S. S Jain Subodh P.G. (Autonomous) College

SUBJECT -DATABASE MANAGEMENT SYSTEM

TITLE –NORMALIZATION BY-VANDANA NIGAM

Presented By
Vandana Nigam

Normalization

Normalization is a process that "improves" a database design by generating relations that are of higher normal forms.

The *objective* of normalization:
"*to create relations where every dependency is on the key, the whole key, and nothing but the key*".

Normalization

The main goal of Database Normalization is to restructure the logical data model of a database to:

1. Eliminate redundancy
2. Organize data efficiently
3. Reduce the potential for data anomalies.

Data Anomalies

Data anomalies are inconsistencies in the data stored in a database as a result of an operation such as update, insertion, and/or deletion.
Such inconsistencies may arise when have a particular record stored in multiple locations and not all of the copies are updated.

Three types of anomalies are generally present-

1. Insert Anomaly
2. Update/ Modification Anomaly
3. Delete Anomaly

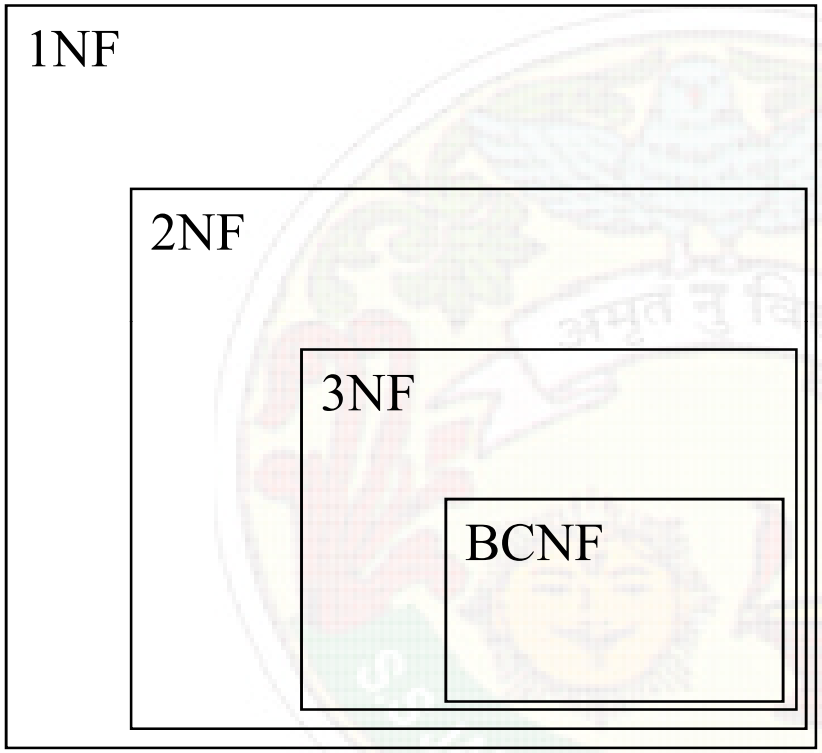
We can prevent such anomalies by implementing 7 different level of normalization called Normal Forms (NF)

Normalization

There is a sequence to normal forms:
1NF is considered the weakest,
2NF is stronger than 1NF,
3NF is stronger than 2NF, and
BCNF is considered the strongest

Also,
any relation that is in BCNF, is in 3NF;
any relation in 3NF is in 2NF; and
any relation in 2NF is in 1NF.

Normalization



a relation in BCNF, is also in 3NF

a relation in 3NF is also in 2NF

a relation in 2NF is also in 1NF

Normalization

We consider a relation in BCNF to be fully normalized.

The benefit of higher normal forms is that update semantics for the affected data are simplified.

This means that applications required to maintain the database are simpler.

A design that has a lower normal form than another design has more redundancy. Uncontrolled redundancy can lead to data integrity problems.

