MODULE-3

SCHEMA REFINEMENT AND NORMAL FORMS

FUNCTIONAL DEPENDENCIES

- 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 → Y

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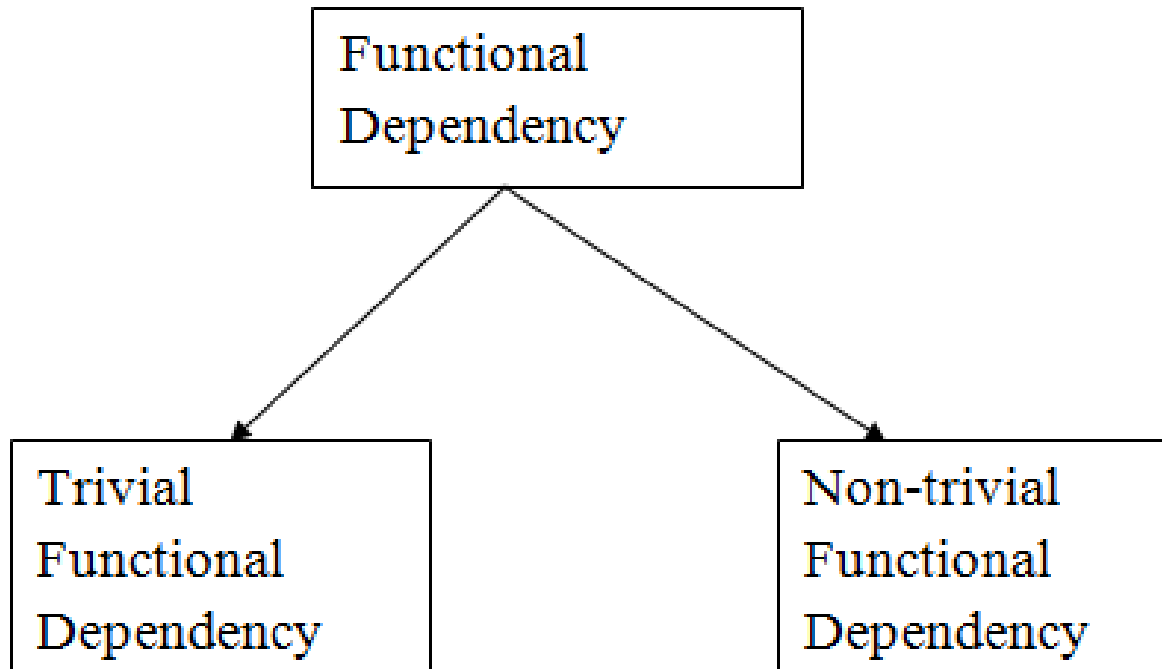
EXAMPLE

- 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 → Emp_Name

- We can say that Emp_Name is functionally dependent on Emp_Id.

Types of Functional dependency

```
┌─────────────────┐
│ Functional      │
│ Dependency      │
└─────────────────┘
      ╱        ╲
     ╱          ╲
    ▼            ▼
┌──────────┐  ┌──────────────┐
│ Trivial  │  │ Non-trivial  │
│ Functional│ │ Functional   │
│ Dependency│ │ Dependency   │
└──────────┘  └──────────────┘
```

Trivial functional dependency

- A → B has trivial functional dependency if B is a subset of A.

- The following dependencies are also trivial like:

  A → A, B → B

- **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.

Non-trivial functional 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

Completely non-trivial functional dependency

- ⦿ If XàY and X∩Y=Φ(null) then it is called completely non-trivial functional dependency.

Prime and non-prime attributes

- ⦿ Attributes which are parts of any candidate key of relation are called as prime attribute, others are non-prime attributes.

Reasoning about functional dependencies

Armstrong Axioms:

- ⦿ Armstrong axioms define the set of rules for reasoning about functional dependencies and also to infer all the functional dependencies on a relational database.

Various axioms rules or inference rules:

- ⊙ Primary axioms:

Rule 1	**Reflexivity**
	If A is a set of attributes and B is a subset of A, then A holds B. { A → B }
Rule 2	**Augmentation**
	If A hold B and C is a set of attributes, then AC holds BC. {AC → BC}
	It means that attribute in dependencies does not change the basic dependencies.
Rule 3	**Transitivity**
	If A holds B and B holds C, then A holds C.
	If {A → B} and {B → C}, then {A → C}
	A holds B {A → B} means that A functionally determines B.

Secondary or derived axioms:

Rule 1	Union
	If A holds B and A holds C, then A holds BC.
	If$\{A \rightarrow B\}$ and $\{A \rightarrow C\}$, then $\{A \rightarrow BC\}$
Rule 2	Decomposition
	If A holds BC and A holds B, then A holds C.
	If$\{A \rightarrow BC\}$ and $\{A \rightarrow B\}$, then $\{A \rightarrow C\}$
Rule 3	Pseudo Transitivity
	If A holds B and BC holds D, then AC holds D.
	If$\{A \rightarrow B\}$ and $\{BC \rightarrow D\}$, then $\{AC \rightarrow D\}$

Attribute closure

- Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.

NOTE:

- To find attribute closure of an attribute set-

- 1) add elements of attribute set to the result set.

