UNIT-4

Schema refinement,
Problems Caused by redundancy,
Decompositions,
Problem related to decomposition,
reasoning about FDS,
FIRST, SECOND, THIRD Normal forms,
BCNF,
Lossless join Decomposition,
Dependency preserving Decomposition,
Multi valued Dependencies,
FORTH Normal Form

Data Redundancy:

- Duplication of data or repetition of data in a table

Problems Caused by Redundancy

Update anomalies: If one copy of data is updated, an inconsistency is created unless all copies are similarly updated

Insertion anomalies: It may not be possible to store some information unless some other information is stored as well

Deletion anomalies: It may not be possible to delete some information without losing some other information as well

- Consider a relation Hourly_Emps

Hourly_Emps(*ssn*, *name*, *lot*, *rating*, *hourly_wages*, *hours_worked)*

- For example, we will refer to the Hourly_Emps schema as *SNLRWH (W denotes the hourly wages attribute)*

- The *key* for Hourly_Emps is *ssn*

- Suppose *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 IC (Integrity Constraint) is an example of a *functional dependency*

- It leads to possible redundancy in the relation Hourly_Emps, as illustrated in Figure 15.1

- If the same value appears in the *rating* column of two tuples, the IC tells us that the same value must appear in the *hourly_wages* column as well

ssn	*name*	*lot*	*rating*	*hourly_wages*	*hours_worked*
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

Figure 15.1 An Instance of the Hourly_Emps Relation

- This redundancy leads to potential inconsistency

- For example, the *hourly_wages* in the first tuple could be updated without making a similar change in the second tuple, which is an example of an *update anomaly*

- Also, we cannot insert a tuple for an employee unless we know the hourly wage for the employee's rating value, which is an example of an *insertion anomaly*

- If we delete all tuples with a given rating value (e.g., we delete the tuples for Smethurst and Guldu) we lose the association between that *rating* value and its *hourly_wage* value *(deletion anomaly)*

Example

Employees' Skills

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

- An **update anomaly**. Employee 519 is shown as having different addresses on different records

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201
424	Dr. Newsome	29-Mar-2007	?

- An **insertion anomaly**. Until the new faculty member, Dr. Newsome, is assigned to teach at least one course, his details cannot be recorded.

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201

DELETE

- A **deletion anomaly**. All information about Dr. Giddens is lost when he temporarily ceases to be assigned to any courses

Null Values

- Let us consider whether the use of *null* values can address some of these problems

- Clearly, *null* values cannot help eliminate update anomalies

- Insertion Anomaly: For example, we cannot record the hourly_wage for a rating unless there is an employee with that rating, because we cannot store a null value in the *ssn* field, which is a primary key field

- Deletion anomaly, we might consider storing a tuple with *null* values in all fields except *rating* and *hourly_wages* if the last tuple with a given rating would otherwise be deleted

- However, this solution will not work because it requires the *ssn* value to be null, and primary key fields cannot be null

- Thus, null values do not provide a general solution to the problems of redundancy

Decompositions

- *decomposition* is dividing the larger relation into the smaller relations

- Each of the smaller relations contains a (strict) subset of the attributes of the original relation

- We can deal with the redundancy in Hourly_Emps by decomposing it into two relations:

  Hourly_Emps2(*ssn*, *name*, *lot*, *rating*, *hours_worked)*
  Wages(*rating*, *hourly_wages)*

- The instances of these relations corresponding to the instance of Hourly_Emps relation in Figure 15.1 is shown in Figure 15.2

ssn	*name*	*lot*	*rating*	*hours_worked*
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

rating	*hourly_wages*
8	10
5	7

Figure 15.2 Instances of Hourly_Emps2 and Wages

- We can easily record the hourly wage for any rating simply by adding a tuple to Wages, even if no employee with that rating appears in the current instance of Hourly_Emps. So insertion anomalies are eliminated.

- Changing the wage associated with a rating involves updating a single Wages tuple. So updation anomalies are eliminated.

- After deleting tuples from the Hourly_Emps, still the information rating and hourly wages available in the Wages. So deletion anomalies are eliminated.

Problems with Decompositions

- There are three potential problems:

- Some queries become more expensive

  e.g., How much did sailor Joe earn? (salary = W*H)

- Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!

  Fortunately, not in the SNLRWH example

- Checking some dependencies may require joining the instances of the decomposed relations

  Fortunately, not in the SNLRWH example

Functional Dependencies

- A **functional dependency** (FD) is a constraint between two sets of attributes in a relation from a database

- Let *R* be a relation schema and let *X* and *Y* be nonempty sets of attributes in *R*

- FD $X \rightarrow Y$ holds if the following holds for every pair of tuples t_1 and t_2: If *t1.X* = *t2.X*, then *t1.Y* = *t2.Y*

- Note: $X \rightarrow Y$ is read as *X functionally determines Y*, or simply as *X determines Y*

Example1:

The following table illustrates A ⟹ B:

A	B
1	1
2	4
3	9
4	16
2	4
7	9

Example2:

Two same Roll_No cannot have different values against Name attribute.

Roll_No	Name	Marks
1.	Anoop	20
1.	Anurag	30
2.	Saurav	40
3.	Rakesh	30
4.	Pritesh	10
5.	Anoop	40

Here, Roll_No $\not\Rightarrow$ Name

- Example3:

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

An Instance that Satisfies $AB \rightarrow C$

- Recall that a *legal* instance of a relation must satisfy all specified ICs, including all specified FDs

- Note: A primary key is a special case of an FD. The attributes in the key play the role of *X*, and the set of all attributes in the relation plays the role of *Y*

Reasoning About Functional Dependencies

- As an example, consider:

Workers(*ssn*, *name*, *lot*, *did*, *since)*

- We know that $ssn \rightarrow did$ holds, since *ssn* is the key, and FD $did \rightarrow lot$ is given to hold

- Thus, the FD $ssn \rightarrow lot$ also holds on Workers

Armstrong's Axioms (Properties) of Functional Dependencies

▪We use *X*, *Y*, and *Z* to denote sets of attributes over a relation schema *R*:

- **Reflexivity:** If $X \supseteq Y$, then $X \rightarrow Y$.
- **Augmentation:** If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z.
- **Transitivity:** If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.
- **Union:** If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$.
- **Decomposition:** If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$.

