CST204 Database Management System

Module 4

Introduction

Syllabus – Overview

- Module 1 : Introduction and Entity Relationship (ER) model
- Module 2 : Relational Model
- Module 3 : SQL DML and Physical Data Organization
- Module 4 : Normalization
- Module 5: Transaction, concurrency and recovery, recent topics

Introduction

Syllabus – Module 4

- Different anomalies in designing a database
- The idea of normalization
- Functional dependency
- Armstrong’s Axioms (proofs not required)
- Closures and their computation
- Equivalence of Functional Dependencies (FD)
- Minimal Cover (proofs not required)
- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce Codd Normal Form (BCNF)
- Lossless join and dependency preserving decomposition
- Algorithms for checking Lossless Join (LJ) and Dependency Preserving (DP) properties

Normalization

Anomalies

- Anomalies are caused when there is too much redundancy in the database's information
- Anomalies can often be caused when the tables that make up the database suffer from poor construction
- Poor table design will become evident if, when the designer creates the database, he doesn't identify the entities that depend on each other for existence
- There are three types of Data Anomalies: Update Anomalies, Insertion Anomalies, and Deletion Anomalies

Normalization

Types of Anomalies

Update Anomalies

- It happen when the person charged with the task of keeping all the records current and accurate, is asked, for example, to change an employee’s title due to a promotion
- If the data is stored redundantly in the same table, and the person misses any of them, then there will be multiple titles associated with the employee
- The end user has no way of knowing which is the correct title
- Let say we have 10 columns in a table out of which 2 are called employee Name and employee address

Normalization

Types of Anomalies

Update Anomalies

- Now if one employee changes it’s location then we would have to update the table
- But the problem is, if the table is not free from anomalies one employee can have multiple entries and while updating all of those entries one of them might get missed

Normalization

Types of Anomalies

Insert Anomalies

- It happen when inserting vital data into the database is not possible because other data is not already there
- For example, if a system is designed to require that a customer be on file before a sale can be made to that customer, but we cannot add a customer until they have bought something, then we have an insert anomaly
- Another example is, suppose Jerry is a new Student with department id 6
- There is no Department with this Dept_ID 6
- Hence , the anomaly
- The usual behaviour should be a new department id with 6 and only then student could have it

Normalization

Types of Anomalies

Deletion Anomalies

- Deletion Anomalies happen when the deletion of unwanted information causes desired information to be deleted as well
- For example, if a single database record contains information about a particular product along with information about a salesperson for the company and the salesperson quits, then information about the product is deleted along with salesperson information
- Let’s say we have student’s information and courses they have taken as follows (student ID,Student Name, Course, address)

Normalization

Types of Anomalies

Deletion Anomalies

- If any student leaves the school then the entry related to that student will be deleted
- However, that deletion will also delete the course information even though course depends upon the school and not the student

Normalization

Types of Anomalies

Deletion Anomalies

- If any student leaves the school then the entry related to that student will be deleted
- However, that deletion will also delete the course information even though course depends upon the school and not the student
- Redundancy is the root cause of all anomalies existing in database
- In order to escape from anomalies, we use a technique called normalization
- Normalization is the process of organizing the data in the database

Normalization

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
- Redundancy means the duplication of data i.e., same data is occurring multiple times in database
- For normalization we use two concepts – keys and functional dependency

Normalization

Normalization - Keys

- Keys in DBMS are used to identify each rows/tuples/records uniquely
- With respect to below table, we have following keys

Name	Marks	Dept	Course
a	78	CS	C1
b	60	EE	C1
a	78	CS	C2
b	60	EE	C3
c	80	IT	C2

Super Key

- They are used to identify each rows/tuples/records uniquely
- Maximum super keys in any relation is $2^n - 1$

Normalization

Normalization - Keys

Candidate Key

- It is a super key whose proper subset is not a super key
- It is a minimal super key

Primary Key

- It is the one candidate key which has no null value
- Remaining candidate keys are called secondary key or alternate keys
- In every relation we have many super keys and candidate keys but only one primary key

Normalization

Normalization - Keys

- In a relation R(A, B, C); A is the candidate key. Find the number of super keys in the relation
- If A is the candidate key

