpur	450000	Unmesh	Sayali	1	200000
G P Road	Nagpur	450000	Unmesh	Ashmit	4	500000
G P Road	Nagpur	450000	Unmesh	Unmesh	1	200000
G P Road	Nagpur	450000	Unmesh	Sayali	2	100000

Fig. 4.8

- cust_name is the common attribute. So select common cust_name which will give the natural join.

branch_name	branch_city	assets	cust_name	loan_no	amount
F C Road	Pune	400000	Sayali	1	200000
F C Road	Pune	400000	Sayali	2	100000
M G Road	Pune	3700000	Ashmit	4	500000
G P Road	Nagpur	450000	Unmesh	1	200000

Fig. 4.9: Result of natural join

- Here, the result has one additional tuple i.e. (F C Road, Pune, 400000, Sayali, 2, 100000).

Problems occur in decomposition:

1. Loss of information called lossy decomposition.
2. Lossing some of functional dependency relationship in lossy decomposition, so we use lossless decomposition.

4.4 DESIRABLE PROPERTIES OF DECOMPOSITION (LOSSLESS JOIN, LOSSY JOIN, DEPENDENCY PRESERVATION)

Oct.-13, 17; April-15, 17

- A relation scheme R can be decomposed into a collection of relation schemas to eliminate anomalies contained in the original relation scheme R.
- However, any such decomposition requires that the information contained in the original relation be maintained i.e. any decomposition must be:
  1. loss less join decomposition.
  2. dependency preserving.
  3. without any repetition of information.

Oct.-11

4.4.1 Loss-Less Join Decomposition

- *Definition: Let R be relation schema and let F be a set of functional dependencies on R. Let R_1 and R_2 form a decomposition of R. This decomposition is a loss-less join decomposition of R if atleast one of the following functional dependencies are in F^+.*

$$R_1 \cap R_2 \rightarrow R_1$$

$$R_1 \cap R_2 \rightarrow R_2$$

- In other words, if a relation R with n attributes and K records or tuples is decomposed into m different sub-relations ($R_1, R_2, \ldots R_m$) then each subrelation will be having K different records.
- From this, it is clear that the cardinality of original relation and its decomposed relation is same.
- If all m subrelations are joined to form a single relation (R'), then its cardinality (number of tuples) must be same as the original relation.
- Similarly, whatever FDs are available in original relation, same FDs must be there in the resultant new relation (R').
- If the decomposition is not proper, then the cardinality of R and R' may not be same. Similarly, the FD sets of R and R' may not be same.

Example 15: Consider the following relation:

stud_advisor (name, dept., advisor) where the functional dependencies used are:

name → Dept.
name → advisor
Advisor → Dept.

Lets consider some records in this relation, this is shown in Fig. 4.10.

name	Dept.	Advisor
Rina	A	Omkar
Ravina	B	Ashmit
Mina	C	Umesh
Tina	A	Nikita

Fig. 4.10: Relation stud_advisor

In this relation is decomposed into two different relations.

R_1 (name, dept.)

R_2 (Dept., advisor) then in these relations total tuples available are same as the original relations shown in Fig. 4.11 and Fig. 4.12.

R_1

name	Dept.
Rina	A
Ravina	B
Mina	C
Tina	A

Fig. 4.11

R_2

Dept.	advisor
A	Omkar
B	Ashmit
C	Umesh
A	Nikita

Fig. 4.12

If we take natural join of R_1 and R_2 it will give us the resultant relation (R') as shown in Fig. 4.13.

name	dept.	advisor
Rina	A	Omkar
Rina	A	Nikita
Ravina	B	Ashmit
Mina	C	Umesh
Tina	A	Omkar
Tina	A	Nikita

Fig. 4.13: Result

Here, the resultant relation (R') cardinality is greater than original relation (R), hence this decomposition is not proper.

Hence, decomposition of a relation scheme R into the relation schemas R_i ($1 \le i \le n$) is said to be a loss-less join decomposition or simply loss-less if the natural join of all these relations gives the original relation R.

$$R = R_1 \; R_2 \; R_3 \; \text{---} \; R_n$$

Example of Loss-Less Join Decomposition:

- Let R (A, B, C) and F = {A → B}. Then the decomposition of R into R_1 (A, B) and R_2 (A, C) is loss-less because the FD {A → B} is contained in R_1 and the common attribute A is a key of R_1.

i.e. $R_1 \cap R_2 \rightarrow R_1$

4.4.2 Lossy Join Decomposition (Oct. 17)

- **Definition:** The decomposition of relation R into R_1 and R_2 is lossywhen the join of R_1 and R_2 does not yield the same relation as in R.

Example of Lossy Decomposition:

- Let R (A, B, C) and F = {A → B}. Then the decomposition of R into R_1 (A, B) and R_2 (B, C) is not loss-less because the common attribute B does not functionally determine either A or C i.e. it is not a key of R_1 and R_2.
- The above method is suitable only for the number of decomposition for a relation schema R is 2. But, if the number of decomposition is more than 2 then the following algorithm can be used.

Algorithm to check if a decomposition is loss-less:

Input: A relation schema R ($A_1, A_2, \ldots\ldots, A_k$):

- Decomposed into the relation schemas $R_1, R_2, R_3, \ldots\ldots R_n$

Output: Whether the decomposition is loss-less or lossy.

Steps: A table table-lossy (1 ..... n, 1 ..... k) is used to test for the type of decomposition.

Row i is the relation schema R_i of the decomposed relation and column j is for attribute A_j in the original relation.

for each decomposed relation R_i do

if an attribute A_j is included in R_i,

than Table-lossy (i, j) := αA_j (place a symbol αA_j in row i, column j of table)

else Table-lossy (i, j) := $\beta_i A_j$ (place a symbol $\beta_i A_j$ in table-lossy)

change := true

while (change) do

for each FD X → Y in F do

if rows i and column j exist such that the same symbol appears in each column corresponding to the attribute of X

then

if one of the symbol in the Y column is α_r then make the other α_r

else if the symbols use $\beta_{pm}\beta_{qm}$ then make both of them, say β_{pm}

else

change := false

i := 1

lossy := true

while (lossy and i ≤ n) do

for each row i of table-lossy

if all symbols are α_s

thenlossy := false

elsei := i + 1

- Finally, if lossy is false then given decomposition is loss-less else lossy.

Example 16: Given R (A, B, C, D) and the set of FDs.

F = {A → B, A → C, C → D} and the decomposition D {R_1 R_2}

R_1 = {A, B, C} R_2 = {C, D}

- To check whether D has a loss less join property:
- Create a table-lossy of size 2 × 4 and fill α's and β's,

	A	B	C	D
R_1	α_A	α_B	α_C	β_1 D
R_2	β_2A	β_2B	α_C	α_D

- For each FD X → Y in F do the following:
  - → Consider FD, A → B. In column A, as all values are not α, no action.
  - → Consider FD, A → C. In column A, as all values are not α, no action.
  - → Consider FD, C → D.
- In column C, all α values are present (same) and in column D, one value is α, hence, replace other values of column D by α_D we get,

	A	B	C	D
R_1	α_A	α_B	α_C	α_D
R_2	β_2A	β_2B	α_C	α_D

- As table-lossy is having 1st row will all α values, it indicates that given decomposition is loss-less.

4.4.3 Dependency Preserving Decomposition

- **Definition:** Given a relation schema R (S, F) where F is a set of all FDs present in R on the attributes in S, R is decomposed into the relations schemas $R_1, R_2 \ldots\ldots R_n$ with FDs, $F_1, F_2, \ldots\ldots F_n$.
- Then the decomposition is dependency preserving if the closure of F':

(where $F' = F_1 \cup F_2 \ldots\ldots \cup F_n$) is equal to F' i.e. $F'^+ = F^+$.

Example of Loss-Less and Dependency Preserving:

- Given R (A, B, C, D) with the FDs F = {A → B, A → C, C → D}, consider the decomposition R into R_1 = (A, B, C) with FDs, F_1 = {A → B, A → C} and R_2 = {C, with FD's F_2 = {C → D}.
- In this decomposition all the original FDs can be logically derived from F_1 and F_2,

i.e. $F' = F_1 \cup F_2$ = {A → B, A → C, C → D}.

- Hence, the decomposition is dependency preserving.
- Also, the common attribute 'C' forms a key of R_2. The decomposition of R into R_1 and R_2 is loss-less and dependency preserving.

Example of Lossy and Not Dependency Preserving:

- Given R = (A, B, C, D) with the FDs F = {A → B, A → C, A → D}, the decomposition of R into R_1 = (A, B, D) with the FDs, F_1 = {A → B, A → D} and R_2 = (B, C) with FDs F_2 = { } is lossy because the common attribute B is not a candidate key of either R_1 or R_2.

- In addition, the FD $A \rightarrow C$ is not implied by any FDs in R_1 and R_2. Thus, the decomposition is not dependency preserving and not loss-less.

