4. Functional Dependencies & Normal Forms

1. Pitfalls in relational database
   Decomposing bad schema
   Need for Decomposition
   Desirable Properties of Decomposition

2. 1NF
   Super Key

3. Functional dependency:
   Closure of Functional Dependency Set
   Closure of Attribute Set
   Minimal Functional Dependency Set

4. 2NF
   BCNF
   3 NF
   Denormalization

5. Practical Database Design
   Alternative Design Techniques

PITFALLS IN RELATIONAL DATABASE:

→ Relational database design requires good collection of relational schemas. A bad design may lead to

- Repetition of Information
- Inability to represent certain information

→ Design Goals :

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

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

branch - name	branch - city	customer - name	Loan - number	Amount
Ramapuram	Chennai	Jones	L-17	1000
Anna Nagar	Chennai	Smith	L-23	2000
Adyar	Chennai	Mark	L-14	1500

→ Problems :

- Redundancy :
  - Data for branch-name, branch-city are repeated for each loan that branch makes.
  - Waste 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

DECOMPOSING BAD SCHEMA:

→ Decomposition is the process of breaking down in parts or elements.

→ It replaces a relation with a collection of smaller relations

→ It breaks the table into multiple tables in a database

→ It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.

→ If there is no proper decomposition of the relation, then it may lead to problems like loss of information.

NEED FOR DECOMPOSITION:

→ A functional decomposition is the process of breaking down the functions of an organization into progressively greater level of details.

→ In decomposition, one function is described in greater detail by a set of other supporting functions.

→ The decomposition of a relation schema R consists of replacing the relation schema by two or more relation schemas that each contain a subset of the attributes of R and together include all attribute in R.

→ Decomposition helps in eliminating some of the problems of bad design such as redundancy, inconsistencies and anomalies.

There are two types of decomposition:

1. Lossy Decomposition
2. Loss less Join Decomposition.

Lossless Decomposition:

→ Decomposition is lossless if it is feasible to reconstruct relation R from decomposed tables using Joins.

→ The information will not lose from the relation when decomposed.

→ The join would result in the same original relation.

<EmpInfo>

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
001	Jacob	29	Alabama	Dpt 1	Operations
002	Henry	32	Alabama	Dpt 2	HR
003	Tom	22	Texas	Dpt 3	Finance

Decompose the above table into 2 tables.

<EmpDetails>

Emp_ID	Emp_Name	Emp_Age	Emp_Location
001	Jacob	29	Alabama
002	Henry	32	Alabama
003	Tom	22	Texas

* Now, Natural Join is applied on the above two tables.
* The result will be **<EmpInfo>**

<DeptDetails>

Dept_ID	Emp_ID	Dept_Name
Dpt 1	001	Operations
Dpt 2	002	HR
Dpt 3	003	Finance

Lassy Decomposition:

* When a relation is decomposed into two or more relational schemas, the loss of information is unavoidable when the original relation is retrieved.

* The table <EmpInfo>, decompose the table into two, <Emp Detail>

Emp_ID	Emp-Name	Emp_Age	Emp_Location
001	Jacob	29	Alabama
002	Henry	32	Alabama
003	Tom	22	Texas.

< Dept Details >

Dept_ID	Dept_Name
Dpt 1	Operations
Dpt 2	HR
Dpt 3	Finance.

* Now, we wont be able to join the above tables, since Emp_ID isn't part of DeptDetails relation.

* Therefore, the above relation has lossy decomposition.

Rules of Functional Dependencies:

1. Reflexive Rule :

In the reflexive rule, if Y is a subset of X, then X determines Y.

If $X \supseteq Y$ then $X \rightarrow Y$

Eg : $X = \{a, b, c, d, e\}$

$Y = \{a, b, c\}$

2. Augmentation Rule:

The augmentation is also called as a partial dependency. In augmentation, if x determines Y, then XZ determines YZ for any Z.

If $X \rightarrow Y$ then $XZ \rightarrow YZ$

Eg : For R(ABCD), if $A \rightarrow B$ then $AC \rightarrow BC$

3. Transitive Rule:

In the transitive rule, if x determines Y and Y determine Z, then x must also determine Z.

If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

4. Union Rule:

Union rule says, if x determines Y and x determines Z, then X must also determine Y and Z.

If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

5. Decomposition Rule:

Decomposition rule is also known as project rule. It is the reverse of union rule.

This rule says, if X determines Y and Z, then X determine Y and X determines Z separately.

If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

6. Pseudo Transitive Rule:

In Pseudo transitive rule, if X determines Y and YZ determines W, then XZ determines W.

If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$.

Types of Functional Dependencies:

1. Multivalued Dependency:

* Multivalued dependency occurs in the situation where there are multiple independent multivalued attributes in a single table.

