Unit 5

Relational-Database Design

DBMS-Prachiti Pimple

Pitfalls in Relational-Database Design

- Relational database design requires that we find a “good” collection of relation schemas.
- A bad design may lead to
  - Repetition of Information.
  - Inability to represent certain information.

- Design Goals:

1. Avoid redundant data .
2. Ensure that relationships among attributes are represented .
3. Facilitate the checking of updates for violation of database integrity constraints.

- Example
- Consider the relation schema:

Lending-schema = (branch-name, branch-city, assets, customer-name,loan-number, amount)

branch-name	branch-city	assets	customer-name	loan-number	amount
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

- Problems:

Redundancy:

. Data for branch-name, branch-city, assets are repeated for each loan that a branch makes

. Wastes space

. Complicates updating

Null values .

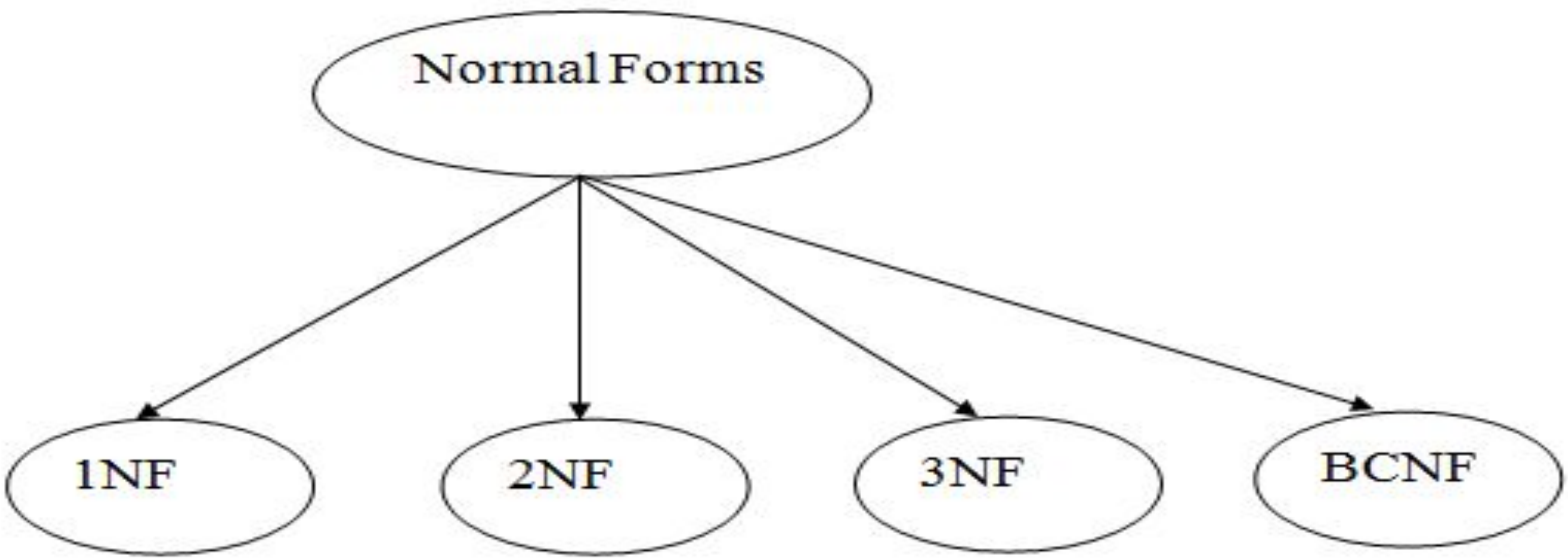
- Cannot store information about a branch if no loans exist .
- Can use null values, but they are difficult to handle.

- In the given example the database design is faulty which makes the above pitfalls in database. So we observe that in relational database design if the design is not good then there will be faults in databases.

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 the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

- Types of Normal Forms
- There are the four types of normal forms:



Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

Problems Without Normalization

- If a table is not properly normalized and have data redundancy then it will not only eat up extra memory space but will also make it difficult to handle and update the database, without facing data loss. Insertion, Updation and Deletion Anomalies are very frequent if database is not normalized.

- **Student**

rollno	name	branch	hod	office_tel
401	Akon	CSE	Mr. X	53337
402	Bkon	CSE	Mr. X	53337
403	Ckon	CSE	Mr. X	53337
404	Dkon	CSE	Mr. X	53337

In the table above, we have data of 4 Computer Sci. students. As we can see, data for the fields branch, hod(Head of Department) and office_tel is repeated for the students who are in the same branch in the college, this is **Data Redundancy**.

