Module II

DATA BASE DESIGN

NORMALIZATION

- Normalization is the process of efficiently organizing data in a database.
- **E.F Codd** proposed the concept of normalization.
- Normalization removes redundant data from the tables to improve the storage efficiency ,data integrity and scalability.

Need for normalization

- Normalization is the process of converting a relation into a standard form.
- The problem in an unnormalized relation are as follows:-
  - Data redundancy
  - Update anomalies
  - Deletion anomalies
  - Insertion anomalies

Need for normalization

Data redundancy:-

- In an unnormalizaed table design some information may be stored repeatedly.
- In the below example ,**student table** the branch information ,hod, office telephone number is repeated.
- This information is known as redundant data.

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

What is an Anomaly?

- Definition
  - Problems that can occur in poorly planned, un-normalized databases where all the data is store in one table (a flat-file database).

Types of Anomalies

- Insert
- Delete
- Update

Insert Anomaly

- An **Insert Anomaly** occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

Insert Anomaly

Course _no	Tutor	Room	Room_size	En_limit
353	Smith	A532	45	40
351	Smith	C320	100	60
355	Clark	H940	400	300
456	Turner	H940	400	45

e.g. we have built a new room (e.g. B123) but it has not yet been timetabled for any courses or members of staff.

Delete Anomaly

- A **Delete Anomaly** exists when certain attributes are lost because of the deletion of other attributes.

Delete Anomaly

Course_no	Tutor	Room	Room_size	En_limit
353	Smith	A532	45	40
351	Smith	C320	100	60
355	Clark	H940	400	300
456	Turner	H940	400	45

e.g. if we remove the entity, course_no:351 from the above table, the details of room C320 get deleted. Which implies the corresponding course will also get deleted.

Update Anomaly

- An **Update Anomaly** exists when one or more instances of duplicated data is updated, but not all.

Update Anomaly

Course_no	Tutor	Room	Room_size	En_limit
353	Smith	A532	45	40
351	Smith	C320	100	60
355	Clark	H940	400	300
456	Turner	H940	400	45

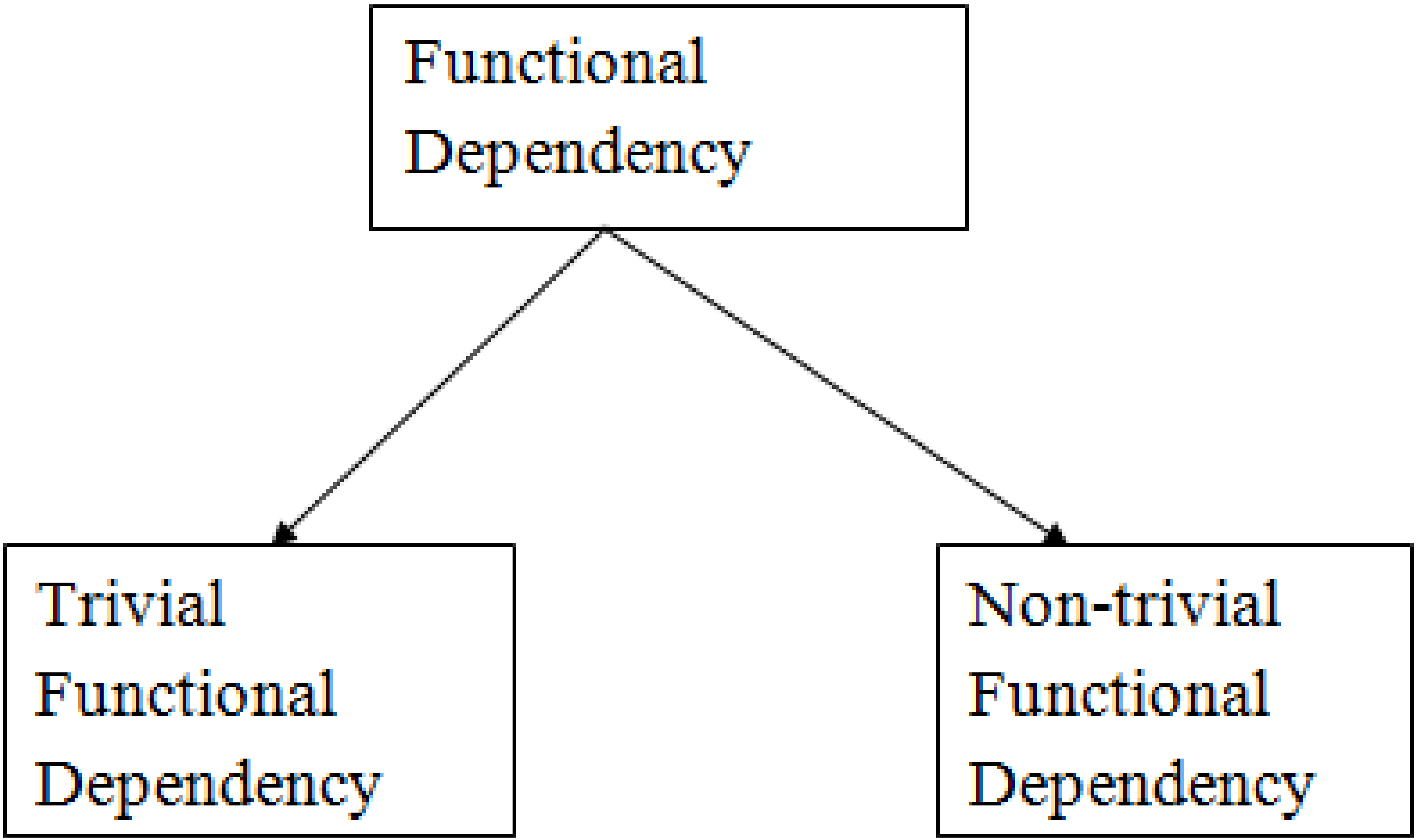
e.g. Room H940 has been improved, it is now of RSize = 500. For updating a single entity, we have to update all other columns where room=H940.