First we introduce the concept of ***functional dependency***

Functional Dependencies

Functional Dependencies

We say an attribute, B, has a *functional dependency* on another attribute, A, if for any two records, which have the same value for A, then the values for B in these two records must be the same. We illustrate this as:

A → B

Example: Suppose we keep track of employee email addresses, and we only track one email address for each employee. Suppose each employee is identified by their unique employee number. We say there is a functional dependency of email address on employee number:

employee number → email address

Functional Dependency

EmpNum	EmpEmail	EmpFname	EmpLname
123	jdoe@abc.com	John	Doe
456	psmith@abc.com	Peter	Smith
555	alee1@abc.com	Alan	Lee
633	pdoe@abc.com	Peter	Doe
787	alee2@abc.com	Alan	Lee

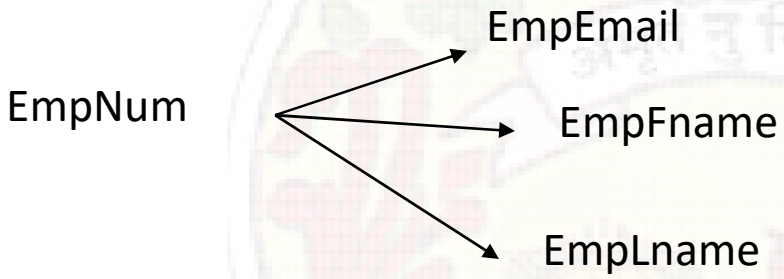
If EmpNum is the Primary Key then the FDs:

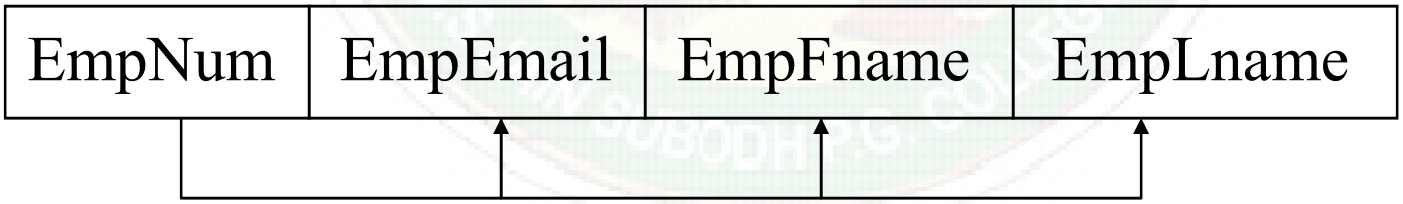
EmpNum → EmpEmail
EmpNum → EmpFname
EmpNum → EmpLname

must exist.

Functional Dependencies

EmpNum → EmpEmail
EmpNum → EmpFname
EmpNum → EmpLname





Determinant

Functional Dependency

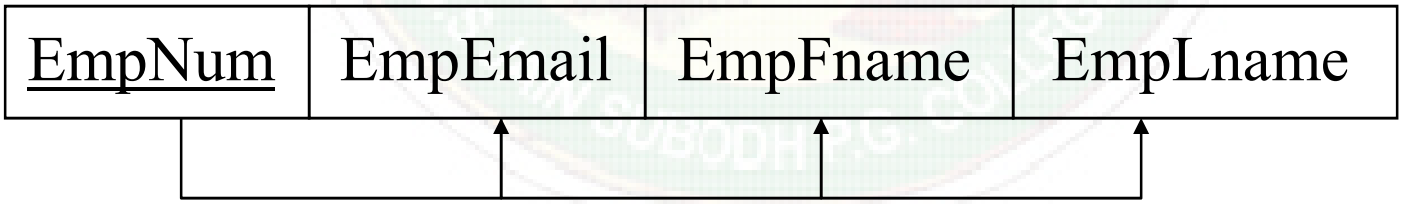
EmpNum → EmpEmail

Attribute on the Left Hand Side is known as the ***determinant***

- EmpNum is a determinant of EmpEmail

Full Functional Dependency

When all non key attributes are dependent on the key attribute, it is called full functional dependency.



