DBMS: Data Normalization

Functional Dependencies

Akhilesh Arya

Determinant/ Dependent

Functional Dependency

EmpNum → EmpEmail

Attribute on the LHS is known as the ***determinant*** and the one on the RHS is Known as ***dependent***

- EmpNum is a determinant of EmpEmail
- And we can say that EmpEmail is dependent on EmpNum

Functional Dependency

FD: x → y
If t1.x = t2.x
Then t1.y = t2.y

Check:
R.No → Name
Name → R.No
R.No → Subject
Dept → Course
Course → Dept
Marks → Dept
Names → Marks
R.No, Name → Subject

R.No.	Name	Subject	Marks	Dept	Course
101	Ajay	Python	78	CSE	C1
102	Amit	Java	60	ME	C1
103	Abhay	CPP	65	CSE	C2
104	Ajay	C#	78	EEE	C3
105	Aniket	DBMS	60	EEE	C3
106	Chanchal	Java	92	CSE	C2
107	Chitvan	Python	87	ME	C4
108	Deepak	DBMS	65	Civil	C5
109	Pateek	TOC	55	CSE	C6
110	Aniket	ML	60	ME	C4

Cont..

Defination

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 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

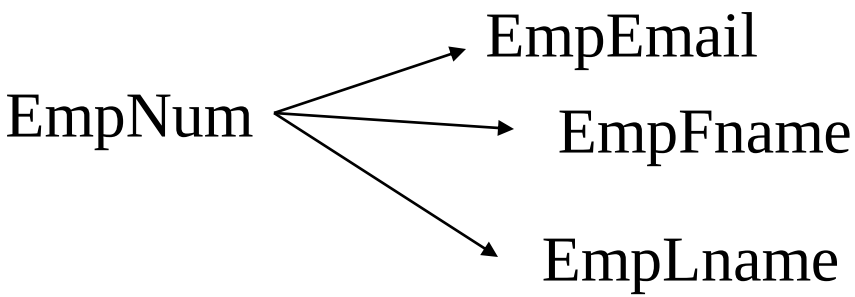
EmpNum → EmpLname

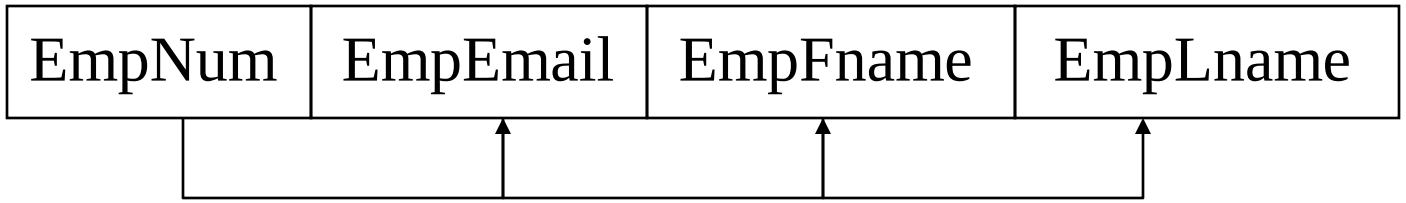
must exist.

Functional Dependencies

EmpNum → EmpEmail
EmpNum → EmpFname
EmpNum → EmpLname

3 different ways you might see FDs depicted





Attribute Closer or Closer Set

- Attribute closure, also known as functional dependency closure, is a property that determines the set of attributes that can be determined or inferred from a given set of attributes in a relational database.
- It is used to identify all the attributes that are functionally dependent on a particular set of attributes.

Example:

- For example, consider a database schema with attributes
  - (A, B, C, D, E) and
  - functional dependencies (A → B, B → C, C → D, D → E).
- The attribute closure of A, denoted as A+, would be {A, B, C, D, E}, because all attributes B, C, D and E can be determined from the value of A through the given functional dependencies.
- Find the closer of B and AD?

Cont..

- Consider a relation R(A, B, C, D, E)
- And functional dependency as fd{A → B, D → E}
  - Find the closer of:
    - ABCDE
    - ABDE
    - ACDE
    - ACD
  - Find candidate key and super key from these closer sets.

Cont..