Functional Dependency

- A ***functional dependency*** **(FD)** is a relationship between two attributes, typically between the PK and other non-key attributes within a table.
- For any relation R, attribute Y is functionally dependent on attribute X (usually the PK), if for every valid instance of X, that value of X uniquely determines the value of Y.
- This relationship is indicated by the representation below :
- **X** ———————> **Y**

- The left side of the above FD diagram is called the ***determinant***, and the right side is the ***dependent.***
- **SIN ———-> Name, Address, Birthdate**
- SIN determines Name, Address and Birthdate.
- **SIN, Course ———> DateCompleted**
  - SIN and Course determine the date completed (DateCompleted). This must also work for a composite PK.

Types of Functional dependency



1. Trivial functional dependency

- $A \rightarrow B$ has trivial functional dependency if B is a subset of A.
- Consider a table with two columns Employee_Id and Employee_Name.
- {Employee_id, Employee_Name} $\rightarrow$ Employee_Id is a trivial functional dependency as
- Employee_Id is a subset of {Employee_Id, Employee_Name}.

2. Non-trivial functional dependency

- $A \rightarrow B$ has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then $A \rightarrow B$ is called as complete non-trivial.
- ID $\rightarrow$ Name

Inference Rules

- *Armstrong's axioms* are a set of inference rules used to infer all the functional dependencies on a relational database.
- They were developed by William W. Armstrong.
- **Axiom of reflexivity**
- This axiom says, if Y is a subset of X, then X determines Y

If $Y \subseteq X$, then $X \rightarrow Y$

- **Axiom of augmentation**
- The axiom of augmentation, also known as a partial dependency, says if X determines Y, then XZ determines YZ for any Z

$$\text{If } X \rightarrow Y \text{, then } XZ \rightarrow YZ \text{ for any } Z$$

- **prime and non-prime attributes**
- attributes of candidate key, are called prime attributes. And rest of the attributes of the relation are non prime.

