

Normalization

Prepared By: N K Shukla

Approach why normalization?

- Consider the following table account and branch for account and branch information.

ACCOUNT		
ANO	BALANCE	BNAME
A01	5000	VVN
A02	6000	KSAD
A03	7000	ANAND
A04	8000	VVN
A05	6000	KSAD

BRANCH	
BNAME	BADDRESS
VVN	MOTA BAZAR
KSAD	CHHOTA BAZAR
ANAND	PATEL COLONY

- ▶ Instead of using two table if information is combined in one table Account_Branch then..

ANO	BALANCE	BNAME	BADDRESS
A01	5000	VVN	MOTA BAZAR
A02	6000	KSAD	CHHOTA BAZAR
A03	7000	ANAND	PATEL COLONY
A04	8000	VVN	MOTA BAZAR
A05	6000	KSAD	CHHOTA BAZAR

- ▶ This poor database design has several pit falls or anomalies listed below:-
 - 1. Redundancy
 - 2. Update Anomalies
 - 3. Insert Anomalies
 - 4. Delete Anomalies

- **1. Redundancy**

Account_branch having repeated address of same branch again and again. It occupy more memories in database.

- **2. Update Anomalies**

if we update address of branch VVN in Account_branch table and if few tuples will not updated then its poor database.

- **3. Insert Anomalies**

if we open new branch ‘SURAT’ but account is not created in that branch then we can enter record in Account_branch table.

- **4. Delete Anomalies**

if we delete particular account from Account_branch table and that is the last account of that branch then branch record will be deleted from table hence branch is there but no record will available in DB.

NORMALIZATION

- ▶ **Normalization is process of removing all redundancy form database.**
- ▶ It is the application of a set of simple rules called first, second & third normal form. When database design is fully normalized ,there is no repletion of data across tables.
- ▶ The advantages of data normalization can be very large in terms of storage space as well Increased efficiency with which data can be updated & maintained.
- ▶ The rules of normalization or normal forms define exactly what kind of information can placed in each table and how this information relates to the fields.

Goal of Normalization :

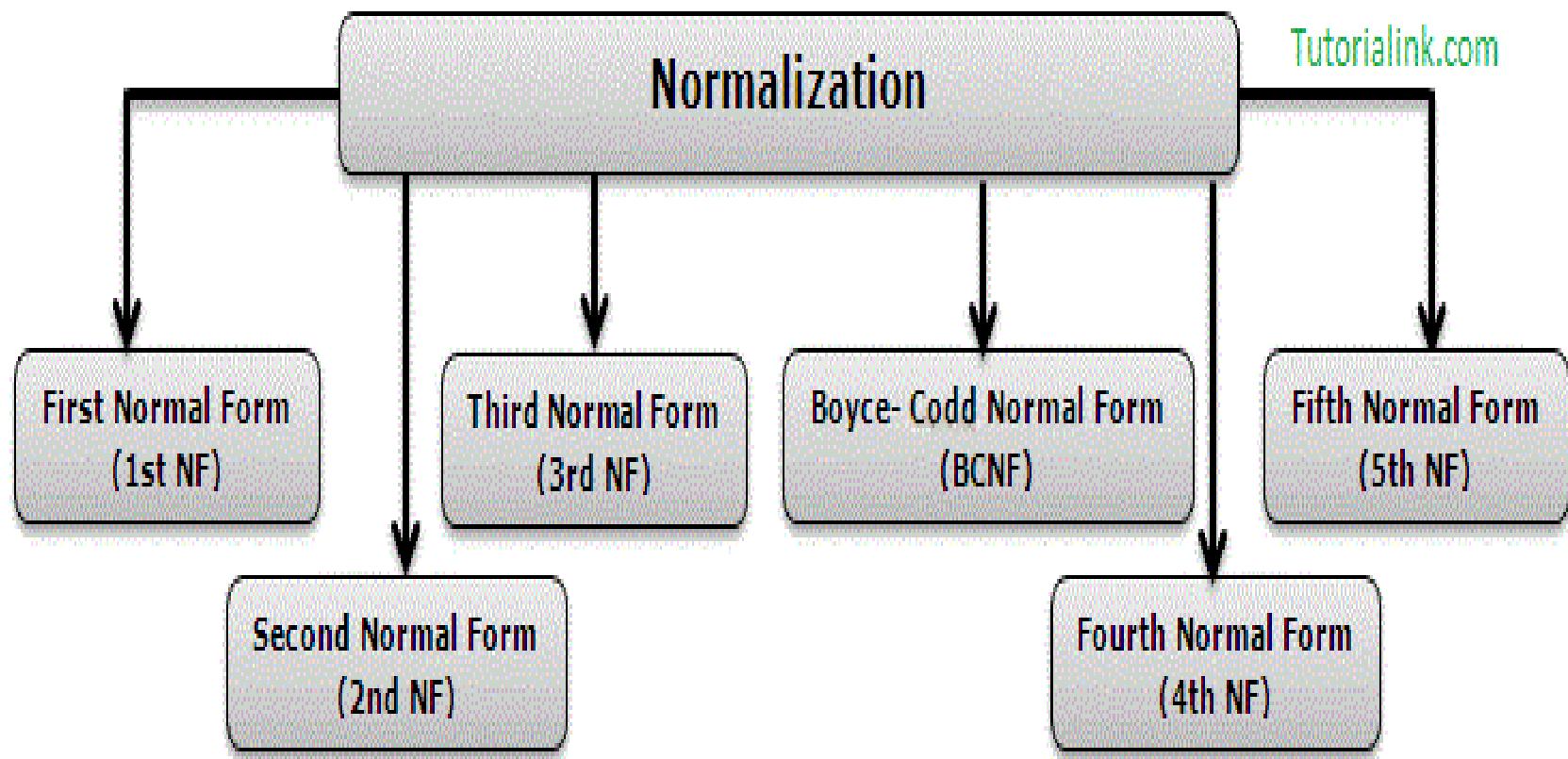
- ▶ To minimize data redundancy
- ▶ To minimize update, insert and delete anomalies