Closure of a Set of FDs

- The set of all FDs implied by a given set (*F)* of FDs is called the **closure of F** and is denoted as F^+

- Armstrong's Axioms, can be applied repeatedly to infer all FDs implied by a set (*F)* of FDs

Example: Suppose we are given a Relation schema R with attributes A, B, C, D, E, F, and the FDs A → BC, B → E, CD → EF. Show that the FD AD → F holds for R and is thus a member of the closure of the given set:

Ans:

1. A → BC (given)
2. A → C (1, decomposition)
3. AD → CD (2, augmentation)
4. CD → EF (given)
5. AD → EF (3 and 4, transitivity)
6. AD → F (5, decomposition)

Attribute Closure

- Attribute closure is used find out the candidate keys in a relation

- If X is a set of attributes, then the attribute closure of X is X^+

$closure = X;$
repeat until there is no change: {
    if there is an FD $U \rightarrow V$ in F such that $U \subseteq closure$,
        then set $closure = closure \cup V$
}

Computing the Attribute Closure of Attribute Set X

- This algorithm starts with attribute *X* and stops as soon as there is no change in the *closure*

- By varying the starting attribute and the order in which the algorithm considers FDs, we can obtain all candidate keys

Example: Suppose we are given a relation schema R with attributes A, B, C, D, E, F and FDs: A → BC, E → CF, B → E, CD → EF. Compute the closure $\{A,B\}^+$ of the set of attributes {A,B} under this set of FDs

Ans:

1. Initialize the *closure* to {A,B}
2. Go through inner loop 4 times, once for each of the given FDs

   i) FD A → BC, here A is a subset of *closure*, so add B and C to *closure* (*closure* ={ A,B,C})

   ii) FD E → CF, do not add C and F to *closure*, because E is not a subset of *closure*

iii) FD B → E, add E to *closure* (*closure* ={ A,B,C,E})

iv) FD CD → EF, do not add F to *closure*

now *closure*={ A,B,C,E}

3. Go through inner loop 4 times, once for each of the given FDs

i) FD A → BC, no change in *closure*

ii) FD E → CF, *closure* ={ A,B,C,E,F}

iii) FD B → E, no change in *closure*

iv) FD CD → EF, no change in *closure*

4. Go through inner loop 4 times, once for each of the given FDs. Closure does not change so the process terminates.

so closure of $\{A,B\}^+$ ={ A,B,C,E,F}

Example: For a relation schema R(A, B, C, D, E). The set of functional dependencies are A → BC, CD → E, B → D, E → A. List the candidate keys for R.

Ans:

Attribute closure:

A → ABCDE

B → BD

D → D

E → ABCDE

BC → ABCDE

CD → ABCDE

The candidate keys are A, E, BC and CD

Differetnt keys in DBMS

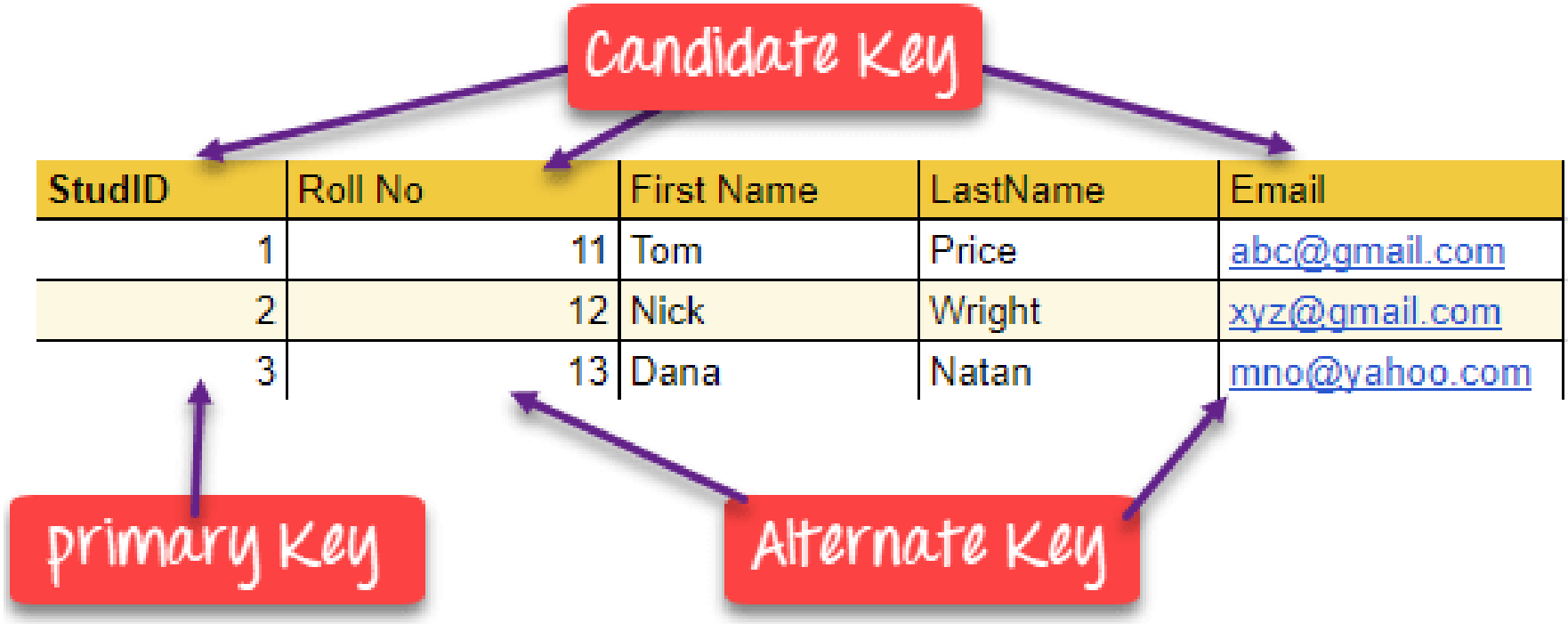
Super key: It is a minimum set of attributes, which uniquely identifies each record in a relation.

Candidate key: It is a minimal super key. (It is a super key, whose proper subset is not a super key.)

Primary key: Choose one of the candidate keys as a primary key.

Composite key:
A primary key having two or more attributes is called composite key or composite primary key.

Alternate key (or Secondary key): The leftout candidate keys after selecting the primary key.



StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

Suit	Value	No. times played
Hearts	Ace	5
Diamonds	Three	2
Hearts	Jack	3
Clubs	Three	5
Spades	Five	1

composite key – {Suit, Value}

Example1: R(ABCD)

	Super key	Candidate key
$A \rightarrow BCD$	A is super key	A is candidate key
$AB \rightarrow CD$	AB is super key	AB is not
$ABC \rightarrow D$	ABC is super key	ABC is not
$BD \rightarrow AC$	BD is super key	BD is candidate key
$C \rightarrow AD$	C is not super key	C is not

- Choose either A or BD as primary key. The remaining key becomes alternate key.

- BD is an example of composite key.