- **Axiom of transitivity**
- The axiom of transitivity says if X determines Y, and Y determines Z, then X must also determine Z

$$\text{If } X \rightarrow Y \text{ and } Y \rightarrow Z \text{, then } X \rightarrow Z$$

- **Secondary Rules –**
- These rules can be derived from the axioms.

1. **Union** – If $A \rightarrow B$ holds and $A \rightarrow C$ holds, then $A \rightarrow BC$ holds. If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
2. **Composition** – If $A \rightarrow B$ and $X \rightarrow Y$ holds, then $AX \rightarrow BY$ holds.
3. **Decomposition** – If $A \rightarrow BC$ holds then $A \rightarrow B$ and $A \rightarrow C$ hold. If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$
4. **Pseudo Transitivity** – If $A \rightarrow B$ holds and $BC \rightarrow D$ holds, then $AC \rightarrow D$ holds. If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$.

Functional Dependency Set

- Functional Dependency set or FD set of a relation is the set of all FDs present in the relation.
- { STUD_NO->STUD_NAME, STUD_NO->STUD_PHONE, STUD_NO->STUD_STATE, STUD_NO->STUD_COUNTRY, STUD_NO -> STUD_AGE, STUD_STATE->STUD_COUNTRY }

Attribute Closure:

- Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.
- To find attribute closure of an attribute set:
- Add elements of attribute set to the result set.
- Recursively add elements to the result set which can be functionally determined from the elements of the result set

- If attribute closure of an attribute set contains all attributes of relation, the attribute set will be super key of the relation.
- Question 1:
- Given relational schema R(P Q R S T) having following attributes P Q R S and T, also there is a set of functional dependency denoted by FD = { P->QR, RS->T, Q->S, T-> P }.
- Determine Closure of $(T)^+$

- FD = { P->QR, RS->T, Q->S, T-> P }.
- T+={ T,P,Q,R,S,T}

- **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?**
- A. {E, F}
  B. {E, F, H}
  C. {E}

- **{{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L}, K -> {M}, L -> {N}**
- {E, F}+={ E,F,G,I,J}

{E, F, H}+={E,F,H,G,I,J,K,L,M,N }

- {E}+={ E}

Canonical Cover of Functional Dependencies/Minimal set of Functional dependency

- A canonical cover of a set of functional dependencies F is a simplified set of functional dependencies that has the same closure as the original set F.
- **Extraneous attributes:** An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies.

- A canonical cover F_c of a set of functional dependencies F such that ALL the following properties are satisfied:
- F logically implies all dependencies in F_c .
- F_c logically implies all dependencies in F.
- No functional dependency in contains an extraneous attribute.
- Each left side of a functional dependency in Fc is unique.

- **Finding Canonical Cover**

repeat

1. Use the union rule to replace any dependencies in $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1\beta_2$.
2. Find a functional dependency $\alpha \rightarrow \beta$ with an extraneous attribute either in α or in β.
3. If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$.

until F does not change

Example1:

Consider the following set *F* of functional dependencies:

F= {

A → BC

B → C

A → B

AB → C

}

Steps to find canonical cover:

1. There are two functional dependencies with the same set of attributes on the left:

   A → BC

   A → B

   These two can be combined to get

   A → BC.

   Now, the revised set F becomes:

   F= {

   A → BC

   B → C

   AB → C

   }

here is an extraneous attribute in AB $\longrightarrow$ C because even after removing AB $\longrightarrow$ C
om the set F, we get the same closures. This is because B $\longrightarrow$ C is already a part o

ow, the revised set F becomes:

= {

→ BC

→ C

is an extraneous attribute in A $\longrightarrow$ BC, also A $\longrightarrow$ B is logically implied by A $\longrightarrow$ B
nd B $\longrightarrow$ C (by transitivity).

= {

→ B

→ C

4. After this step, F does not change anymore. So,

Hence the required canonical cover is,

F_c= {

A $\longrightarrow$ B

B $\longrightarrow$ C

}

- **Let F = {A $\rightarrow$ B, A $\rightarrow$ C, BC $\rightarrow$ D}. Can A determine D uniquely?**

Consider the given functional dependencies-

$$AB \rightarrow CD$$

$$AF \rightarrow D$$

$$DE \rightarrow F$$

$$C \rightarrow G$$

$$F \rightarrow E$$

$$G \rightarrow A$$

Which of the following options is false?