Algorithm to check if a decomposition is dependency preserving:

Input: A relation scheme and a set F of FDs, A projection (R_1, R_2, R_3 ..... R_n) of R with the FDs (F_1, F_2, F_3 ..... F_n).

Output: Whether the decomposition is dependency preserving or not.

Steps: $F' := \phi$

For $i := 1$ to n do

$F' := F' \cup F_i$

For each FD $X \rightarrow Y \rightarrow F$ and while (flag) do

compute X^+, the closure of X under F'

if $Y \rightarrow X^+$, then flag := false.

Finally, if flag is false, then the decomposition is not FD preserving.

Example 17: Consider a relation R = (A, B, C, D) with the FDs, F = {A → B, A → C, C → D} and its decomposition into R_1 (A B C) with FDs.

F_1 = {A → B, A → C} and

R_2 = (C, D) with FDs

F_2 = {C → D}

Consider : $F' := \phi$

$F' := F_1 \cup F_2$

F' = {A → B, A → C, C → D}, flag := true.

Consider each FD one by one.

$A \rightarrow C$, A^+ :=ABCD

as $C \subseteq A^+$, no change in flag

$A \rightarrow B$, A^+ :=ABCD

as $B \subseteq A^+$, no change in flag

$C \rightarrow D$, C^+ :=CD

as $D \subseteq C^+$, no change in flag.

Finally, as flag is true, given decomposition is dependency preserving.

4.5 NORMAL FORMS (1NF, 2NF, 3NF AND BCNF) (Oct. 16)

- Normalization is simply the process of distilling the structure of database to the point where user removes repeated group of data into separate tables. In the normalization process, grouping of data can be done.
- The process of identifying and eliminating anomalies is called normalization.
- Using normalization, we start with a collection of tables, apply sets of rules to eliminate anomalies, and produce a new collection of problem-free tables. The sets of rules are called normal forms.
- Database operations is affect, when add, edit or delete records in the inadequate normalized table is called Database Anomalies.
- Anomaly problem can be categories into insert anomaly, update anomaly and delete anomaly.

1. **Insert Anomaly** is happen when add record(s) into inadequate quantity of attributes table structure. **For example:** A record is hold 15 information and the table is desinged with 13 attributes. When insert new record(s), the database operation is affect and occur error.
2. **Update Anomaly** is happen when update record(s) into inadequate normalized and redundant records. **For example:** An employee table structure does not have primary key to identity unique

record and duplicate employee information stored in the particular table. When, upda particular employee information, the database operation is affect and update wrong information

3. Delete Anomaly is happen when delete record(s) into inadequate normalized and redundar records. For example: an employee table structure does not have primary key and foreign key t identify unique record and duplicate employee information stored in the particular table. Whe delete particular employee information, the database operation is affect and delete wron information.

Goals of Normalization:

1. Eliminate redundant data for example; storing the same data in more than one table has to bi avoided.
2. Ensure data dependencies if present in a relation then it makes sense for example, only storing related data in a table.

- Normalization is a process of reducing redundancies of data in a database.

Definition of Normalization:

- Database normalization is, "the process of efficiently organizing data in a database". OR
- Database normalization is, "the process of organizing the fields and tables of relational database to minimize redundancy".
- The components of normalization are normal forms. Relations can fall into one or more categories (or classes) called Normal Forms.
- A database's level of normalization is determined by the normal form. Fig. 4.15 shows types of normal forms. (Oct. 16)

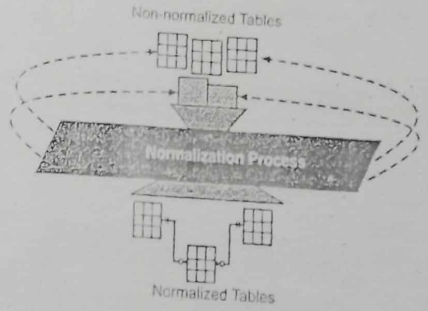


Fig. 4.14: A Graphic Representation of the General Normalization Process

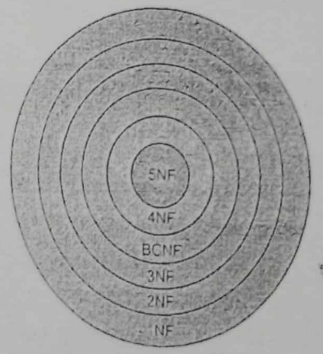


Fig. 4.15

4.5.1 First Normal Form (1NF)

- The 1NF states that every tuple must contain exactly one value for each attributes from the domain of that attribute. That is, multi-valued attributes, composite attributes and their cominations are not allowed in 1NF.
- **Definition:** A relation schema R is said to be in the 1NF if and only if the domains of all attributes of R contain atomic (or indivisible) values only.
- The steps for converting a database to 1NF are:

Step 1: Find and remove attributes that allow redundant and multiple values in a relation.

Step 2: Group the removed items in another table.

Step 3: Assign the new table a key which is the primary key of the old relation.

- **For example:** Inarelation DEPARTMENT D_Name, D_Mgrssn and D_Locations is fully functionally dependent on D_Number and is represented as: D_NUMBER {D_NAME, D_MGRSSN, D_LOACATIONS} as shown in Fig. 4.16 (a). It is possible that one particular Department may be located at multiple locations; this means D_Location is a multi-valued attribute.
- Therefore, the DEPARTMENT relation shown in Fig. 4.16 (b) is not in 1NF. In order to store atomic values (single value) in each cell, there is an attempt made in Fig. 4.16 (c) to store D_Locations at each individual tuple but this results in to repetition of information where department name, number and mgrssn has to be replicated.
- To bring the DEPARTMENT relation into 1NF, we need to remove the attribute that violates 1NF and place it in a separate relation along with the primary key of the relation.
- Here, the larger department relation is broken down into two separate relation one for where the violating attribute D_Locations is put in one relation along with Primary Key of the department relation and other relation contains all other attribute except the violating attribute.

DEPARTMENT

D_NAME	D-NUMBER	D_MGRSSN	D_LOCATIONS

(a)

DEPARTMENT

D_NAME	D-NUMBER	D_MGRSSN	D_LOCATIONS
Sales	6	34455	Satara, Kolhapur, Sangli
Administration	3	76543	Pune
Headquarters	1	86655	Mumbai

(b)

DEPARTMENT

D_NAME	D-NUMBER	D_MGRSSN	D_LOCATIONS
Sales	6	34455	Satara
Sales	6	34455	Kolhapur
Sales	6	34455	Sangli
Administration	3	76543	Pune
Headquarters	1	86655	Mumbai

(c)

Fig. 4.16: 1NF using DEPARTMENT Relation

- The solution shown in Fig. 4.17 is considered the best because it does not suffer from redundancy and it is completely general having no limit placed on a maximum number of values to be inserted as an individual record in a table.

DEPARTMENT

D_NAME	D-NUMBER	D_MGRSSN

PK (Primary Key)

DEPARTMENT_LOCATIONS

D-NUMBER	D_LOCATIONS

PK (Primary Key)

Fig. 4.17: Department Relation Normalization into 1NF

4.5.2 Second Normal Form (2NF)

April 17

- The 2NF is based on the concept of full functional dependency.
- **Definition:** A relational schema R is said to be in the 2NF, if every non-key attribute A in R is fully functionally dependent on the primary key. An attribute is non-key or non-prime attribute, if it is not a part of the candidate key of R.
- The steps for converting a database to 2NF are:

  Step 1: Find and remove attributes that are related to only a part of the key.

  Step 2: Group the removed items in another table.

  Step 3: Assign the new table a key that consists of that part of the old composite key.
- **For Example:** A relation EMP_PROJ{SSN, P-ID, HOURS, E_NAME, P_NAME, and P_LOCATION} with SSN and P_NUMBER as a composite primary key of the relation is in 1NF.
- Here, {SSN, P_ID → {HOURS} is an example of full functional dependencies. Since, neither SSN → HOURS nor P_ID → HOURS holds out alone i.e., removing any one key will not give any information about hours a particular employee has spent for a specified project.
- Also {P_ID} → {P_NAME, P_LOCATION} is an example of full functional dependencies, since it P_NAME and P_LOCATION is fully dependent on a single key. But here {SSN} → E_NAME is not a full FD (it is called a partial dependency) since, SSN → E_NAME only holds i.e. it would give information about a particular employee based on it SSN even if P_NUMBER information is not present. So the Relation EMP_PROJ is not in a 2NF.
- If the relation is not decomposed into smaller relation satisfying 2NF then this situation could lead to the following problems:
  1. **Deletion:** If an employee completes work on a project, the employee's record will be deleted. The information regarding the project the employee was belonging to will also be lost.
  2. **Insertion:** The record of employee cannot be entered until the employee is assigned a project.
  3. **Updating:** For a given employee, the employee code and different project number is repeated several times.