- Find all the candidate key in the relation given
  - R(A, B, C, D, E, F)
  - F = {AB → C, C → DE, E → F, D → A, C → B}

Trivial functional dependency

- Trivial Functional Dependency: A functional dependency X -> Y is considered trivial if Y is a subset of X.

$$Y \subset X$$

- In other words, Y can be determined directly from X without any additional information.
- Trivial functional dependencies are not considered interesting or useful because they do not provide any meaningful constraint on the data.

Example:

- FirstName → FirstName is a trivial dependency
- ID, FirstName → FirstName is also a trivial dependency as FirstName is subset of ID, FirstName

ID	FirstName	LastName	Age	Department
101	Ajay	Kumar	32	CSE
102	Amit	Jain	24	ECE
103	Abhay	Singhvi	35	ME
104	Chanchal	Ahuja	25	ECE
105	Prateek	Surana	27	ME

Non- Trivial functional dependency

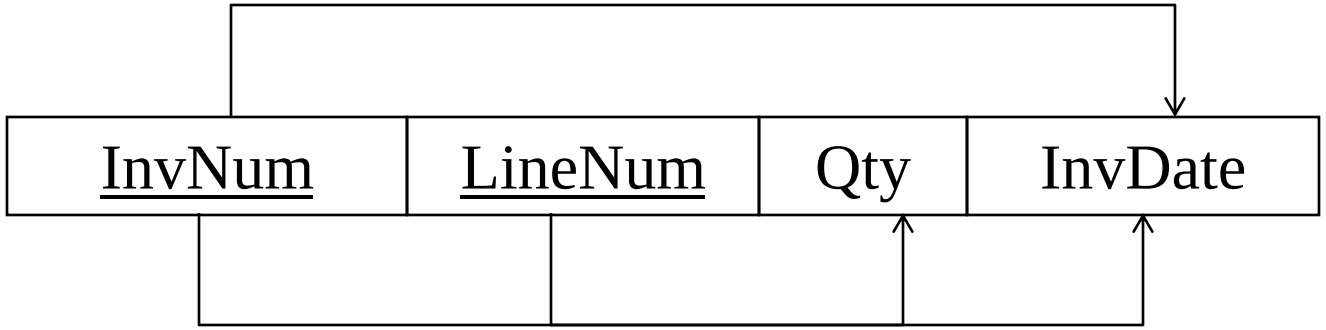
- Non-trivial Functional Dependency: A functional dependency X -> Y is considered non-trivial if Y is not a subset of X.

$$Y \cap X = \emptyset$$

- In other words, there is nothing common in X and Y.
- In such case we have to check with the table that even the functional dependency will exist or not.

Partial dependency

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



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

Transitive dependency

Transitive dependency

Consider attributes A, B, and C, and where

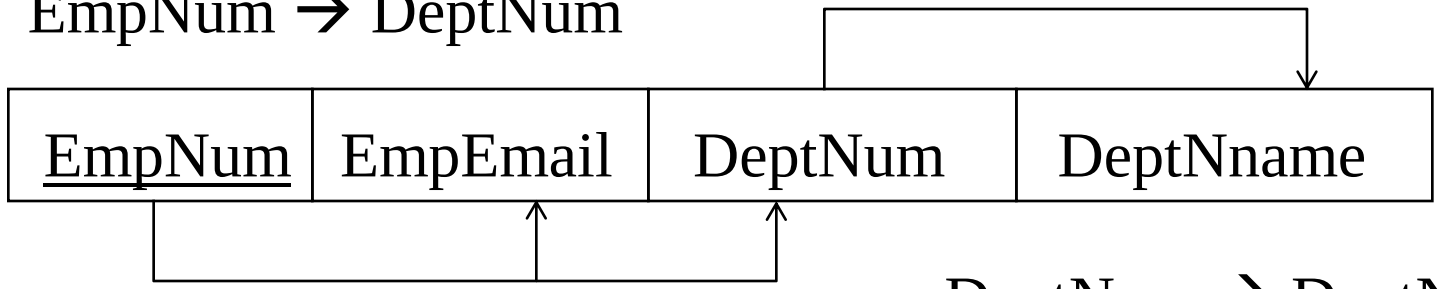
A → B and B → C.

