# ADITYA ENGINEERING COLLEGE (A)

# DBMS

# Schema Refinement & Normalization

By

**Dr. S Rama Sree**
Professor in CSE & Dean(Academics)
Aditya Engineering College(A)
Surampalem.

# Contents

- Purpose of Normalization or schema refinement
- Concept of functional dependency
- Normal forms based on functional dependency(1NF, 2NF and 3 NF)
- Concept of surrogate key
- Boyce-codd normal form(BCNF)
- Lossless join and dependency preserving decomposition
- Fourth normal form(4NF)
- Fifth Normal Form (5NF).

# Problems Caused by Redundancy

Storing the same information redundantly, that is, in more than one place within a database, can lead to several problems

- **Redundant Storage:** Some information is stored repeatedly
- **Update Anomalies:** If one copy of such repeated data is updated, an inconsistency is created unless all copies are similarly updated.
- **Insertion Anomalies:** It may not be possible to store certain information unless some other, unrelated, information is stored as well.
- **Deletion Anomalies:** It may not be possible to delete certain information without losing some other, unrelated, information as well.

# Definitions

**Hourly Emps(<u>ssn</u>, name, lot, rating, hourly wages, hours _worked) SNLRWH**

| S | N | L | R | W | H |
|---|---|---|---|---|---|
| 123-22-3666 | Attishoo | 48 | 8 | 10 | 40 |
| 231-31-5368 | Smiley | 22 | 8 | 10 | 30 |
| 131-24-3650 | Smethurst | 35 | 5 | 7 | 30 |
| 434-26-3751 | Guldu | 35 | 5 | 7 | 32 |
| 612-67-4134 | Madayan | 35 | 8 | 10 | 40 |

- Suppose that the hourly_wages attribute is determined by the Rating attribute. That is, for a given Rating value, there is only one permissible hourly_wages value.
- This redundancy has the same negative consequences as before:
- **Redundant Storage**: The rating value 8 corresponds to the hourly wage 10, and this association is repeated three times.
- **Update Anomalies**: The hourly_wages in the first tuple could be updated without making a similar change in the second tuple
- **Insertion Anomalies**: We cannot insert a tuple for an employee unless we know the hourly wage for the employee's rating value.
- **Deletion Anomalies**: If we delete all tuples with a given rating value (e.g., we delete the tuples for Smethurst and Guldu) we loose the association between that Rating value and its hourly_wage value.

# Decompositions

- Redundancy arises when a relational schema forces an association between attributes that is not natural.
- The essential idea is that many problems arising from redundancy can be addressed by replacing a relation 'with a collection of smaller' relations.
- A decomposition of a relation schema R consists of replacing the relation schema by two9or more) relation schemas that each contain a subset of the attributes of R and together include all attributes in R.
- We can decompose Hourly_Emps into two relations:

Hourly_Emps2(<u>ssn</u>, name, lot, rating, hours_worked)

Wages(<u>rating</u>, hourly_wages)

| S | N | L | R | H |
|---|---|---|---|---|
| 123-22-3666 | Attishoo | 48 | 8 | 40 |
| 231-31-5368 | Smiley | 22 | 8 | 30 |
| 131-24-3650 | Smethurst | 35 | 5 | 30 |
| 434-26-3751 | Guldu | 35 | 5 | 32 |
| 612-67-4134 | Madayan | 35 | 8 | 40 |

| R | W |
|---|---|
| 8 | 10 |
| 5 | 7 |

- After decomposition, we can easily record the hourly wage for any rating simply by adding a tuple to Wages relation, even if no employee with that rating appears in the current instance of Hourly_Emps.
- Changing the wage associated with a rating involves updating a single Wages tuple. This is more efficient than updating several tuples.
- From a performance standpoint, queries over the original relation may require us to join the decomposed relations. If such queries are common, the performance penalty of decomposing the relation may not be acceptable. In this case, we may choose to live with some of the problems of redundancy and not decompose the relation.

# Functional Dependency

- A functional dependency (FD) is a kind of Integrity Constraint(IC) that generalizes the concept of a key.

Let R be a relation schema and let X and Y be nonempty sets of attributes in R. We say that an instance r of R satisfies the FD $X \rightarrow Y$ (read as X functionally determines Y) if the following holds for every pair of tuples $t_1$ and $t_2$ in r.

