Mahatma Gandhi Charitable Trust Managed

Shri Labhubhai Trivedi Institute of Engineering & Technology

CREATING CREATORS - SLTIET

Relational Database Management Systems

RDBMS - Subject Code: 4330702

Prof. Rinkal Umaraniya

CSE Department

SLTIET, Rajkot

CREATING CREATORS - SLTIET

Unit – 5

Normalization

CREATING CREATORS - SLTIET

Sub Topics:

5.1 Basics of Normalization
5.2 Dependencies among attributes
5.3 First Normal Form (1NF)
5.4 Second Normal Form (2NF)
5.5 Third Normal Form (3NF)
5.6 Advantages and Disadvantages of Normalization

5.1 Basics of Normalization

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

- Making relations very large.
- It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.
- The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

What is Normalization?

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

Data modification anomalies can be categorized into three types:

- **Insertion Anomaly:** Insertion anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the loss of some other important data.
- **Updation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Anomaly

Take one example to understand anomaly :

Example : **Tabel : Student**

s_id	s_name	cor_id	cor_name	f_id	f_name	salary
1	Ram	C1	RDBMS	F1	John	30000
2	Ravi	C2	OS	F2	Bob	40000
3	Nitin	C1	RDBMS	F1	John	30000
4	Amrit	C1	RDBMS	F1	John	30000

Insertion Anomaly:

- Insertion anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- In our example, we introduces new course JAVA but currently no any students take this course. That's why we can not insert the our new course details in our database because s_id is primary key and it’s not allowed to add null value in s_id column.
- Such type of problem is known as ‘Insertion Anomaly’.

Deletion Anomaly:

- These anomalies occur when deleting a record from a database and can result in the unintentional loss of data.
- In our example, if we delete student those s_id is 1 then it does not any problem occur in student table.
- But if we delete s_id = 2, then problem is occur because delete all the course related information of student 2. Now we enroll one new student in course 2 (OS) then it not possible to insert this new student in student table.
- Because course 2 related all information was deleted like course id, course name, faculty id, faculty name and salary.
- Such type of problem is known as ‘Deletion Anomaly’.

Update Anomaly:

- These anomalies occur when modifying data in a database and can result in inconsistencies or errors.
- In our example, if we update faculty F1 salary then it required to update rows where s_id = 1,3 and 4.
- Problem occur if we forget to update all above mention rows.
- If we failure to update some of the tuples may result in inconsistent state of the database.
- Such type of problem is known as ‘Update Anomaly’.

5.2 Dependencies among attributes

Prime & Non-prime Attributes :

Account :

Acc_No	Balance	Bank_Name

Here,

1) Acc_No is a primary key; hence it is considered a prime attribute.

2) On the other hand balance and Bank_Name are not members of primary key; hence they are considered non-prime attributes.

Continue…

Here, we consider one example.

Account

a_no	balance	b_name
A01	5000	vvn
A02	6000	ksad
A03	7000	anand
A04	8000	ksad
A05	6000	vvn

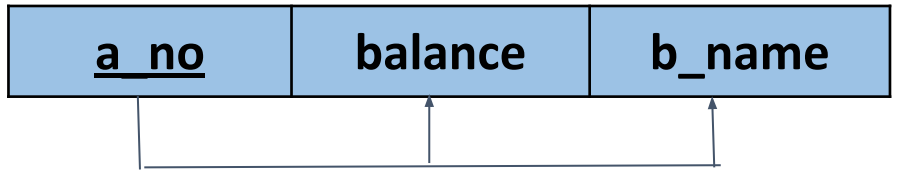
Branch

b_name	b_address
vvn	Mota bazar, VV Nagar
ksad	Chhota bazar, Karmsad
anand	Nana bazar, Anand

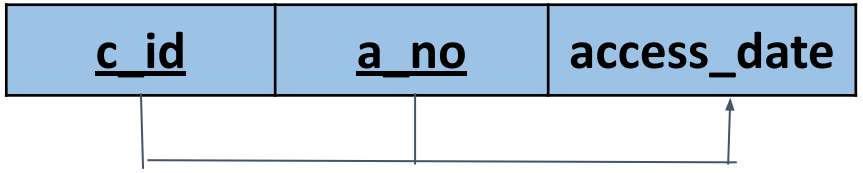
Continue…

Functional Dependency :

- Consider the relation Account given in table.
- In this relation, for any a_no, say 'A02' there is a unique set of balance and b_name. In other words, a_no can uniquely determine balance and b_name. So, there is a functional dependency from a_no to balance and b_name.
- This can be denoted by **a_no -> balance** and **a_no -> b_name** or in combination, **a_no -> {balance, b_name}.**
- Here, balance functionally depends on the a_no. But, the reverse is not valid. For any balance there is no unique set of a_no.
- For example, for balance 6000 there are two account number, 'A02' and 'A05'. So there is no functional dependency from balance to a_no.