Example2: Find super keys, candidate keys and primary key

Emp_SSN	Emp_Number	Emp_Name
123456789	226	Steve
999999321	227	Ajeet
888997212	228	Chaitanya
777778888	229	Robert

Super keys:

{Emp_SSN}

{Emp_Number}

{Emp_SSN, Emp_Number}

{Emp_SSN, Emp_Name}

{Emp_SSN, Emp_Number, Emp_Name}

{Emp_Number, Emp_Name}

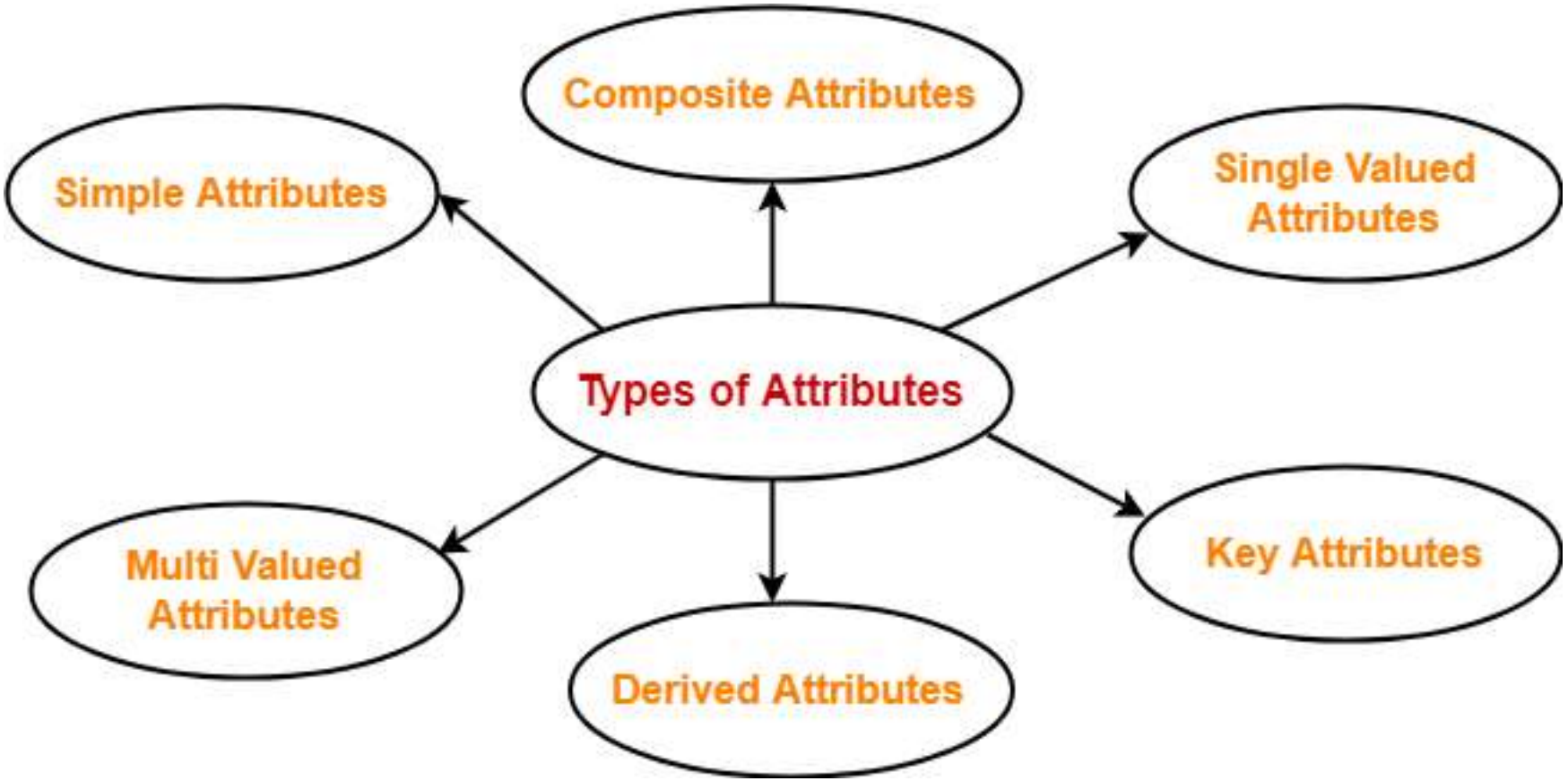
Candidate keys:
{Emp_SSN}
{Emp_Number}

Primary key:
Choose either {Emp_SSN} or {Emp_Number}

Alternate key:
If primary key is {Emp_SSN} then Alternate key is {Emp_Number}

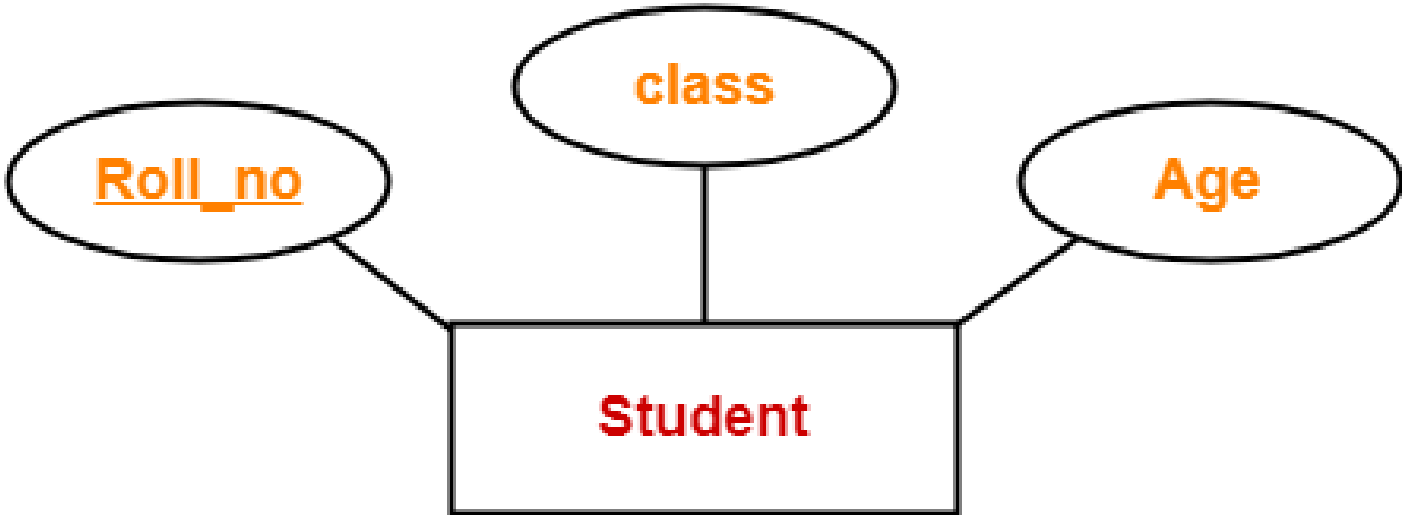
Types of Attributes

- In ER diagram, attributes associated with an entity set may be of the following types