(A) $\{ CF \}^+ = \{ A , C , D , E , F , G \}$

(B) $\{ BG \}^+ = \{ A , B , C , D , G \}$

(C) $\{ AF \}^+ = \{ A , C , D , E , F , G \}$

- **Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A–>B, BC–> D, E–>C, D–>A}. What are the candidate keys of R? [GATE 2005]**
  (a) AE, BE
  (b) AE, BE, DE
  (c) AEH, BEH, BCH
  (d) AEH, BEH, DEH

Functional Dependencies and Normalization for Relational Databases

PART 2

Normalization

- Normalization is the process of efficiently organizing data in a database with two goals in mind
- First goal: eliminate redundant data
  - for example, storing the same data in more than one table
- Second Goal: ensure data dependencies make sense
  - for example, only storing related data in a table

Benefits of Normalization

- Less storage space
- Quicker updates
- Less data inconsistency
- Clearer data relationships
- Easier to add data
- Flexible Structure

The Solution: Normal Forms

- Bad database designs results in:
  - redundancy: inefficient storage.
  - anomalies: data inconsistency, difficulties in maintenance
- 1NF, 2NF, 3NF, BCNF are some of the early forms in the list that address this problem

Brief History/Overview

- Database Normalization was first proposed by Edgar F. Codd.
- Codd defined the first three Normal Forms.
- One of the key requirements to remember is that Normal Forms are progressive. That is, in order to have 3rd NF we must have 2nd NF and in order to have 2nd NF we must have 1st NF.

1st Normal Form The Requirements

- The requirements to satisfy the 1st NF:
  - The values in each column of a table are atomic (No multi-value attributes allowed).
  - There are no repeating groups: two columns do not store similar information in the same table.

1) First normal form -1NF

- **1NF : if all attribute values are atomic: no repeating group, no multivalued attributes.**
- The following table is not in 1NF

DPT_NO	MG_NO	EMP_NO	EMP_NM
D101	12345	20000 20001 20002	Carl Sagan Mag James Larry Bird
D102	13456	30000 30001	Jim Carter Paul Simon

Table in 1NF

DPT_NO	MG_NO	EMP_NO	EMP_NM
D101	12345	20000	Carl Sagan
D101	12345	20001	Mag James
D101	12345	20002	Larry Bird
D102	13456	30000	Jim Carter
D102	13456	30001	Paul Simon

- all attribute values are atomic because there are no repeating group and no composite attributes.

Second Normal Form

- Uses the concepts of **FDs, primary key**
- Definitions
  - **Prime attribute:** An attribute that is member of the primary key K.
  - **Non Prime attribute**: An attribute that is not a member of the primary key K.
  - **Full functional dependency:** a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

Second Normal Form

- Examples:
  - {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold
  - {SSN, PNUMBER} -> ENAME is not a full FD (it is called a partial dependency) since SSN -> ENAME also holds

Partial FDs and 2NF

- Partial FDs:
  - A FD, $A \rightarrow B$ is a partial FD, if some attribute of A can be removed and the FD still holds
  - Formally, there is some proper subset of A,

  $C \subset A$, such that $C \rightarrow B$
- Let us call attributes which are part of some candidate key, key attributes, and the rest non-key attributes.

Second normal form:

- A relation is in second normal form (2NF) if it is in 1NF and no non-key attribute is partially dependent on a candidate key.
- In other words, no $C \rightarrow B$ where C is a strict subset of a candidate key and B is a non-key attribute.

Second Normal Form (2)

- A relation schema R is in **second normal form (2NF)** if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key.
- A relation in 2NF will not have any partial dependencies.
- R can be decomposed into 2NF relations via the process of 2NF normalization

Second Normal Form

Consider this **Order** table (in 1NF)**:**

Order no	item code	Order date	Qty	Price_ per_unit

orderno, itemcode → Order_date

orderno, itemcode → Qty

orderno, itemcode → Price_per_unit

Item code → Price_per_unit

Order no → Order date

Order is **not 2NF** since there is a **partial dependency** of **Item code on Price_per_unit**.