* A multivalued dependency is a complete constraint between two sets of attributes in a relation.

PROPERTIES OF DECOMPOSITION:

1. Lossless Decomposition
2. Dependency Preservation
3. Lake of Data Redundancy

Dependency Preservation:

→ Dependency is an important constraint on the database.

→ Every dependency must be satisfied by at least one decomposed table.

→ If $\{A \rightarrow B\}$ holds, then two sets are functional dependent. And, it becomes more useful for checking the dependency easily if both sets in same relation.

→ This decomposition property can only be done by maintaining the functional dependency.

→ In this property, it allows to check the updates without computing the natural join of the database structure.

Lack of Data Redundancy:

→ The proper decomposition should not suffer from any data redundancy.

→ The lack of data redundancy property may be achieved by Normalization process.

→ It is also known as Repetition of Information.

FUNCTIONAL DEPENDENCY:

* Functional dependency is a relationship that exists when one attribute uniquely determines another attribute.

* If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as $X \rightarrow Y$, which specifies Y is functionally dependent on X.

* X is a determinant set and Y is a dependent attribute. Each value of X is associated with precisely one Y value.

Example:

Employee Number	Employee Name	Salary	City
1.	Dan	50000	New York
2.	Francis	38000	London
3.	Andrew	25000	Tokyo

* In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary.

* By this, we can say that the city, Employee Name and salary are functionally depended on Employee number.

* A functional dependency is denoted by an arrow →

Car Model	Maf_Year	Color
H001	2017	Metallic
H001	2017	Green
H010	2015	Blue
H033	2012	Gray

* In this example, maf_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:

car_model → maf_year

car_model → colour.

2. 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.

* Consider a table with two columns emp_id and emp_name. {Emp_id, emp_name} → emp_id is a trivial functional dependency as emp_id is a subset of {emp_id, emp_name}.

3. 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.

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

{Company} → {CEO}, but CEO is not a subset of Company and hence it is non-trivial functional dependency.

Advantages of Functional Dependency:

* Functional Dependency avoids data redundancy. Therefore same data do not repeat at multiple locations in that database.
* It helps to maintain the quality of data in the database.
* It helps to define meanings and constraints of databases.
* It helps to identify bad designs
* It helps to find the facts regarding the database design.

CLOSURE OF change DEPENDENCY:

* 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 $\{F\}^+$.
* There are 3 steps to calculate closure of functional dependency.

  Step 1: Add the attributes which are present on Left Hand Side in the original FD.

  Step 2: Now, add the attributes present on RHS of the FD.

Step 3: With the help of attributes present on RHS, check the other attributes that can be derived from the other given FD.

Repeat this process until all the possible attributes which can be derived are added in the closure.

Eg: Student_details having Roll_No, Name, Marks, Location as the attributes and having 2 FD.

FD1 : Roll_No → Name, Marks

FD2 : Name → Marks, Location

Step 1 : $\{Roll\_no\}^+ = \{Roll\_No\}$

Step 2 : $\{Roll\_no\}^+ = \{Roll\_No, Marks\}$

Step 3 : $\{Roll\_no\}^+ = \{Roll\_No, Marks, Name, Location\}$.

Closure for "Name"

Step 1 : $\{Name\}^+ = \{Name\}$

Step 2 : $\{Name\}^+ = \{Name, Marks, Location\}$

Step 3 : $\{Name\}^+ = \{Name, Marks, Location\}$

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

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

CLOSURE OF AN ATTRIBUTE SET:

* Given a set of attributes A and a set of FDs F, closure of A under F is the set of all attributes implied by A.

Example:

$R = (A, B, C, G, H, I)$

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

$(AG)^+ = ?$

Step 1 : AG

2 : $ABCG$ ($A \rightarrow C$ and $A \rightarrow B$)

3 : $ABCGH$ ($CG \rightarrow H$ and $CG \subseteq AGBC$)

4 : $ABCGHI$ ($CG \rightarrow I$ and $CG \subseteq AGBCH$)

- AG is a super key
- $(A^+) = BC$, $(G^+) = G$

Uses:

* Determining if $A \rightarrow B$ is a valid FD
* Determining superkeys and candidate keys
* Can be used to compute F^+

MINIMAL FUNCTIONAL DEPENDENCY SET:

* It is also called as canonical cover, it is a simplified and reduced version of the given set of FDs.

* Since it is a reduced version, it is also called as Irreducible set.

Steps To Find Canonical Cover:

Step 1: Write the given set of FD in such a way that each FD contains exactly one attribute on its right side

Eg: FD $X \rightarrow YZ$ will be written as

$X \rightarrow Y$

$X \rightarrow Z$

Step 2: Consider each FD one by one from the set. Determine whether it is essential or non-essential.