1. Simple Attributes

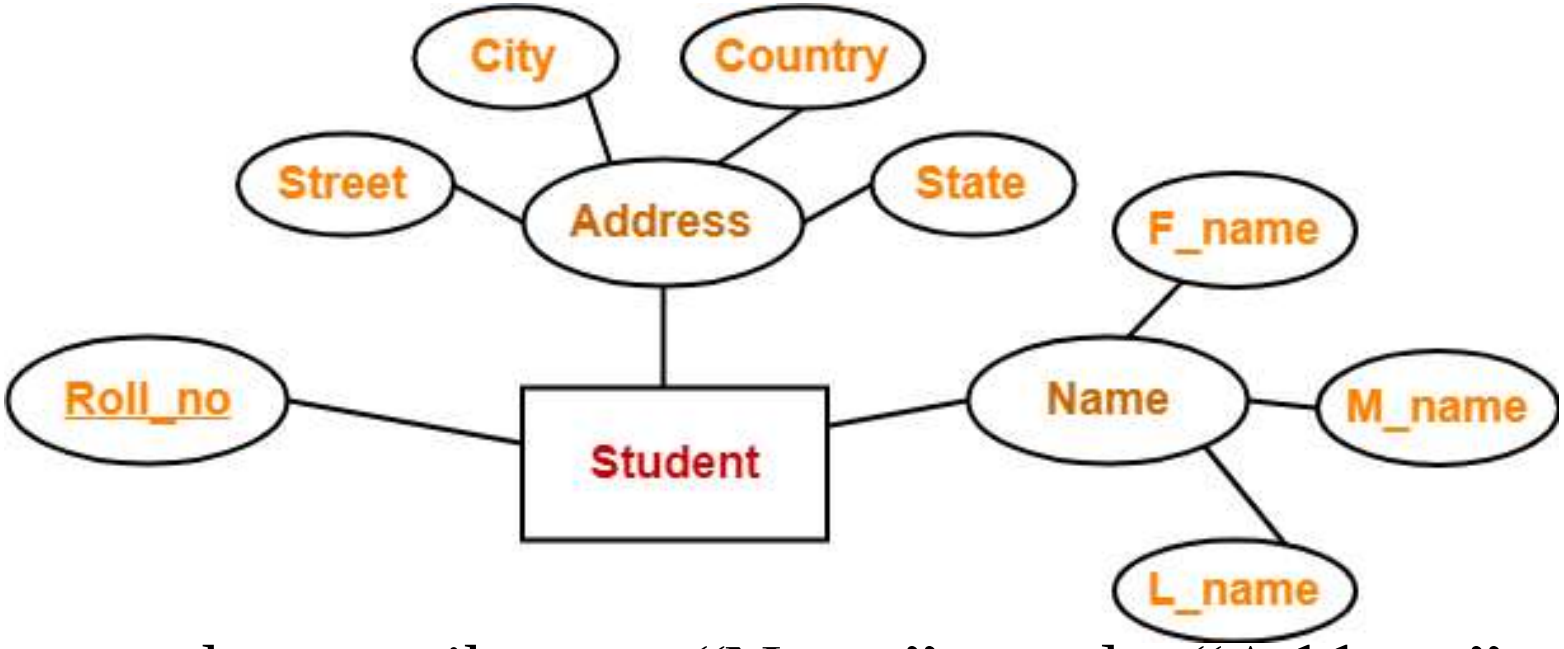
- Simple attributes are those attributes which can not be divided further.
- **Example:**



- Here, all attributes are simple attributes as they can not be divided further.

2. Composite Attributes

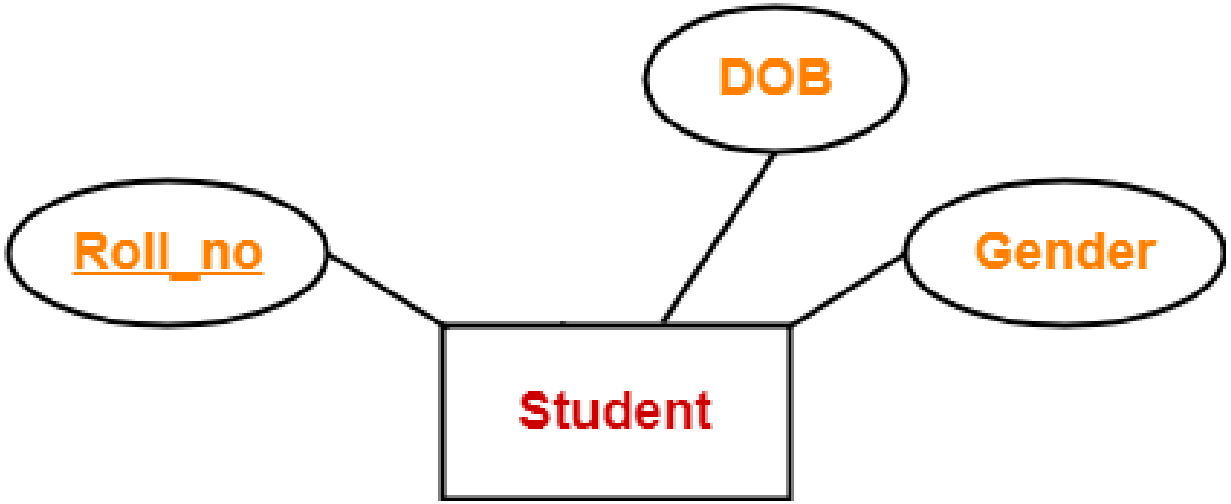
- Composite attributes are those attributes which are composed of many other simple attributes.
- **Example**



- Here, the attributes "Name" and "Address" are composite attributes as they are composed of many other simple attributes.

3. Single Valued Attributes

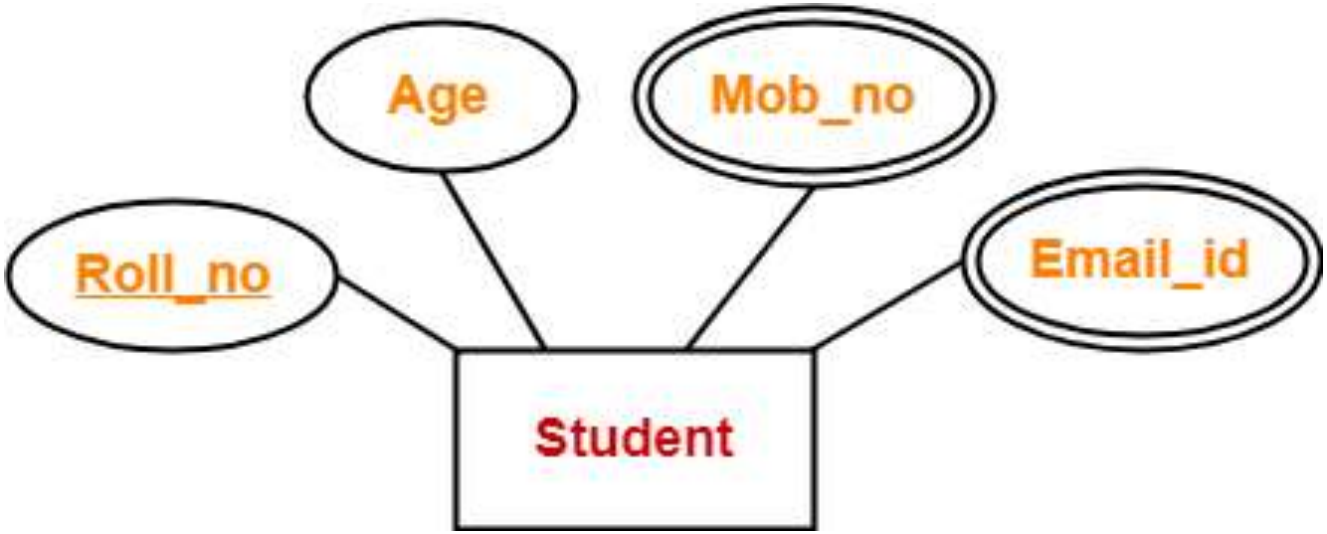
- Single valued attributes are those attributes which can take only one value for a given entity from an entity set.
- **Example**



- Here, all attributes are single valued attributes as they can take only one specific value for each entity.