Transitive dependency

Consider attributes A, B, and C, and where

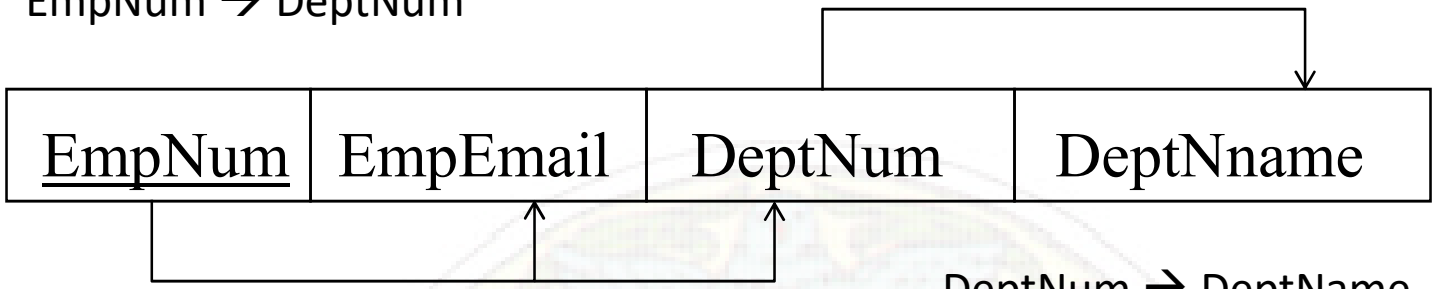
A → B and B → C.

Functional dependencies are transitive, which means that we also have the functional dependency A → C

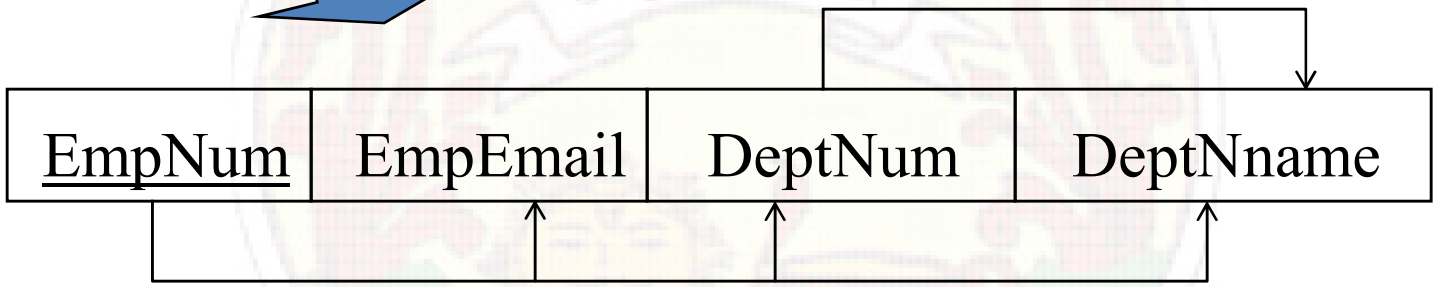
We say that C is transitively dependent on A through B.

Transitive Dependency

EmpNum → DeptNum



DeptNum → DeptName

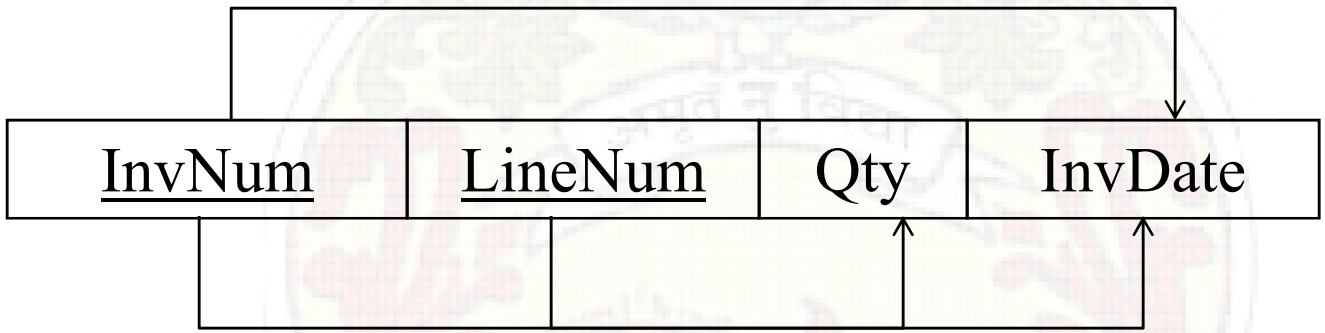


DeptName is *transitively dependent* on EmpNum via DeptNum

EmpNum → DeptName

Partial Dependency

A **partial dependency** exists when an attribute B is functionally dependent on an attribute A, and A is a component of a multipart candidate key.



Candidate keys: {InvNum, LineNum} InvDate is *partially dependent* on {InvNum, LineNum} as InvNum is a determinant of InvDate and InvNum is part of a candidate key

First Normal Form

First Normal Form

We say a relation is in **1NF** if all values stored in the relation are single-valued and atomic.