Second Normal Form

Consider this **Order** table (in 1NF)**:**

Order no	item code	Order date	Qty	Price_ per_unit

We can *improve* the database by decomposing the relation into three relations:

Order no	Order date

item code	Price_ per_unit

Order no	item code	Qty

Third Normal Form

Third Normal Form

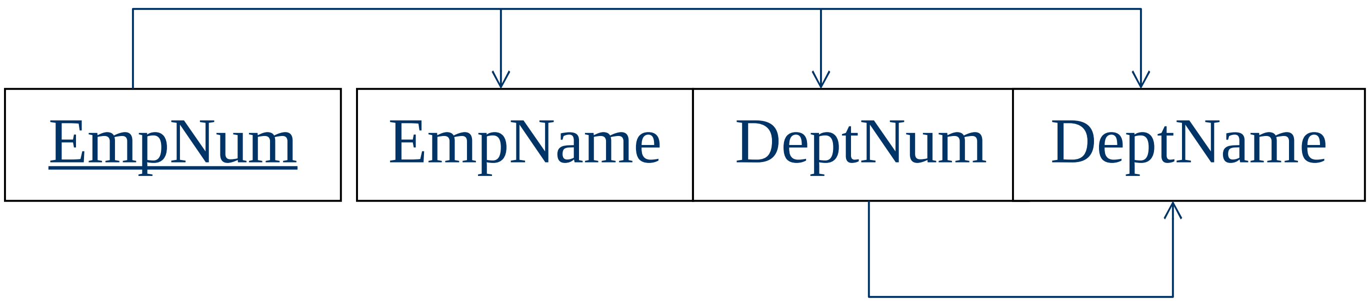
- A relation in 3NF will not have any transitive dependencies of non-key attribute on a candidate key through another non-key attribute.

Third Normal Form

- Let R be a relation schema, F be the set of FDs given to hold over R, X be a subset of the attributes of R, and A be an attribute of R. R is in third normal form if, for every FD….
- A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency $X \rightarrow Y$.
- X is a super key.
- Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Third Normal Form

Consider this **Employee** relation



EmpName, DeptNum, and DeptName are non-key attributes.

DeptNum determines DeptName, a non-key attribute.

Is the relation in 3NF? … no

Is the relation in 2NF? … yes

Third Normal Form

<u>EmpNum</u>	EmpName	DeptNum	DeptName

We correct the situation by decomposing the original relation into two 3NF relations. Note the decomposition is *lossless*.

EmpNum	EmpName	DeptNum

DeptNum	DeptName

Verify these two relations are in 3NF.

relation is in third normal form, if there is **no transitive dependency** for non-prime tributes as well as it is in second normal form.

relation is in 3NF if **at least one of the following condition holds** in every non-ivial function dependency X -> Y

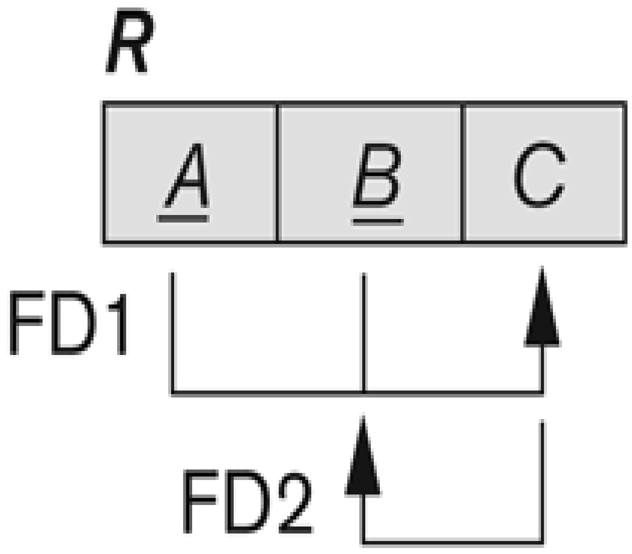
. X is a super key.

