International Islamic University Chittagong (IIUC)

Department of Computer and Communication Engineering

Program: **B.sc (Eng.)**
Course Code: **CCE–3505**
Type: **Theory**
Segment: **Final Term-Segment 5**

Baizid MD Ashadzzaman
Designation: **Adjunct Lecturer**
Email: **baizid.md.ashadzzaman@gmail.com**
Phone: **8801862420119**

Course Contents:

Contents	Note

Functional Dependency and Normalization

Functional Dependency

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.

For example:

Assumewe 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 Dependencies in DBMS

1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Multivalued functional dependency
4. Transitive functional dependency

1. Trivial Functional Dependency

In **Trivial Functional Dependency**, a dependent is always a subset of the determinant. i.e. If **X → Y** and **Y is the subset of X**, then it is called trivial functional dependency **Example:**

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, **{roll_no, name} → name** is a trivial functional dependency, since the dependent **name** is a subset of determinant set **{roll_no, name}.** Similarly, **roll_no → roll_no** is also an example of trivial functional dependency.

2. Non-trivial Functional Dependency

In **Non-trivial functional dependency**, the dependent is not a subset of the determinant. i.e. If **X → Y** and **Y is not a subset of X**, then it is called Non-trivial functional dependency.

Example:

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, **roll_no → name** is a non-trivial functional dependency, since the dependent **name** is **not a subset of** determinant **roll_no.** Similarly, **{roll_no, name} → age** is also a non-trivial functional dependency, since **age** is **not a subset of**

{roll_no, name}

3. Multivalued Functional Dependency

In Multivalued functional dependency, entities of the dependent set are not dependent on each other. i.e. If **a → {b, c}** and there exists **no functional dependency** between **b and c**, then it is called a **multivalued functional dependency. For example,**

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18
45	abc	19

Here, **roll_no → {name, age}** is a multivalued functional dependency, since the dependents **name** & **age** are **not dependent** on each other(i.e. **name → age** or **age → name doesn't exist !**)

4. Transitive Functional Dependency

In transitive functional dependency, dependent is indirectly dependent on determinant. i.e. If **a → b** & **b → c**, then according to axiom of transitivity, **a → c**. This is a **transitive functional dependency.**

For example,

enrol_no	name	dept	building_no
42	abc	CO	4
43	pqr	EC	2
44	xyz	IT	1
45	abc	EC	2

Here, **enrol_no → dept** and **dept → building_no.** Hence, according to the axiom of transitivity, **enrol_no → building_no** is a valid functional dependency. This is an indirect functional dependency, hence called Transitive functional dependency.

Normalization?

- Normalization is the process of organizing the data in the database.
- Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships. o The normal form is used to reduce redundancy from the database table.

Objective of Normalization

1. It is used to remove the duplicate data and database anomalies from the relational table.
2. Normalization helps to reduce redundancy and complexity by examining new data types used in the table.
3. It is helpful to divide the large database table into smaller tables and link them using relationship.
4. It avoids duplicate data or no repeating groups into a table.
5. It reduces the chances for anomalies to occur in a database

Types of Normalization

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce and Codd Normal Form (BCNF)

First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following 4 rules:

- It should only have single(**atomic or Single**) valued attributes/columns.
- Values stored in a column should be of the same domain.
- All the columns in a table should have unique names.
- And the order in which data is stored should not matter.
- 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

Or

EMP_ID	EMP_NAME	EMP_PHONE-1	EMP_PHONE-2	EMP_STATE
14	John	7272826385	9064738238	UP
20	Harry	8574783832		Bihar
12	Sam	7390372389	8589830302	Punjab

For more practice see this video :https://www.youtube.com/watch?v=pPLIhlzPH0g **Second Normal Form (2NF)**

- It should be in the First Normal form.
- It should not have **Partial Dependency.** In the second normal form, all non-key attributes are fully functional dependent on the primary key

What is Partial Dependency?

When a table has a primary key that is made up of two or more columns, then all the columns in that table should depend on the entire primary key or on a part of it. If any column depends on a part of the primary key then we say that we have Partial dependency in the table.

Example:

student_id	student_name	branch	Subject_id	Subject_name	marks	teacher_name
1	Akon	CSE	1	C Language	70	Miss. C
2	Bkon	Mechanical	2	DSA	82	Mr. D
3	Ckon	BBA	3	Operating System	66	Mr. A

To convert the given table into 2NF, we decompose it into two tables:

Updated **Subject** table

subject_id	subject_name	teacher_name
1	C Language	Miss. C
2	DSA	Mr. D
3	Operating System	Mr. Op

Updated **Score** table:

student_id	subject_id	marks
1	1	70
2	2	82
3	3	66

Third Normal Form (3NF)

A table is said to be in the Third Normal Form when,

- It satisfies the First Normal Form and the Second Normal form. And, it doesn't have Transitive Dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

What is Transitive Dependency?