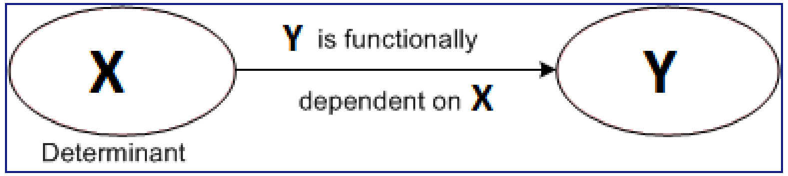
If $t_1.X = t_2.X$, then $t_1.Y = t_2.Y$

- An attribute is functionally dependent on another, If we can use the value of one attribute to determine the value of another.

y is functionally dependent on x

or

x functionally determines y




Functional dependency between X and Y

Ex:-ssn → Employee_name

ssn can be used to uniquely identify an employee_name

# Functional Dependency

| FID | FNAME | CID | CNAME |
|---|---|---|---|
| 3102 | SHARMA | C01 | DBMS |
| 2352 | PHANI | C02 | CO |
| 1201 | HEMA | C03 | PPL |
| 353 | SASTRY | C04 | JAVA |

FID → FNAME

CID → CNAME

# Closure of a set of Functional Dependencies

- A **Closure** is a **set of all FDs** implied by a given set F of **FDs.** This **closure set** is denoted by $F^+$.
- We can infer or compute the closure of given set F of FDs by following 3 rules called **Armstrong Axioms** that can be applied repeatedly to infer all FDs implied by a set F of FDs.
- Let X, Y & Z denote set of attributes over a relation R

<u>Reflexivity</u>: If $X \subseteq Y$, then $Y \to X$ or If $X \supseteq Y$, then $X \to Y$

<u>Augmentation</u>: If $X \to Y$, then $XZ \to YZ$ for any Z

<u>Transitivity</u>: If $X \to Y$ and $Y \to Z$, then $X \to Z$

It is convenient to use some additional rules while reasoning about $F^+$

<u>Union</u>: If $X \to Y$ and $X \to Z$, then $X \to YZ$

<u>Decomposition</u>: If $X \to YZ$, then $X \to Y$ and $X \to Z$

# Closure of a set of Functional Dependencies

**Example: Compute the closure F + for a given R and a set of FDs**

R = (A, B, C, G, H, I), and the set of FDS (A → B, A → C, CG → H, CG → I, B → H )

Step 1:

A → B and B → H

∴ By transitivity rule.

A → H

Step2:

CG → H and CG → I

∴ By union rule

CG → HI

Step 3:

A → C

∴ By augmentation rule

Step 4:

AG → CG and CG → I

∴ By transitivity rule

AG → I

Therefore, finally F+ includes
(A → H, CG → HI, AG →I, A →BC)

# Attribute Closure

**Let α be a set of attributes. We call the set of all attributes functionally determined by α under a set F of FDs i.e the closure of α under F denoted as α+ . To compute α+, the algorithm used is given below. The algorithm is used to check if the given closure $\alpha^+$ is a superkey or not. The closure $\alpha^+$ is a superkey if the result includes all the attributes of R.**

Input:-Set of Functional dependencies(F) and set of attributes(α)
Output:-stored in a variable result.

Algorithm to compute $\alpha^+$

```
result := α;
while (changes to result) do
    for each β → γ in F do
        begin
            if β ⊆ result then  result := result ∪ γ
        end;
```

# Attribute Closure

Example: Given the following, check if (AG)+ is a super key or not.

Consider the following:

$R = (A, B, C, G, H, I)$
$F = \{A \to B, CG \to H, A \to C, CG \to I, B \to H\}$

$(AG)^+$

| | |
|---|---|
| AG | |
| ABG | $A \to B$ |
| ABCG | $A \to C$ |
| ABCGH | $CG \to H$ |
| ABCGHI | $CG \to I$ |

1) result = AG
2) $A \to B$ causes
$A \subseteq$ result
$A \subseteq AG$
$\therefore$ result $= AG \cup B$
$= ABG$

(AG)+ is a super key as result variable contains all the attributes in R

- **GATE Question: Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, N} and the set of functional dependencies {{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L}, K -> {M}, L -> {N} on R. What is the key for R? (GATE-CS-2014)**
  A. {E, F}
  B. {E, F, H}
  C. {E, F, H, K, L}
  D. {E}
- **Answer:** Finding attribute closure of all given options, we get:
  {E,F}+ = {EFGIJ}
  {E,F,H}+ = {EFHGIJKLMN}
  {E,F,H,K,L}+ = {{EFHGIJKLMN}
  {E}+ = {E}
  {EFH}+ and {EFHKL}+ results in set of all attributes, but EFH is minimal. So it will be candidate key. So correct option is (B).

# Normalization

- Normalization is a process of making relations by decomposing them into smaller relations to

1) Reduce redundancy
2) Eliminate update anomaly
3) Eliminate delete anomaly
4) Eliminate insert anomaly

- Given a relation schema, we need to decide whether it is a good design or we need to decompose it into smaller relations. To provide such guidance, several normal forms have been proposed.
- If a relation schema is in one of these normal forms, we know that certain kinds of problems cannot arise.
- The normal forms based on FDs are first normal form (1NF), second normal form (2NF), third normal form (3NF), and Boyce-Codd normal form (BCNF).