Advantages of Normalization :

- ▶ Greater overall database organization design will be gained.
- ▶ The amount of unnecessary redundant data reduced.
- ▶ Data integrity is easily maintained within the database.
- ▶ The database & application design processes are much for flexible.
- ▶ Security is easier to maintain or manage.

- ▶ **Disadvantages of Normalization :**
- ▶ The disadvantage of normalization is that it produces a lot of tables with a relatively small number of columns. These column then have to be joined using their primary/foreign key relation ship.
- ▶ This has two disadvantages.
 - Performance: all the joins required to merge data slow processing & place additional stress on your hardware.
 - Complex queries: developers have to code complex queries in order to merge data from different tables.

Various normal form :



FIRST NORMAL FORM (1NF) :

- ▶ "A relation schema R is in 1NF, if it does not have any **composite** attributes,**multivalued** attribute or their combination.“
- ▶ The objective of first normal form is that the table should contain no repeating groups of data. Data is divided into logical units called entities or tables
- ▶ All attributes (column) in the entity (table) must be single valued.
- ▶ Repeating or multi valued attributes are moved into a separate entity (table) & a relationship is established between the two tables or entities.

Example of the first normal form :

Consider the following **Customer** table.

Customer :

cid	name	address		contact_no
		society	city	
C01	aaa	Amul avas,Anand		{123456798 8}
C02	bbb	near parimal garden,abad		{123,333,44 45}
C03	ccc	sardar colony , surat		

- ▶ Here, address is a **composite attribute** , which is further subdivided into two column society and city. And attribute **contact_no** is multivalued attribute.

Problems with this relation are -

- ▶ It is not possible to store multiple values in a single field in a relation. so, if any customer has more than one contact number, it is not possible to store those numbers.
- ▶ Another problem is related to information retrieval . Suppose, here, if there is a need to find out all customers belonging to some particular city, it is very difficult to retrieve. The reason is: city name for all customers are combined with society names and stored whole as an address.

Solution for composite attribute

- Separate attributes in a relation for each sub-attribute of a composite attribute.

In our example, insert two separate attributes for Society and city in a relation in place of single composite attributes address. Now, insert data values separately for Society and City for all tuples.

Customer :

cid	name	society	city	contact_no
C01	aaa	Amul avas	Anand	{1234567988}
C02	bbb	near parimal garden	abad	{123,333,4445}
C03	ccc	sardar colony	surat	

Solution for Multi-valued attribute

- Two approaches are available to solve problem of multi-valued attribute

1. First Approach:

In a First approach, determine maximum allowable values for a multi-valued attributes. In our case, if maximum two numbers are allowed to store, insert two separate attributes to store contact numbers as shown.