4. Multi Valued Attributes

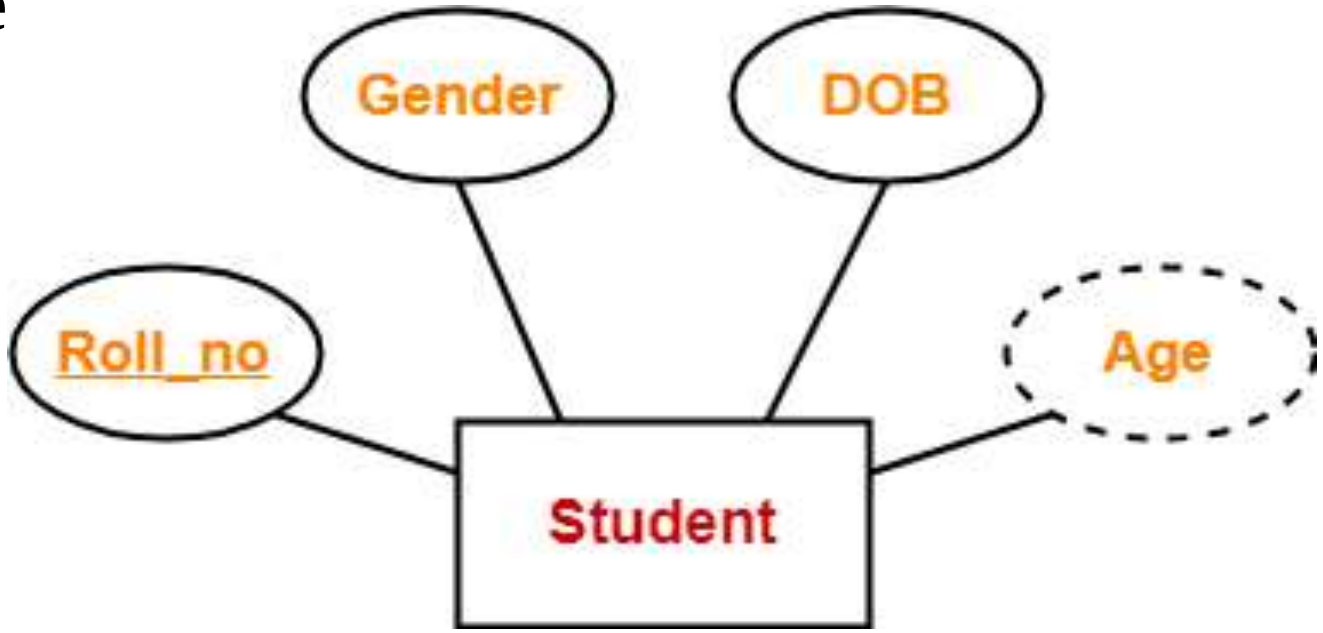
- Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.
- **Example**



- Here, the attributes “Mob_no” and “Email_id” are multi valued attributes as they can take more than one values for a given entity.

5. Derived Attributes

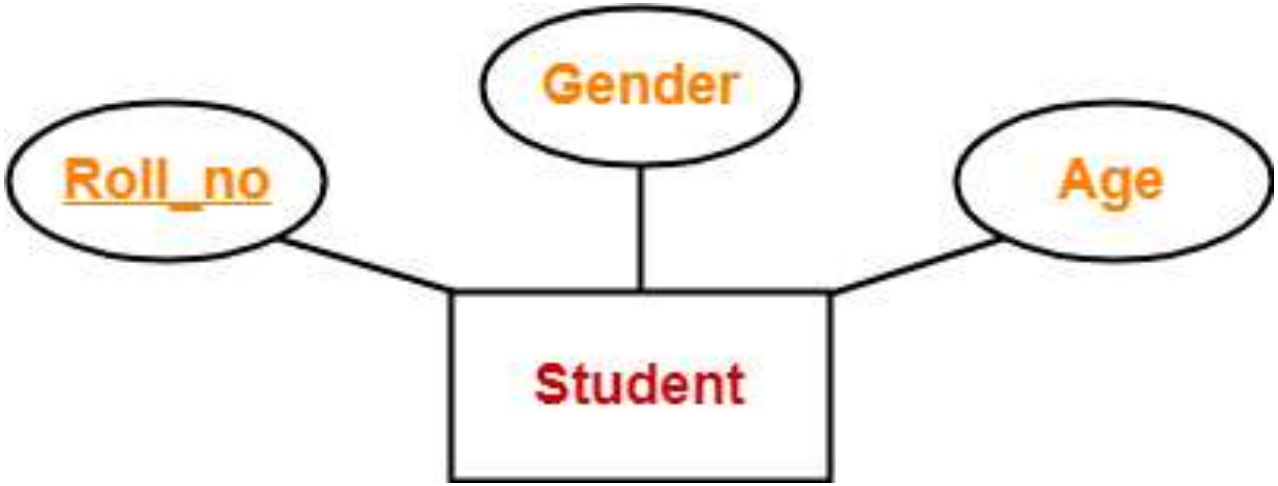
- Derived attributes are those attributes which can be derived from other attribute(s).
- **Example**



- Here, the attribute "Age" is a derived attribute as it can be derived from the attribute "DOB".

6. Key Attributes

- Key attributes are those attributes which can identify an entity uniquely in an entity set.
- **Example**



- Here, the attribute "Roll_no" is a key attribute as it can identify any student uniquely.

Normalization

- ***Normalization of data*** is a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anomalies

- First, second, third normal form and Boyce-Codd normal forms (BCNF) are based on the ***functional dependencies*** among the attributes of a relation

- Fourth normal form (4NF) is based on **multi valued dependency**

- Fifth normal form (5NF) is based on **join dependency**

- Relation which does not satisfy the normal form test is decomposed into smaller relations

- Another point is that the database designers *need not* normalize to the highest possible normal form

- Every relation in BCNF is also in 3NF, every relation in 3NF is also in 2NF, and every relation in 2NF is in 1NF

- Note: In a relational database, a relation is always in First Normal Form (1NF) at least.