# Normal Forms

**First Normal Form :-A relation is in 1NF if every field contains only atomic values i.e no lists or sets.**

Students

| FirstName | LastName | Knowledge |
|---|---|---|
| Thomas | Mueller | Java, C++, PHP |
| Ursula | Meier | PHP, Java |
| Igor | Mueller | C++, Java |

Startsituation

Result after Normalisation

Students

| FirstName | LastName | Knowledge |
|---|---|---|
| Thomas | Mueller | C++ |
| Thomas | Mueller | PHP |
| Thomas | Mueller | Java |
| Ursula | Meier | Java |
| Ursula | Meier | PHP |
| Igor | Mueller | Java |
| Igor | Mueller | C++ |

Act

# First Normal Form

| SUPPLIER_NO | STATUS | CITY | PART_NO | QUANTITY |
|---|---|---|---|---|
| S1 | 20 | BOMBAY | P1 | 300 |
| S1 | 20 | BOMBAY | P2 | 200 |
| S1 | 20 | BOMBAY | P3 | 400 |
| S2 | 10 | CHENNAI | P1 | 300 |
| S2 | 10 | CHENNAI | P2 | 200 |
| S3 | 10 | CHENNAI | P3 | 400 |
| S4 | 20 | BOMBAY | P2 | 200 |
| S4 | 20 | BOMBAY | P1 | 300 |

FIRST(SUPPLIER_NO,STATUS,CITY,PART_NO,QUANTITY)
PRIMARY KEY(SUPPLIER_NO,PART_NO)
FDs:- SUPPLIER_NO→CITY
CITY→STATUS
PART_NO→QUANTITY

# First Normal Form

- Redundancy exists
- Insert Anomaly:-we cannot insert supplier information until that supplier supplies atleast one part.
- Delete Anomaly:- If we delete a tuple in FIRST with supplier_no s3 and part_no p3, we lose the information that s3 is located in chennai.
- Update Anomaly:- If supplier s1 moves from bombay to ahmedabad, to find every tuple s1 and bombay and to replace it.
- **Table FIRST contains too much information packed together. Its better to Unpack the data, to place shipment information in one table and supplier information in another table. This can be done using the 2NF.**

# Second Normal Form

- **A relation is in 2NF if and only if it is in 1NF and every nonkey attribute is dependent on the primary key.(i.e no partial dependency).**
- It means that no non-prime attribute is dependent on subset of any candidate key of the table. The Primary key attributes are Supplier_no,Part_no and Nonkey attributes are status,city,quantity.
- In the FIRST table, there is partial dependency. i.e SUPPLIER_NO→CITY. So, now we need to remove the partial dependency by decomposing into 2 tables; SECOND, SHIPMENTS.

| SUPPLIER_NO | STATUS | CITY |
|---|---|---|
| S1 | 20 | BOMBAY |
| S2 | 10 | CHENNAI |
| S3 | 10 | CHENNAI |
| S4 | 20 | BOMBAY |

SECOND(SUPPLIER_NO,STATUS,CITY)

PRIMARY KEY(SUPPLIER_NO)

**FDs:** SUPPLIER_NO→CITY

CITY→STATUS

SUPPLIER_NO, PART_NO→QUANTITY

| SUPPLIER_NO | PART_NO | QUANTITY |
|---|---|---|
| S1 | P1 | 300 |
| S1 | P2 | 200 |
| S1 | P3 | 400 |
| S2 | P1 | 300 |
| S2 | P2 | 200 |
| S3 | P3 | 400 |
| S4 | P2 | 200 |
| S4 | P1 | 300 |

SHIPMENTS(SUPPLIER_NO,PART_NO,QUANTITY)

PRIMARY KEY(SUPPLIER_NO,PART_NO)

FOREIGN KEY(SUPPLIER_NO) REFERENCES SECOND(SUPPLIER_NO)

# Second Normal Form

- Redundancy still exists
- Insert Anomaly:-we can insert information that s5 is located in Delhi,even s5 doesn’t currently supply any parts.
- Delete Anomaly:-we can delete s3p3 in shipments table, we don't lose information that s3 is located in chennai.
- Update Anomaly:-we can change the city for s1 from bombay to ahmedabad by changing it once in SECOND table.
- If we want to change the status of suppliers who resides in Chennai from 10 to 15, we change in one place and forget to change in other, it leads to inconsistency.
- **Although Second Normal Form (2NF) relations have less redundancy than those in 1NF, they may still suffer from update anomalies.**
- **If we update only one tuple and not the other, the database would be in an inconsistent state. This update anomaly is caused by a transitive dependency. We need to remove such dependencies by progressing to Third Normal Form (3NF).**

# Third Normal Form

- **A relation is in 3NF if & only if it is in 2NF and every non key attribute is non transitively dependent on the primary key.(No transitive dependency).**
- In the table SECOND, there exists a transitive dependency, which is to be removed.

  Supplier_no→City

  City→Status

  Then Supplier_no→status (transitive dependency)
- To convert to 3NF, unpack/decompose the table SECOND into supplier information in one table & city information in another and the SHIPMENTS table is as it is.
- Insert Anomaly:-we cannot insert a supplier in bangalore into the SC table who have status of 50 until some supplier is actually located in that city(i.e Bangalore status 50 is inserted into CS table) .
- Delete Anomaly:-we can delete supplier s5 without losing status for delhi is 30.
- Update Anomaly:-we need to change the status for bombay from 20 to 40 we need not find every tuple for bombay for changing it.