In a table we have some column that acts as the primary key and other columns depends on this column. But what if a column that is not the primary key depends on another column that is also not a primary key or part of it? Then we have Transitive dependency in our table.

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

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_ZIP). 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. o For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example 1: 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:

EMP_ID → EMP_COUNTRY

EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into 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:

EMP_ID → EMP_COUNTRY

EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID
For the second table: EMP_DEPT
For the third table: {EMP_ID, EMP_DEPT}

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

Example 2: Consider a relation R with attributes (student, subject, teacher).

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

(student, Teacher) -> subject (student, subject) -> Teacher

Teacher -> subject

Candidate keys are (student, teacher) and (student, subject).

A relation R is in BCNF if for every non-trivial FD X->Y, X must be a key. So the above relation is not in BCNF, because in the FD (teacher->subject), teacher is not a key.

To convert BCNF R is divided into two relations R1(Teacher, subject) and R2(student, Teacher).

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

Closure of Functional Dependency in DBMS

Closure of a functional dependency in database design refers to the set of functional dependencies that can be inferred from a given set of functional dependencies. In other words, it is the process of finding all the functional dependencies that can be derived from the original set of functional dependencies, using the properties of functional dependencies. The closure of functional dependencies is important in normalization, as it helps identify redundant data and improve the efficiency of database operations.

Example:

Consider a relation R(A,B,C,D,E) having below mentioned functional dependencies.

FD1 : A → BC

FD2 : C → B FD3 : D → E FD4 : E → D

Now, we need to calculate the closure of attributes of the relation R. The closures will be:

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

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

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

$\{D\}^+ = \{D, E\}$

$\{E\}^+ = \{E\}$

Closure Of Functional Dependency : Calculating Candidate Key

A Candidate Key of a relation is an attribute or set of attributes that can determine the whole relation or contains all the attributes in its closure.

Example-1 : Consider the relation R(A,B,C) with given functional dependencies :

FD1 : A → B

FD2 : B → C

Now, calculating the closure of the attributes as :

{A}+ = {A, B, C}

{B}+ = {B, C}

{C}+ = {C}

Clearly, “A” is the candidate key as, its closure contains all the attributes present in the relation “R”.

Example-2 : Consider another relation R(A, B, C, D, E) having the Functional dependencies :

FD1 : A → BC

FD2 : C → B FD3 : D → E FD4 : E → D

Now, calculating the closure of the attributes as :

{A}+ = {A, B, C}

{B}+ = {B}

{C}+ = {C, B}

{D}+ = {E, D}

{E}+ = {E, D}

In this case, a single attribute is unable to determine all the attribute on its own like in previous example. Here, we need to combine two or more attributes to determine the candidate keys.

{A, D}+ = {A, B, C, D, E}

{A, E}+ = {A, B, C, D, E}

Hence, "AD" and "AE" are the two possible keys of the given relation “R”. Any other combination other than these two would have acted as extraneous attributes.

NOTE : Any relation “R” can have either single or multiple candidate keys.

Redundancy

Redundancy means having multiple copies of the same data in the database. This problem arises when a database is not normalized. Suppose a table of student details attributes is: student Id, student name, college name, college rank, and course opted.

Id	Name	Contact	College	Course	Rank
1	A	213	IIUC	CSE	3
2	B	123	IIUC	CSE	3
3	C	433	IIUC	CSE	3

4	D	223	IIUC	CSE	3

The values of attribute college name, college rank, and course is being repeated which can lead to problems. Problems caused due to redundancy are:

- Insertion anomaly
- Deletion anomaly
- Updation anomaly

Insertion anomaly: If a student detail has to be inserted whose rank is not being decided yet then insertion will not be possible till the time rank is decided for the student.

Id	Name	Contact	College	Course	Rank
1	A	213	IIUC	CSE	

Deletion Anomaly: If the details of students in this table are deleted then the details of the college will also get deleted which should not occur by common sense. This anomaly happens when the deletion of a data record results in losing some unrelated information that was stored as part of the record that was deleted from a table.

It is not possible to delete some information without losing some other information in the table as well.

Updation Anomaly: Suppose the rank of the college changes then changes will have to be all over the database which will be time-consuming and computationally costly.

Functional Dependency

In a relational database management, functional dependency is a concept that specifies the relationship between two sets of attributes where one attribute determines the value of another attribute. It is denoted as X → Y, where the attribute set on the left side of the arrow, X is called Determinant, and Y is called the Dependent.