Customer:

<u>cid</u>	name	Society	city	contact_n o1	contact_n o2	contact_n o3
C01	aaa	Amul avas	Anand	1234567 988		
C02	bbb	near parimal garden	abad	123	333	4445
C03	ccc	sardar colony	surat			

- ▶ Now, if customer has only one contact number or no any contact number, then keep the related field empty for tuple of that customer.
- ▶ If customer has two contact numbers, store both number in related fields. If customer has more than two contact numbers, store two numbers and ignore all other numbers.

2. Second Approach:

- ▶ In a second approach, remove the multi-valued attribute that violates 1NF and place it in a separate relation along with the primary key of given original relation.
- ▶ The primary key of new relation is the combination of multi-valued attribute and primary key of old relation. for example, in our case, remove the contact_no attribute and place it with cid in a separate relation customer_contact.
- ▶ Primary Key for relation Customer_contact will be combination of cid and contact_no.

customer

<u>cid</u>	<u>name</u>	<u>Society</u>	<u>city</u>
C01	aaa	Amul avas	Anand
C02	bbb	near parimal garden	abad
C03	ccc	sardar colony	surat

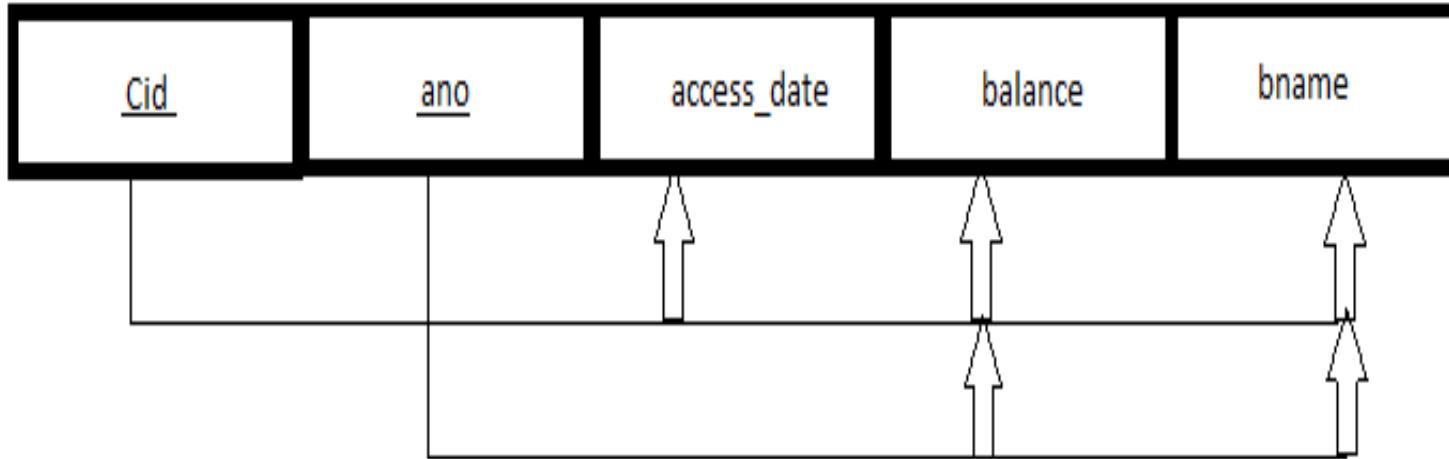
Customer_contact

<u>cid</u>	<u>contact_no</u>
C01	1234567988
C02	123
C02	333
C02	4445

- ▶ First approach is simple. But, it is not always possible to put restriction on maximum allowable values. It also introduces null values formant fields.
- ▶ Second approach is superior as it does not suffer from draw backs of first approach. But, it is some what complicated one. For example, to display all information about any/all customers, two relations - Customer and Customer_contact - need to be accessed.

SECOND NORMAL FORM (2NF) :

- ▶ "A relation schema R is in 2NF, if It is in First Normal Form, and every non_prime attribute of relation is fully functionally dependent on primary key.“
- ▶ A relation can violate 2NF only when it has more than one attribute in combination as a primary key. if relation has only single attribute as a primary key, then, relation will definitely be in 2NF.
- ▶ *Example:*
- ▶ consider the folllowing relation Table
Depositor_Account