1NF places restrictions on the structure of relations. Values must be simple.

1st Normal Form Requirements

The requirements to satisfy the 1st NF:

Each table has a primary key: minimal set of attributes which can uniquely identify a record

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.

First Normal Form

The following in **not** in 1NF (First Example)

EmpNum	EmpPhone	EmpDegrees
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

EmpDegrees is a multi-valued field:

employee 679 has two degrees: *BSc* and *MSc*

employee 333 has three degrees: *BA, BSc, PhD*

First Normal Form

EmpNum	EmpPhone	EmpDegrees
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

To obtain 1NF relations we must, without loss of information, replace the above with two relations

First Normal Form

Employee

EmpNum	EmpPhone
123	233-9876
333	233-1231
679	233-1231

EmployeeDegree

EmpNum	EmpDegree
333	BA
333	BSc
333	PhD
679	BSc
679	MSc

An outer join between Employee and EmployeeDegree will produce the information we saw before

1st Normal Form
Second Example

Un-normalized Students table:

Student#	AdvID	AdvName	AdvRoom	Class1	Class2
123	123A	James	555	102-8	104-9
124	123B	Smith	467	209-0	102-8

Normalized Students table:

Student#	AdvID	AdvName	AdvRoom	Class#
123	123A	James	555	102-8
123	123A	James	555	104-9
124	123B	Smith	467	209-0
124	123B	Smith	467	102-8

Second Normal Form

Second Normal Form

A relation is in **2NF** if it is in 1NF, and every non-key attribute is fully dependent on each candidate key. (That is, we don't have any partial functional dependency.)

- 2NF (and 3NF) both involve the concepts of key and non-key attributes.
- A *key attribute* is any attribute that is part of a key; any attribute that is not a key attribute, is a *non-key attribute*.
- Relations that are not in BCNF have data redundancies
- A relation in 2NF will not have any partial dependencies

Requirement of Second Normal Form

The requirements to satisfy the 2nd NF:

All requirements for 1st NF must be met.

Redundant data across multiple rows of a table must be moved to a separate table.

The resulting tables must be related to each other by use of foreign key.

Second Normal Form

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

InvNum	LineNum	ProdNum	Qty	InvDate

InvNum, LineNum ⟶ ProdNum, Qty

There are two candidate keys.

Qty is the only non-key attribute, and it is dependent on InvNum

InvNum ⟶ InvDate

Since there is a determinant that is not a candidate key, InvLine is **not BCNF**

InvLine is **not 2NF** since there is a partial dependency of InvDate on InvNum

InvLine is only in **1NF**

Second Normal Form

InvLine

InvNum	LineNum	ProdNum	Qty	InvDate

The above relation has redundancies: the invoice date is repeated on each invoice line.

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

InvNum	LineNum	ProdNum	Qty

InvNum	InvDate

Question: What is the highest normal form for these relations? 2NF? 3NF? BCNF?