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

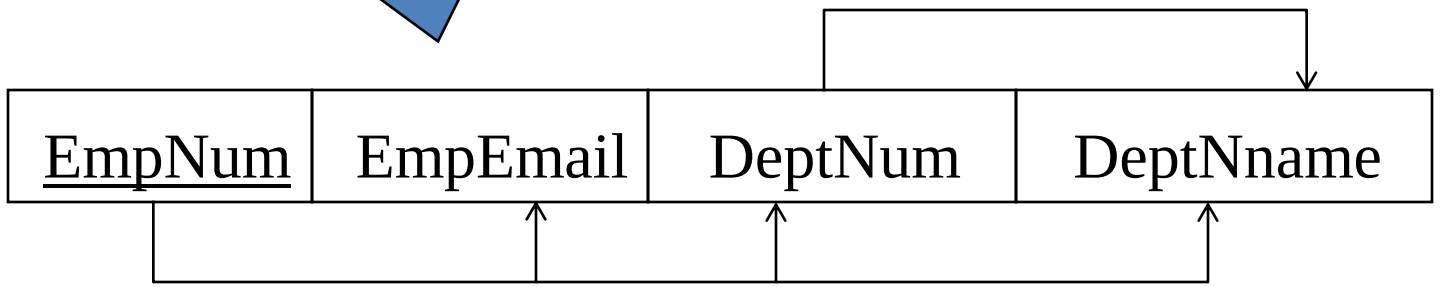
We say that C is transitively dependent on A through B.

Transitive dependency

EmpNum → DeptNum



DeptNum → DeptName



DeptName is *transitively dependent* on EmpNum via DeptNum

EmpNum → DeptName

The main objectives of normalization are:

- **Eliminating redundancy:** Redundancy refers to the repetition of data in a database. By normalizing a database, redundant data is eliminated, and data is stored in a structured and efficient manner. This helps in reducing storage space and improves data consistency.

- **Eliminating data anomalies:** Data anomalies are inconsistencies or irregularities that may occur in a database due to redundant or poorly structured data. Normalization helps to eliminate these anomalies, such as insertion anomalies, deletion anomalies, and update anomalies, by organizing the data in a systematic way.

- **Ensuring data integrity:** Data integrity refers to the accuracy, consistency, and reliability of data in a database. Normalization helps in maintaining data integrity by ensuring that data is stored in a consistent and non-redundant manner, and by establishing relationships between tables using primary keys and foreign keys.

- **Improving performance:** Normalized databases are usually more efficient in terms of storage and retrieval of data compared to denormalized databases. Normalization helps in improving the performance of a database by reducing the amount of redundant data, optimizing queries, and simplifying database operations.

- **Facilitating database maintenance and scalability:** Normalization makes it easier to maintain and update a database, as changes to data only need to be made in one place. It also facilitates scalability, as new data can be added without disrupting the existing structure

Well-Structured Relations

- A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
- Goal is to avoid anomalies
  - **Insertion Anomaly** – adding new rows forces user to create duplicate data
  - **Deletion Anomaly** – deleting rows may cause a loss of data that would be needed for other future rows
  - **Modification Anomaly** – changing data in a row forces changes to other rows because of duplication

Example – Figure 5.2b

EMPLOYEE2

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	SPSS	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

Question – Is this a relation?

Answer – Yes: unique rows and no multivalued attributes

Question – What's the primary key?

Answer – Composite: Emp_ID, Course_Title

Anomalies in this Table

- **Insertion** – can't enter a new employee without having the employee take a class
- **Deletion** – if we remove employee 140, we lose information about the existence of a Tax Acc class
- **Modification** – giving a salary increase to employee 100 forces us to update multiple records

Why do these anomalies exist?

Because we've combined two themes (entity types) into one relation. This results in duplication, and an unnecessary dependency between the entities