Account

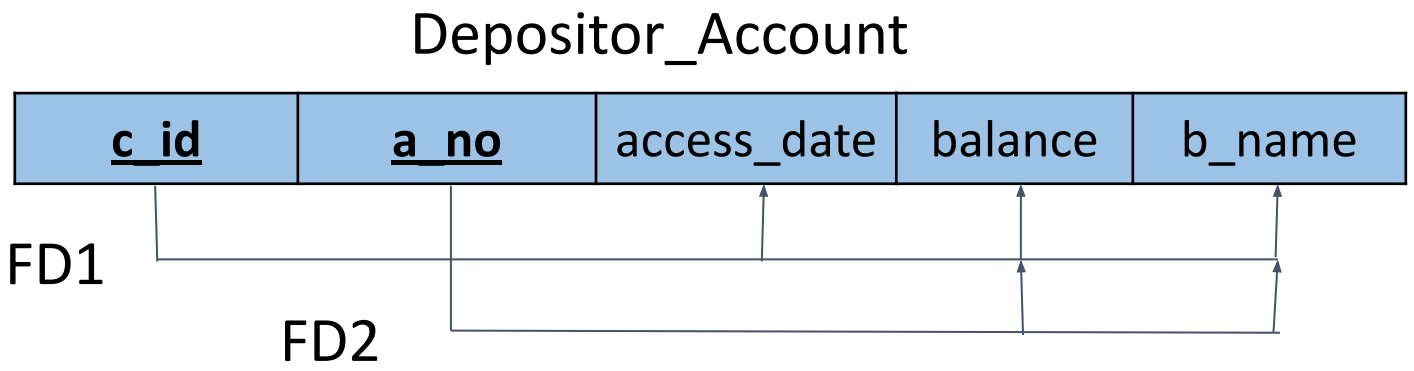


Account_Holder

Continue…

Fully Functional Dependency :

- Consider the following relation schema Depositor_Account.
- Here, is c_id or a_no is removed from the primary key, access_date cannot be determined uniquely.
- So, access_date is fully functional dependant on c_id and a_no, as depicted in FD1.
- But, balance and b_name depend only on the a_no as depicted in FD2.
- If c__id is removed from the primary key, even then the dependency of balance and b_name on holds.
- So, balance and b_name do not fully functionally depend on the primary key, i.e. c_id and a_no.



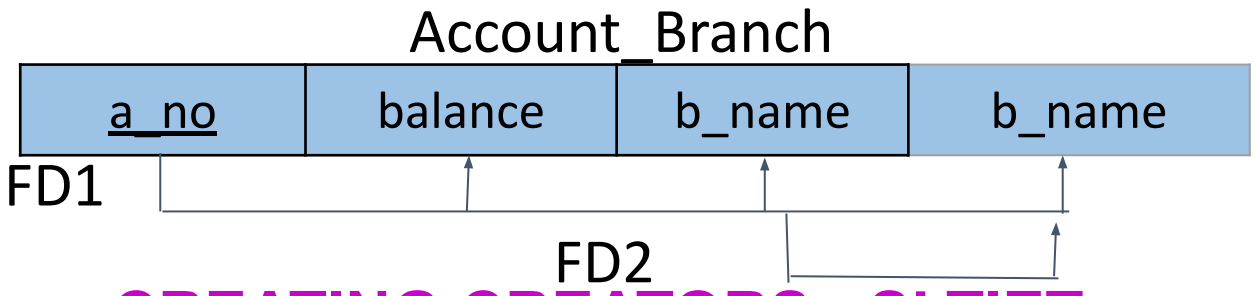
Continue…

Partial Functional Dependency :

- Consider the following relation schema Depositor_Account.
- Here, is c_id or a_no is removed from the primary key, the dependency of balance and b_name on a_no holds.
- So, balance and b_name partially functional dependency on the primary key, i.e. c_id and a_no.