    Then the super keys will be A, AB, AC, ABC = 4
- If AC is the candidate key

    Then the super keys will be AC, ACB = 2

Normalization

Normalization – Functional Dependencies

- Functional Dependency is a set of constraints on various attributes of a relational
- It is used to specify a formal measure of the 'goodness' of relational designs
- It is a constraint between two sets of attributes
- Functional Dependency and keys are used to define normal forms for relations
- Let R = {A1, A2, …., An}
- A set of attributes X functionally determines a set of attributes Y if the value of X determines a unique value for Y

Normalization

Normalization – Functional Dependencies

- It is denoted by X -> Y
- It means between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples
- For example, employee_id → name means employee_id functionally determines the name of the employee
- As another example in a timetable database, {student_id, time} → {lecture_room}, student ID and time determine the lecture room where the student should be
- A function dependency A → B means for all instances of a particular value of A, there is the same value of B

Normalization

Normalization – Functional Dependencies

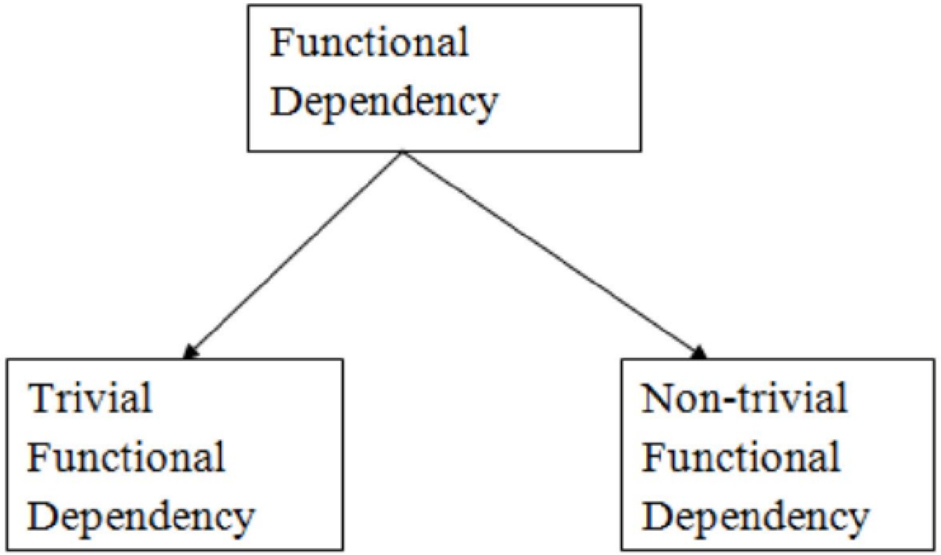
- For example in the below table $A \longrightarrow B$ is true, but $B \longrightarrow A$ is not true as there are different values of A for B = 3

A	B
1	3
2	3
4	0
1	3
4	0

Normalization

Normalization – Functional Dependencies

- There are two types of functional dependencies – trivial functional dependency and non – trivial functional dependency



Normalization

Normalization – Trivial Functional Dependencies

- $A \rightarrow B$ has trivial functional dependency if B is a subset of A
- The following dependencies are also trivial like: $A \rightarrow A$, $B \rightarrow B$
- Consider a table with two columns Employee_Id and Employee_Name then

{Employee_id, Employee_Name} $\rightarrow$ Employee_Id is a trivial functional dependency as Employee_Id is a subset of {Employee_Id, Employee_Name}

- Also, Employee_Id $\rightarrow$ Employee_Id and Employee_Name $\rightarrow$ Employee_Name are trivial dependency

Normalization

Normalization – Non -Trivial Functional Dependencies

- $A \longrightarrow B$ has a non-trivial functional dependency if B is not a subset of A
- For example, ID $\longrightarrow$ Name and Name $\longrightarrow$ DOB is non – trivial functional dependency

Normalization

Normalization – Armstrong’s Axioms

- The Armstrong's axioms are the basic inference rule
- Armstrong's axioms are used to conclude functional dependencies on a relational database
- The inference rule is a type of assertion
- It can apply to a set of FD(functional dependency) to derive other FD
- Using the inference rule, we can derive additional functional dependency from the initial set
- The Functional dependency has 6 types of inference rule

Normalization

Normalization – Armstrong's Axioms – Inference Rules

Reflexive Rule (IR1)

- In the reflexive rule, if Y is a subset of X, then X determines Y
- If $X \supseteq Y$ then $X \rightarrow Y$
- Example:
- X = {a, b, c, d, e}
- Y = {a, b, c}