Normalization Types

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

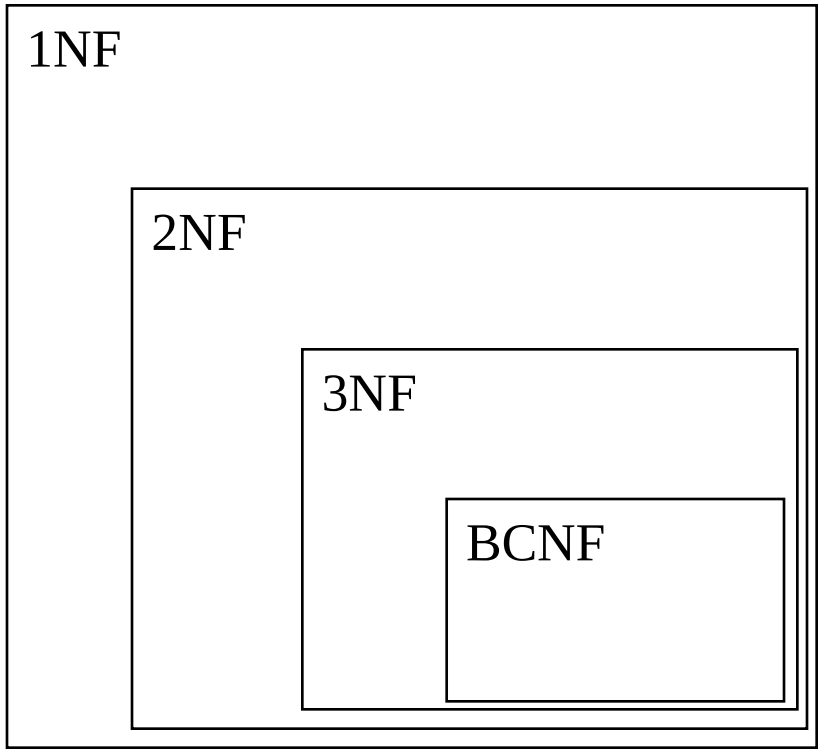
There is a sequence to normal forms:

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

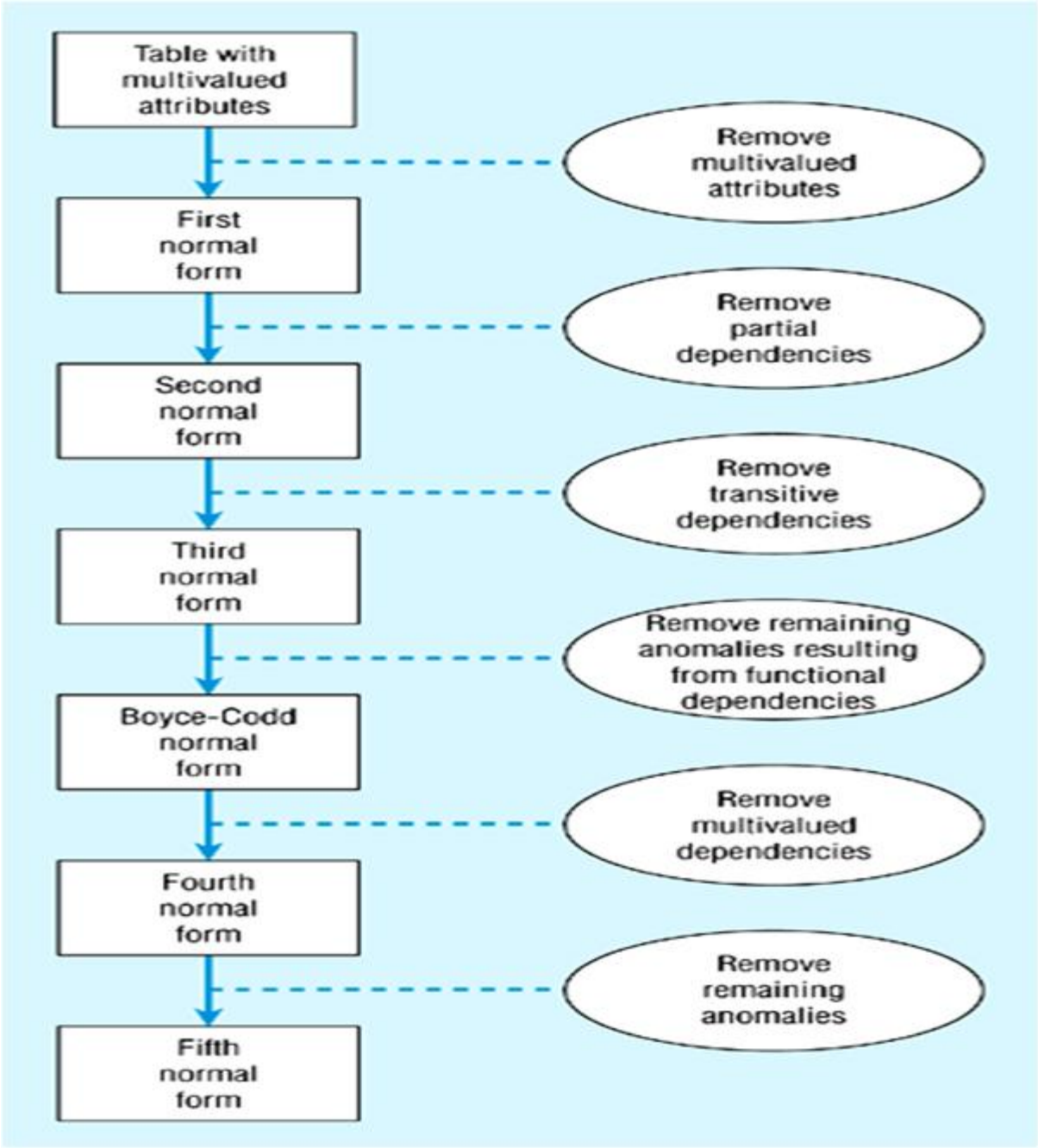
•We consider a relation in BCNF to be fully normalized.

