Normalization

Normalization

We discuss four normal forms: first, second, third, and Boyce-Codd normal forms
1NF, 2NF, 3NF, and BCNF

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

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

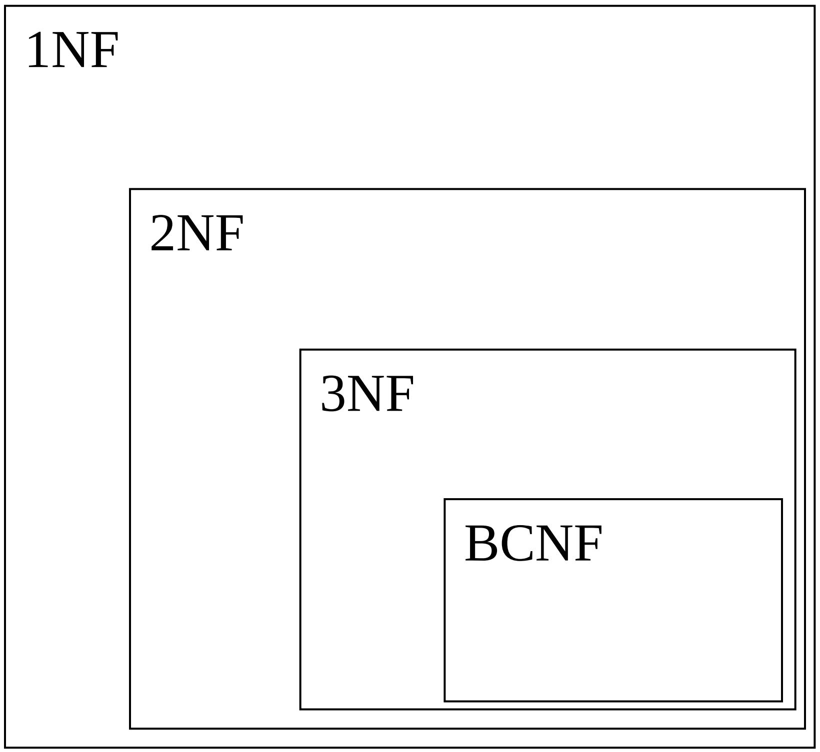
Also,

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***

NORMALIZATION is a database design technique that organizes tables in a manner that reduces redundancy and dependency of data. Normalization divides larger tables into smaller tables and links them using relationships. The purpose of Normalization is to eliminate redundant (useless) data and ensure data is stored logically.

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:

$A \rightarrow B$

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 $\rightarrow$ email address

Functional Dependencies

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 PK then the FDs:

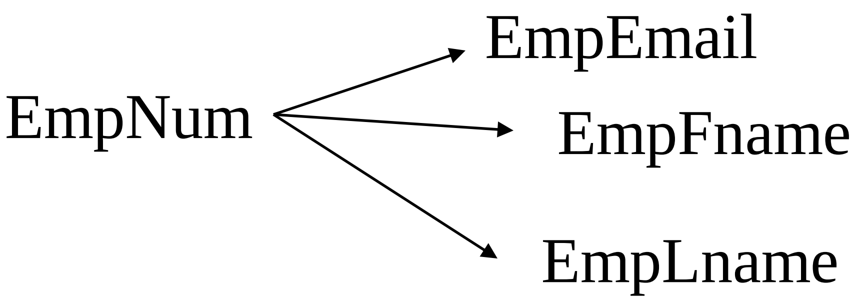
EmpNum → EmpEmail
EmpNum → EmpFname
EmpNum → EmpLname

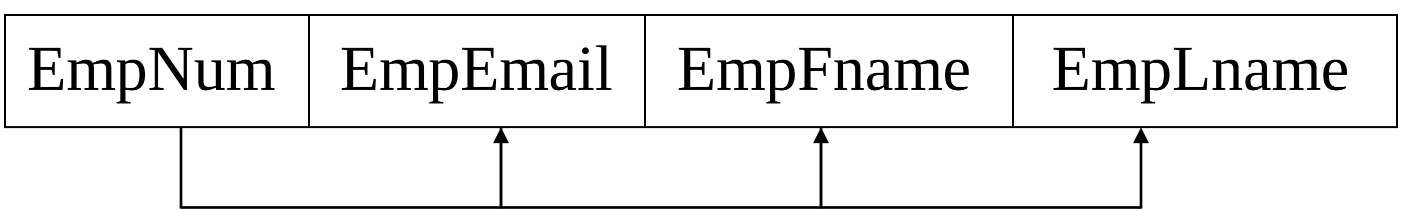
must exist.