Transitive Functional Dependency :

- Consider the following relation schema Account_Branch.
- Here, functional dependency a_no -> b_name in FD1.
- And another functional dependency b_name -> b_address in FD2.
- Moreover, b_name is a non-prime attribute. So, there is a transitive dependency from a_no to b_address, denoted by a_no -> b_address.



5.3 First Normal Form (1NF)

- **A relation R is in first normal form (1NF) :** if and only if it does not contain any composite or multi valued attributes or their combinations.

C_ID	Name	Address		Contact_No
		Society	City	
101	Mr. Mehta	SD Road	Ahmedabad	9654713258, 8563247895
102	Kartik	Kalawad Road	Rajkot	6524871365
103	Kumud	Sanala Road	Morbi	8532479654

- Above relation has four attributes C_ID, Name, Address, Contact_No. Here Address is a composite attribute which is further divided into sub attributes as Society and City.
- Another attribute Contact_no is a multi-valued attribute which can store more than one value. So Above relation is not in 1NF.

Continue…

Problem:

- Suppose we want to find all customers for some particular city then it is difficult to retrieve. Reason is city name is combined with society name and stored whole as address.
- It is not possible to store multiple values in single field in table. So, if any customer has more than one contact number, it is impossible to store those numbers.

Continue…

Solution for composite attribute :

- Insert separate attribute for each sub attribute of composite attribute.
- In our example insert two separate attribute for society and city in table instead of single attribute composite address.
- So, above table can be created as follows:

C_ID	Name	Society	City	Contact_No1
101	Mr. Mehta	SD Road	Ahmedabad	9654713258, 8563247895
102	Kartik	Kalawad Road	Rajkot	6524871365
103	Kumud	Sanala Road	Morbi	8532479654

Continue…

Solution for multi-valued attribute :

1) First Approach:

- Determine maximum allowable values for multi-valued attribute.
- Insert separate attribute for multi valued attribute and insert only one value on one attribute and other in other attribute.
- So, above table can be created as follows:

C_ID	Name	Society	City	Contact_No1	Contact_No2
101	Mr. Mehta	SD Road	Ahmedabad	9654713258	8563247895
102	Kartik	Kalawad Road	Rajkot	6524871365	
103	Kumud	Sanala Road	Morbi	8532479654	

Continue…

2) Second Approach:

- Remove multi valued attributes and place it in a separate relation along with the primary key of a given original relation.
- So, above table can be created as follows:

Customer :

C_ID	Name	Society	City
101	Mr. Mehta	SD Road	Ahmedabad
102	Kartik	Kalawad Road	Rajkot
103	Kumud	Sanala Road	Morbi

Customer_Contact:

C_ID	Contact_No
101	9654713258
101	8563247895
102	6524871365
103	8532479654

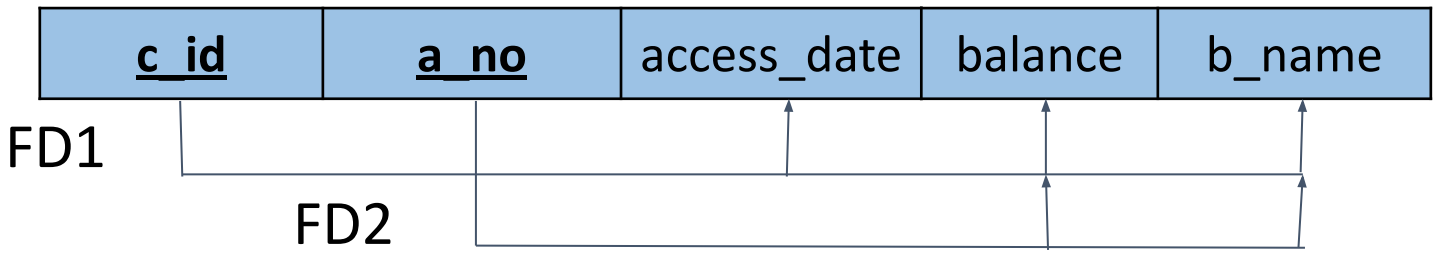
5.3 Second Normal Form (2NF)

- **A relation R is in second normal form (2NF) :** if and only if it is in 1NF and every non-prime attribute of relation is fully functionally dependent on the primary key.
- **A relation R is in second normal form (2NF)** : if and only if it is in 1NF and no any non-prime attribute is partially dependent on the primary key.
- Below relation has five attributes c_id, a_no, access_date, balance, b_name and two FDS:

  FD1: {c_id, a_no} -> {access_date, balance, b_name} and

  FD2: a_no -> {balance, b_name}

Depositor_Account



Continue…

We have C_ID and Acc_No as primary key.