Normalization

Normalization – Armstrong's Axioms – Inference Rules

Augmentation Rule (IR2)

- 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 $XZ \rightarrow YZ$
- Example:
- For R(ABCD), if $A \rightarrow B$ then $AC \rightarrow BC$

Normalization

Normalization – Armstrong's Axioms – Inference Rules

Transitive Rule (IR3)

- 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 $X \rightarrow Z$

Union Rule (IR4)

- 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 $X \rightarrow YZ$

Normalization

Normalization – Armstrong's Axioms – Inference Rules

Decomposition Rule (IR5)

- 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 determines Y and X determines Z separately
- If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Pseudo transitive Rule (IR6)

- 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 $XZ \rightarrow W$

Normalization

- **Normalization – Closures**

- 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 axioms
- If F is a functional dependency then closure of functional dependency can be denoted using {F}+

Normalization

Normalization – Closures

- Find the closures for relations R(A, B, C, D, E) with following functional dependencies FD {$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$}

Normalization

Normalization – Closures

- Consider a relation R (A , B , C , D , E , F , G) with the functional dependencies-
- $A \longrightarrow BC$
- $BC \longrightarrow DE$
- $D \longrightarrow F$
- $CF \longrightarrow G$

Normalization

Normalization – Closures

A+ = { A }

- = { A , B , C } (Using A ⟶ BC)
- = { A , B , C , D , E } (Using BC ⟶ DE)
- = { A , B , C , D , E , F } (Using D ⟶ F)
- = { A , B , C , D , E , F , G } (Using CF ⟶ G)
- Thus,
- A+ = { A , B , C , D , E , F , G }

Normalization

Normalization – Closures

D+ = { D }

= { D , F } (Using D $\longrightarrow$ F)

We can not determine any other attribute using attributes D and F contained in the result set.

Thus,

D+ = { D , F }

Normalization

Normalization – Closures

$\{ B , C \}^+ = \{ B , C \}$

$= \{ B , C , D , E \}$ (Using $BC \longrightarrow DE$)

$= \{ B , C , D , E , F \}$ (Using $D \longrightarrow F$)

$= \{ B , C , D , E , F , G \}$ (Using $CF \longrightarrow G$)

Thus

$\{ B , C \}^+ = \{ B , C , D , E , F , G \}$

Normalization

Normalization – Closures

Finding the Keys Using Closure

- Super Key
- If the closure result of an attribute set contains all the attributes of the relation, then that attribute set is called as a super key of that relation
- Thus, we can say that the closure of a super key is the entire relation schema
- In the previous example, the closure of attribute A is the entire relation schema
- Thus, attribute A is a super key for that relation

Normalization

Normalization – Closures

- Candidate Key-
- If there exists no subset of an attribute set whose closure contains all the attributes of the relation, then that attribute set is called as a candidate key of that relation
- In the previous example,
- No subset of attribute A contains all the attributes of the relation
- Thus, attribute A is also a candidate key for that relation

Normalization

Normalization – Equivalence of Functional Dependency

- In DBMS, two different sets of functional dependencies for a given relation may or may not be equivalent
- If R1 and R2 are the two sets of functional dependencies, then following 3 cases are possible-

1. If all Functional dependencies of R1 can be derived from Functional dependencies present in R2, we can say that R2 is a subset of R1 ($R2 \supset R1$).
2. If all Functional dependencies of R2 can be derived from Functional dependencies present in R1, we can say that R1 is a subset of R2 ($R1 \supset R2$).
3. If 1 and 2 both are satisfied, then R1 = R2.

Normalization

Normalization – Equivalence of Functional Dependency

Case 1) Determining Whether R2 ⊃ R1 or not

- Step 1)
- Take into consideration, the functional dependencies of set R1
- For every functional dependency P→ Q, find by using the functional dependencies of set R1, the closure of P
- Step 2)
- Take into consideration, the functional dependencies of set R2
- For every functional dependency P→ Q, find by using the functional dependencies of set R2, the closure of P
- Step 3)
- Compare the results of Step 1 and Step 2
- If the functional dependency of set R2 has determined all those attributes that were determined by the functional dependencies of set R1, then it means R2 ⊃ R1
- Thus, we conclude R2 is a subset of R1 (R2 ⊃ R1) otherwise not