To determine whether a FD is essential or not, compute the closure of its left side.

Once by considering that the particular FD is present or not in the set

The following two cases are possible

Case 1: Results Come Out to be Same

- It means that the presence or absence of that FD does not create any difference.
- It is non-essential, eliminate that FD from the set.

Case 2: Results Come Out to be Different

- It means that the presence or absence of that FD creates a difference.
- It is essential, do not eliminate it

Step 3: Consider the newly obtained set of FD.
Check if there is any FD that contains more than one attribute on its left side.

Then following two cases are possible

Case 1: No

- There exists no FD containing more than one attribute on its LHS
- Then, the set obtained in step 2 is canonical cover

Case 2: Yes

- There exists at least one functional dependency containing more than one attribute on its LHS.
- Then, consider all such FD one by one.
- Check if their left side can be reduced.

Use the following steps to check

- Consider a FD, compute the closure of all the possible subsets of the left side of that FD.
- If any of the subsets produce the same closure result as produced by the entire left side, then replace the left side with that subset.

After this step is complete, the set obtained is the canonical cover.

Example 1

R(w, x, y, z) FD: $x \to w$, $wz \to xy$, $y \to wxz$

Step 1: $x \to w$
$wz \to x$
$wz \to y$
$y \to w$
$y \to x$
$y \to z$

Step 2

For $x \to w$

Considering $x \to w$,
$x^+ = \{x, w\}$

Ignoring $x \to w$,
$x^+ = \{x\}$

Results are different

$x \to w$ is essential

For $wz \to x$

Considering $wz \to x$,
$(wz)^+ = \{w, x, y, z\}$.

Ignoring $wz \to x$,
$(wz)^+ = \{w, x, y, z\}$.

Results are same

$wz \to x$ can be eliminated.

Now our FD is

$x \to w$
$wz \to y$
$y \to w$
$y \to x$
$y \to z$

For $wz \to y$:

Considering $wz \to y$,

$(wz)^+ = \{w, x, y, z\}$

Ignoring $wz \to y$,

$(wz)^+ = \{w, z\}$

Results are different

$wz \to y$ is essential.

For $y \to w$:

Considering $y \to w$,

$y^+ = \{x, y, z, w\}$

Ignoring $y \to w$,

$y^+ = \{w, x, y, z\}$

Results are same.

$y \to w$ can be eliminated.

Now our FD is,

$x \to w$

$wz \to y$

$y \to x$

$y \to z$

For $y \to x$

Considering $y \to x$,

$y^+ = \{w, x, y, z\}$

Ignoring $y \to x$,

$y^+ = \{y, z\}$

Results are different.

$y \to x$ is essential

For $y \to z$:

Considering $y \to z$,

$y^+ = \{w, x, y, z\}$

Ignoring $y \to z$,

$y^+ = \{w, x, y\}$

Results are different

$y \to z$ is essential.

Our essential FD

$x \to w$

$wz \to y$

$y \to x$

$y \to z$

Step 3

$wz \to y$ contains more than one attribute on its left side.

Considering $wz \to y$,

$(wz)^+ = \{w, z, x, y\}$

Consider all possible subset of wz

$w^+ = w$

$z^+ = z$

None of subset have same closure

The Canonical Cover is

$x \to w$ $\quad$ $y \to x$

$wz \to y$ $\quad$ $y \to z$

Example 2 :

FD : $A \rightarrow BC$

$B \rightarrow C$

$A \rightarrow B$

$AB \rightarrow C$

1. FD $A \rightarrow BC$

$A \rightarrow B$

with same set of attribute, these two can be combined to get $A \rightarrow BC$

Now revised FD

$A \rightarrow BC$

$B \rightarrow C$

$AB \rightarrow C$

2. There is an extraneous attribute in $AB \rightarrow C$ because even after removing $AB \rightarrow C$ from the set F, we get the same closure.

Now revised FD

$A \rightarrow BC$

$B \rightarrow C$

3. C is an extraneous attribute in $A \rightarrow BC$ by transitivity

~~$A \rightarrow B$~~ , ~~$A \rightarrow C$~~

$A \rightarrow B$

$B \rightarrow C$

4. Hence req Canonical cover is

$A \rightarrow B$

$B \rightarrow C$

NORMALIZATION:

- Normalization is the process of organizing the data in the database.
- It 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.
- It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.

Example:

Emp_id	Emp_Name	Emp_Address	Emp_Dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

Update Anomaly:

In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the ~~address of Rick then we have to update~~ same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert Anomaly:

Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

Delete Anomaly:

Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

To overcome these anomalies we need to normalize the data.

Types of Normal Forms:

There are four types of normal forms

1. 1NF
2. 2NF
3. 3NF
4. BCNF

FIRST NORMAL FORM:

- 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.
- 1NF disallows the multi-valued attribute, composite attribute and their combinations.