- In this relation schema, access_date, balance & b_name are non-prime attributes. Among all these 3 attributes, access_date is fully dependent on primary key (c_id, a_no)
- But balance and b_name do not fully dependent on primary key. Instead, they depend on ano only. This his attributes are partial dependent on primary key. So below relation is not in 2NF.

Problem:

- For example, in case of a joint account multiple customers have a common account.
- If some account says ‘A02’ is jointly owned by two customers ‘C02’ and ‘C04’ then data values for attributes balance and b_name will be duplicated in two different tuples of customers ‘C02’ and ‘C04’.

Continue…

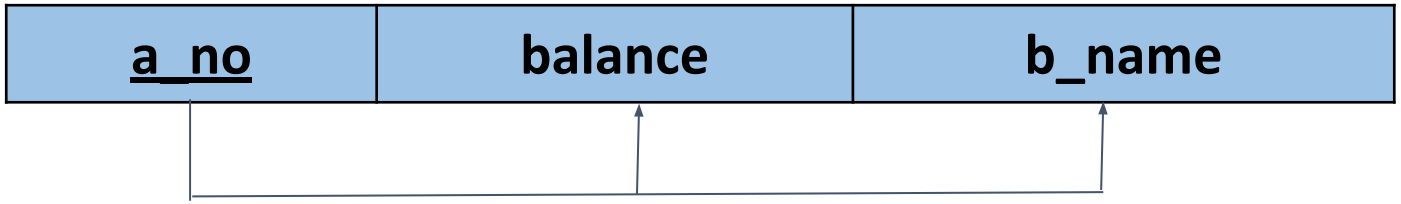
Solution:

- Decompose relation in such a way that resultant relation does not have any partial FD.
- For this purpose remove partial dependent attributes that violate 2NF from relation. Place them in separate new relation along with the prime attribute on which they are fully dependent.
- The primary key of a new relation will be the attribute on which they are fully dependent.
- Keep other attributes the same as in that table with the same primary key.

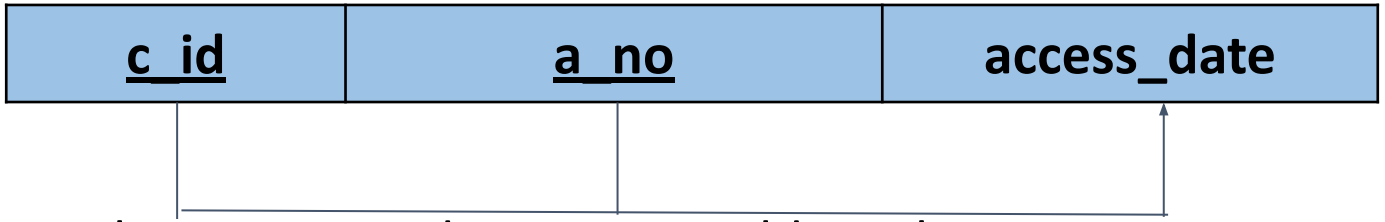
Continue…

So, above table Depositor_Account can be decomposed into Account and Account_Holder table as per following.

Account:



Account_Holder:



So, Both Account and Account_Holder relations are in second normal form.