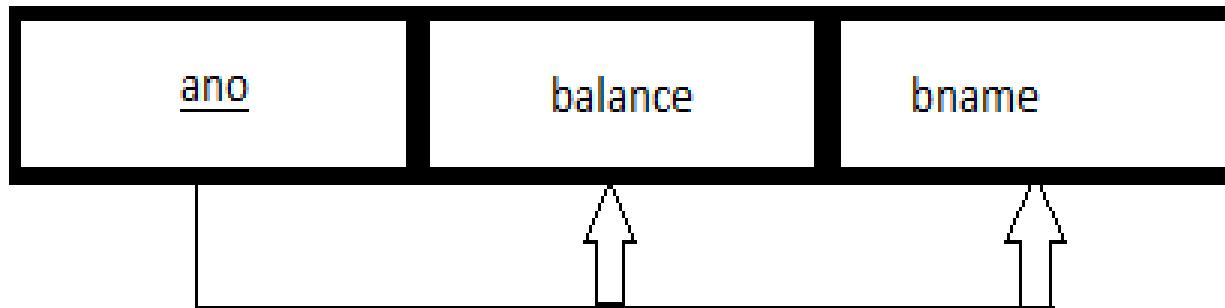


- In this relation schema, access_date, balance and bname are non - prime attributes. Among all these three attributes, access_date is fully dependent on primary key (Cid and ano). But balance and bname are not fully dependent on primary key. they depend on ano only.
So, this relation is not in Second normal form. Such kind of partial dependencies result in data redundancy.

- ▶ *Solution:*
- ▶ Decompose the relation such that, resultant relations do not have any partial functional dependency. For this purpose, remove the partial dependent non-prime attributes that violates 2NF in relation.
- ▶ Place them in a separate new relation along with the prime attribute on which they fully depend.
In our example, balance and bname are partial dependant attribute on primary key.
- ▶ so, remove them and place in separate relation called account alog with prime attribute ano. For relation Account, ano will be a Primary key.
- ▶ The Depositor_account relation will be decamped in two separate relations, called Account_holder and Account.

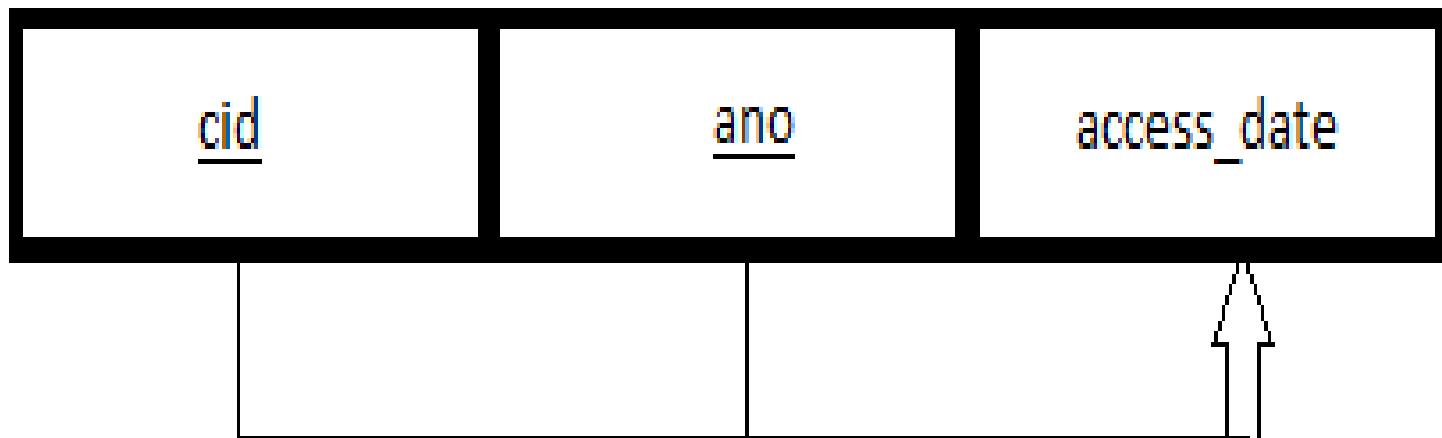
► Account.

Tutoriallink.com



► Account_Holder:

Tutoriallink.com

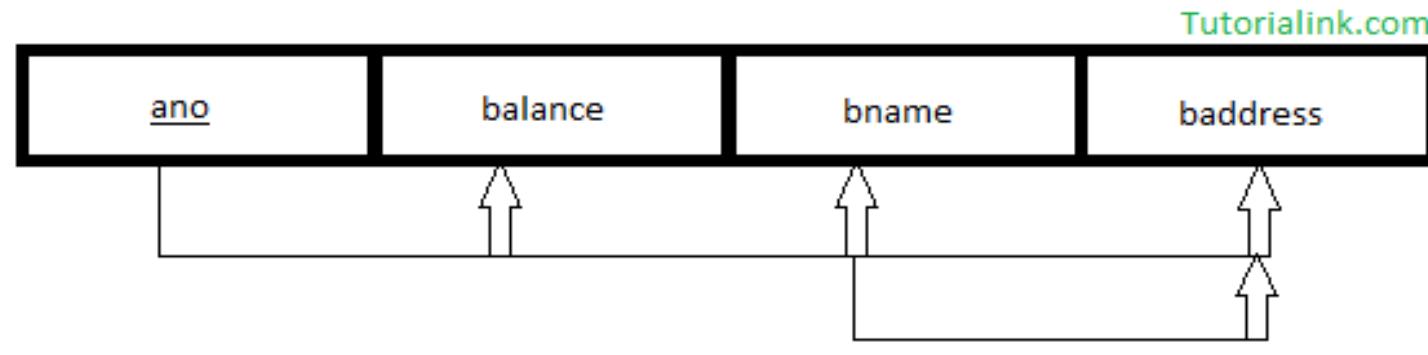


THIRD NORMAL FORM (3NF) :

- ▶ "A relation schema R is in 3NF, if it is in Second normal form, and no any non-prime attribute of relation is transitively dependent on primary key.“
- ▶ Third normal form ensures that the relation does not have any non-prime attribute transitively dependent on primary key. In other words, It ensures that all the non-prime attributes of relation directly depend on the primary key.

Example:

- ▶ Consider the following relation schema **Account_Branch**
Account_Branch:



- ▶ This relation contains following functional dependencies.
FD1 : $\text{ano} \rightarrow \{\text{balance}, \text{bname}, \text{baddress}\}$, and
FD2 : $\text{bname} \rightarrow \text{baddress}$
- ▶ In this relation schema, there is a functional dependency $\text{ano} \rightarrow \text{bname}$ between ano & bname as shown in FD1. also, there is another functional dependency $\text{bname} \rightarrow \text{baddress}$ between bname & baddress as shown in FD2.
- ▶ more over bname is a non-prime attribute. So, there is a transitive dependency from ano to baddress , denoted by $\text{ano} \rightarrow \text{baddress}$.
- ▶ Such kind of transitive dependencies result in data redundancy. In this relation, branch address will be stored repeatedly for each account of the same branch, occupying more amount of memory.

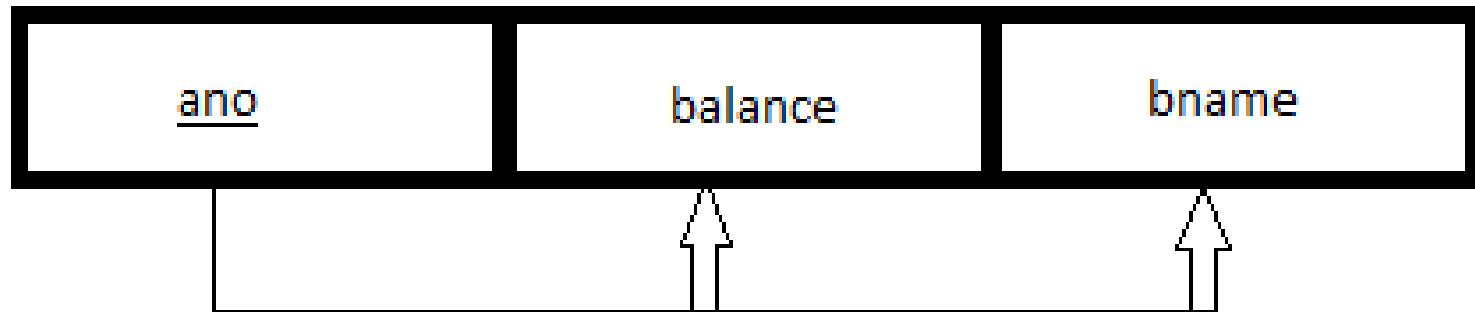
- ▶ Solution:
- ▶ Decompose the relation in such a way that, resultant relations do not have any non-prime attribute transitively dependent on primary key. For this purpose, remove the transitively dependent non-prime attributes that violates 3NF from relation. Place them in a separate new relation along with the non-prime attribute due to which transitive dependency occurred. The primary key of new relation will be the non-prime attribute.

In our example, baddress is transitively dependent on ano due to non-prime attribute bname. So, remove baddress and place it in separate relation called Branch along with the non-prime attribute bname. for relation Branch, bname will be a primary key.

The Account_Branch relation will be decomposed in two separate relations called Account and Branch.

► Account:

Tutorialink.com



► Branch:

Tutorialink.com