# Third Normal Form

| SUPPLIER_NO | CITY |
|---|---|
| S1 | BOMBAY |
| S2 | CHENNAI |
| S3 | CHENNAI |
| S4 | BOMBAY |
| S5 | DELHI |

SC(SUPPLIER_NO,CITY)
PRIMARY KEY(SUPPLIER_NO)
FOREIGN KEY(CITY) REFERENCES CS(CITY)
**FDs:** SUPPLIER_NO→CITY

| CITY | STATUS |
|---|---|
| BOMBAY | 20 |
| CHENNAI | 10 |
| DELHI | 30 |
| BANGALORE | 50 |

CS(CITY,STATUS)
PRIMARY KEY(CITY)
FDs: CITY→STATUS

| SUPPLIER_NO | PART_NO | QUANTITY |
|---|---|---|
| S1 | P1 | 300 |
| S1 | P2 | 200 |
| S1 | P3 | 400 |
| S2 | P1 | 300 |
| S2 | P2 | 200 |
| S3 | P3 | 400 |
| S4 | P2 | 200 |
| S4 | P1 | 300 |

SHIPMENTS(SUPPLIER_NO,PART_NO,QUANTITY)
PRIMARY KEY(SUPPLIER_NO,PART_NO)
FDs: SUPPLIER_NO, PART_NO→QUANTITY

# Boyce-Codd normal form

- Despite these additional constraints in 3NF, dependencies can still exist that will cause redundancy to be present in 3NF relations. This weakness in 3NF, resulted in the presentation of a stronger normal form called Boyce–Codd Normal Form
- BCNF was developed in 1974 by Raymond F. Boyce and Edgar F. Codd
- **A relation is in BCNF, if and only if, the relation is in 3NF and every determinant is a candidate key.**
- The left side of the Functional Dependency(FD) is called the determinant, and the right side is the dependent
- In FD, Supplier_no→city; the Supplier_no is called determinant and city is called dependent.

- A relation is in BCNF iff, the relation is in 3NF and X is superkey for every functional dependency (FD) X → Y in a given relation i.e for every FD, LHS is super key.
- To test whether a relation is in BCNF, we identify all the determinants and check if they are candidate keys.

| SUPPLIER_NO | CITY | PART_NO | QUANTITY |
|---|---|---|---|
| S1 | BOMBAY | P1 | 300 |
| S1 | BOMBAY | P2 | 200 |
| S2 | CHENNAI | P1 | 300 |
| S2 | CHENNAI | P3 | 400 |
| S3 | LUCKNOW | P4 | 500 |

Supp(supplier_no,city,part_no,quantity) FDs: Supplier_no→city, part_no → quantity

# Boyce-codd normal form

- The table **Supp** is not in BCNF because in the FD1, the LHS Supplier_no is not a key and in FD2, the LHS part_no is also not a key. To convert Supp table into BCNF, split it into 2 tables SUPPL & SHIPMENT.

NOTE: To find if an attribute is a key or not, we need to find the attribute closure. Refer slide 12.

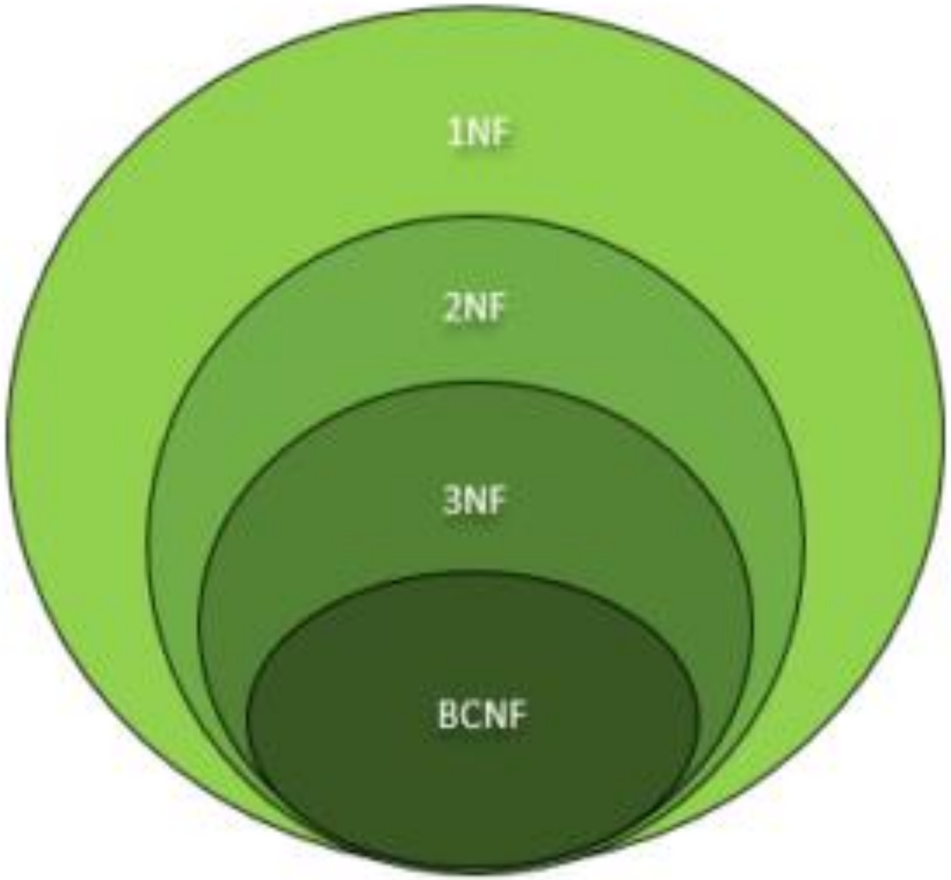
| SUPPLIER_NO | CITY |
|---|---|
| S1 | BOMBAY |
| S2 | CHENNAI |
| S3 | LUCKNOW |