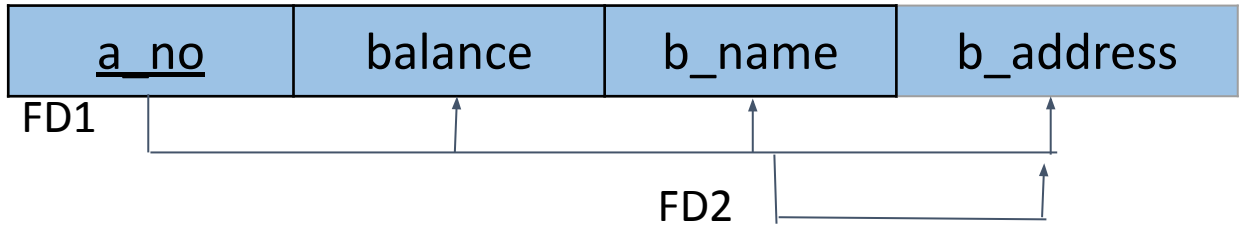
5.4 Third Normal Form (3NF)

- **A relation R is in third normal form (3NF) :** If and only if it is in 2NF and no any non- prime attribute of a relation is transitively dependent on the primary key.
- Below relation has four attributes Acc_No, Balance, Bank_Name, Bank_Add and two FDS:

  FD1 : a_no {balance, b_name, b_address} and

  FD2 : b_name {b_address}
- In this relation schema, there is functional dependency a_no -> b_name as shown in FD1.
- Also, there is another functional dependency b_name -> b_address as shown in FD2.
- Moreover, b_name is a non-prime attribute. So, there is a transitive dependency from a_no to b_address denoted by a_no -> b_address.
- So, there is a non-prime attribute b_address which is transitively dependent on primary key a_no.
- So, the above relation is not in 3NF.

Account_Branch



Continue…

Problem:

- Transitively dependency results in data redundancy.
- In this relation bank address will be stored repeatedly for each account of the same branch which occupies more space.

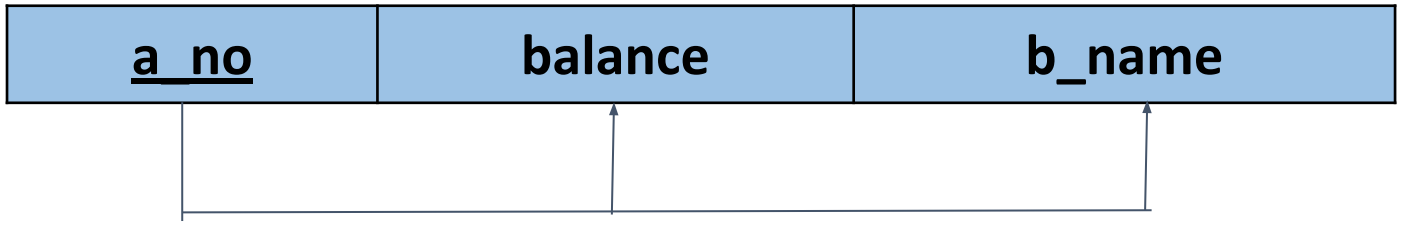
Solution:

- Decompose relation in such a way that resultant relation does not have any non-prime attribute that are transitively dependent on primary key.
- For this purpose remove transitively dependent attribute that violate 3NF from relation.
- Place them in separate new relation along with the non-prime attribute due to which transitive dependency occurred. The primary key of new relation will be this non-prime attribute.
- Keep other attributes the same as in that table with the same primary key.

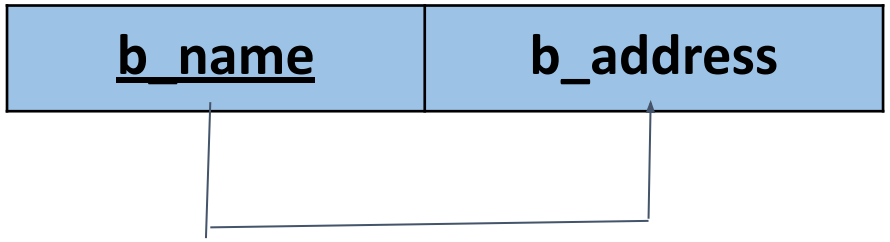
Continue…

So, above table Account_Branch can be decomposed into Account and Branch table as per following:

Account:



Branch:



- Now, there is no any transitive dependency in account and branch relations.
- So, these both relations are in third normal form.

5.5 Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

5.6 Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- Difficult and expensive
- It requires a detailed database design
- Maintenance overhead

Questions

Q-1) What is Normalization?

Q-2) Explain 1NF with example.

Q-3) What is functional dependency, fully functional dependency, partial functional dependency and transitive dependency?

Q-3) Write about 2NF and 3NF with example.

Q-4) Advantages and Disadvantages of Normalization.

SLTIET
आत्मदीपोभव

Thank You

CREATING CREATORS - SLTIET