First Normal Form (1NF)

- *A relation is said to be in **1NF** if and only if each attribute of the relation is atomic* (i.e. *the value of any attribute in a tuple must be a single value)*

- It disallow multi valued attributes, composite attributes, and their combinations

Example1:

Student_id	Name	Subjects
100	Akshay	Computer Networks, Operating Systems
101	Aman	Database Management Systems
102	Anjali	Automata, Compiler Design

- This relation is not in 1NF, convert it in to 1NF

Student_id	Name	Subjects
100	Akshay	Computer Networks
100	Akshay	Operating Systems
101	Aman	Database Management Systems
102	Anjali	Automata
102	Anjali	Compiler Design

Relation is in 1NF

Example2:

Manager	Persons
Vinod	Shaju , Manoj, Ashok, Naveen
Rajiv	Saravana, Salil, Kannan

Relation is not in 1NF

Manager	Persons
Vinod	Shaju
Vinod	Manoj
Vinod	Ashok
Vinod	Naveen
Rajiv	Kannan
Rajiv	Salil
Rajiv	Saravana

Relation is in 1NF

Example3:

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

Relation is not in 1NF

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

Relation is in 1NF

Second Normal Form (2NF)

- *A relation R is in 2NF if it is in 1NF and every nonprime attribute in R is fully functionally dependent on the primary key of R*

- A functional dependency X → Y is a ***full functional dependency*** if removal of any attribute *A* from *X* means that the dependency does not hold any more;

  That is, for any attribute A ∈ X, (X - {A}) does not functionally determine Y

- A functional dependency X → Y is a ***partial dependency*** if some attribute A ∈ X can be removed from X and the dependency still holds;

  That is, for some A ∈ X, (X - {A}) → Y

Prime attribute: An attribute of relation schema R is called a prime attribute of R if it is a member of some candidate key of R

Nonprime attribute (or Non key attribute): An attribute is called nonprime attribute if it is not a member of any candidate key

Example: In a relation schema R with attributes A, B, C, D, E, F and if $\{A,B\}^+ = \{A,B,C,D,E,F\}$. Then A and B are prime attributes. C, D, E and F are non prime attributes.

Note: If primary key in a relation contains a single attribute, directly we can say that relation is in 2NF.

Example1:

TABLE_PURCHASE_DETAIL

Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

- The relation is in 1NF.
- This table has a composite primary key {Customer ID, Store ID}.
- The non-key attribute is {Purchase Location}.

- {Customer ID, Store ID} → {Purchase Location} is a partial dependency, because

{Store ID} → {Purchase Location}

- That means {Purchase Location} only depends on {Store ID}, which is only part of the primary key.

- Therefore, this table is not in second normal form.

- So, decompose the relation in to two relations.

- In one relation, take attributes which are in partial dependency. (part of candidate key → non key attribute).

R1(Store ID, Purchase Location)

- In another relation, take remaining attributes from the original relation (Customer ID) and one common attribute (Store ID) between the decomposed relations.

R2(Customer ID, Store ID)

R1

TABLE_STORE

Store ID	Purchase Location
1	Los Angeles
2	New York
3	San Francisco

R2

TABLE_PURCHASE

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

Example2:

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith	ProductX	Boston
123456789	2	7.5	Smith	ProductY	Sugarland
666884444	3	40.0	Narayan	ProductZ	Houston
453453453	1	20.0	Joyce	ProductX	Boston
453453453	2	20.0	Joyce	ProductY	Sugarland
333445555	2	10.0	Franklin	ProductY	Sugarland

- The relation is in 1NF.

- {SSN, PNUMBER} → HOURS is a full dependency (neither SSN → HOURS nor PNUMBER → HOURS holds)

- The dependency {SSN, PNUMBER} → ENAME is partial because SSN → ENAME holds

- The dependency {SSN,PNUMBER} → {PNAME, PLOCATION} is partial, because

  PNUMBER → {PNAME, PLOCATION}

- We have the following FDs

  FD1 {SSN,PNUMBER} → HOURS

  FD2 SSN → ENAME

  FD3 PNUMBER → {PNAME, PLOCATION}

- So, relation is not in 2NF, because relation has partial dependencies

- Because of 3 FDs, now decompose the relation into 3 relations as shown below

R1

SSN	**PNUMBER**	**HOURS**

R2

SSN	**ENAME**

R3

PNUMBER	**PNAME**	**PLOCATION**

Third Normal Form (3NF)

- *A relation R is in 3NF if it is in 2NF and no nonprime attribute of R is transitively dependent on the primary key*

- A functional dependency X → Y in a relation R is a *transitive dependency* if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold

Example1:

EMPNO	**ENAME**	**DNUMBER**	**DNAME**
1	Kevin	201	R&D
2	Jones	224	IT
3	Jake	201	R&D

- The dependency EMPNO → DNAME (non prime attribute) is transitive through DNUMBER because both the dependencies

EMPNO → DNUMBER

and

DNUMBER → DNAME hold

and DNUMBER is neither a key itself nor subset of the key

- The relation is in 2NF but it is not in 3NF because relation has transitive dependency

- Now decompose the relation into two relations, in one relation take the attributes which are in the FD DNUMBER → DNAME (FD with non prime attributes)

R1(DNUMBER, DNAME)

- In another relation take the remaining attributes from the original relation (EMPNO, ENAME) and one common attribute (DNUMBER) between the decomposed relations

R2(EMPNO, ENAME, DNUMBER)

R1

DNUMBER	**DNAME**
201	R&D
224	IT

R2

EMPNO	**ENAME**	**DNUMBER**
1	Kevin	201
2	Jones	224
3	Jake	201

Example:

TABLE_BOOK_DETAIL

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

- The relation is in 2NF
- Book ID → Genre ID and
  Genre ID → Genre Type (nonprime attribute)
  Therefore, Book ID → Genre Type
- This relation has transitive dependency. So this relation is not in 3NF
- Now decompose the relation into the following relations

TABLE_GENRE

R1

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

TABLE_BOOK

R2

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

1. Consider a relation R(A,B,C,D,E,F,G) and the following functional dependencies:

$\{B \rightarrow D,$
$D \rightarrow E,$
$AB \rightarrow C,$
$AE \rightarrow F,$
$C \rightarrow A,$
$F \rightarrow G\}.$

Identify candidate key(s) of R:

Ans:

-Look at the missing attributes on the RHS of the FDs. Here B is missing. So B must be a part of the candidate key.

-AB^+ = ABCDEFG. So AB is a candidate key.

-Next check any replacement of either A or B in AB^+.

-There is one given FD $C \rightarrow A$, means A can be determined from C. So A can be replaced with C.

$CB^+ = ABCDEFG$. So CB is a candidate key.

-Candidate keys are AB and BC

2. Consider a relation R(A,B,C,D,E,F) and the following functional dependencies:

$AB \rightarrow C$,

$C \rightarrow ABDE$,

$ADE \rightarrow F$

Identify candidate key(s) of R:

Ans: There is no missing attribute on the RHF of the FDs.
-By looking at the FDs, we can say C is a possible candidate key.

- C^+ = ABCDEF, So C is a candidate key

-From the given FD AB → C, C can be replaced with AB

-AB^+ = ABCDEF. So AB is a candidate key.