Example:

Emp_ID	Emp_Name	Emp_Phone	Emp_State
14	John	7272826385 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab

- Relation Employee is not in 1NF because of multi valued attribute EMP_PHONE.

Emp_ID	Emp_Name	Emp_Phone	Emp_State
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab

SECOND NORMAL FORM:

- In the 2NF, relational must be in 1NF
- All non-key attributes are fully FD on the primary key

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_ID	Teacher_Age
25	30
47	35
83	38

Teacher_ID	Subject
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

THIRD NORMAL FORM:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.
- An attribute that is not part of any candidate key is known as non-prime attribute.

Example:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_street
1001	John	282005	TN	Chennai	Central
1002	Ajeet	222008	MH	Mumbai	Meadows
1006	Lora	282007	Delhi	Delhi	Greenways
1101	Lilly	292008	WB	Kolkata	Babar
1201	Steve	222999	KA	Banglore	Canal

Candidate Key: emp_id

Non prime attribute: all attribute except emp-id

Here, emp_state, emp_city & emp_street dependent on emp_zip and emp_zip dependent on emp-id that makes non-prime attributes transitively dependent on super key. It violates the rule.

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

emp_zip	emp_state	emp_city	emp_street
282005	TN	Chennai	Central
222008	MH	Mumbai	Meadows
282007	Delhi	Delhi	Greenways
292008	WB	Kolkata	Babar
222999	KA	Banglore	Canal

BOYCE CODD NORMAL FORM:

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every FD $X \rightarrow Y$, 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:

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 FD 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 3 tables.

Emp - Country Table

Emp - ID	Emp_ Country
264	India
364	UK

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
264	Designing
264	Testing
364	Stores
364	Developing

FD : Emp_ ID → Emp_ Country
Emp_Dept → {Dept_Type, Emp_Dept_No}

Candidate Keys :
1. Emp_ID
2. Emp_Dept
3. Emp_ID, Emp_Dept.

Now, this is in BCNF because left side part of both the functional dependencies is a Key.

DENORMALIZATION :

- Denormalization is a database optimization technique in which we add redundant data to one or more tables.
- This can help us to avoid costly joins in a relational database.
- It does not means not doing normalization. It is an optimization technique that is applied after doing normalization.

- In a traditional normalized database, we store data in separate logical tables and attempt to minimize redundant data.
- For eg, in a normalized database, we have a Course table and Teachers table.
- Each entry in Courses would store the teacher ID for a course but not the teacher Name.
- When we need to retrieve a list of all courses with the Teacher Name, we would do a join b/w these two tables.
- If a teacher changes his or her name, we have to update only name in one place, if tables are large we may spend long time on the table.
- In denormalization, in order to get the efficiency advantage of fewer joins, redundancy is permitted.

Advantage:

1. Retrieving data is faster since we do fewer joins
2. Queries to retrieve can be simpler, since we need to look at fewer table.

Disadvantage:

1. Update and insert are more expensive
2. Data may be inconsistent
3. Data redundancy necessities more storage.

PRACTICAL DATABASE DESIGN:

→ Most database users do not use a query language.

→ An application program acts as the intermediary between users and the database.

→ Applications split into

- front end
- middle layer
- back end

→ Front-end : user interface

- Forms
- Graphical user interfaces
- Many interfaces are Web-based.

WEB INTERFACE:

→ Web browsers have become truly the standard user interface to databases.

→ Enable large numbers of users to access databases from anywhere.

→ Avoid the need for downloading / installing specialized code, while providing a good graphical user interface.

- Javascript, Flash and other scripting languages run in browser, but are downloaded transparently.

→ Example : banks, airline and rental car reservations, university course registration and grading.

Application Architecture:

- Application layers
  - model - view - controller (MVC) architecture
    - model : business logic
    - view : presentation of data, depends on display device
    - controller : receives events, executes action and returns a view to the user.
  - business - logic layer
    - provides high level view of data and actions on data
    - hides details of data storage schema
  - data access layer
    - interfaces between business logic layer and the underlying database.
    - provides mapping from object model of business layer to relational model of database.

ALTERNATIVE DESIGN TECHNIQUES:

→ Database widely support encryption

→ Different levels of encryption:

- disk block
  - every disk block encrypted using key available in database system software.
  - even if attacker gets access to database data, decryption cannot be done without access to the key.

- Entire relations or specific attributes of relations
  - non-sensitive relations, or non-sensitive attributes of relations need not be encrypted.
  - however, attributes involved in primary / foreign key constraints cannot be encrypted.

- → Storage of encryption or decryption keys
  - typically, single master key used to protect multiple encryption decryption keys stored in database.