- 2) recursively add elements to the result set which can be functionally determined from the elements of result set.

Input : Let F be a set of FDs for relation R.

Steps:
        1. $X^+ = X$ //initialize X^+ to X

        2. For each FD : Y -> Z in F Do
            If $Y \subseteq X^+$ Then   //If Y is contained in X^+
            $X^+ = X^+ \cup Z$ //add Z to X^+
            End If
            End For
        3. Return X^+ //Return closure of X

Output : Closure X^+ of X under F

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.

NORMAL FORMS

1) First normal form (1NF)

2) Second normal form (2NF)

3) Third normal form (3NF)

4) Boyce coded normal form (BCNF)

5) Forth normal form (4NF)

6) Fifth normal form (5NF)

7) Sixth normal form (6NF)

8) Domain key normal form.

Normal Forms

1st Normal Form	No repeating data groups
2nd Normal Form	No partial key dependency
3rd Normal Form	No transitive dependency
Boyce-Codd Normal Form	Reduce keys dependency
4th Normal Form	No multi-valued dependency
5th Normal Form	No join dependency

$$1NF \supset 2NF \supset 3NF \supset BCNF \supset 4NF \supset 5NF$$

Property	3NF	BCNF	4NF
Eliminates redundancy due to FD's	Most	Yes	Yes
Eliminates redundancy due to MVD's	No	No	Yes
Preserves FD's	Yes	Maybe	Maybe
Preserves MVD's	Maybe	Maybe	Maybe

Properties of normal forms and their decompositions

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.

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

SECOND NORMAL FORM (2NF)

- o In the 2NF, relational must be in 1NF.

- o In the second normal form, all non-key attributes are fully functional dependent on the primary key

Example: 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_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)

- o A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- o 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- o 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.

1. X is a super key.

2. 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

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:

1. EMP_ID → EMP_COUNTRY
2. 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

Functional dependencies:

1. EMP_ID $\rightarrow$ EMP_COUNTRY
2. EMP_DEPT $\rightarrow$ {DEPT_TYPE, EMP_DEPT_NO}

FOURTH NORMAL FORM (4NF)

- o A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.

- o For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

- The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

- In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

- So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

FIFTH NORMAL FORM (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.

- 5NF is also known as Project-join normal form (PJ/NF).

Example

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

- In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

- Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

- So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

RELATIONAL DECOMPOSITION

- o When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.

- o In a database, it breaks the table into multiple tables.

- o If the relation has no proper decomposition, then it may lead to problems like loss of information.

- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

Types of Decomposition

```
        ┌─────────────────────┐
        │    Decomposition     │
        └─────────────────────┘
           ╱              ╲
          ╱                ╲
 ┌──────────────┐    ┌──────────────┐
 │ Lossless     │    │ Dependency   │
 │ Decomposition│    │ Preserving   │
 └──────────────┘    └──────────────┘
```
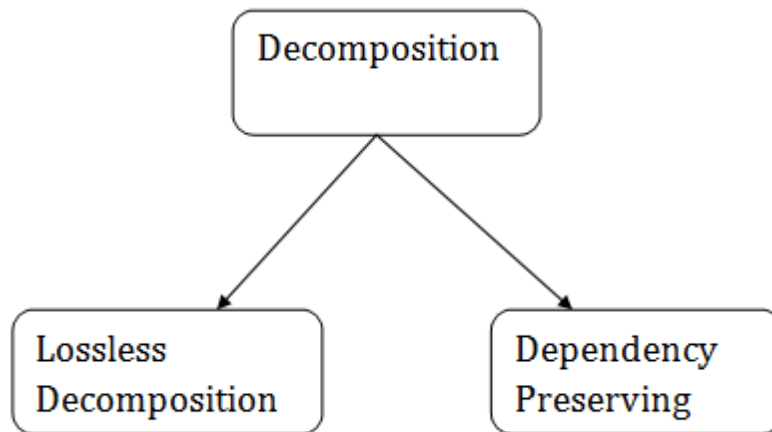
Lossless Decomposition

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.

- The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.

- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

Example:

EMPLOYEE_DEPARTMENT table:

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

The above relation is decomposed into two relations EMPLOYEE and DEPARTMENT

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY
22	Denim	28	Mumbai
33	Alina	25	Delhi
46	Stephan	30	Bangalore
52	Katherine	36	Mumbai
60	Jack	40	Noida

DEPARTMENT table

DEPT_ID	EMP_ID	DEPT_NAME
827	22	Sales
438	33	Marketing
869	46	Finance
575	52	Production
678	60	Testing

Now, when these two relations are joined on the common column "EMP_ID", then the resultant relation will look like:

Employee ⋈ Department

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

Hence, the decomposition is Lossless join decomposition.

Dependency Preserving

- It is an important constraint of the database.

- In the dependency preservation, at least one decomposed table must satisfy every dependency.

- If a relation R is decomposed into relation R1 and R2, then the dependencies of R either must be a part of R1 or R2 or must be derivable from the combination of functional dependencies of R1 and R2.

- For example, suppose there is a relation R (A, B, C, D) with functional dependency set (A->BC). The relational R is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of relation R1(ABC).