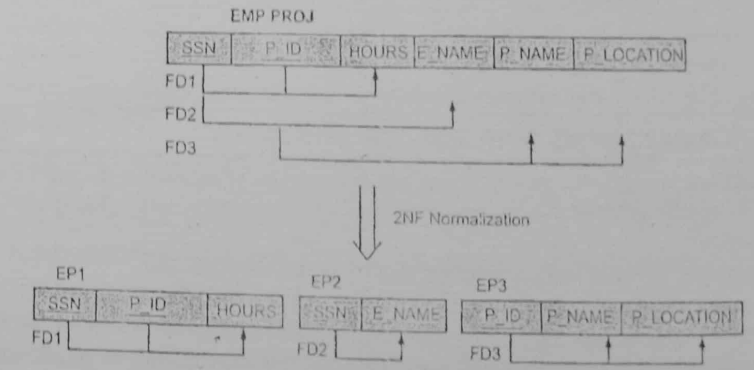


Fig. 4.18: 2NF using EMP_PROJ

- In order to bring the relation in 2NF it can be decomposed into relations; where attributes depending on only a part of a key is removed and assigned to a new relation with part of primary key on which it is depending.
  1. {SSN, P-ID) → {HOURS} (No partial dependency, as HOURS depend on both Primary keys), So create EP1 Relation = {SSN, ID, HOURS}

2. {SSN} → E_NAME (Partial functional dependencies, as ENAME depends on only part of key SSN). So remove the other part of the key on which right hand side attribute is not depending and create EP2 Relation = {SSN, E_NAME}
3. {P_ID} → {P_NAME, P_LOCATION} (No partial dependency, as P_NAME and P_LOCATION both depend on the Primary KEY P-ID). So Create EP3 Relation = {P_ID, P_NAME, P_LOCATION}

- Now, EP1, EP2 AND EP3 satisfy 2NF.

4.5.3 Third Normal Form (3NF)

April-15, 17, Oct.-16, 17

- The normal form that is based on the concept of transitive dependency is the 3NF.
- A relation is in 3NF when it is in 2NF (i.e. fully functionally dependent on every Primary Key of relation) and no non-key attribute in relation is transitively dependent on the primary key.
- **Definition:** A relation schema R with a set F of functional dependencies is in the 3NF, if, for every FD $X \rightarrow Y$ in F, where X is any subset of attributes of R and Y is any single attribute of R, at least one of the following statements hold:
  1. $X \rightarrow Y$ is a trivial FD i.e., $Y \subseteq X$.
  2. X is a super key for R.
  3. Y is contained in a candidate key for R.
- The steps for converting a database to 3NF are:

  Step 1: Find and remove attributes that are dependent on non-key attribute.

  Step 2: Group the removed attributes with non-key attributes on which it is depending in new table.

  Step 3: Make the non-key attribute as a Primary Key (PK) of a new table on which the removed attribute from the old table is depending.
- **For Example:** A relation EMP_DEPT {E_NAME, SSN, B_DATE, ADDRESS, D_ID, D_NAME and DMGR_SSN} with SSN as a primary key of the relation here {SSN} → {E_NAME, B_DATE, ADDRESS, D_ID} is an example of full functional dependencies. Since none of these attributes stand out alone i.e. changing the SSN will also likely to change these attribute values.
- Here,{D_ID} → {D_NAME, DMGR_SSN} is an example of transitive functional dependency, since D_ID is a non-key attribute on which D_NAME and DMGR_SSN is dependent, which in turn is dependent on the primary key SSN. So the Relation EMP_DEPT is not in a 3NF.
- If the relation is not decomposed into smaller relation satisfying 3NF, then this situation could lead to the following problems:
  1. **Deletion:** If a particular employee's record is deleted, the information regarding the Manager of the department would be lost.
  2. **Insertion:** The department Manager of a new department that does not have any employees as yet cannot be entered.
  3. **Updating:** For a given department, the particular Manager's SSN is prepared several times. Hence, if a department Manager moves to another department, the changes will have to be made consistently across the table.

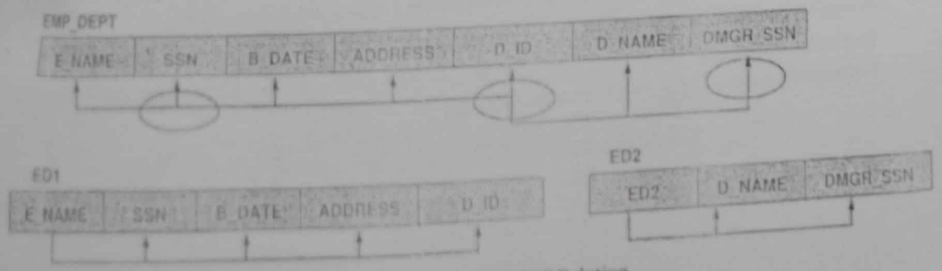


Fig. 4.19: 3NF using EMP_DEPT Relation

- In order to bring the relation in 3NF it can be decomposed into relations: where attributes depe on non-key attribute is removed and assigned to a new relation with non-key attribute on whic depending:
  1. {SSN} → {E_NAME, B_DATE, ADDRESS, D_ID} (No partial and transitive dependency, as all attributes are fully functionally dependent on the Primary Key SSN). So create ED1 Relati {E_NAME, SSN, B_DATE, ADDRESS, D_ID}
  2. {D_ID}→ {D_NAME, DMGR_SSN} (Transitive dependencies, as D_NAME and DMGR_SSN dep on non-key attribute D_ID). So remove D_NAME and DMGR_SSN with non-key attribute D make D_ID as primary key of this new relation. Create ED2 Relation = {D_ID, D_NA DMGR_SSN}
- Now, ED1 and ED2 satisfy 3NF.

4.5.4 Boyce-Codd Normal Form (BCNF)

April

- A relationship is said to be in BCNF if it is already in 3NF and the left hand side of every depende is a candidate key.
- The BCNF was proposed to deal with the relations having two or more candidate keys that composite and that overlap.
- **Definition:** A relation schema R is in a BCNF if for every Functional dependency $X \rightarrow A$ in F, where is the subset of the attributes of R, and A is an attributes of R, one of the following statements holds:
  1. $X \rightarrow Y$ is a trivial FD, i.e., $Y \subseteq X$.
  2. X is a super key.
- In simple terms, 3NF can be stated as, a relation schema R is in BCNF, if and only if every non-trivi FD has a candidate key as its determinant.
- The steps for converting a database to BCNF are:

  Step 1: Find and remove attributes that are dependent on an attribute which is not a candidate key.

  Step 2: Group the removed attributes with non-candidate key attribute on which it is depending i new table.
- **For Example:** In the STUD_DEPT relation (STUD_ID, DEPT_ID) is the candidate key of the relation There are three determinants (STUD_ID, DEPT_ID and MARKS) but a determinant mark is not candidate key. So the relation STUD-DEPT is not in BCNF.

STUD-DEPT

STUD_ID	STUD_NAME	MARKS	GRADE	DEPT_ID	DEPT_NAME
1	Mangesh	78	Distinction	1	Civil
2	Bhavesh	64	First Class	2	IT
3	Kiran	75	Distinction	3	Computer
4	Amar	50	Second Class	1	Civil

Fig. 4.20: BCNF using STUD_DEPT Relation

- To transform the relation to BCNF, we must remove the attributes that are violating functional dependency by creating new relations as follows:
  1. {STUD_ID} → {STUD_NAME, DEPT_ID} (As STUD_NAME and DEPT_ID are fully functionally dependent on STUD_ID, which is a candidte key as well). So Create SD1 Relation = {STUD_ID, STUD_NAME, DEPT_ID}
  2. {MARKS} → {GRADE} (As grade is dependent on MARKS and if marks changes grade will also change, but here MARKS is not a candidate key). As per the definition of BCNF every determinant has to be a candidate key, here in this example MARKS is not a candidate key. So create SD2 Relation = {STUD_ID, MARKS, GRADE}

3. (DEPT_ID) → (DEPT_NAME) (As DEPT_NAME is fully functionally dependent on only DEPT_ID, which is a possible candidate key of a relation). So create SD3 Relation = (DEPT_ID, DEPT_NAME)

- Now, SD1, SD2 and SD3 satisfy BCNF.

SD1

STUD_ID	STUD_NAME	DEPT_ID
1	Mangesh	1
2	Bhavesh	2
3	Kiran	3
4	Amar	1

SD2

STUD_ID	MARKS	GRADE
1	78	Distinction
2	64	First Class
3	75	Distinction
4	50	Second Class

SD3

DEPT_ID	DEPT_NAME
1	Civil
2	IT
3	Computer
1	Civil

Fig. 4.21: STUD_DEPT Relation Transformed into BCNF

Comparison between 3NF and BCNF:

Sr. No.	3NF	BCNF
1.	For a functional dependency X → Y, 3NF allows this dependency in a relation if X is not a candidate key and Y is either a primary-key attribute or a non-key attribute depending on X.	For a functional dependency X → Y, BCNF insists that for this dependency to remain in a relation, X must be a candidate key.
2.	3NF has some redundancy	BCNF does not have any redundancy.
3.	A relation R in 3NF when it is in 2NF and no non-key attribute in relation is transitively dependent on the primary key.	A realtion R is said to be in BCNF, if and only if every determinant is a candidate key.
4.	3NF is dependency preserving.	BCNF is not dependency preserving.