SUPPL(SUPPLIER_NO,CITY)
PRIMARY KEY(SUPPLIER_NO)
FD1: Supplier_no→city

| SUPPLIER_NO | PART_NO | QUANTITY |
|---|---|---|
| S1 | P1 | 300 |
| S1 | P2 | 200 |
| S2 | P1 | 300 |
| S2 | P3 | 400 |
| S3 | P4 | 500 |

SHIPMENT(SUPPLIER_NO,PART_NO,QUANTITY)
PRIMARY KEY(SUPPLIER_NO,PART_NO)
FOREIGN KEY(SUPPLIER_NO) REFERENCES SUPPL(SUPPLIER_NO)
FD2: supplier_no,part_no→quantity

Considering the 2 tables, and the FDs, we can say that supplier_no in FD1 is key and supplier_no,part_no is also a key. Therefore, we can say that these relations are in BCNF.




*The Normal Form Hierarchy*

**Example: Consider a relation R with attributes (Student, Teacher, Subject).**

FDs: { (student, Teacher) -> subject
(Teacher, subject) -> student
Teacher -> subject }

| Student | Teacher | Subject |
|---|---|---|
| Jhansi | P.Naresh | Database |
| jhansi | K.Das | C |
| subbu | P.Naresh | Database |
| subbu | R.Prasad | C |

- The above relation is not in BCNF, because in the FD (teacher->subject), teacher is not a key. (student,teacher) and (teacher, subject) are keys.
- Teacher-> subject violates BCNF [since teacher is not a candidate key].
- For example, if we try to delete the student Subbu, we will lose the information that R. Prasad teaches C.

- If X->Y violates BCNF then divide R into R1(X, Y) and R2(R-Y).[ R2 includes all attributes in R except (minus) the attribute Y ]
- So R is divided into two relations R1(Teacher, subject) and R2(student, Teacher).
- All the anomalies which were present in R, are now removed in the above two relations.

**R1**

| Teacher | Subject |
|---|---|
| P.Naresh | database |
| K.DAS | C |
| R.Prasad | C |

**R2**

| Student | Teacher |
|---|---|
| Jhansi | P.Naresh |
| Jhansi | K.Das |
| Subbu | P.Naresh |
| Subbu | R.Prasad |

# Difference between 3NF and BCNF

| S.NO. | 3NF | BCNF |
|---|---|---|
| 1. | In 3NF there should be no transitive dependency that is no non prime attribute should be transitively dependent on the candidate key. | In BCNF for any relation A->B, A should be a super key of relation. |
| 2. | It is less stronger than BCNF. | It is comparatively more stronger than 3NF. |
| 3. | In 3NF the functional dependencies are already in 1NF and 2NF. | In BCNF the functional dependencies are already in 1NF, 2NF and 3NF. |
| 4. | The redundancy is high in 3NF. | The redundancy is comparatively low in BCNF. |
| 5. | In 3NF there is preservation of all functional dependencies. | In BCNF there may or may not be preservation of all functional dependencies. |
| 6. | It is comparatively easier to achieve. | It is difficult to achieve. |
| 7. | Lossless decomposition can be achieved by 3NF. | Lossless decomposition is hard to achieve in BCNF. |

# Surrogate key

- Surrogate key also called a synthetic primary key, is generated when a new record is inserted into a table automatically by a database that can be declared as the primary key of that table.
- It is the sequential number outside of the database that is made available to the user and the application or it acts as an object that is present in the database but is not visible to the user or application.
- We can say that , in case we do not have a natural primary key in a table, then we need to artificially create one in order to uniquely identify a row in the table , this key is called the surrogate key or synthetic primary key of the table. However , surrogate key is not always the primary key .

- **Features of the surrogate key :**
- It is automatically generated by the system.
- It holds anonymous integer.
- It contains unique value for all records of the table.
- The value can never be modified by the user or application.