Normalization

Normalization – Equivalence of Functional Dependency

Case 2) Determining Whether R1 $\supset$ R2 or not

- Step 1)
- Take into consideration the functional dependencies of set R2
- For every functional dependency P $\rightarrow$ Q, find by using the functional dependencies of set R2, the closure of P
- Step 2)
- Take into consideration the functional dependencies of set R1
- For every functional dependency P $\rightarrow$ Q, find by using the functional dependencies of set R1, the closure of P
- Step 3)
- Compare the results of Step 1 and Step 2
- If the functional dependency of set R1 has determined all those attributes that were determined by the functional dependencies of set R2, then it means R1 $\supset$ R2
- Thus, we conclude that R1 is a subset of R2 (R1 $\supset$ R2) otherwise not

Normalization

Normalization – Equivalence of Functional Dependency

- Case 2) Determining Whether R1 ⊃ R2 or not
- Case 3) Determining Whether Both R1 and R2 satisfy each other or not
- If R2 is a subset of R1 and R1 is a subset of R2, then both R1 and R2 satisfied each other
- Thus, if both the above cases satisfied, we conclude that R1 = R2

Normalization

Normalization – Equivalence of Functional Dependency

A relation R (P, Q, U, S, and T) is having two functional dependencies sets R1 and R2, which is shown as

Set R1:	Set R2:
$P \longrightarrow C$	$P \longrightarrow QU$
$PQ \longrightarrow U$	$S \longrightarrow PT$
$S \longrightarrow T$	

Normalization

Normalization – Equivalence of Functional Dependency

Case 1) Determining Whether R2 $\supset$ R1 or not

Step 1)

(P)+ = {P, Q, U} // closure of left side of P $\longrightarrow$ QU using set R1

(S)+ = {P, Q, U, S, T} // closure of left side of S $\longrightarrow$ PT using set R1

Step 2)

(P)+ = {P, Q, U} // closure of left side of P $\longrightarrow$ QU using set R2

(S)+ = {P, Q, U, S, T} // closure of left side of S $\longrightarrow$ PT using set R2

Step 3)

Comparing the results of Step 1 and Step 2, we find,

Functional dependencies of set R2 can determine all the attributes which have been determined by the functional dependencies of set R1

Thus, we conclude R2 is a subset of R1 i.e. R2 $\supset$ R1

Normalization

Normalization – Equivalence of Functional Dependency

Case 2) Determining Whether R1 ⊃ R2 or not

Step 1)

(P)+ = {P, Q, U} // closure of left side of P→ Q using set R2

(PQ)+ = {P, Q, U} // closure of left side of PQ → U using set R2

(S)+ = {P, Q, U, S, T} // closure of left side of S → PU and S → T using set R2

Step 2)

(P)+ = {P, Q, U} // closure of left side of P→ Q using set R1.

(PQ)+ = {P, Q, U} // closure of left side of PQ → U using set R1.

(S)+ = {P, Q, U, S, T} // closure of left side of S → PU and S → T using set R1

Step 3)

Comparing the results of Step 1 and Step 2, we find,

Functional dependencies of set R1 can determine all the attributes which have been determined by the functional dependencies of set R2