4.5.5 Case Studies/Examples (Oct. 17, April 18)

Case Study 1: Normalize the following table upto 3 NF step by step. The database is related to order-report shown in Fig. 4.22.

Order no.	Order date	Invoice no.	Invoice date	Product no.	Product name	Quantity ordered
25	17/3/13	1231	30/4/13	1	Computer	5
130	17/3/13	1232	30/4/13	2	Printer	5
520	28/3/13	1233	30/4/13	2	Printer	1
				4	A4 pages	10
				5	Main sheets	70
				6	Mouse	10

Fig. 4.22: Consolidated order report

Solution:

1NF:

The given date is in unnormalized form. First convert it into 1 NF. In 1 NF no repeating groups.

For this, partition each data structure containing repeating groups which accomplishes the same purpose. Fig. 4.23 is the result of 1NF.

Order no.	Order date	Invoice no.	Invoice date	Product no.	Product name	Quantity ordered
25	17/3/13	1231	30/4/13	1	Computer	5
130	17/3/13	1232	30/4/13	2	Printer	5
520	28/3/13	1233	30/4/13	2	Printer	1
520	28/3/13	1233	30/4/13	4	A4 pages	10
520	28/3/13	1233	30/4/13	5	Main sheets	70
520	38/3/13	1233	30/4/13	6	Mouse	10

Fig. 4.23

- We remove the four repeating elements and group them into a new table called order-record. [Refer Fig. 4.24 (a)].
- Rest of the elements are added into a new table called order-item. (Refer Fig. 4.24 (b)).

order no.	order date	invoice no.	invoice date
25	13/3/13	1231	30/4/13
130	17/3/13	1232	30/4/13
520	28/3/13	1233	30/4/13

Fig. 4.24: (a) Order-record

order no.	product no.	product name	quantity
25	1	computer	5
130	2	printer	5
520	2	printer	1
	4	A4 pages	10
	5	mainsheets	70
	6	mouse	10

Fig. 4.24: (b) Order-item

Both the relations order-record and order-item are in 1NF. But order-item is not in its ideal form. Some of the attributes are not functionally dependent on primary key [order no, product no]. Some nonkey attributes such as product name, quantity are dependent only on product_no and not on the key which we have considered.

2NF:

All the nonkey fields are depend on the primary key (composite key). For this, verify that each non-key field is in first normal form and fully functionally dependent on primary key.

Product no → product description is the functional dependency.

To convert order-item into 2 NF, we separate the table (decompose) such that the above functional dependency satisfies correctly. [Refer Fig. 4.25 (a), 4.25 (b), 4.25 (c)]

order no.	order date	invoice no.	invoice date
25	13/3/13	1231	30/4/13
130	17/3/13	1232	30/4/13
520	28/3/13	1233	30/4/13

Fig. 4.25 (a): Order-record

order no	product no	quantity ordered
24	1	5
130	2	5
520	2	1
	4	10
	5	70
	6	10

Fig. 4.25 (b): Order-item

product no	product description
1	computer
2	printer
4	A4 pages
5	mainsheets
6	mouse

Fig. 4.25: (c) Product

3NF:

When all non-key fields are independent of one another, then relation is in 3NF. There is check for redundancy within the relation.

Duplicate data elements which can be derived from other elements are removed at 3NF. For example, in order-record relation we have invoice date which is dependent on both i.e. order-no and invoice-no.

To convert the ORDER-RECORD into 3NF we have to remove invoice-date and group it with invoice-no as the key in separate table.

So we have following 3 NF relations, [See Fig. 4.26 (a), (b), (c), (d)].

order no	order date	invoice no
25	13/3/13	1231
130	17/3/13	1232
520	28/3/13	1233

Fig. 4.26 (a): Order-record

order no	product no	quantity ordered
25	1	5
130	2	5
520	2	1
520	4	10
520	5	70
520	6	10

Fig. 7.26 (b): Order item

product no	product description
1	computer
2	printer
4	A4 pages
5	mainsheets
6	mouse

Fig. 4.26 (c): Product

invoice no	invoice-date
1231	30/4/13
1232	30/4/13
1233	30/4/13

Fig. 4.26 (d): Invoice

Therefore the relationship is transformed into following relations (3NF):

Order-record – order-no – primary key
order-date
invoice-no – foreign key

Product – product no – primary key
product description

Order-item – order no – primary key
product no – foreign key
quantity-ordere

Invoice – invoice no – primary key
invoice-date

Case Study 2: The given sales-report in un-normalized relation since it has repeating groups. Normalize this relation upto 3NF. The sales report is shown in Fig. 4.27.

sales person no	sales person name	sales area	cust-no	cust-name	warehouse no	warehouse location	sales amount
501	Vinit	Aundh	101	CMS	4	Mumbai	2 lakh
			102	IBM	3	Thane	2.5 lakh
			301	Lavasa	4	Mumbai	4 lakh
345	Pritesh	Warje	45	Infosys	2	Chennai	55 Thousand
			127	Microland	1	Bangalore	1.2 lakh

Fig. 4.27: Sales report

Solution: The normalized sales report is as shown in Fig. 4.28.

sales person no	sales person name	sales area	cust-no	cust-name	warehouse no	w location	sales amount
501	Vinit	Aundh	101	CMS	4	Mumbai	2 lakh
501	Vinit	Aundh	102	IBM	3	Thane	2.5 lakh
501	Vinit	Aundh	301	Lavasa	4	Mumbai	4 lakh
345	Pritesh	Warje	45	Infosys	2	Chennai	55 thousands
345	Pritesh	Warje	127	Microland	1	Banglore	1.2 lakh

Fig. 4.28: Normalized sales-report

1NF:

Create two separate structures sales person (Refer Fig. 4.29) and sales-customer (Refer Fig. 4.30).

Sales person no	Sales person name	Sales are
501	Vinit	Aundh
345	Pritesh	Warje

Fig. 4.29: Sales-person

Sales person no	cust-no	cust-name	warehouse no	w location	Sales amount
501	101	CMS	4	Mumabi	2 lakh
501	102	IBM	3	Thane	2.5 lakh
501	301	Lavasa	4	Mumabi	4 lakh
345	45	Infosys	2	Chennai	55 thousand
345	127	Microland	1	Banglore	1.2 lakh

Fig. 4.30: Sales-customer

In sales-customer, some of the attributes are not functionally dependent on primary key – sales person no, cust-no.

2NF:

Sales-person table is as it is, [See Fig. 4.31(a)].

Sales-customer will be decompose into two more relations i.e. sales [Refer Fig. 4.31 (b)] and warehouse [Refer Fig. 4.31 (c)].

Sales person no	Sales person name	Sales area
501	Vinit	Aundh
345	Pritesh	Warje

Fig. 4.31 (a): Salesperson

Sales person no	cust-no	Sales Amount
501	101	2 lakh
501	102	2.5 lakh
501	301	4 lakh
345	45	55 thousand
345	127	1.2 lakh

Fig. 4.31 (b): Sales

cust-no	cust-name	warehouse no	warehouse location
101	CMS	4	Mumbai
102	IBM	3	Thane
301	Lavasa	4	Mumbai
45	Infosys	2	Chennai
127	Microland	1	Banglore

Fig. 4.31 (c): Warehouse

Warehouse location (warehouse location) is not fully dependant on cust-no. i.e. it has non-key dependencies.

We will go for 3NF.

3NF:

Sales person and sales relations are as it is. [Refer Fig. 4.32 (a), Fig. 4.32 (b)].

Warehouse is decomposed into customer [Refer Fig. 4.32 (c)] and warehouse [Fig. 4.32 (d)].

Sales person no	Sales person name	Sales area
501	Vinit	Aundh
345	Pritesh	Warje

Fig. 4.32 (a): Sales person

Sales person no	cust-no	Sales amount
501	101	2 lakh
501	101	2.5 lakh
501	301	4 lakh
345	45	55 thousand
345	127	1.5 lakh

Fig. 4.32 (b): Sales

cust-no	cust name	warehouse no
25	CMS	4
102	IBM	3
301	Lavasa	4
45	Infosys	2
127	Microland	

Fig. 4.32 (c): Customer

warehouse no	warehouse location
4	Mumabi
3	Thane
2	Chennai
1	Banglore

Fig 4.32 (d): Warehouse

Therefore, the sales report relations is transformed into following 3NF relations:

Sales person	– Sales person no	–	primary key
	Sales person name		
	Sales area		
Sales	– cust-no	–	foreign key
	sales amount		
	sales person no	–	foreign key
Customer	– cust no	–	primary key
	cust name		
	warehouse no	–	foreign key
Warehouse	– warehouse-no	–	primary key
	warehouse location		

Case Study 3: Normalize the following relation:

Land system

Soil type

Erodibility (decaying of plant)

Solution: Erodibility is uniquely determined by soil-type.