Functional Dependencies

EmpNum → EmpEmail
EmpNum → EmpFname
EmpNum → EmpLname

3 different ways you might see FDs depicted





Determinant

Functional Dependency

EmpNum → EmpEmail

Attribute on the LHS is known as the ***determinant***

- EmpNum is a determinant of EmpEmail

Transitive dependency

Transitive dependency

Consider attributes A, B, and C, and where

A → B and B → C.

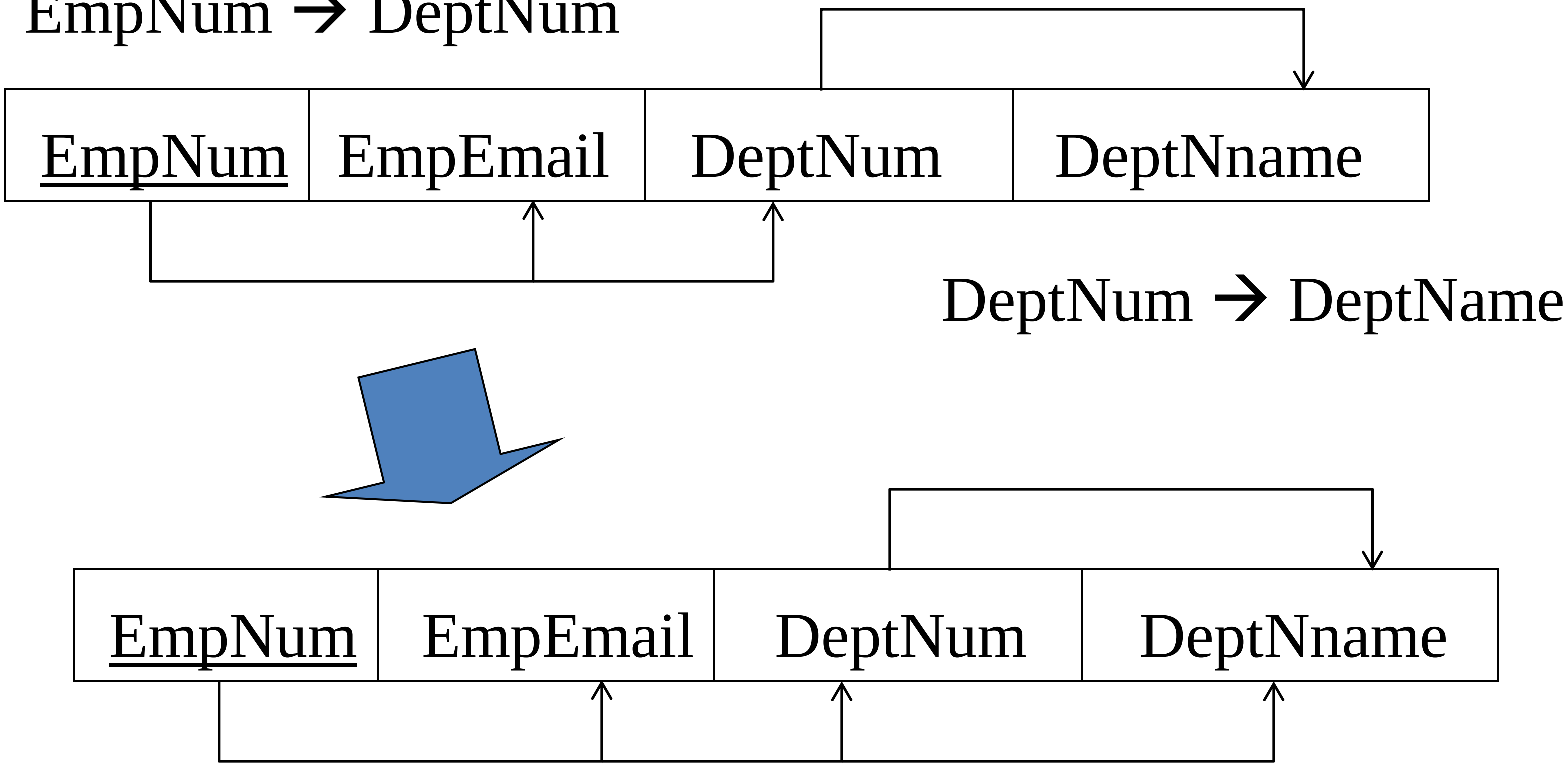
Functional dependencies are transitive, which means that we also have the functional dependency