2nd Normal Form
Second Example

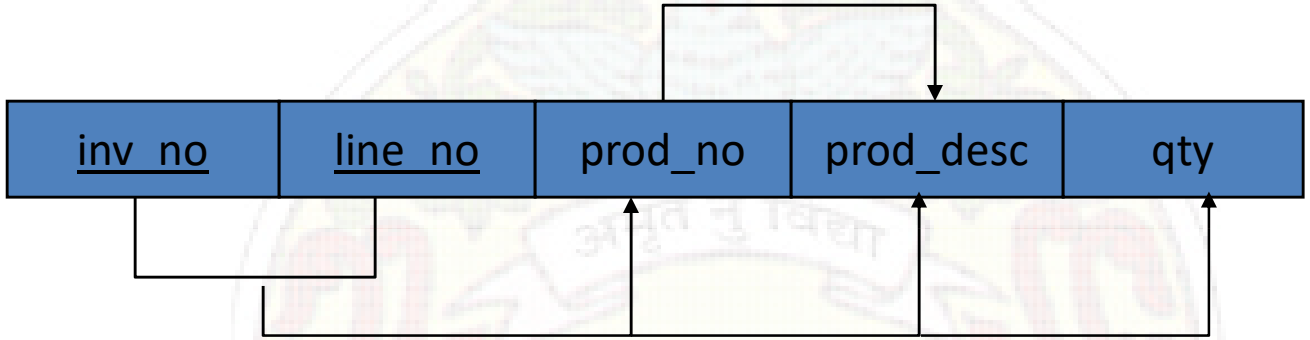
Students table

Student#	AdvID	AdvName	AdvRoom
123	123A	James	555
124	123B	Smith	467

Registration table

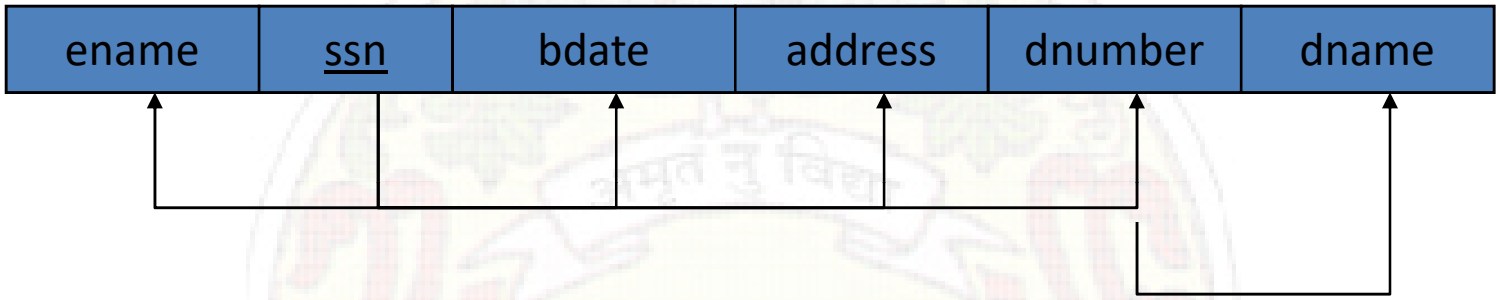
Student#	Class#
123	102-8
123	104-9
124	209-0
124	102-8

Is the following relation in 2NF?



2NF, but not in 3NF, nor in BCNF:

EmployeeDept



since dnumber is not a candidate key and we have:

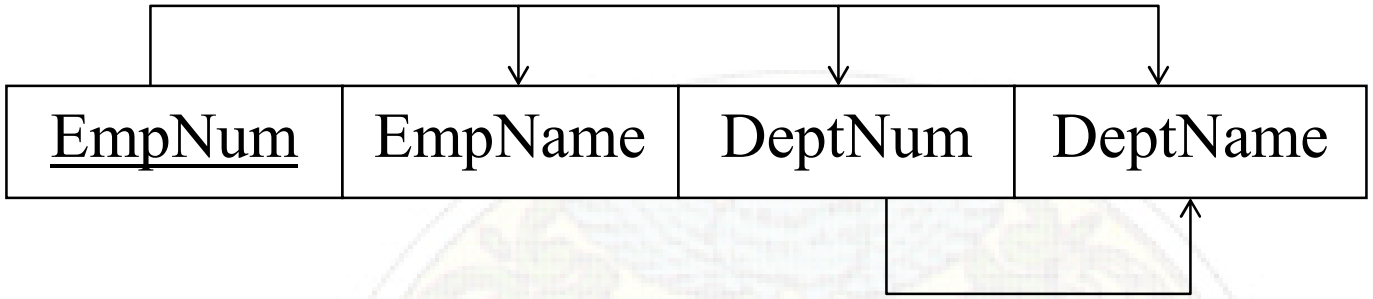
dnumber → dname.

Third Normal Form

- A relation is in **3NF** if the relation is in 1NF and all determinants of *non-key* attributes are candidate keys
  That is, for any functional dependency: X → Y, where Y is a non-key attribute (or a set of non-key attributes), X is a candidate key.
- This definition of 3NF differs from BCNF only in the specification of non-key attributes - 3NF is weaker than BCNF. (BCNF requires all determinants to be candidate keys.)
- A relation in 3NF will not have any transitive dependencies of non-key attribute on a candidate key through another non-key attribute.
- Eliminate fields that do not depend on the primary key; That is, any field that is dependent not only on the primary key but also on another field must be moved to another table.