Let consider some record into unnormalized relations shown in Fig. 4.33.

Land system	Soil type	Erodibility
Faraway	Loamy sand	0.10
Limestone	Sandy loam	0.25
Mundooka	Loamy sand	0.10

Fig. 4.33: Unnormalized Relation

1NF: Separate repeating groups

Soil table

land system	primary key
soil type	

Erosion

soil type	primary key
erodability	

This is normalized relation so need to go till 3 NF because at the start only it is normalized.

Case Study 4: Normalize the following data:

project_no
project_name
emp_no
emp_name
rate_category
hourly_rate

Solution: Above is the unnormalized data.

1NF: Separate repeating groups:

Employee-project Table

project_no	primary key
project_name	
emp_no	primary key
emp_name	
rate category	
hourly rate	

2NF: Find the elements depending upon primary keys or partial keys:

Employee Project Table

project_no	primary key
emp_no	primary key

Employee table

emp_no	primary key
emp_name	
rate category	
hourly rate	

Project table

project_no	primary key
project_name	

3NF: Find non-key attributes. Emp_name is not dependent on either rate category or hourly rate. Same is applies to rate category. But, hourly rate is dependent on rate category.

Employee project table

project_no	primary key
emp_no	primary key

Employee table

emp_no	primary key
emp_name	
rate category	

Rate table

rate category	primary key
hourly rate	

Project table

project_no	primary key
project_name	

All above table are in 3 NF and ready to implement.

Case Study 5: Normalize the following table of attributes upto 3NF. The database is related to cric association.

The fields are:

Team_code	skill_name	club_city
player_code	club_code	club_district
team_name	umpire_code	
player_name	club_name	
player_age	umpire_name	
player_skill_code	club_address	

Following assumptions to be made –

1. A player can belong to one team and has only one skill.
2. An umpire can belong to one club only.
3. Club arranges matches between various teams and appoint umpires for the matches.

Solution: Unnormalized data:

club_code		team_code
club_name	team_name	
club_address	player_code	
club_district	player_name	
club_city		player_age
umpire_code	player_skill_code	
umpire_name	skill_name	

1NF: Separate out the repeating group and find out key field.

Club table

club_code	primary key
team_code	secondary key
team_name	
player_code	
player_name	
player_age	
player_skill_code	
skill_name	

Umpire table

club_code	primary key
club_name	
club_address	
club_district	
club_city	
umpire_code	
umpire_name	

2NF: Find out the attributes depending upon the whole key and also the attributes depending upon the partial key.

CT table

club_code	primary key
team_code	secondary (partial key)

Team table

team_code	primary key
team_name	

Player table

team_code	primary key
player_code	
player_name	
player_age	
player_skill_code	
skill_name	

Club-table

club_code	primary key
club_name	
club_age	
club_district	
club_city	

Umpire table

club_code	primary key
umpire_code	
umpire_name	

3NF: Find out the non-key elements depending upon some other non-key element. And extract the elements which can be directly computed.

From the player structure player age, player_skill_code, skill_name can be separated out as they are non_key elements depends upon the non_key elements namely player_name.

Similarly, club_name is a non_key element.

∴ **table-1**

player_name	primary key
player_age	
player_skill_code	
skill_name	

and table-2

club_name	primary key
club_age	
club_district	
club_city	

The key elements from the new structures are found out. In the player structure player_name is set out as a key field. And in the club structure club_name is set out as a key field.

Case Study 6: Normalize the following data to 3NF using appropriate relation.

Invoice

cust_code :	Mina	Inv. no :	011	Inv. date	: 05/03/13
cust_name :	ABCL Ltd.	Po. no :	005	Po date	: 05/03/13
cust_address :	Camp	Ch. no :	002	Ch_date	: 05/03/13

Item no	Desc	Quantity	Rate	Amount
A005				
			[Discount]	—
			Inv.value	—

Solution: Unnormalized data:

Inv_no	cust_code
Inv_date	cust_name
Inv_value	cust_address
Item_no	po_no (purchase order no.)
Desc	po_date
Rate	Ch_no (Challan no.)
Qty	Ch_date

1NF: Repeating group separated.

Invoice table

Inv_no	primary key
Item_no	
Desc	
Qty	
Rate	
Amt	

Customer table

Inv_no	primary key
Inv_date	
cust_code	
cust_name	
cust_address	
po_no	
po_date	
Ch_no	
Ch_date	
Inv_value	

2NF: Find out elements depending upon the main key and partial key.

Item-1 table

Inv_no	primary key
Item_no	
Qty	

Item-2 table

Item_no	primary key
Desc	

Rate
Amt

Invoice table
Inv_no primary key
Inv_date
Inv_value

C-1 table
inv_no primary key
cust_code

C-2 table
cust_code primary key
cust_name
cust_address

C-3 table
Inv_no primary key
Po_no

C-4 table
Po_no primary key
Po_date

C-5 table
Inv_no primary key
Ch_no

C-6 table
Ch_no primary key
Ch_date

2NF: We have to separate out the non-key elements depending upon some other non-key element.
In above example, only the cust-address is a non-key element which is depending upon other non-key element i.e. cust_name.

C-7 table
cust_name primary key
cust_address

Case Study 7: Normalize the following data to 3NF using appropriate relation. Give brief explanation for each level.

Kaviman Ltd.

cust_no	:	001	order no	:	1250
cust_name		Shekhar	order date	:	5/3/13
address	:	camp	delivery date	:	10/3/13

item_no	description	rate	quantity
001	chair	65	100
014	table	1200	3
015	Washing machine	14000	2

Remark : Delivery at Pune H.O.

Solution:

Step 0: Firstly, we have to write down the data in the unnormalized form. For this, the data eleme related to each other are to be listed out one after another.

Unnormalized data:

cust_no
cust_name
address
order_no
order_date
delivery_date
item no
desc
rate
quantity

Step 1:

To achieve 1NF, remove all the repeating groups.
A repeating group is actually another relation.
Hence, it is removed from the record and treated as an additional record structure or relation.
In this example item_no, description, rate and quantity, etc. repeats with each customer. Hence, we repeat it.
The main key from the original structure is to be brought into repeating group structure.
The new structure is named with appropriate name.

Item table

cust_no	primary key
item_no	partial key (secondary)
desc	
rate quantity	

Customer table

cust_no	primary key
cust_name	
address	
order_no	
order_date	
delivery date	

Step 2:

As we know the above relation is in 1NF so we will go for 2NF we are looking for functional dependencies.
For example, description of item is dependent or item_no, rate is dependent on item_no.
Similarly, cust_name is dependent on cust_no., cust_address is dependent on cust_name and so on. We have to group such functionally dependent attributes and achieve 2NF.

So 2NF will have the following tables:

item-1 table

cust_no	primary key
item_no	primary key
quantity	

item-2 table

item_no	primary key
desc	
rate	

cust-1 table

cust_no	primary key
cust_name	
address	

cust-2 table

cust_no	primary key
order no	

cust-3 table

order_no	primary key
order_date	
delivery_date	

Step 3: In 3 NF we have to find out the non-key elements depending upon non-key elements. In the above structure address is a non-key element depends upon the cust_name which is also a non-key element. So, it can be defined as a new data structure and in which cust_name can be defined as a key field.

4.6 KEYS CONCEPT WITH EXAMPLES (SUPER KEY, CANDIDATE KEY AND COMPOSITE KEY)

4.6.1 Super Key

Oct.-16

- A super key is a set of one or more attributes which taken collectively, allow us to identify uniquely an entity in the entity set.
- Consider example of employee table. Employee table contains two attributes or two columns - Empno and Dept_id. Emp_name can be repeated and not unique.

  Empno, Dept_id is a super key.

  Empno is a key and Dept_id is a key.
- Here, two attributes Empno and Dept_id together makes a super key. Empno is a key because it contains minimal set of attributes. This concept is not sufficient.

4.6.2 Candidate Key

April-17

- A table can have more than one set of columns that could be chosen as the key called candidate key.
- In other words, a set of attributes that uniquely identifies a tuple according to a key which is minimal set of the columns in the table.