Types of Functional Dependencies in DBMS

Trivial functional dependency

Non-Trivial functional dependency

Multivalued functional dependency

Transitive functional dependency

1. Trivial Functional Dependency

In Trivial Functional Dependency, a dependent is always a subset of the determinant. i.e. If X →

Y and Y is the subset of X, then it is called trivial functional dependency

Example: Table: [roll, name, age]

Here, {roll_no, name} → name is a trivial functional dependency, since the dependent name is a subset of determinant set {roll_no, name}. Similarly, roll_no → roll_no is also an example of trivial functional dependency.

2. Non-trivial Functional Dependency

In Non-trivial functional dependency, the dependent is strictly not a subset of the determinant. i.e. If X → Y and Y is not a subset of X, then it is called Non-trivial functional dependency.

Here, roll_no → name is a non-trivial functional dependency, since the dependent name is not a subset of determinant roll_no. Similarly, {roll_no, name} → age is also a non-trivial functional dependency, since age is not a subset of {roll_no, name}

3. Multivalued Functional Dependency

In Multivalued functional dependency, entities of the dependent set are not dependent on each other. i.e. If a → {b, c} and there exists no functional dependency between b and c, then it is called a multivalued functional dependency

Here, roll_no → {name, age} is a multivalued functional dependency, since the dependents name & age are not dependent on each other(i.e. name → age or age → name doesn't exist !)

4. Transitive Functional Dependency

In transitive functional dependency, dependent is indirectly dependent on determinant. i.e. If a

→ b & b → c, then according to axiom of transitivity, a → c. This is a transitive functional dependency.

Example: table [enrol_no name dept building_no]

Here, enrol_no → dept and dept → building_no. Hence, according to the axiom of transitivity, enrol_no → building_no is a valid functional dependency. This is an indirect functional dependency, hence called Transitive functional dependency.

Closure of Functional Dependency in DBMS

Closure of a functional dependency in database design refers to the set of functional dependencies that can be inferred from a given set of functional dependencies. In other words, it is the process of finding all the functional dependencies that can be derived from the original set of functional dependencies, using the properties of functional dependencies. The closure of functional dependencies is important in normalization, as it helps identify redundant data and improve the efficiency of database operations.

Example:

Consider a relation R(A,B,C,D,E) having below mentioned functional dependencies.

FD1 : A BC

FD2 : C →→→→ B

FD3 : DE FD4 : E D

Now, we need to calculate the closure of attributes of the relation R. The closures will be:

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

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

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

$\{D\}^+ = \{D, E\}$

$\{E\}^+ = \{E\}$

Closure Of Functional Dependency : Calculating Candidate Key

A Candidate Key of a relation is an attribute or set of attributes that can determine the whole relation or contains all the attributes in its closure.

Example-1 : Consider the relation R(A,B,C) with given functional dependencies :

FD1 : A B

FD2 : B →→ C

Now, calculating the closure of the attributes as :

{A}+ = {A, B, C}

{B}+ = {B, C}

{C}+ = {C}

Clearly, “A” is the candidate key as, its closure contains all the attributes present in the relation “R”.

Example-2 : Consider another relation R(A, B, C, D, E) having the Functional dependencies :

FD1 : A BC

FD2 : C →→→ B

FD3 : D E

FD4 : E → D

Now, calculating the closure of the attributes as :

{A}+ = {A, B, C}

{B}+ = {B}

{C}+ = {C, B}

{D}+ = {E, D}

{E}+ = {E, D}

In this case, a single attribute is unable to determine all the attribute on its own like in previous example. Here, we need to combine two or more attributes to determine the candidate keys.

{A, D}+ = {A, B, C, D, E}

{A, E}+ = {A, B, C, D, E}

Hence, "AD" and "AE" are the two possible keys of the given relation “R”. Any other combination other than these two would have acted as extraneous attributes.

NOTE : Any relation “R” can have either single or multiple candidate keys.

Closure Of Functional Dependency : Key Definitions

Prime Attributes : Attributes which are indispensable part of candidate keys. For example : “A, D, E” attributes are prime attributes in above example-2.

Non-Prime Attributes : Attributes other than prime attributes which does not take part in formation of candidate keys. For example.

For example : Consider the relation R(A, B, C, D) with functional dependencies :

FD1 : A BC

FD2 : B C

FD3 : D C

Here, Candidate key can be “AD” only. Hence, Prime Attributes : A,
D.

Non-Prime Attributes : B, C

Redundancy

Redundancy means having multiple copies of the same data in the database. This problem arises when a database is not normalized. Suppose a table of student details attributes is: student Id, student name, college name, college rank, and course opted.

Id	Name	Contact	College	Course	Rank
1	A	213	IIUC	CSE	3
2	B	123	IIUC	CSE	3
3	C	433	IIUC	CSE	3
4	D	223	IIUC	CSE	3

As it can be observed that values of attribute college name, college rank, and course is being repeated which can lead to problems. Problems caused due to redundancy are:

Insertion anomaly

Deletion anomaly

Updation anomaly

Insertion anomaly: If a student detail has to be inserted whose rank is not being decided yet then insertion will not be possible till the time rank is decided for the student.