- Insertion Anomaly

Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as **NULL**.

- Also, if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students.
- These scenarios are nothing but **Insertion anomalies**.

- Updation Anomaly

What if Mr. X leaves the college? or is no longer the HOD of computer science department? In that case all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency. This is Updation anomaly.

- Deletion Anomaly

In our **Student** table, two different information's are kept together, Student information and Branch information. Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information. This is Deletion anomaly.

First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

- Rules for First Normal Form

Rule 1: Single Valued Attributes

Each column of your table should be single valued which means they should not contain multiple values.

Rule 2: Attribute Domain should not change

In each column the values stored must be of the same kind or type.

For example: If you have a column dob to save date of births of a set of people, then you cannot or you must not save 'names' of some of them in that column along with 'date of birth' of others in that column. It should hold only 'date of birth' for all the records/rows.

Rule 3: Unique name for Attributes/Columns

This rule expects that each column in a table should have a unique name.

Rule 4: Order doesn't matters

This rule says that the order in which you store the data in your table doesn't matter

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

- The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Example 2:

- Assume, a video library maintains a database of movies rented out. Without any normalization in database, all information is stored in one table as shown below.

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

- **1NF Example**

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Functional Dependency

- The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

$$X \rightarrow Y$$

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

example:

- Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.

- Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

- Functional dependency can be written as:

$$Emp\_Id \rightarrow Emp\_Name$$

Types of Functional Dependencies

1. Multivalued Dependency
2. Trivial Functional Dependency
3. Non-Trivial Functional Dependency
4. Transitive Dependency

1.Trivial Functional Dependency

- The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.
- So, X -> Y is a trivial functional dependency if Y is a subset of X.

Example:

Consider a table with two columns Employee_Id and Employee_Name.

{Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as Employee_Id is a subset of {Employee_Id, Employee_Name}
.

Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial dependencies too

2. Non-trivial functional dependency

Functional dependency which also known as a nontrivial dependency occurs when A->B holds true where B is not a subset of A.

In a relationship, if attribute B is not a subset of attribute A, then it is considered as a non-trivial dependency.

A → B has a non-trivial functional dependency if B is not a subset of A.

When A intersection B is NULL, then A → B is called as complete non-trivial.

Example:

ID → Name

Name → DOB

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Apple	Tim Cook	57

Example:

(Company} -> {CEO}

(if we know the Company, we knows the CEO name)
But CEO is not a subset of Company, and hence it’s non-trivial functional dependency.

3.Multivalued Dependency in DBMS

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

Example:

Car_model	Manufacturing year	Color
H001	2017	Metallic
H001	2017	Green
H005	2018	Metallic
H005	2018	Blue
H010	2015	Metallic
H033	2012	Gray

In this example, year and color are independent of each other but dependent on car_model. In this example, these two columns are said to be multivalue dependent on car_model.

This dependence can be represented like this:

car_model -> Manufacturing year

car_model-> colour

4.Transitive Dependency in DBMS

- A Transitive Dependency is a type of functional dependency which happens when "t" is indirectly formed by two functional dependencies.

- Consider a relation R(A B C) where A, B and C are the attributes of the relation R.

A Transitive dependency exists when we have the following functional dependency pattern A→B and B → C; therefore, A → C

Example:

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Alibaba	Jack Ma	54

- {Company} -> {CEO}

(if we know the company, we know its CEO's name)

- {CEO } -> {Age}

If we know the CEO, we know the Age

- Therefore according to the rule of rule of transitive dependency:
- { Company} -> {Age} should hold, that makes sense because if we know the company name, we can know his age.

Second Normal Form (2NF)

- The second step in Normalization is 2NF.
- A table is in 2NF, only if a relation is in 1NF and meet all the rules, and every non-key attribute is fully dependent on primary key.
- The Second Normal Form eliminates partial dependencies on primary keys.

Example

<StudentProject>

StudentID	**ProjectID**	**StudentName**	**ProjectName**
S89	P09	Olivia	Geo Location
S76	P07	Olivia	Geo Location
S56	P03	Ava	IoT Devices
S92	P05	Alexandra	Cloud Deployment

- In the above table, we have partial dependency;
- The prime key attributes are StudentID and ProjectID.
- The non-prime attributes
  i.e. StudentName and ProjectName should be functionally dependent on part of a candidate key, to be Partial Dependent.
- The StudentName can be determined by StudentID, which makes the relation Partial Dependent.
- The ProjectName can be determined by ProjectID, which makes the relation Partial Dependent.
- Therefore, the <StudentProject> relation violates the 2NF in Normalization and is considered a bad database design.