- Candidate key is defined by two parts :
  1. Two distinct tuples (records) cannot have same value in all the fields of a key.
  2. No subset of the set of fields (attributes) in a key is a unique identifier or a tuple.
- First part of definition means the attributes are uniquely identified.
- For example: If {studid, studname} is a candidate key then the DBMS will not allow the students relation to contain two tuples describing different students with the same name.
- The second part of definition means, e.g. the set of attributes {studid, studname} is not a key for students because the set properly contains the key studid.
- Since, studid alone can be super key, so we cannot include in the candidate key.
- By definition, no proper subset is a super key, such minimal super keys are called candidate keys. Therefore, {studid, studname} is not a candidate key since attribute studid alone is a candidate key.
- Another example, suppose we consider two attributes from customer table which are custname and street.
- These attributes are sufficient to distinguish among members of the customer entity set. So {custname and street} is a candidate key.
- Similarly, the attribute social-security is candidate key.
- Although the attributes custname and social-security together can distinguish customer entities, their combination does not form a candidate key, since the attribute social-security alone is a candidate key.
- Another example of salesman table containing following columns or attributes Sno, Sname, region, commission.

  Sno → Candidate key.

  Sname → Cannot be a candidate key.
- If we add attribute passportno in this table, then we can identify person uniquely by passport number.

  Passportno → Candidate key.

4.6.3 Composite Key

- There are situations when one attribute cannot form a key, because single attribute cannot uniquely identify every tuple or row in the table. So we need two or more attributes to identify tuple in the table.
- The key consists of two or more columns is called as composite key.
- For example: Consider a table of supplier which tells us the parts which supplier sells. Table is shown below:

Supid	Partid	Quantity	Price
S1	P1	12	300
S1	P2	10	100
S2	P2	5	50
S3	P3	7	200

- Here, neither the supid nor the partid can identify a row in the table uniquely.
- However, two of them together can easily identify a tuple or row in the table uniquely.
- Hence, composite key is (supid, partid).

4.7 ALGORITHM TO FIND CANDIDATE KEY / PRIMARY KEY FOR A RELATION

- First we will see the algorithm to find candidate keys from functional dependencies. The input is functional dependencies.

Algorithm:

1. Find the attributes that are neither on the left and right side
2. Find attributes that are only on the right side
3. Find attributes that are only on the left side
4. Combine the attributes on step 1 and 3
5. Test if the closures of attributes on step 4 constitutes all the attributes. If yes it is a candidate key.
6. If not, find the relation exteriors, that is the attributes not included in step 4 and step 2.
7. Now test the closures of attributes on step 4 + one attribute in step 6 one at a time. All those combinations are candidate keys if their closures constitute all the attributes.

Example: Consider the following relation and functional dependencies as,

R = (ABCDE)

A → C

E → D

B → C

Find out all candidate keys.

Steps:

1. Find the attributes that are neither on the left and right side → (none).
2. Find attributes that are only on the right side → C, D
3. Find attributes that are only on the left side → A, B, E
4. Combine the attributes on step 1 and 3 → since step 1 has no attributes, it's just A, B, E.
5. Test if the closures of attributes of ABE, it contains all the attributes of relation R.

So we have only one candidate key, that is ABE.

Solved Examples:

Example 1: Relation treatment is given as follows:

treatment (doctor_name, doct_no, pat_no, pat_name, salary) and the functional dependencies are given as:

F = {
doct_no→doctor_name
doct_no→pat_name
doct_name→ salary
}

Find the key ?

Solution: Let us find the closure.

(doct_no)⁺.

as doct_no → doctor_name and
doct_no → pat_name

(doct_no)⁺ = {
doct_no, doct_name, pat_name
}

doct_name→ salary

∴ (doct_no)⁺ = {
doct no, doct name, pat_name, salary
}

It has not given the whole relation. So doct_no cannot be a key.

Now find closure of (doct_no, pat_no)⁺.

result = doct_no, pat_no
doct_no→doctor_name and
doct_no→pat_name

∴ result = doct_no, pat_no, doctor-name, pat_name

See the functional dependency

doct_name → salary

∴ result = doct_no, doct_name, pat_no, pat_name, salary

∴ (doct_no, pat_no) can work as key.

Example 2: The relation film is given as follows:

film (movie_no, name, release_year, actor_id, actor_name)

and the functional dependencies are given as

F = {
movie_no→release_year,
actor_id→actor_name, movie_no,
name →release_year
}

Find the key ?

Solution: Find the closure of (movie_no) i.e. (movie_no)⁺

result = movie_no

but movie no → release year

result = movie_no, release year

∴ (movie_no)⁺ = {movie_no, release year}

Only movie_no cannot work as key.

Now find the closure of movie_no, actor_id i.e. (movie_no, actor_id)⁺.

result = movie_no, actor_id

but movie_no → release year

and actor_id → actor_name, movie_no

result = movieno, release year, actor_id, actor_name

Now find closure of movie_no, name i.e. (movie_no, name)⁺.

∴ result = movie_no, name

but movie no → release year

name → release year

∴ result = movie_no, name, release year

(movie_no, name)⁺and (movie_no)⁺ are not providing all the attributes of a relation. But (movie_no, actor_id)⁺ provide all the attributes. So it can be used as key.

Example 3: The relation R is given as follows:

R = (name, price, category)

and the functional dependencies are given as

F = {name → price, name → category}

and key is {name). Check after decomposition R is dependency is preserving

Solution: Relation R is decomposed as

R_1 = (name, price)

R_2 = (name, category)

If we place the records and perform Cartesian product, join operation we get lossless join decomposition and dependency is preserved.

Example 4: The relation R is given as follows:

R = (name, price, category)

and the functional dependencies are given as:

F = {name → price, price → category)

and key is {name). Check after decomposition R is dependency is preserving

Solution: Relation R is decomposed as

R_1 = (name, price)

R_2 = (price, category)

- To understand it better, let's solve it by placing some values.

R_1

name	price
pen	10
duster	25

R_2

price	category
5	write
25	rub

- Find Cartesian product (⋈).

name	price	price	category
pen	10	5	write
pen	25	25	rub
duster	10	5	write
duster	25	25	rub

- Find common price and find natural join (⋈).

name	price	category
pen	25	rub
duster	25	rub

- But pen's price is 10 in relation R_1.
- That is why it is lossy decomposition and dependency is also not preserved.

Example 5: Find the Candidate Key

R(ABCDEFG) and the functional dependencies are as,

AB → F

AD → E

F → G

Solution:

Steps:

1. Find the attributes that are neither on the left and right side → C.
2. Find attributes that are only on the right side → EG.
3. Find attributes that are only on the left side → ABD.
4. Combine the attributes on step 1 and 3 → since step 1 has no attributes, it's just ABCD.
5. The closures of ABCD is ABCDEFG, so ABCD is our candidate key this time.

Example 6: Find the candidate key.

R (ABCD) and the functional dependencies are as,

ABC → D

D → A

Solution:

Steps:

1. Find the attributes that are neither on the left and right side → None.
2. Find attributes that are only on the right side → None.
3. Find attributes that are only on the left side → BC.
4. Combine the attributes on step 1 and 3 → BC.
5. The closure of BC is only BC, we should find the relation exterior.
6. Find the relation exteriors, that is the attributes not included in step 4 and step 2. → in this example it is AD.
7. Now test the closures of attributes on step 4 + one attribute in step 6 one at a time. → ABC closures are ABCD, so it is a candidate key → BCD closures are ABCD, so it is also a candidate key> so in this case we have two candidate keys, they are ABC and BCD.

PRACTICE QUESTIONS

Q. I: Multiple Choice Questions:

1. Which design organizes data in tables or relations?
   (a) Relational database (b) Object database
   (c) Network database (d) All of these
2. Which sign is used to represent a functional dependency?
   (a) + (b) →
   (c) * (d) All of these
3. Types of Functional Dependency includes _____.
   (a) Multi-valued (b) Non-trivial
   (c) Transitive Functional (d) All of these
4. X → Y is trivial if?
   (a) X ⊂ Y (b) Y ⊂ X
   (c) X ⊇ Y (d) None of the mentioned
5. If F is a set of functional dependencies, then the closure of F is denoted by?
   (a) F^* (b) F^0
   (c) F^+ (d) F
6. If the decomposition is able to represnt all the facts about the relation then such a decompositon is called as?
   (a) Lossless decomposition (b) Lossy decomposition
   (c) Insecure decomposition (d) Secure decomposition
7. Database Anomalies includes _____.
   (a) Insert (b) Update
   (c) Delete (d) All of these
8. Goals of normalization includes _____.
   (a) Ensure data dependencies (b) Eliminate redundant data
   (c) both (a) and (b) (d) None of these
9. We say that a decomposition having the property $F^+ = F^+$ is a _____ decomposition.
   (a) Dependency losing (b) Dependency preserving
   (c) Lossless (d) None of the mentioned

10. If the decomposition is unable to represent certain important facts about the relation, then such a decomposition is called as?
(a) Lossless decomposition (b) Lossy decomposition
(c) Innocent decomposition (d) Secure decomposition
11. Which of the following is not a condition for X → Y in Boyce Codd Normal Form?
(a) X → Y is trivial
(b) X is the superkey for the relational schema R
(c) Y is the superkey for the relational schema R
(d) All of the mentioned
12. F⁺ is called as the ___ of F.
(a) Closure (b) Sum
(c) Cartesian product (d) None of the mentioned
13. In the _____ normal form, a composite attribute is converted to individual attributes.
(a) First (b) Second
(c) Third (d) Fourth
14. Which forms simplifies and ensures that there are minimal data aggregates and repetitive groups:
a) 1NF (b) 2NF
(c) 3NF (d) All of the mentioned