Thus, we conclude R1 is a subset of R2 i.e. R1 ⊃ R2

Normalization

Normalization – Equivalence of Functional Dependency

Case 3) Determining Whether Both R1 and R2 satisfy each other or not

From Step 1, we conclude R2 $\supset$ R1

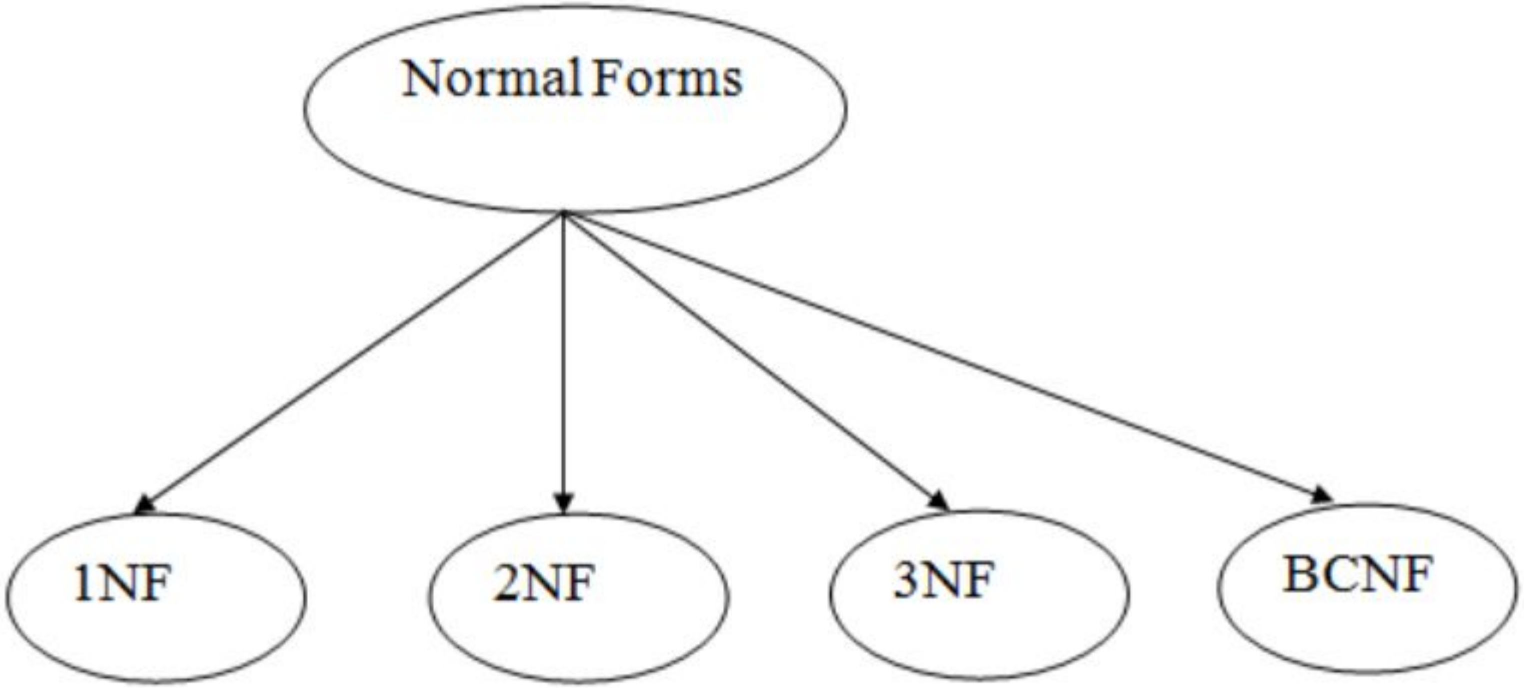
From Step 2, we conclude R1 $\supset$ R2

Thus, we conclude that both R1 and R2 satisfied each other i.e. R1 = R2

Normalization

Normalization – Normal Forms

- There are four types of Normalization



Normalization

Normalization - 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

Normalization

Normalization - First Normal Form (1NF)

EMPLOYEE table:

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

Normalization

Normalization - First Normal Form (1NF)

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

Normalization

Normalization - First Normal Form (1NF)

Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD PHONE. Its decomposition into 1NF has been shown in table 2

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	INDIA
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

Normalization

Normalization - Second Normal Form (2NF)

- To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency
- A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table
- Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency

Normalization

Normalization - Second Normal Form (2NF)

- Example 1 – Consider table-3 as following below

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000

- Note that, there are many courses having the same course fee

Normalization

Normalization - Second Normal Form (2NF)

- Here, COURSE_FEE cannot alone decide the value of COURSE_NO or STUD_NO;
- COURSE_FEE together with STUD_NO cannot decide the value of COURSE_NO;
- COURSE_FEE together with COURSE_NO cannot decide the value of STUD_NO;
- Hence, COURSE_FEE would be a non-prime attribute, as it does not belong to the one only candidate key {STUD_NO, COURSE_NO} ;
- But, COURSE_NO -> COURSE_FEE , i.e., COURSE_FEE is dependent on COURSE_NO, which is a proper subset of the candidate key
- Non-prime attribute COURSE_FEE is dependent on a proper subset of the candidate key, which is a partial dependency and so this relation is not in 2NF

Normalization

Normalization - Second Normal Form (2NF)

- To convert the above relation to 2NF, we need to split the table into two tables such as :
- Table 1: STUD_NO, COURSE_NO
- Table 2: COURSE_NO, COURSE_FEE