-Candidate keys are AB and C

Boyce-Codd Normal Form (BCNF)

- *A relational R is in **BCNF** if it is in 3NF and for every FD X → Y if one of the following statements is true:*
  - *X → Y is a trivial functional dependency (i.e., Y is a subset of X), or*
  - *X is a super key*

Example1: Consider a relation R(A, B, C, D) with the FDs A → BCD, BC → AD and D → B.

Ans: The candidate keys are A and BC.

Relation is in 3NF

In A → BCD, A is super key.

In BC → AD, BC is super key.

In D → B, D is not super key. So relation is not in BCNF

- Now decompose the relation into the following relations.
- In one relation take attributes from the FD, which violates the BCNF rule

R1(D, B) (D → B holds)

- In another relation take remaining attributes from the original relation and one common attribute between the decomposed relations

R2(A, C, D) (A → CD holds)

- **BCNF relations are R1(D, B), R2(A, C, D)**

Question1:

1. Consider a relation R(ABCDEFGHIJ) with the FDs AB → C, A → DE, B → F, F → GH and D → IJ. Decompose the relation up to BCNF.

Ans:

1NF: Given FDs AB → C,
A → DE,
B → F,
F → GH,
D → IJ

Find the candidate keys: Look at the missing attributes on the RHS of the FDs. Here A and B are missing. So AB must be a part of the candidate key.

AB^+ = ABCDEFGHIJ. So AB is a candidate key.

-Next check any replacement of either A or B in AB^+. No.
-So only AB is a candidate key.
-Relation is in 1NF, because relation has a candidate key.

2NF:

-Consider the given FDs AB → C, A → DE, B → F, F → GH and D → IJ
-A → DE is a partial dependency. So relation is not in 2NF.
-Next compute A^+, So A^+ = ADEIJ
-Now decompose the given relation into **R1(ADEIJ)** and **R2(ABCFGH)**.
-R1 is in 2NF. Check R2 is in 2NF or not.
-R2(ABCFGH) with FDs AB → C, B → F and F → GH
-Candidate key is only AB.
- B → F is a partial dependency. So R2 is not in 2NF

-Next compute B^+, So B^+ = BFGH

-Now decompose R2 into **R21(BFGH)** and **R22(ABC)**.

-R21 and R22 are in 2NF.

-So 2NF relations are

R1(ADEIJ), R21(BFGH) and **R22(ABC)**

3NF:

-Consider the 2NF relations R1(ADEIJ), R21(BFGH) and R22(ABC)

-Consider R1(ADEIJ) with FDs A → DE and D → IJ. Candidate key is A.

-A → D and D → IJ implies A → IJ. R1 has transitive dependency. So R1 is not in 3NF.

-Decompose R1 into **R11(DIJ)** and **R12(ADE)**.

-Consider R21(BFGH) with FDs B → F and F → GH. Candidate key is B.

-B → F and F → GH implies B → GH. R21 has transitive dependency. So R21 is not in 3NF.
-Decompose R21 into **R211(FGH)** and **R212(BF)**.
-Consider R22(ABC) with FD AB → C. R22 is in 3NF.
-So 3NF relations are
R11(DIJ), R12(ADE), R211(FGH), R212(BF) and R22(ABC)

BCNF:

-D is super key in R11, A is super key in R12, F is super key in R211, B is super key in R212 and AB is super key in R22. so no violation of BCNF rule.
-All relations are in BCNF.

Question2:

1. Consider a relation R(ABCDEF) with the FDs A → BC, D → E and AD → F. Decompose the relation up to BCNF.

Ans:

1NF: Given FDs A → BC,
D → E,
AD → F

Find the candidate keys: Look at the missing attributes on the RHS of the FDs. Here A and D are missing. So AD must be a part of the candidate key.

AD^+ = ABCDEF. So AD is a candidate key.

-Next check any replacement of either A or D on the LHS. No

-So only AD is a candidate key.

-Relation is in 1NF, because relation has a candidate key.

2NF:

-Consider the given FDs A → BC, D → E and AD → F.

-A → BC is a partial dependency. So relation is not in 2NF.

-Next compute A^+, So A^+ = ABC.

-Now decompose the given relation into **R1(ABC)** and **R2(ADEF)**.

-R1 is in 2NF. Check R2 is in 2NF or not.

-R2(DEFA) with FDs D → E and AD → F

-Candidate key is only AD.

-D → E is a partial dependency. So R2 is not in 2NF.

-Next compute D^+, So D^+ = DE

-Now decompose the relation R2 into **R21(DE)** and **R22(ADF)**

- R21 and R22 are in 2NF. So 2NF relations are

  R1(ABC), R21(DE) and **R22(ADF)**

3NF:

-Consider the 2NF relations R1(ABC), R21(DE) and R22(ADF).

-Consider R1(ABC) with FD A → BC. It is not transitive dependency, so R1 is in 3NF.

-Consider R21(DE) with FD D → E. It is not transitive dependency, so R21 is in 3NF.

-Consider R22(ADF) with FD AD → F. It is not transitive dependency, so R22 is in 3NF.

BCNF:

-Consider the 3NF relations R1(ABC), R21(DE) and R22(ADF).

-A is super key in R1, D is super key in R21 and AD is super key in R22. so no violation of BCNF rule.
-All relations are in BCNF.

Note: Any relation can be decomposed only into two sub relations at a time.

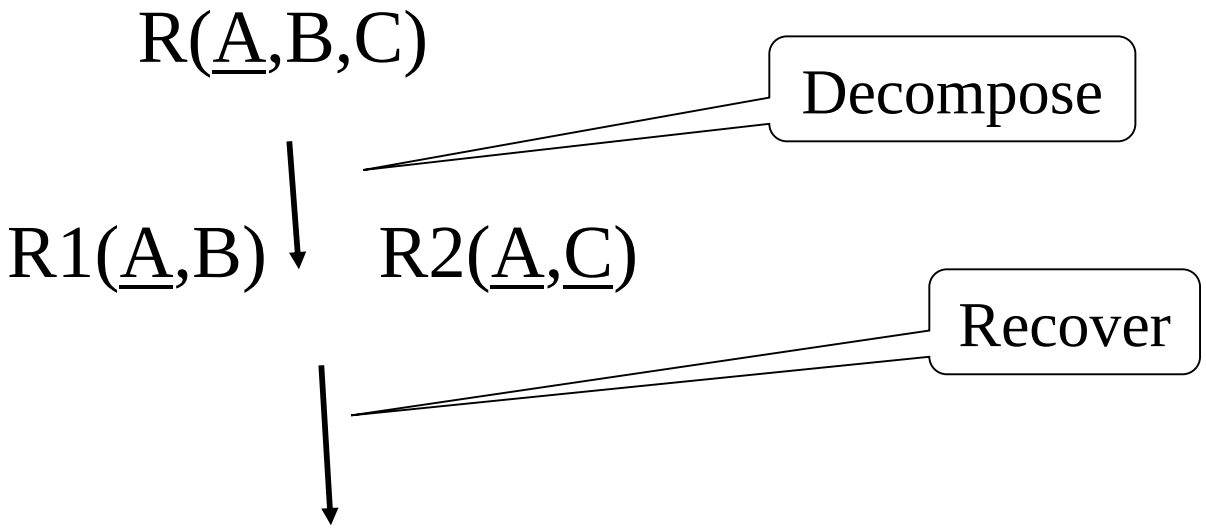
Types of decomposition

▪When a relation is decomposing into smaller relations, that decomposition must satisfy two properties.