Third Normal Form

Consider this **Employee** relation



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

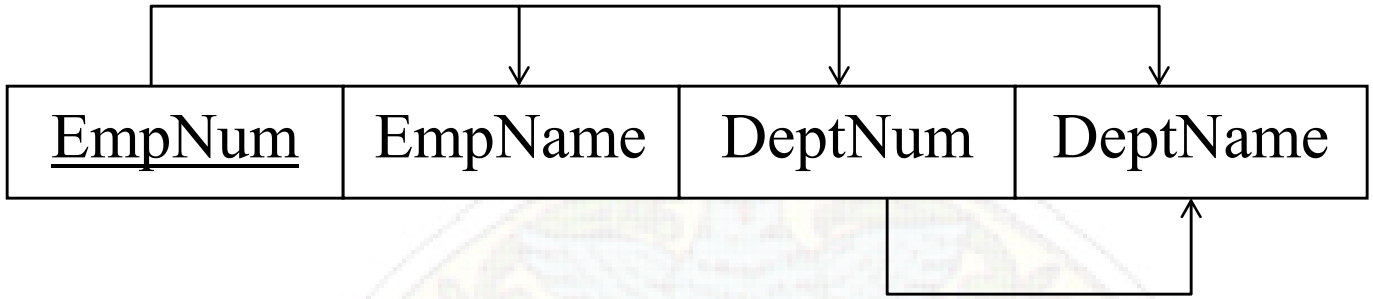
DeptNum determines DeptName, a non-key attribute, and DeptNum is not a candidate key.

Is the relation in 3NF? … no

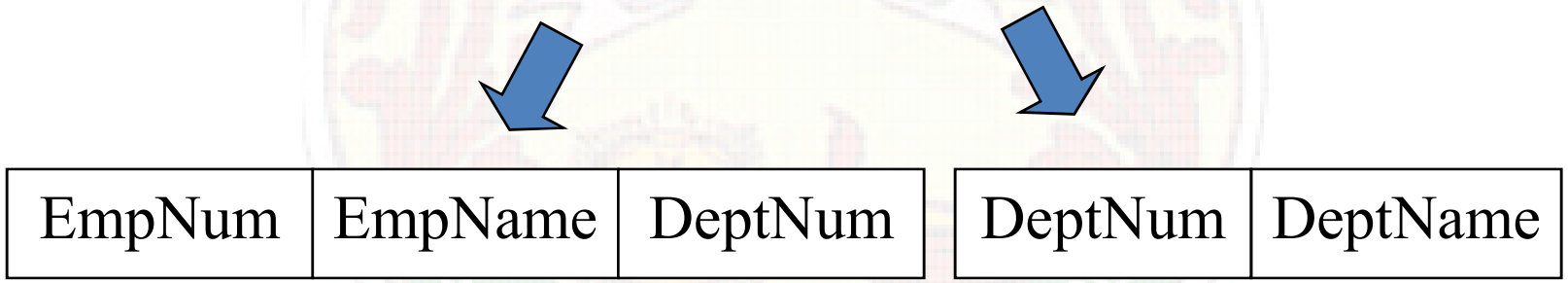
Is the relation in BCNF? … no

Is the relation in 2NF? … yes

Third Normal Form



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



Verify these two relations are in 3NF.

Boyce-Codd Normal Form

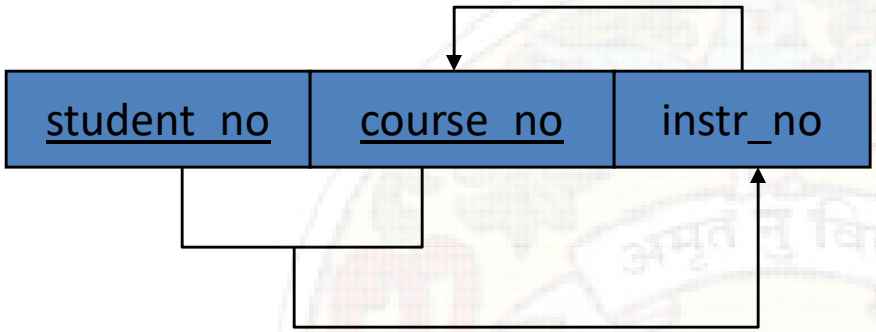
Boyce-Codd Normal Form

BCNF is the simpler form of 3CNF and eliminates all the problems of 3NF.
The difference between 3NF and BCNF is that for a functional dependency A → B, 3NF allows this dependency in a relation if B is a primary key attribute and A is not a candidate key.