Id	Name	Contact	College	Course	Rank
1	A	213	IIUC	CSE	

Deletion Anomaly: If the details of students in this table are deleted then the details of the college will also get deleted which should not occur by common sense. This anomaly happens when the deletion of a data record results in losing some unrelated information that was stored as part of the record that was deleted from a table.

It is not possible to delete some information without losing some other information in the table as well.

Updation Anomaly: Suppose the rank of the college changes then changes will have to be all over the database which will be time-consuming and computationally costly.

Normalization:
Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. So, it helps to minimize the redundancy in relations. Normal forms are used to eliminate or reduce redundancy in database tables.

What is 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 undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships. ● The normal form is used to reduce redundancy from the database table.

Types of Normal Forms:

- **Unnormalized form or UNF**

- **First Normal Form or 1NF**
- **Second Normal Form or 2NF**
- **Third Normal Form or 3NF**
- Elementary key normal form or EKNF
- **Boyce Codd Normal Form or BCNF**
- Fourth normal form or 4NF
- Essential tuple normal form or ETNF
- Fifth normal form or 5NF
- Domain-key normal form or DKNF
- Sixth normal form or 6NF

1. Unnormalized form or UNF:

It is the simplest database model also known as non-first normal form (NF2).A UNF model will suffer problems like data redundancy thus it lacks the efficiency of database normalization.

Id	Name	Course
1	A	Physics Chemistry
2	B	Math Physics
3	C	Math English

In this above example data in unnormalized-form because this table contains multivalued attributes in the course tuple. But there are several advantages also present for the unnormalized form:

- UNF can deal with the complex data structures,
- querying in UNF is simpler, ● Restructuring the data is easier.

2. First Normal Form or 1NF:

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

Id	Name	Course
1	A	C1, C2
2	B	C1
3	C	C2

In the above table Course is a multi-valued attribute so it is not in 1NF. Below Table is in 1NF as there is no multi-valued attribute **Convert above table to 1NF:**

Id	Name	Course

1	A	C1
1	A	C2
2	B	C1
3	C	C2

3. Second Normal Form or 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.

Table: Grade

Id	Name	Address	C_id	C_name	Credit	Grade
1	A	CH	101	C	2	A
1	A	CH	102	JAVA	3	B
2	B	KH	101	C	2	A
2	B	KH	102	JAVA	3	C

Above this table, Primary Key- (Id+C_id)

Partial Dependency: C_name, Credit only Depends on C_id. And also Name, Address only depends on Id **but not both Id and C_id equally**.

Convert above table to 2NF:

Id	Name
1	A
2	B

Table: Student Primary Key: Id

C_id	C_name	Credit
101	C	2
102	JAVA	3

Table: Course Primary Key: C_id

Id	C_id	Grade
1	2	A

1	3	B
2	2	A
2	3	C

Table: Grade Primary Key: (Id,C_id)

4. Third Normal Form or 3NF:

A relation is said to be in third normal form, if we **did not have any transitive dependency** for non-prime attributes. The basic condition with the Third Normal Form is that, the relation must be in **Second Normal Form**.

Transitive: When an indirect relationship causes functional dependency it is called Transitive Dependency.

If P -> Q and Q -> R is true, then P-> R is a transitive dependency

C_name	Teacher_id	Teacher_name	Credit
C	1	JAA	3
C Lab	2	MNA	1
Java	1	JAA	3
Java Lab	2	MNA	1

The above table is not in 3NF because it has a transitive functional dependency −

[C_name -> Teacher_id] [Teacher_id -> Teacher_name]

Therefore, the following has transitive functional dependency. [C_name -> Teacher_name]

Above tabe convert 3NF:

Table: Teacher

Table: Teaches

C_name	Teacher_id	Credit
C	1	3
C Lab	2	1
Java	1	3
Java Lab	2	1

Teacher_id	Teacher_name
1	JAA
2	MNA

4. Boyce-Codd Normal Form (BCNF) It should be in the Third Normal Form.

And, for any dependency A → B, A should be a super key.

The second point sounds a bit tricky, right? In simple words, it means, that for a dependency A → B, A cannot be a non-prime attribute, if B is a prime attribute.

Studernt_id	Course	Instructor
1	Java	JAA
1	C	MNA
2	Java	MSU
3	C++	TH
4	Java	JAA

Above this table, if you closely observe that, Instructor can find the subject uniquely, which is basically non prime attribute.

(Student_id, Course) -> Instructor …….

Instructor -> Course; But Instructor is non-prime attribute.

Above tabe convert BCNF:

Student_id	Instructor_id

Instructor_id	Instructor	Course