. Y is a prime attribute (each element of Y is part of some candidate key).

Boyce Codd Normal Form

- A relation R is in BCNF if R is in Third Normal Form
- Let *R be a relation schema, F be the set of FD's given to hold over R, .X be a* subset of the attributes of R, and *A be an attribute of R. R is in Boyce*-*Codd* normal form if, for every FD X → *A in F, one of the following statements is* true:
- • *A E X; that is, it is a trivial FD, or*
- • *X is a super key.*

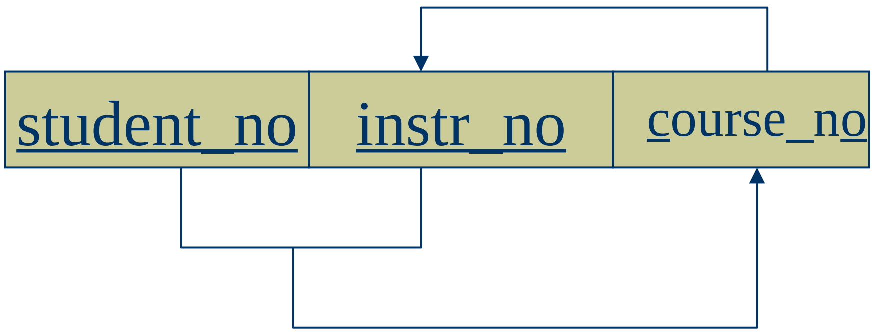
3NF, Not in BCNF…….



Boyce-Code Normal Form (BCNF)

- A relation is in BCNF if every determinant is a candidate key.

In 3NF, but not in BCNF:



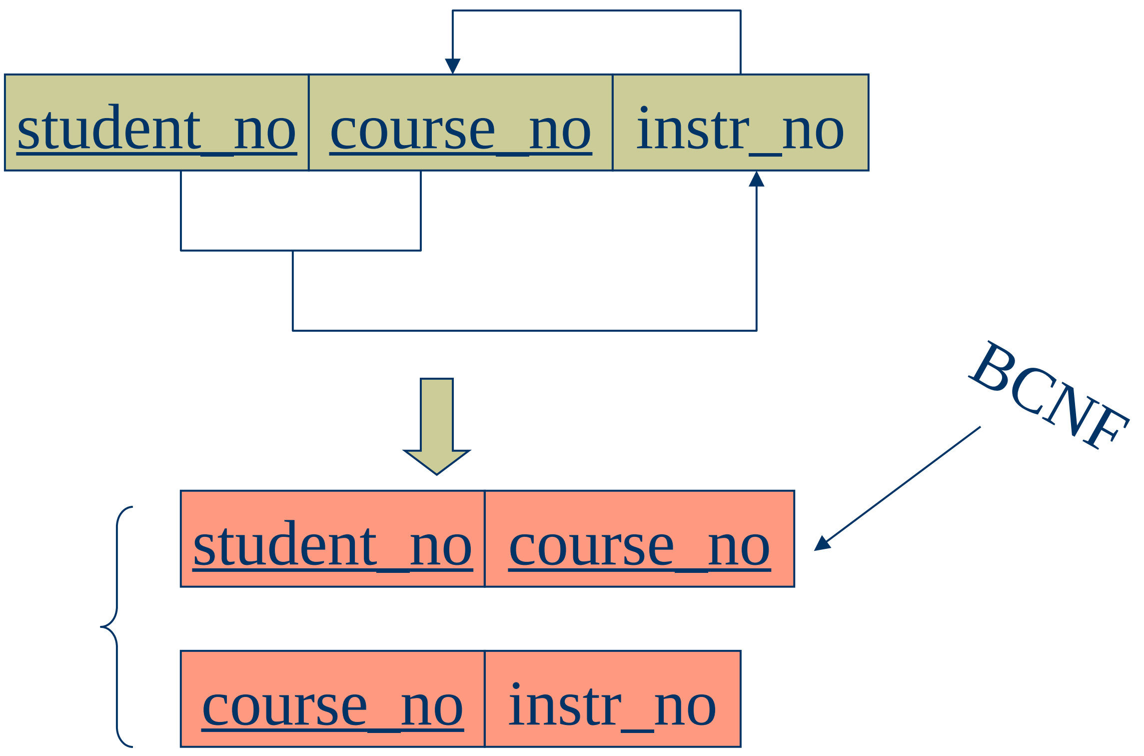
Instructor teaches one course only.