•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.

Steps in normalization



First Normal Form

The key characteristics of a table in 1NF are:

No duplicate rows: Each row in the table must be unique. This is achieved by having a primary key column that uniquely identifies each row in the table. No two rows should be identical.

Atomic values: Each column in a table should contain only atomic values, which are indivisible and cannot be further broken down. This means that a column should not contain multiple values or arrays of values. If a column contains multiple values, it should be split into separate columns.

Composite attributes: Each column in a table should contain non composite attributes only

First Normal Form

The following in **not** in 1NF

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

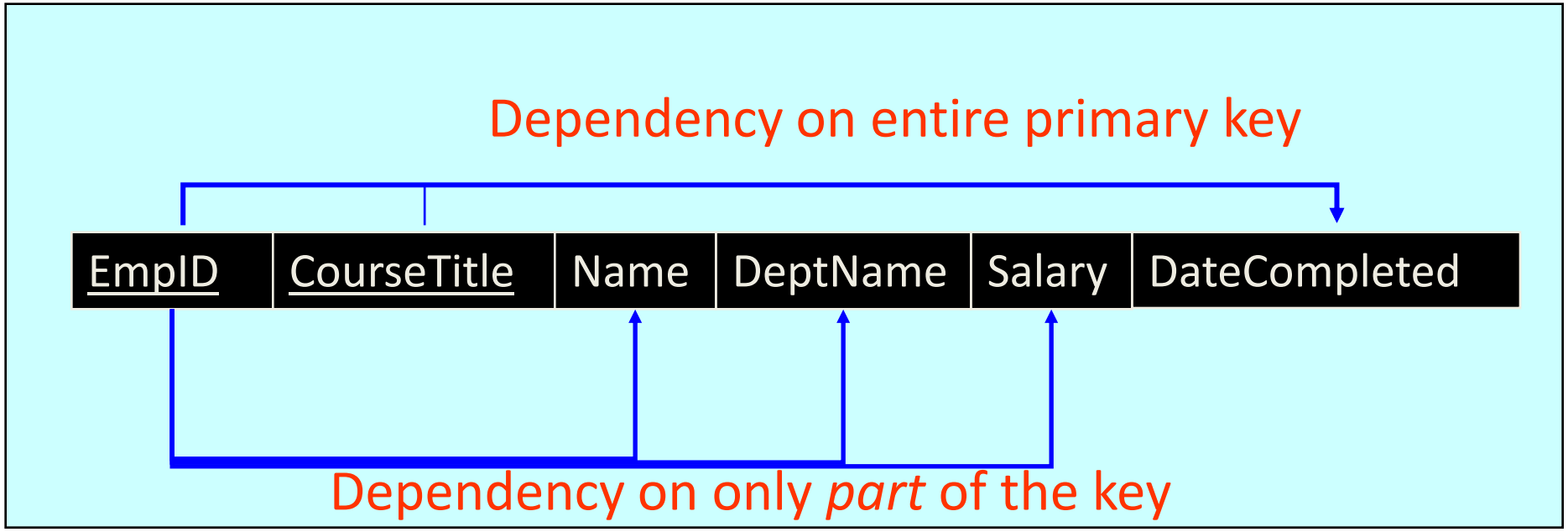
EmpNum	EmpDegree
333	BA
333	BSc
333	PhD
679	BSc
679	MSc