A → C

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

Transitive dependency

EmpNum → DeptNum

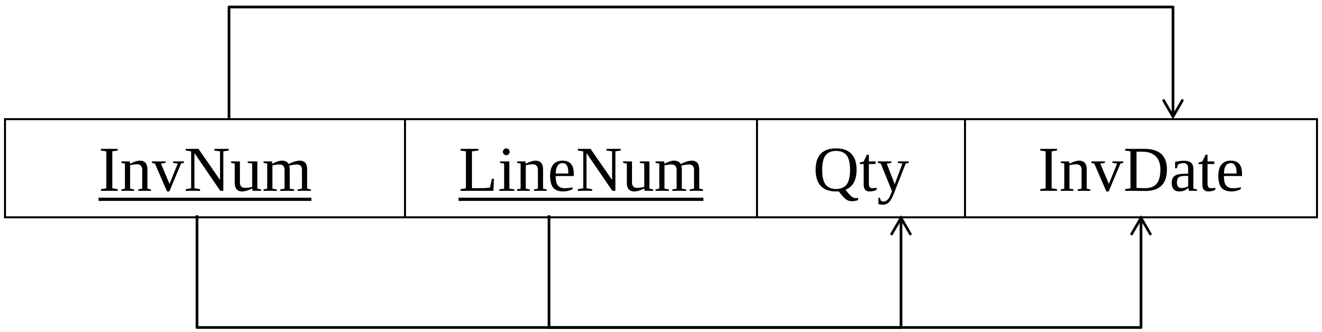


DeptName is *transitively dependent* on EmpNum via DeptNum

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.

First Normal Form

The following in **not** in 1NF

<u>EmpNum</u>	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

<u>EmpNum</u>	**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 - see next slide

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

Boyce-Codd Normal Form

Boyce-Codd Normal Form

BCNF is defined very simply:

> a relation is in BCNF if it is in 1NF and if every determinant is a candidate key.

If our database will be used for OLTP (on line transaction processing), then BCNF is our target. Usually, we meet this objective. However, we might denormalize (3NF, 2NF, or 1NF) for performance reasons.

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

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

<u>InvNum</u>	<u>LineNum</u>	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:

→

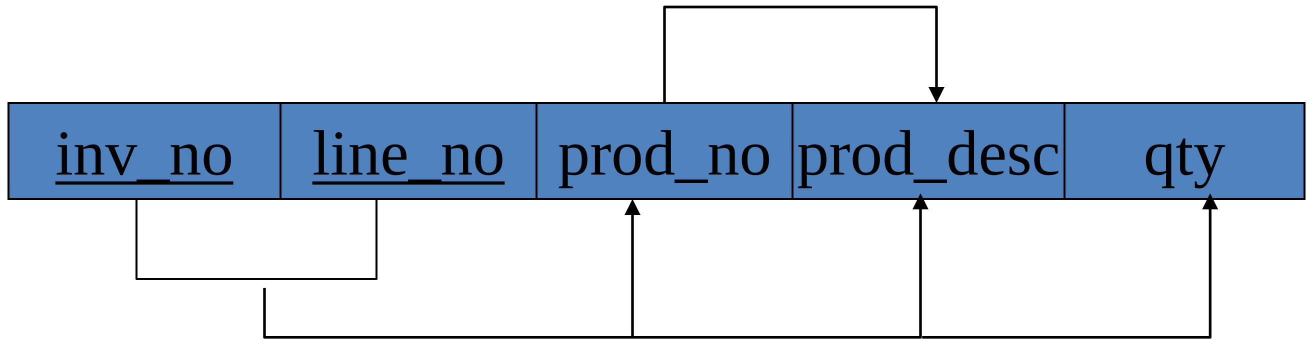
<u>InvNum</u>	<u>LineNum</u>	ProdNum	Qty