Table 1

STUD_NO	COURSE_NO
1	C1
2	C2
1	C4
4	C3
4	C1

Table 2

COURSE_NO	COURSE_FEE
C1	1000
C2	1500
C3	1000
C4	2000
C5	2000

Normalization

Normalization - Second Normal Form (2NF)

- 2NF tries to reduce the redundant data getting stored in memory
- For instance, if there are 100 students taking C1 course, we don't need to store its Fee as 1000 for all the 100 records, instead once we can store it in the second table as the course fee for C1 is 1000

Normalization

Normalization - Third Normal Form (3NF)

- A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form
- A relation is in 3NF if at least one of the following condition holds in every non-trivial functional dependency X –> Y
  - X is a super key
  - Y is a prime attribute (each element of Y is part of some candidate key)
- Transitive dependency – If A->B and B->C are two FDs then A->C is called transitive dependency

Normalization

Normalization - Third Normal Form (3NF)

- Consider the table below:

STUD_NO	STUD_NAME	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	HARYANA	INDIA	20
2	RAM	PUNJAB	INDIA	19
3	SURESH	PUNJAB	INDIA	21

Table 4

- FD set: {STUD_NO -> STUD_NAME, STUD_NO -> STUD_STATE, STUD_STATE -> STUD_COUNTRY, STUD_NO -> STUD_AGE}
- Candidate Key: {STUD_NO}

Normalization

Normalization - Third Normal Form (3NF)

- For this relation in table 4, STUD_NO -> STUD_STATE and STUD_STATE -> STUD_COUNTRY are true
- So STUD_COUNTRY is transitively dependent on STUD_NO
- It violates the third normal form. To convert it in third normal form, we will decompose the relation STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY_STUD_AGE) as:
- STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_AGE)
- STATE_COUNTRY (STATE, COUNTRY)

Normalization

Normalization - Boyce - Codd Normal Form (BCNF)

- BCNF is also called 3.5NF as it is an upgraded version of 3NF
- A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key
- A relation is in BCNF iff in every non-trivial functional dependency X –> Y, X is a super key
- Until now we have seen prime attribute → non prime attribute (case of functional dependency); part of primary key → non prime attribute (case of partial dependency); non prime attribute → non prime attribute (case of transitive dependency)
- What will happen is non prime attribute → prime attribute ? (then the relation is not in BCNF)

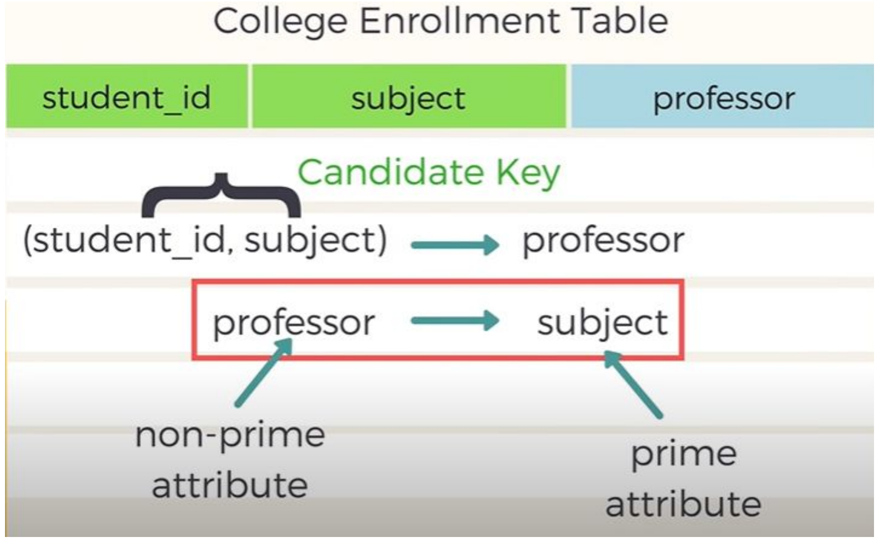
Normalization

Normalization - Boyce - Codd Normal Form (BCNF)

- Example: consider the following table

College Enrollment Table

student_id	subject	professor
101	Java	P. Java
101	C++	P. Cpp
102	Java	P. Java2
103	C#	P. Chash
104	Java	P. Java



- Here, professor (non-prime attribute) determines subject (prime attribute), so the table is not in BCNF

Normalization

Normalization - Boyce - Codd Normal Form (BCNF)

- To convert the table in to BCNF, we have to split the table into two