# Properties of Decomposition

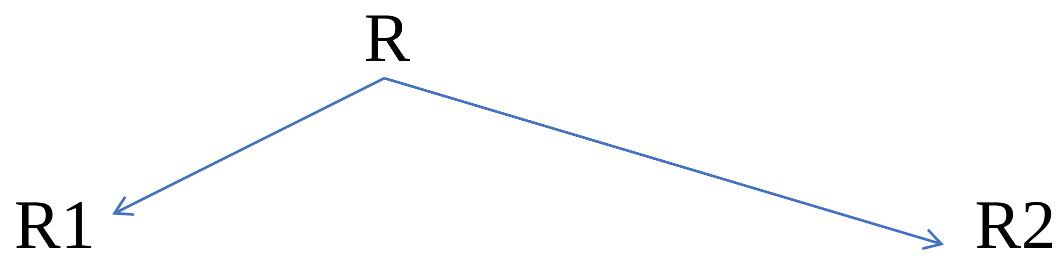
There are 2 types of Decompositions

1) Lossless Decomposition
2) Dependency Preservation

**Loss-less join decomposition/ Lossless Decomposition**

Let R be a relation schema and let F be a set of FDs over R. A decomposition of R into two schemas is said to be loss less join, if we recover original relation from decomposed relation. Otherwise it is lossy join.

We take projections of a relation and recombine them using natural join, to obtain original relation.



# Loss-less join decomposition

| A | B | C |
|---|---|---|
| $a_1$ | $b_1$ | $c_1$ |
| $a_1$ | $b_2$ | $c_2$ |

decompose

| A | B |
|---|---|
| $a_1$ | $b_1$ |
| $a_1$ | $b_2$ |

| A | C |
|---|---|
| $a_1$ | $c_1$ |
| $a_1$ | $c_2$ |

join

| A | B | C |
|---|---|---|
| $a_1$ | $b_1$ | $c_1$ |
| $a_1$ | $b_1$ | $c_2$ |
| $a_1$ | $b_2$ | $c_1$ |
| $a_1$ | $b_2$ | $c_2$ |

| A | B | C |
|---|---|---|
| $a_1$ | $b_1$ | $c_1$ |
| $a_1$ | $b_2$ | $c_2$ |

decompose

| A | C |
|---|---|
| $a_1$ | $c_1$ |
| $a_1$ | $c_2$ |

| B | C |
|---|---|
| $b_1$ | $c_1$ |
| $b_2$ | $c_2$ |

join

| A | B | C |
|---|---|---|
| $a_1$ | $b_1$ | $c_1$ |
| $a_1$ | $b_2$ | $c_2$ |

# Dependency Preservation

- 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.
- It allows us to enforce all FDs by examining a single relation on each insertion or modification of a tuple.

Ex:-Let R(SUPPLIER_NO,STATUS,CITY,PART_NO,QUANTITY) be a relation.

FDs:- FD1: SUPPLIER_NO→CITY

FD2: CITY→STATUS

FD3: PART_NO→QUANTITY

R is decomposed into R1 & R2 such that

R1(SUPPLIER_NO,STATUS,CITY) which satisfies FD1,FD2

R2(SUPPLIER_NO,PART_NO,QUANTITY) which satisfies FD3

We can say that, the decomposition of R into R1 & R2 is dependency preservation.

| PROPERTIES | 1NF | 2NF | 3NF | BCNF |
|---|---|---|---|---|
| DEFINITION | Every attribute contains single value | Relation is in 1 NF and All the non-key attributes of the table are fully functionally dependent on the Primary key of the table. | Relation is in 2 NF and every non key attribute is non transitively dependent on the primary key. | A relation is in 3NF and every determinant is a candidate key |
| Loss less join decomposition | Always achievable | Always achievable | Always achievable | Sometimes not achievable |
| Dependency preservation | - | - | Possible | Either loss less join or dependency preservation is possible. not both |
| Anomalies | May allow some anomalies | May allow some anomalies | May allow some anomalies | Always eliminates anomalies |

# Fourth Normal Form

- Fourth Normal Form comes into picture when Multi-valued Dependency occur in any relation.
- **A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency(MVD).**
- A table is said to have multi-valued dependency, if the following conditions are true,
  1. For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency(MVD) denoted as A→→B and read as A multi determines B.
  2. Also, a table should have at-least 3 columns for it to have a multi-valued dependency.
  3. And, for a relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

| EMP_NO | PROJECT_NO | SKILL |
|---|---|---|
| E1 | 1 | ANALYSIS |
| E1 | 1 | DESIGN |
| E1 | 1 | PROGRAM |
| E2 | 2 | ANALYSIS |
| E2 | 2 | DESIGN |
| E2 | 2 | PROGRAM |

**EMPLOYEE(EMP_NO,PROJECT_NO,SKILL)**

MVDs: EMPNO→→SKILL
EMPNO→→PROJECT_NO

Clearly, the table EMPLOYEE is in BCNF , but has MVDs. Therefore, it is not in 4NF.