→

<u>InvNum</u>	InvDate

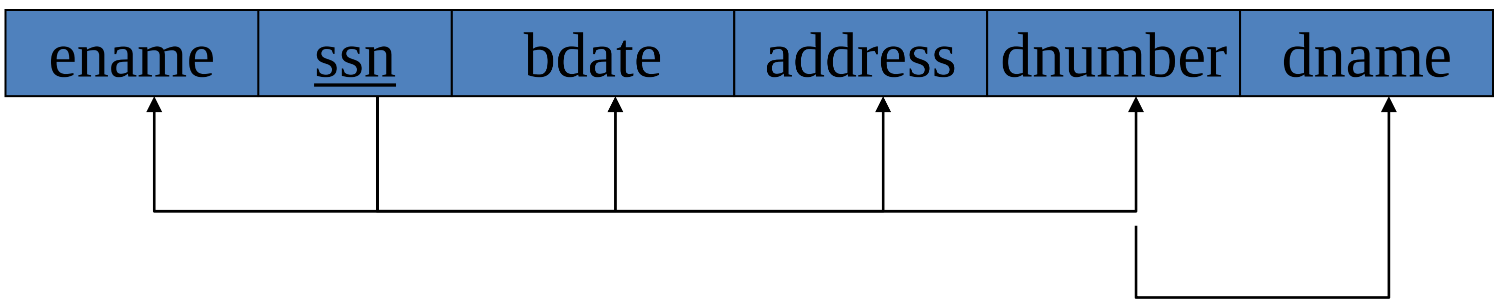
Question: What is the highest normal form for these relations? 2NF? 3NF? BCNF?

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 $\rightarrow$ dname.

Third Normal Form

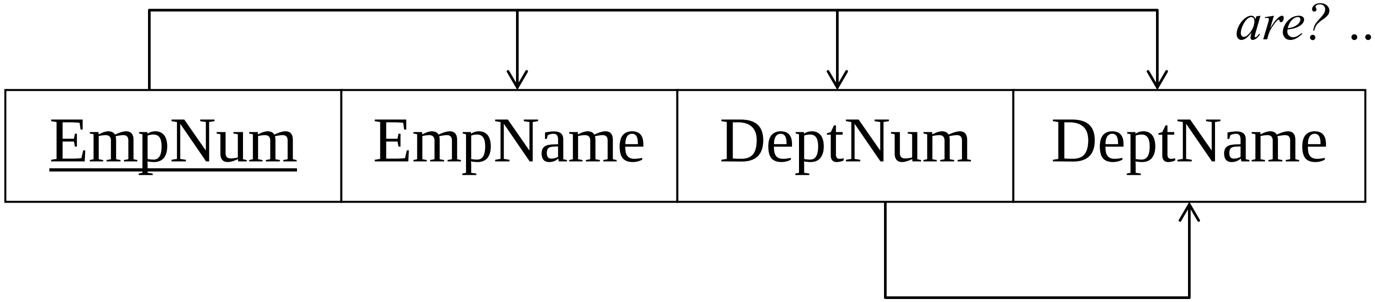
Third Normal Form

- 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: $X \rightarrow Y$, 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.

Third Normal Form

Consider this **Employee** relation

Candidate keys are? …



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

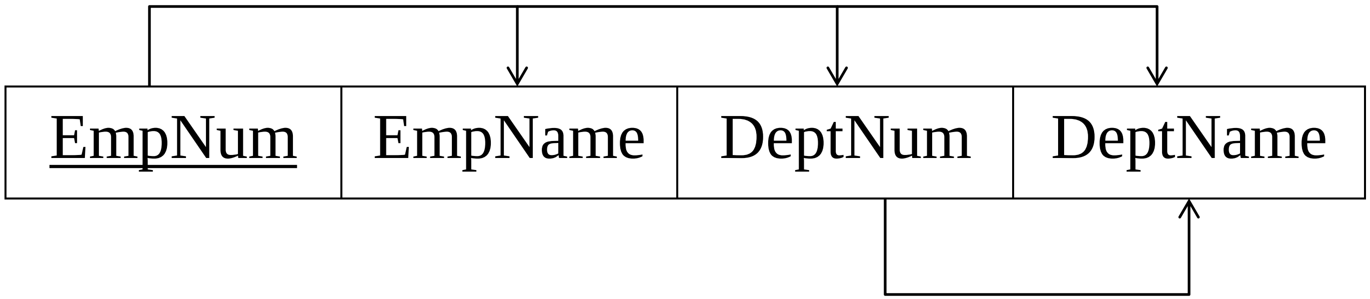
DeptNum determines DeptName, a non-key attribute, and DeptNum is not a candidate key.