Answers

1. (a) 2. (b) 3. (d) 4. (a) 5. (c) 6. (a) 7. (d) 8. (c) 9. (b) 10. (b)
11. (c) 12. (a) 13. (a) 14. (c)

Q.II: Fill in the Blanks:

1. _____ is a set of constraints between two attributes in a relation from a database.
2. _____ are a set of rules, that when applied repeatedly, generates a closure of functional dependencies.
3. Data dependencies are similar to _____.
4. Repetition of data again and again in any relations is _____.
5. Codd classified and defined the anomalies of normal form in the year _____.
6. Breaking up the relation schema into smaller relation schema is called _____.
7. A functional dependency is an association between the _____ of a relation.
8. In decomposition, if information is loosed, it is called _____ decomposition.
9. Closure is important to find _____ in any database or relation.
10. _____ is a property that is desired while decomposition, i.e., no FD of the original relation is lost.
11. Database operations is affect, when add, edit or delete records in the inadequate normalized table is called Database_____.
12. The components of normalization are normal forms. Relations can fall into one or more categories (or classes) called _____.
13. The _____ is a set of all functional and multi values dependencies implied by a set of functional dependencies.

Answers

1. Functional Dependency (FD) 2. Armstrong's Axioms 3. constraints
4. redundancy 5. 1972 6. decomposition
7. column 8. lossy 9. key
10. Dependency preservation 11. anomalies 12. Normal Forms
13. Closure

Q. III: State True or False:

1. FD is particular type of constraint.
2. A functional dependency is an generalization between the columns of a relation.
3. Functional dependencies are transitive.
4. Decomposition means breaking up a relation schema into smaller relation schemas.
5. The process of identifying and eliminating anomalies is called normalization.
6. A database's level of normalization is determined by the normal form.
7. The loss-less joint also known as non-additive join decomposition.
8. A function that has no partial functional dependencies is in 3NF form.
9. A relation is in first normal form if every attribute in that relation is singled valued attribute.
10. A relation is in third normal form, if there is transitive dependency for prime attributes is it is in second normal form.

1. (T) 2. (F) 3. (T) 4. (T) 5. (T) 6. (T) 7. (T) 8. (F) 9. (T) 10. (F)

Q. IV: Answer the following Questions:

(A) Short Answer Questions:

1. What is meant by relational database design?
2. What is functional dependency?
3. Enlist types of functional dependencies.
4. What is meant by normalization?
5. Define decomposition.
6. What is key?
7. What is normal form?
8. Enlist types of normal forms.
9. Define primary key.
10. Explain lossless join decomposition.

(B) Long Answer Questions:

1. Define functional dependency with the help of example.
2. What are the types functional dependencies? Explain with example.
3. State and prove Armstrong's axioms for functional dependency.
4. Define normalization. Specify needs of normalization.
5. What is normalization ?
6. What are the desirable properties of decomposition ?
7. What are the pit falls in relational database design ? Explain with example.
8. Explain different anomalies related with normalization.
9. Explain loss-less join decomposition with example.
10. With the help of example describe decomposition.
11. Write short notes on :
(i) 1NF
(ii) 2 NF
(iii) 3 NF
(iv) BCNF
12. Normalize the following relation to 3NF.
(i) Ename Address
SSN D NUM (dept_num)
BDate D Name (dept_name)

(ii) plant_name cust_code
plant_code no_plant
plant_type cust_address
plant_price
cust_name

(iii) doctor_code hospital_no
doctor_name hospital_name
patient_name
patient_code
ward_no

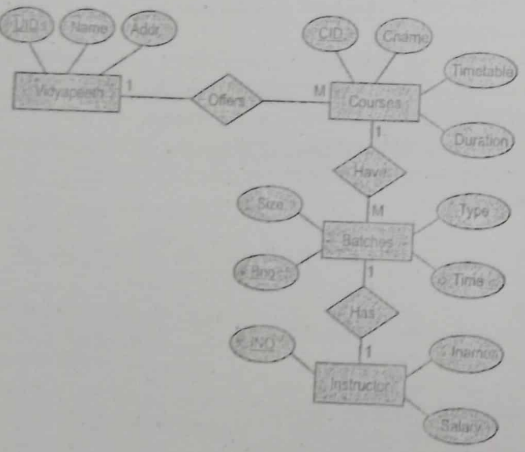
13. Consider a relational database customer:
customer (cno, cname, city, branch, br_no, loan_no, laon_amt, assets)
Decompose this relation into subrelations.

14. Consider a relational database student and normalize it to 3 NF.
student (rollno, name, address, phone_no, class_code, class_name, subject_code, subject_name)

15. Consider the relation R = {A, B, C, D, G, H, I} and set of functional dependencies defined on R, F as {A → B, A → C, CG → H, CG → I, B → H}
Compute closure of F i.e. F⁺ and Compute closure of Attribute Set (AG)⁺.

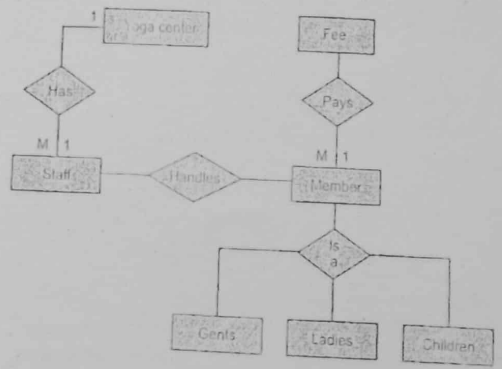
16. ShramikVidhyapit offers 10 courses on 'Information Technology'. Each course is conducted in several batches, where the maximum batch size is of 50 students. For every batch separate instructor is appointed. When for a particular course minimum 25 students get admitted, then the timetable is prepared and separate classrooms and also instructors availability, daily one hour is allocated to each batch. As numbers of students is growing, it is difficult to the management to schedule the course.
Suggest a suitable information system to handle the above problem :
(i) Draw entity-relation diagram for the information system design.
(ii) Convert entity-relationship diagram into relational database in 3NF.

Ans. E-R diagram:



3NF for database:
Vidyapeeth (UID, name, add)
Courses (CID, Cname, timetable, duration, UID)
Batches (Bno, size, type, time, CID)
Instructor (INO, Iname, salary)

17. Shubhangi Yoga Exercise Center having two receptionists who take care that every person who comes in exercise center giving the fees or not ? There are many gents, ladies and children who come to do exercise. The center provides gym, aerobics, swimming, tennis, chess and yoga.
Each section has different well trained coach.
There is Management Committee to manage all the outside things like marketing, government permission, taxes etc.
Suggest a suitable information system to handle the above tasks :
(i)Draw Entity_Relation Diagram for the Information System Design.
(ii) Convert Entity_Relation Diagram Relational Database in 3 NF.
Ans. E-R diagram:



Database :
Staff (staff_id, staff_name, type_id, designation)
Yogacenter (type_id, type of exercise).
Member (member_id, member_name, gender, staff_id).

UNIVERSITY QUESTIONS AND ANSWERS

April 2015

1. What are the desirable properties of decomposition? Explain in brief. **(5-M)**
Ans. Refer to Section 4.4.

2. What are the anomalies that might arise if we have redundant data?
Ans. Refer to Section 4.1.1.

3. Consider the relation R(B, C, D, E, F, G) and the set of FD's,
F = {B → C, DE → G, B → D, DE → F, C → F}
Compute (BE)+. 5 M
Ans. Refer to Section 4.2.

4. Consider the following relations :
Machine (m_no, m_name, m_type, M_cost)

Part (P_no, P-name, P-description)

Machine and part are related with many to many relationship.

Create a relational database in 3NF and solve the following queries in SQL.

(i) List all the machines having the part 'register'.

(ii) Find the machines having more than 5 parts.

(iii) Increase the cost of all machines by 10%. (5 M)

Ans. Refer to Section 4.5.3.

5. Consider the following relations:
Person (P_no , name, address)

Car (C_no , year, model)

Person and car are related with one to many relationship.

Create a relational database in 3NF and solve the following queries in SQL :

(i) List the names of all people who own a 'Indica'.

(ii) Delete all the details of the person 'Mr. Joshi'.

(iii) Find the name of the person who owns maximum no. of cars. (5 M)

Ans. Refer to Section 4.5.3.

6. Consider the following relations:
Supplier (S_id, sname, address)

Parts (P_id, Pname, Colour)

Suppliers and parts are related with many to many relationship with the descriptive attribute cost. Create a relational database in 3NF and solve the following queries in SQL :

(i) Find the names of suppliers who supply parts which are blue or pink in colour.

(ii) Find total cost of all parts supplied by 'Shree Agencies'.