- **Example (Table converted to 2NF)**
- To remove Partial Dependency and violation on 2NF, decompose the above tables −

<StudentInfo>

StudentID	**ProjectID**	**StudentName**
S89	P09	Olivia
S76	P07	Olivia
S56	P03	Ava
S92	P05	Alexandra

<ProjectInfo>

ProjectID	ProjectName
P09	Geo Location
P07	Geo Location
P03	IoT Devices
P05	Cloud Deployment

- Example 2:

CAND_ID	SUBJECT_NO	SUBJECT_FEE
111	S1	1000
222	S2	1500
111	S4	2000
444	S3	1000
444	S1	1000
222	S5	2000

- In this table, you can note that many subjects come with the same subject fee. Three things are happening here:
- The SUBJECT_FEE won't be able to determine the values of CAND_NO or SUBJECT_NO alone;
- The SUBJECT_FEE along with CAND_NO won't be able to determine the values of SUBJECT_NO;
- The SUBJECT_FEE along with SUBJECT_NO won't be able to determine the values of CAND_NO;

Thus,

- We can conclude that the attribute SUBJECT_FEE is a non-prime one since it doesn't belong to the candidate key here {SUBJECT_NO, CAND_ID} ;
- But, on the other hand, the SUBJECT_NO – > SUBJECT_FEE, meaning the SUBJECT_FEE depends directly on the SUBJECT_NO, and it forms the candidate key's proper subset. Here, the SUBJECT_FEE is a non-prime attribute, and it depends directly on the candidate key's proper subset. Thus, it forms a partial dependency.
- ***Conclusion:*** The relation mentioned here does not exist in 2NF.
- Let us now convert it into 2NF. To do this, we will split this very table into two, where:
- Table 1: CAND_NO, SUBJECT_NO and Table 2: SUBJECT_NO, SUBJECT_FEE

CAND_NO	SUBJECT_NO
111	S1
222	S2
111	S4
444	S3
444	S1
222	S5

Table 2

SUBJECT_NO	SUBJECT_FEE
S1	1000
S2	1500
S3	1000
S4	2000
S5	2000

- **Example 3:** Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

- In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.
- To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.
- A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.
- X is a super key.
- Y is a prime attribute, i.e., each element of Y is part of some candidate key.

- **Example:**
- **EMPLOYEE_DETAIL table:**

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

- **Super key in the table above:**

1. {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

- Candidate key: {EMP_ID}

- Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

- Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.
- That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Consider the following example:

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

In the table able, [Book ID] determines [Genre ID], and [Genre ID] determines [Genre Type]. Therefore, [Book ID] determines [Genre Type] via [Genre ID] and we have transitive functional dependency, and this structure does not satisfy third normal form.

To bring this table to third normal form, we split the table into two as follows:

TABLE_BOOK

Book ID	Genre ID	Price
1	1	25.99
2	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99

TABLE_GENRE

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

Now all non-key attributes are fully functional dependent only on the primary key. In [TABLE_BOOK], both [Genre ID] and [Price] are only dependent on [Book ID]. In [TABLE_GENRE], [Genre Type] is only dependent on [Genre ID].

Example (Table violates 3NF)

<MovieListing>

Movie_ID	**Listing_ID**	**Listing_Type**	**DVD_Price** ($)
0089	007	Comedy	100
0090	003	Action	150
0091	007	Comedy	100

The above table is not in 3NF because it has a transitive functional dependency −

.

Movie_ID -> Listing_ID
Listing_ID -> Listing_Type

Therefore, **Movie_ID -> Listing_Type** i.e. transitive functional dependency

- **Example (Table converted to 3NF)**
- To form it in 3NF, you need to split the tables and remove the transitive functional dependency.

<Movie>

Movie_ID	**Listing_ID**	**DVD_Price** ($)
0089	007	100
0090	003	150
0091	007	100

<Listing>

Listing_ID	Listing_Type
007	Comedy
003	Action
007	Comedy

Unnormalized

<MovieListing>

Movie_ID	Listing_ID	Listing_Type	DVD_Price ($)
0089	007	Comedy	100
0090	003	Action	150
0091	007	Comedy	100

Normalized

<Movie>

Movie_ID	Listing_ID	DVD_Price ($)
0089	007	100
0090	003	150
0091	007	100

<Listing>

Listing_ID	Listing_Type
007	Comedy
003	Action

Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

- **Example:** Let's assume there is a company where employees work in more than one department.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

- **In the above table Functional dependencies are as follows:**
- EMP_ID → EMP_COUNTRY
- EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.
To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549