Is the relation in 3NF? … no

Is the relation in BCNF? … no

Is the relation in 2NF? … yes

Third Normal Form



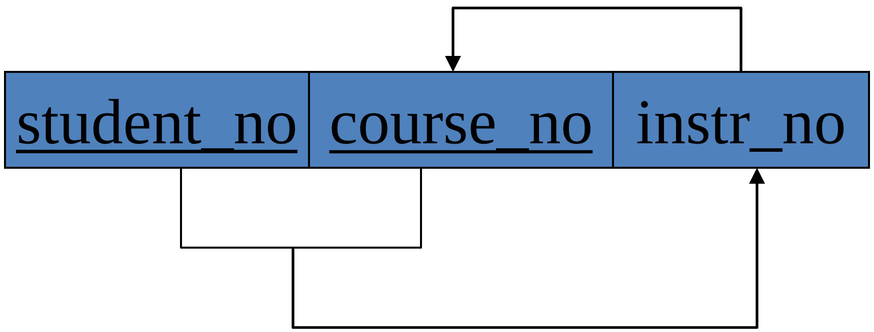
We correct the situation by decomposing the original relation into two 3NF relations. Note the decomposition is *lossless*.

EmpNum	EmpName	DeptNum

DeptNum	DeptName

Verify these two relations are in 3NF.

In 3NF, but not in BCNF:

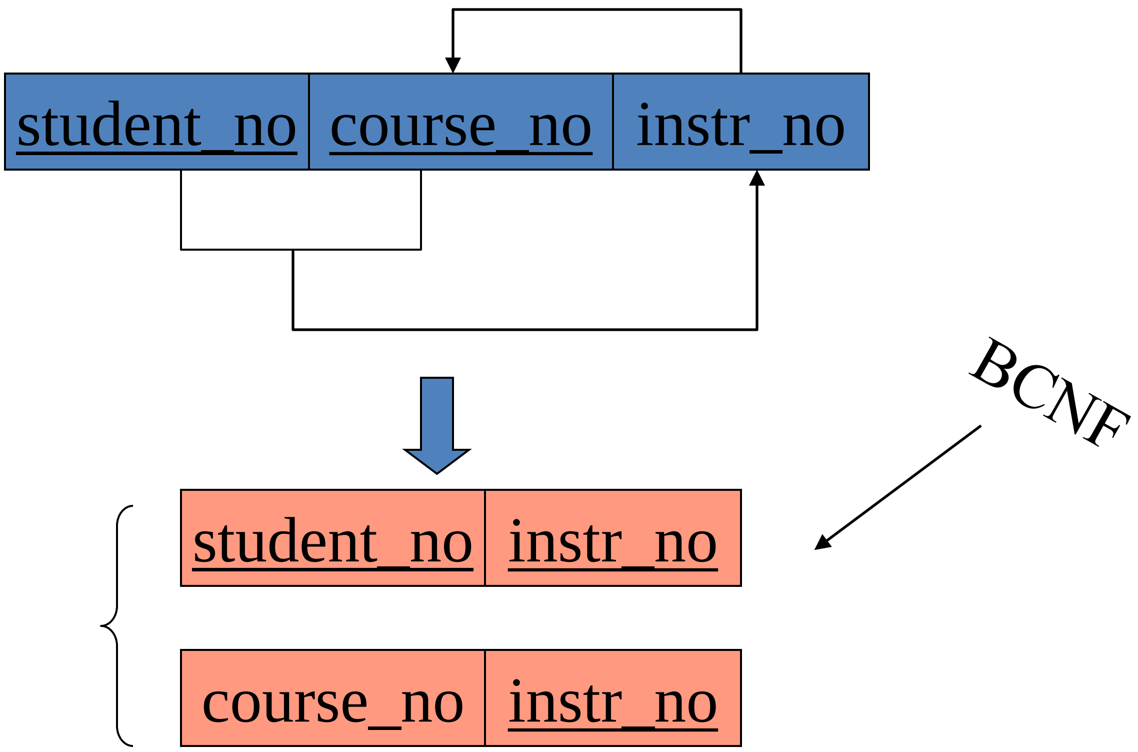


Instructor teaches one course only.

Student takes a course and has one instructor.

{student_no, course_no} $\rightarrow$ instr_no
instr_no $\rightarrow$ course_no

since we have instr_no $\rightarrow$ course-no, but instr_no is not a Candidate key.



{student_no, instr_no} → student_no
{student_no, instr_no} → instr_no
instr_no → course_no