To convert to 4NF, decompose EMPLOYEE into 2 tables EMP_PROJECT & EMP_SKILL

| EMP_NO | SKILL |
|---|---|
| E1 | ANALYSIS |
| E1 | DESIGN |
| E1 | PROGRAM |
| E2 | ANALYSIS |
| E2 | DESIGN |
| E2 | PROGRAM |

EMP_SKILL(EMP_NO,SKILL)

| EMP_NO | PROJECT_NO |
|---|---|
| E1 | 1 |
| E2 | 2 |

EMP_PROJECT(EMP_NO,PROJECT_NO

These tables do not have MVDs.(no 3 columns)

# Fifth Normal Form

- **A relation will be in 5NF if and only 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).
- A relation R is in Fifth Normal Form (5NF) if and only if the following conditions are satisfied simultaneously:

1. It's in 4NF
2. If we can decompose table further to eliminate redundancy and anomaly, and when we re-join the decomposed tables by means of candidate keys, we should not be losing the original data or any new record set should not arise. In simple words, joining two or more decomposed table should not lose records nor create new records.

- Join dependency is a constraint which is similar to functional dependency or multivalued dependency. It is satisfied if and only if the relation concerned is the join of a certain number of projections. Such type of constraint is called join dependency.

Alternatively,

- Join decomposition is a further generalization of Multivalued dependencies. If the join of R1 and R2 over common attribute C is equal to relation R then we can say that a join dependency (JD) exists, where R1 and R2 are the decompositions R1(A, B, C) and R2(C, D) of a given relation R (A, B, C, D).

# Department

| Department | Subject | Student |
|---|---|---|
| CSE | CS101 | Shreya |
| IT | IT501 | Yug |
| CSE | CS102 | Ruthvi |
| CSE | CS103 | Rini |
| ME | ME201 | Sushant |
| EC | EC301 | Ira |

**Consider the relation Department.**

**There are 2 MVDs here. So the relation is not in 4NF**

**Dept $\rightarrow\rightarrow$ Subject**

**Dept $\rightarrow\rightarrow$ Student**

**To convert to 4NF, decompose them into 2 tables and remove the MVDs. The new tables are Dsub & Dstud which are in 4NF.**

DSub

| Department | Subject |
|---|---|
| CSE | CS101 |
| IT | IT501 |
| CSE | CS102 |
| CSE | CS103 |
| ME | ME201 |
| EC | EC301 |

DStud

| Department | Student |
|---|---|
| CSE | Shreya |
| IT | Yug |
| CSE | Ruthvi |
| CSE | Rini |
| ME | Sushant |
| EC | Ira |

# D sub $\bowtie$ D Stu (Join)

| Department | Subject | Student |
|---|---|---|
| CSE | CS101 | Shreya |
| CSE | CS101 | Ruthvi X |
| CSE | CS101 | Rini X |
| IT | IT501 | Yug |
| CSE | CS102 | Shreya X |
| CSE | CS102 | Ruthvi |
| CSE | CS102 | Rini X |
| CSE | CS103 | Shreya X |
| CSE | CS103 | Ruthvi X |
| CSE | CS103 | Rini |
| ME | ME201 | Sushant |
| EC | EC301 | Ira |

X -
After Join , we get 6 Redundant tuples

- Join dependency exists where redundant rows are generated when the tables are joined by using a natural join operation
- We now have to remove the join dependency. Therefore, decompose the main “Department” table into 3 tables instead of 2.

DSub

| **Department** | **Subject** |
|---|---|
| CSE | CS101 |
| IT | IT501 |
| CSE | CS102 |
| CSE | CS103 |
| ME | ME201 |
| EC | EC301 |

**SubStu**

| Sub | Stu |
|---|---|
| CS101 | Shreya |
| IT501 | Yug |
| CS102 | Ruthvi |
| CS103 | Rini |
| ME201 | Sushant |
| EC301 | Ira |

DStud

| **Department** | **Student** |
|---|---|
| CSE | Shreya |
| IT | Yug |
| CSE | Ruthvi |
| CSE | Rini |
| ME | Sushant |
| EC | Ira |

- Now, Natural join all these 3 tables(basing on the common attributes).