Student takes a course and has one instructor. Student can take more than one course.

{student_no, instr_no} → course_no
course_no ->instr_no

since we have course_no ->instr_no
, but Course_no is not a Candidate key.

BCNF: Eg



student_no
course_no
instr_no
student_no
course_no
course_no
instr_no
BCNF

Key points

- BCNF is free from redundancy.
- If a relation is in BCNF, then 3NF is also also satisfied.
- If all attributes of relation are prime attribute, then the relation is always in 3NF.
- A relation in a Relational Database is always and at least in 1NF form.

- Every Binary Relation (a Relation with only 2 attributes) is always in BCNF.
- If a Relation has only singleton candidate keys(i.e. every candidate key consists of only 1 attribute), then the Relation is always in 2NF(because no Partial functional dependency possible).

- Sometimes going for BCNF form may not preserve functional dependency. In that case go for BCNF only if the lost FD(s) is not required, else normalize till 3NF only.
- There are many more Normal forms that exist after BCNF, like 4NF and more. But in real world database systems it’s generally not required to go beyond BCNF.

Multivalued dependency

- Let R be a relation schema and let X and Y be subsets of the attributes of R.The multivalued dependency *X ->->Y is said to hold over R if,in every legal instance r of R, each X value is associated with a set of Y values and this set is independent of the values in the other attributes.*

  OR

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

Multivalued Dependencies

Course	Teacher	Book
Physics101	Green	Electronics
Physics101	Green	Optics
Physics101	Brown	Mechanics
Maths301	Brown	Geometry
Maths301	Green	Vectors
Maths301	Green	Algebra

- Course ->-> Book
  Course ->-> Teacher

Multivalued Dependencies

Course	Book
Physics101	Electronics
Physics101	Optics
Maths301	Geometry
Maths301	Vectors
Maths301	Alegbra

Course	Teacher
Physics101	Green
Physics101	Brown
Maths301	Green

Fourth Normal Form (4NF)

- **4NF** is a direct generalisation of BCNF.
- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- Let R be a relation schema, *X and Y be non empty subsets of the attributes of R, and F' be a set* of dependencies that includes both FDs and MVDs.
- *R is said to be in fourth* normal form (4NF), if, for every MVD *X*->->*Y that holds over R, one of* the following statements is true:
    - *Y E X or XY=R or*
    - *X is a superkey.*

Multivalued dependency

- Let R be a relation schema and let X and Y be subsets of the attributes of R.The multivalued dependency *X ->->Y is said to hold over R if,in every legal instance r of R, each X value is associated with a set of Y values and this set is independent of the values in the other attributes.*

OR

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

Multivalued Dependencies

Course	Teacher	Book
Physics101	Green	Electronics
Physics101	Green	Optics
Physics101	Brown	Mechanics
Maths301	Brown	Geometry
Maths301	Green	Vectors
Maths301	Green	Algebra

- Course ->-> Book
  Course ->-> Teacher

Fourth Normal Form (4NF)

- **4NF** is a direct generalisation of BCNF.
- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- Let R be a relation schema, *X and Y be non empty subsets of the attributes of R, and F' be a set* of dependencies that includes both FDs and MVDs.
- *R is said to be in fourth* normal form (4NF), if, for every MVD *X->->Y that holds over R, one of* the following statements is true:
  - *Y E X or XY=R or*
  - *X is a superkey.*

Fifth normal form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Join Dependency

- Join decomposition is a further generalization of Multivalued dependencies.
- If the join of R1 and R2 over 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 relations R (A, B, C, D).
- Alternatively, R1 and R2 are a lossless decomposition of R.