Whereas BCNF insists that for this dependency to remain in a relation, A must be a candidate key.

So BCNF is stronger than 3NF. A 3NF which does not has multiple overlapping candidate keys is said to be in BCNF.

In 3NF, but not in BCNF:

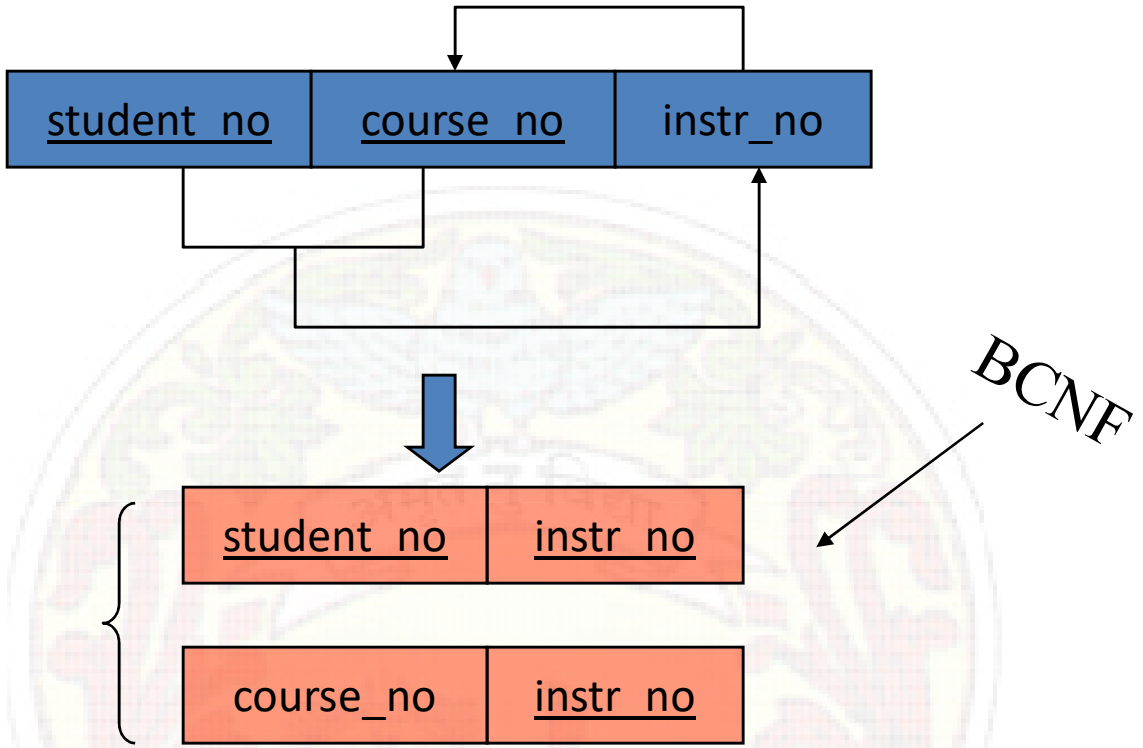


Instructor teaches one course only.

Student takes a course and has one instructor.

{student_no, course_no} → instr_no
instr_no → course_no

since we have instr_no → course-no, but instr_no is not a Candidate key.



$\{student\_no, instr\_no\} \rightarrow student\_no$

$\{student\_no, instr\_no\} \rightarrow instr\_no$

$instr\_no \rightarrow course\_no$

Third Normal Form Example

Students table:

Student#	AdvID	AdvName	AdvRoom
123	123A	James	555
124	123B	Smith	467

Student table:

Student#	AdvID
123	123A
124	123B

Advisor table:

AdvID	AdvName	AdvRoom
123A	James	555
123B	Smith	467

3rd Normal Form Example Cont.

Students table:

Student#	AdvID
123	123A
124	123B

Registration table:

Student#	Class#
123	102-8
123	104-9
124	209-0
124	102-8

Advisor table:

AdvID	AdvName	AdvRoom
123A	James	555
123B	Smith	467

Conclusion

- We have seen how Database Normalization can decrease redundancy, increase efficiency and reduce anomalies by implementing three of seven different levels of normalization called Normal Forms. The first three NF's are usually sufficient for most small to medium size applications.