(iii) Find the names and address of all suppliers who are supplying the item 'Bath towel'. (5 M)

Ans. Refer to Section 4.

7. Consider the following relations:
Musician (m_no , m_name, age, city)

Instrument (i_no , i_name)

Musician and instrument are related with a many to many relationship. Create a relational database in 3NF and solve the following queries in SQL :

(i) list all the 'tabala' players.

(ii) Find all the musicians who study in Pune and play 'flute'.

(iii) List all the instruments that are played by more than 3 musicians. (5 M)

Ans. Refer to Section 4.5.3.

8. A sports institute wants to maintain records of all its sports teams. For each team, the information regarding players in that team is to be maintained. Each player can be a part of more than one team. The details of all matches played and the scores of the teams in each match are to be recorded.

(i) Design an E-R diagram for above scenario, assume attributes if necessary.

(ii) Convert the E-R diagram into a relational database in 3NF. (7 M)

Ans. Refer to Section 4.5.3.

April 2016

1. What is a functional dependency? (1 M)

Ans. Refer to Section 4.2.

2. What are the different anomalies caused by redundant storage ? Explain any two in detail. (5 M)

Ans. Refer to Section 4.1.1.

3. Consider the relation : R(A, B, C, D, E, F) and a set of FD's on R as :
F = {A → C, C → BE, E → F, CD → F, E → D}
Compute closure of F i.e. F+. 5 M

Ans. Refer to Section 4.2.4.

October 2016

1. Define normal form. Explain third normal form (3NF) with example. (5 M)

Ans. Refer to Section 4.5.3.

2. What is decomposition of a relation? What are its desirable properties? (5 M)

Ans. Refer to Section 4.3.

3. Consider relation:
A (A, B, C, D, E, F) and set of FDs defined on R as:
F = {B → C, A → BC, DE → F, C → D, A → F}
Compute closure of F i.e. F+. (5 M)

Ans. Refer to Section 4.2.4.

4. Consider the following relations :
Student (Roll_no, name, address)
Book (Book_no, name, subject, Author)
Student and book are related with many to many relationship along with descriptive attributes- borrow_date, return_date.
Create a relational database for the above, convert it in 3NF and solve the following queries :
(i) Find out the student details who have borrowed books of author 'Y. Kanitkar'.
(ii) Delete all the records of students who have not done any book transaction.
(iii) List the book details authorise. (5 M)

Ans. Refer to Section 4.5.3.

5. Consider the relations :
Employee (Emp_id, Emp_name, salary)
Project (P_id), P_name, start_date, duration)
Employee and project are related with many-to-one.
Create a relational database for the above, convert it in 3NF and solve the following queries :
(i) List the details of employees working on project 'SI'.
(ii) Find out the name of employee earning salary less than that of 'Amit'. (5 M)

Ans. Refer to Section 4.5.3.

April 2017

1. What is decomposition? (1 M)

Ans. Refer to Section 4.3. (1 M)

2. Define second normal form.

Ans. Refer to Section 4.5.2.

3. Consider the following relation:

R(A, B, C, D, E) and the set of FD's defined on R as :

$F = \{A \rightarrow B, CD \rightarrow E, A \rightarrow C, B \rightarrow D, E \rightarrow A\}$

Compute the closure of F i.e. F+. (5 M)

Ans. Refer to Section 4.2.4.

4. State and explain in short rules of inference for functional dependencies. (5 M)

Ans. Refer to Section 4.

5. Consider the following relations:

Emp (eno, ename, salary, commission, designation)

Dept (dno, dname, location)

Emp and Dept are related with many to one relationship. Create a relational database in 3NF and solve the following queries in SQL :

(i) Find out employees who are working at Aurangabad location.

(ii) Find the maximum, minimum and average salary for every designation.

(iii) Update commission for every employee by 6% who belong to botany department. (5 M)

Ans. Refer to Section 4.5.3.

6. Consider the following relations:

Wholesaler (wno, wname, address, city)

Product (Pno, Pname)

Wholesaler and product are related with many to many relationship. Create a relational database in 3NF and solve the following queries in SQL:

(i) List the wholesalers of product 'Mouse'.

(ii) Count the number of wholesaler from 'Pune' city.

(iii) Delete records of wholesaler where product name is 'Scanner'. (5 M)

Ans. Refer to Section 4.5.3.

7. Consider the following relations :

Doctor (dno, dname, address, city)

Patient (Pno, Pname, address, disease)

Doctor and patient are related with many to many relationship. Create a relational database in 3 NF and solve the following in SQL :

(i) Find the number of patients visited by 'Dr. Pawar'.

(ii) Find the number of patients suffering from 'Cancer'.

(iii) Display doctor name and city who gives treatment to the patient 'Mr. Sagar'. (5 M)

Ans. Refer to Section 4.5.3.

8. Consider the following relations :

Movie (mno, mname, budget)

Actor (ano, aname, role)

Movie and Actor are related with one to many relationship. Create a relational database in 3NF and solve the following queries in SQL :

(i) List the names of movie in which 'Salman' has acted.

(ii) List the budgetwise movie.

(iii) Count the number of actors in movie 'PK'.

Ans. Refer to Section 4.5.3. (5 M)

October 2017

1. Define Third Normal Form.

Ans. Refer to Section 4.5.3. (1 M)

2. What is functional dependency?

Ans. Refer to Section 4.2. (2 M)

3. Consider the relation : R (A, B, C, D, G, H, I) and the set of FD's defined on R as :

$F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Compute the closure of F i.e. F+.

Ans. Refer to Section 4.2.4. (5 M)

4. What are the anomalies that might arise if we have redundant data? (5 M)

Ans. Refer to Section 4.1.1.

5. What are desirable properties of decomposition? Explain in brief. (5 M)

Ans. Refer to Section 4.4.

6. Consider the following relations:

Country (c_code, name, capital)

Population (p_code, population)

Country and population are related with one to one relationship. Create a relational database in 3NF and solve the following queries in SQL :

(i) Give the name and population of country whose capital is 'Delhi'.

(ii) List the name of all countries whose population greater than 75,00,000.

(iii) To print country wise population. (5 M)

Ans. Refer to Section 4.5.3.

7. Consider the following relations:

Machine (m_no, m_name, m_type, m_cost)

Part (p_no, p_name, description)

Machine and part are related with one to many relationship.

Create a relational database in 3NF and solve the following queries in SQL :

(i) Increase the cost of machine by 35%.

(ii) List all machine whose cost > 25,000.

(iii) Display machine name and cost having parts gear box and strearing. (5 M)

Ans. Refer to Section 4.5.3.

8. Consider the following relations:

Company (c_id, c_product, c_name, region, state)

Branches (b_id, b_name, b_product, city)

Company and Branches are related with one to many relationship. Create a relational database in 3NF and solve the following in SQL :

(i) List all the cities having branch product 'CPU' and 'MOUSE'.

(ii) List all the states whose branch product is 'Pen Drive'.

(iii) Print citywise branches in descending order. (5 M)

Ans. Refer to Section 4.5.3.

April 2018

1. What is functional dependency? (1 M)

Ans. Refer to Section 4.3.

2. Define BCNF. (1 M)

Ans. Refer to Section 4.5.4.

3. Let R = (A, B, C, D, E) is a relational schema with the following functional dependencies :

   A = {A → BC, CD → E, E → D, E → A}

   List the candidate keys for R. (5 M)

Ans. Refer to Section 4.2.

4. Give a set of functional dependencies for the relation at schema R (A, B, C, D) with primary key AB under which R is in 1NF but not in 2NF. (5 M)

Ans. Refer to Section 4.2.

5. Consider the following relations :

   Supplier (sid, sname, address)

   Parts (pid, pname, color, cost)

   Supplier and parts are related with many to many relationship.

   Create a relational database in 3NF and solve the following queries in SQL:

   (i) List all the suppliers who is supplying some red parts.

   (ii) Find the number of parts supplied by each supplier.

   (iii) Find the supplier names of parts whose cost is more than ₹ 250/-. (5 M)

Ans. Refer to Section 4.5.3.

6. Consider the following relations :

   Country (countrycode, name, capital)

   Population (pcode, pcount)

   Country and population are related with one to one relationship.

   Create a relational database in 3NF and solve the following queries in SQL.

   (i) Find the country name having lowest population.

   (ii) Find the name and population of a country whose capital name starting with a.

   (iii) List the names of all countries whose population is within the range 1,00,000 to 4,00,000. (5 M)

Ans. Refer to Section 4.5.3.

7. Consider the following relations:

   Person (pno, name, address)

   Car (cno, year, model)

   Person and car are related with one to many relationship.

   Create a relational database in 3NF and solve the following in SQL:

   (i) List all the names of people from Kharadi and have Maruti 800.

   (ii) Change address of Mr. Korth to Pune.

   (iii) List the name of people having car before 2010. (5 M)

Ans. Refer to Section 4.5.3.

❖❖❖