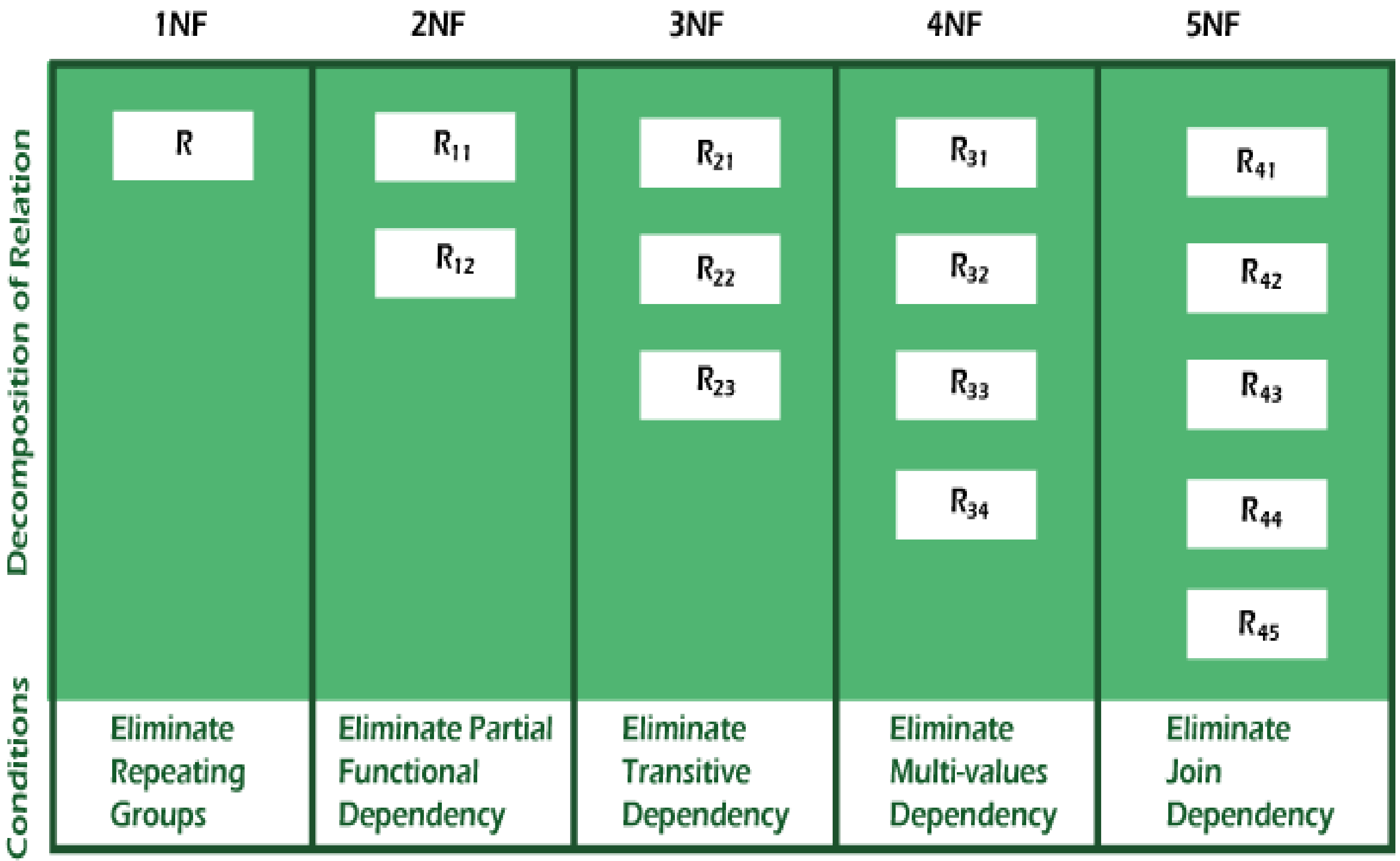
Select d1.dept, d2.sub, d3.dept from Dsub d1, Substu d2, Dstu d3 where d1.subject= d2. subject and d2.student=d3.student and d3. dept=d1.dept;

- We get the original table without loosing tuples or increase in tuples.
- We can say that, the three tables are now in 5NF.

| Department | Subject | Student |
| --- | --- | --- |
| CSE | CS101 | Shreya |
| IT | IT501 | Yug |
| CSE | CS102 | Ruthvi |
| CSE | CS103 | Rini |
| ME | ME201 | Sushant |
| EC | EC301 | Ira |

# Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints. Following are the various 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. |
| BCNF | A stronger definition of 3NF is known as Boyce Codd's normal form. |
| 4NF | A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency. |
| 5NF | A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless. |

# Types of Dependencies

- Functional Dependency
- Total & partial Dependency
- Transitive Dependency
- Multi valued Dependency
- Join Dependency

# References

**Text Books:**


- Database Management Systems, 3/e, Raghurama Krishnan, Johannes Gehrke, TMH.
- Database System Concepts,5/e, Silberschatz, Korth, TMH.


**Reference Books:**


- Introduction to Database Systems, 8/e C J Date, PEA.
- Database Management System, 6/e Ramez Elmasri, Shamkant B. Navathe, PEA
- Database Principles Fundamentals of Design Implementation and Management, Corlos Coronel, Steven Morris, Peter Robb, Cengage Learning.


**Web Links:**

- https://nptel.ac.in/courses/106/105/106105175/
- https://www.geeksforgeeks.org/introduction-to-nosql/
- https://beginnersbook.com/2015/05/normalization-in-dbms/
- https://www.javatpoint.com/dbms-tutorial

# Thank You