1. Lossless join decomposition
2. Dependency preserving decomposition

Lossless Join Decomposition

- A decomposition is *lossless* if we can recover original relation from the decomposed relations



- A decomposition is *lossy* if we can not recover original relation from the decomposed relations

Determining Whether Decomposition Is Lossless Or Lossy:

- Consider a relation R is decomposed into two sub relations R_1 and R_2.

- If all the following conditions satisfy, then the decomposition is lossless.

- If any of these conditions fail, then the decomposition is lossy.

Condition-01:

- Union of both the sub relations must contain all the attributes that are present in the original relation R.

Thus, $\mathbf{R_1 \cup R_2 = R}$

Condition-02:

- Intersection of both the sub relations must not be null. In other words, there must be some common attribute which is present in both the sub relations.

Thus, **R1 ∩ R2 ≠ ∅**

Condition-03:

- Intersection of both the sub relations must be a super key of either R_1 or R_2 or both.

Thus, **$R_1 \cap R_2$ = Super key of R_1 or R_2**

Example1: Consider a relation R(A, B, C, D) with the functional dependencies A → B and C → D. Determine whether the decomposition of R into R_1(A, B) and R_2(C, D) is lossless or lossy.

Ans:

Condition-01:

$R_1 (A , B) \cup R_2 (C , D) = R (A , B , C , D)$

Thus, condition-01 satisfies.

Condition-02:

$R_1 (A , B) \cap R_2 (C , D) = \Phi$

So, condition-02 fails.

Thus, we conclude that the decomposition is lossy.

Example: Consider a relation R(A, B, C, D) with the functional dependency $A \rightarrow BC$. Determine whether the decomposition of R into R_1(A, B, C) and R_2(A, D) is lossless or lossy.

Ans:

Condition-01:

$R_1 (A , B , C) \cup R_2 (A , D) = R (A , B , C , D)$

Thus, condition-01 satisfies.

Condition-02:

$R_1 (A , B , C) \cap R_2 (A , D) = A$

Thus, condition-02 satisfies.

Condition-03:

$R_1 (A , B , C) \cap R_2 (A , D) = A$

A is super key in R_1

Thus, condition-03 satisfies.

- So, decomposition is lossless.

Dependency-Preserving Decomposition

- To define dependency-preserving decomposition precisely, we have to introduce the concept of a projection of FDs

- Let R be a relation schema that is decomposed into two schemas with attribute sets X and Y, and let F be a set of FDs over R

- The **projection of F on X** is the set of FDs in the closure F^+ (not just F) that involve only attributes in X

- We will denote the projection of F on attributes X as F_X

- Note that a dependency $U \rightarrow V$ in F^+ is in F_X only if all the attributes in U and V are in X

- The decomposition of relation schema R with FDs F into schemas with attribute sets X and Y is **dependency-preserving** if $(F_X \cup F_Y)^+ = F^+$

Example: Suppose that a relation R with attributes ABC is decomposed into relations with attributes AB and BC. The set F of FDs over R includes $A \rightarrow B$, $B \rightarrow C$, and $C \rightarrow A$. Is this decomposition dependency-preserving? Is $C \rightarrow A$ preserved?

Ans: given set of FDs $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

$F^+ = F \ U \ \{A \rightarrow C, B \rightarrow A, C \rightarrow B\}$

So $F^+ = \{A \rightarrow B, B \rightarrow C, C \rightarrow A, A \rightarrow C, B \rightarrow A, C \rightarrow B\}$

$F_{AB} = \{A \rightarrow B, B \rightarrow A\}$

$F_{BC} = \{B \rightarrow C, C \rightarrow B\}$

$F_{AB} \ U \ F_{BC} = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B\}$

$(F_{AB} \ U \ F_{BC})^+ = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B, C \rightarrow A, A \rightarrow C\}$

$(F_{AB} \cup F_{BC})^+ = F^+$

So decomposition is dependency preserved

$C \rightarrow A$ is also preserved, because $(F_{AB} \cup F_{BC})^+$ contains $C \rightarrow A$

Multi Valued Dependencies (MVD)

- It is a dependency between attributes (for example, A, B, and C) in a relation, such that for each value of A there is a set of values of B, and a set of values of C. However, the set of values of B and C are independent of each other.

- Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The *multivalued dependency* $\alpha \rightarrow\rightarrow \beta$ holds on R if in any legal relation $r(R)$, for all pairs for tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that:

$$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$$
$$t_3[\beta] = t_1[\beta]$$
$$t_3[R - \beta] = t_2[R - \beta]$$
$$t_4[\beta] = t_2[\beta]$$
$$t_4[R - \beta] = t_1[R - \beta]$$

- Relation shown in Fig 5.12 has two MVDs:
  ENAME $\rightarrow\rightarrow$ PNAME and
  ENAME $\rightarrow\rightarrow$ DNAME

ENAME	PNAME	DNAME
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

Fig 5.12 EMP

- An MVD X →→ Y in R is called a trivial MVD if (a) Y is a subset of X, or (b) X U Y = R

- An MVD that satisfies neither (a) nor (b) is called a nontrivial MVD

- Example: AB →→ B trivial MVD

Fourth Normal Form (4NF)

- A relation is in 4NF if it is in BCNF and contains no MVDs

- BCNF to 4NF involves the removal of the MVDs from the relation by placing the attribute(s) in a new relation along with a copy of the determinant(s)

Example1: The EMP relation of Fig 5.12 is in BCNF but not in 4NF because it contains MVDs ENAME →→ PNAME and ENAME →→ DNAME

- We decompose EMP into EMP_PROJECTS and EMP_DEPENDENTS shown in Fig 5.13

ENAME	PNAME
Smith	X
Smith	Y

EMP_PROJECTS

ENAME	DNAME
Smith	John
Smith	Anna

EMP_DEPENDENTS

Fig 5.13

Example2

Branch_Staff_Client relation

Branch_No	*SName*	*CName*
B3	Ann Beech	Aline Stewart
B3	David Ford	Aline Stewart
B3	Ann Beech	Mike Richie
B3	David Ford	Mike Richie

Branch_Staff relation

Branch_No	*SName*
B3	Ann Beech
B3	David Ford

Branch_Client relation

Branch_No	*CName*
B3	Aline Stewart
B3	Mike Richie

Example3

Employee Name	Skills	Language
Mohan	C Sharp	Hindi
Mohan	Asp.Net	Hindi
Mohan	SQL Server	Hindi
Mohan	C Sharp	English
Mohan	Asp.Net	English
Mohan	SQL Server	English

Employee Name	Skills
Mohan	C Sharp
Mohan	Asp.Net
Mohan	SQL Server

Employee Name	Language
Mohan	Hindi
Mohan	English

4NF