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

Second Normal Form

- 1NF ***plus*** every non-key attribute is fully functionally dependent on the ENTIRE primary key
  - Every non-key attribute must be defined by the entire key, not by only part of the key
  - No **partial functional dependencies**

– Functional Dependencies in EMPLOYEE2



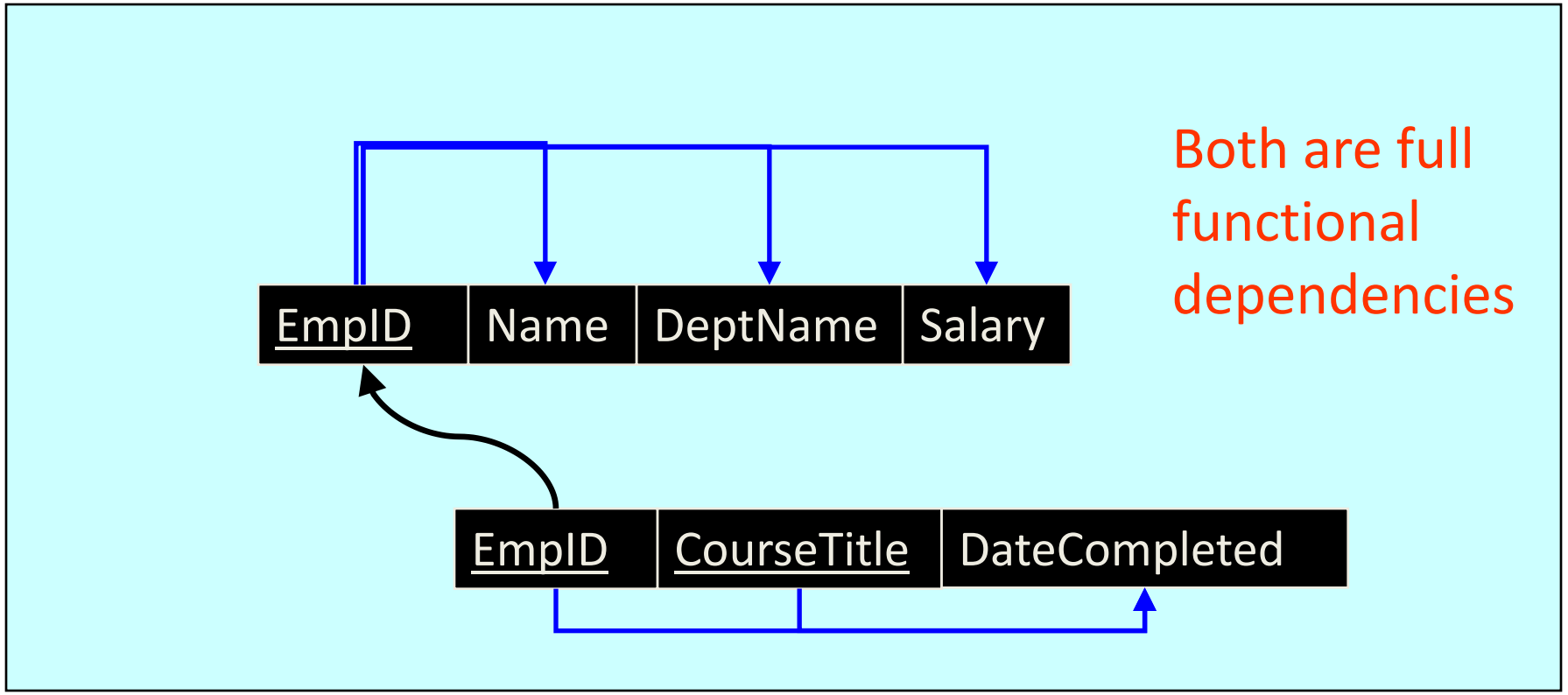
EmpID, CourseTitle ➔ DateCompleted

EmpID ➔ Name, DeptName, Salary

Therefore, NOT in 2nd Normal Form!!

Getting it into 2nd Normal Form

- Decomposed into two separate relations



Second Normal Form

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

Ecode	ProjCode	Dept	Hours

Ecode, ProjCode ⟶ Hours There are two candidate keys.

Ecode ⟶ Dept

Project is **not 2NF** since there is a partial dependency of Dept on Ecode

Second Normal Form

Project

Ecode	ProjCode	Dept	Hours

- The department of a particular employee cannot be recorded until the employee is assigned a project
- If an employee is shifted to another dept this information should be recorded at multiple instances
- If the employee completes work on the project his/her record will be deleted the information regarding the department the employee belongs to will also be lost

Ecode	ProjCode	Hours

Ecode	Dept

Third Normal Form

Third Normal Form

- A relation is in **3NF** if the relation is in 2NF 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.

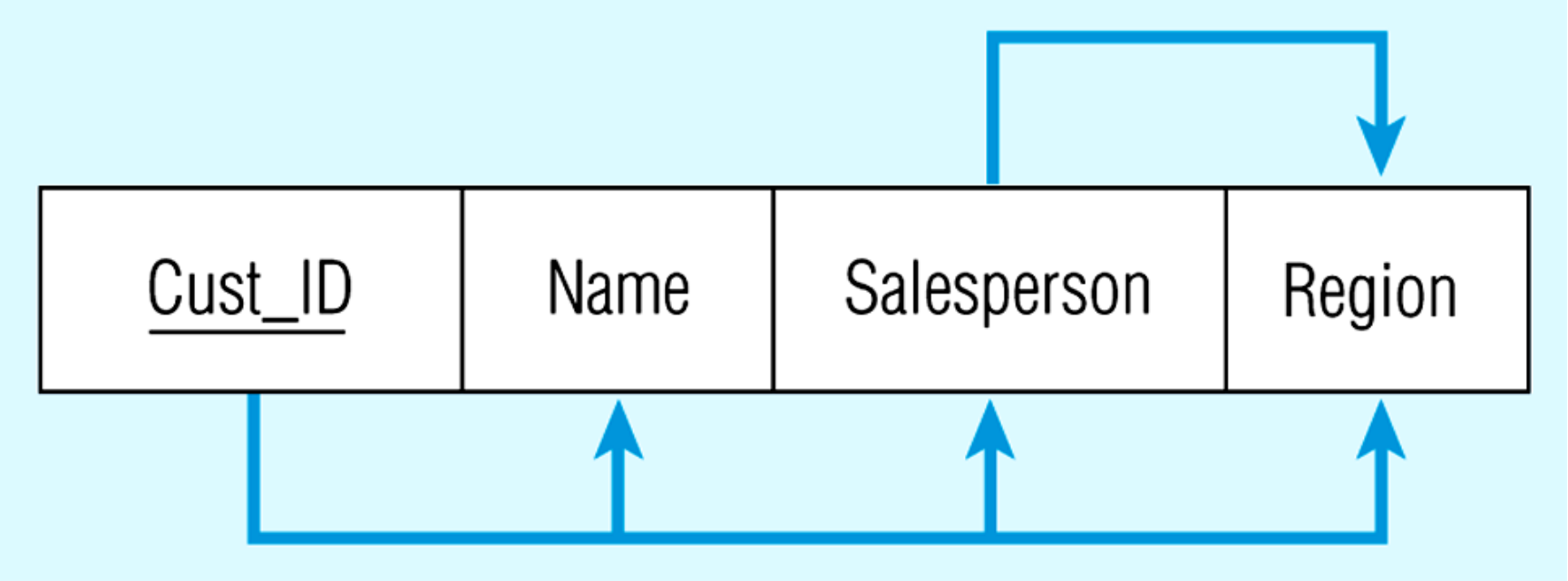
Figure 5-24 -- Relation with transitive dependency

(a) SALES relation with simple data

SALES

Cust_ID	Name	Salesperson	Region
8023	Anderson	Smith	South
9167	Bancroft	Hicks	West
7924	Hobbs	Smith	South
6837	Tucker	Hernandez	East
8596	Eckersley	Hicks	West
7018	Arnold	Faulb	North

Figure 5-24(b) Relation with transitive dependency



CustID → Name
CustID → Salesperson
CustID → Region

All this is OK
(2nd NF)

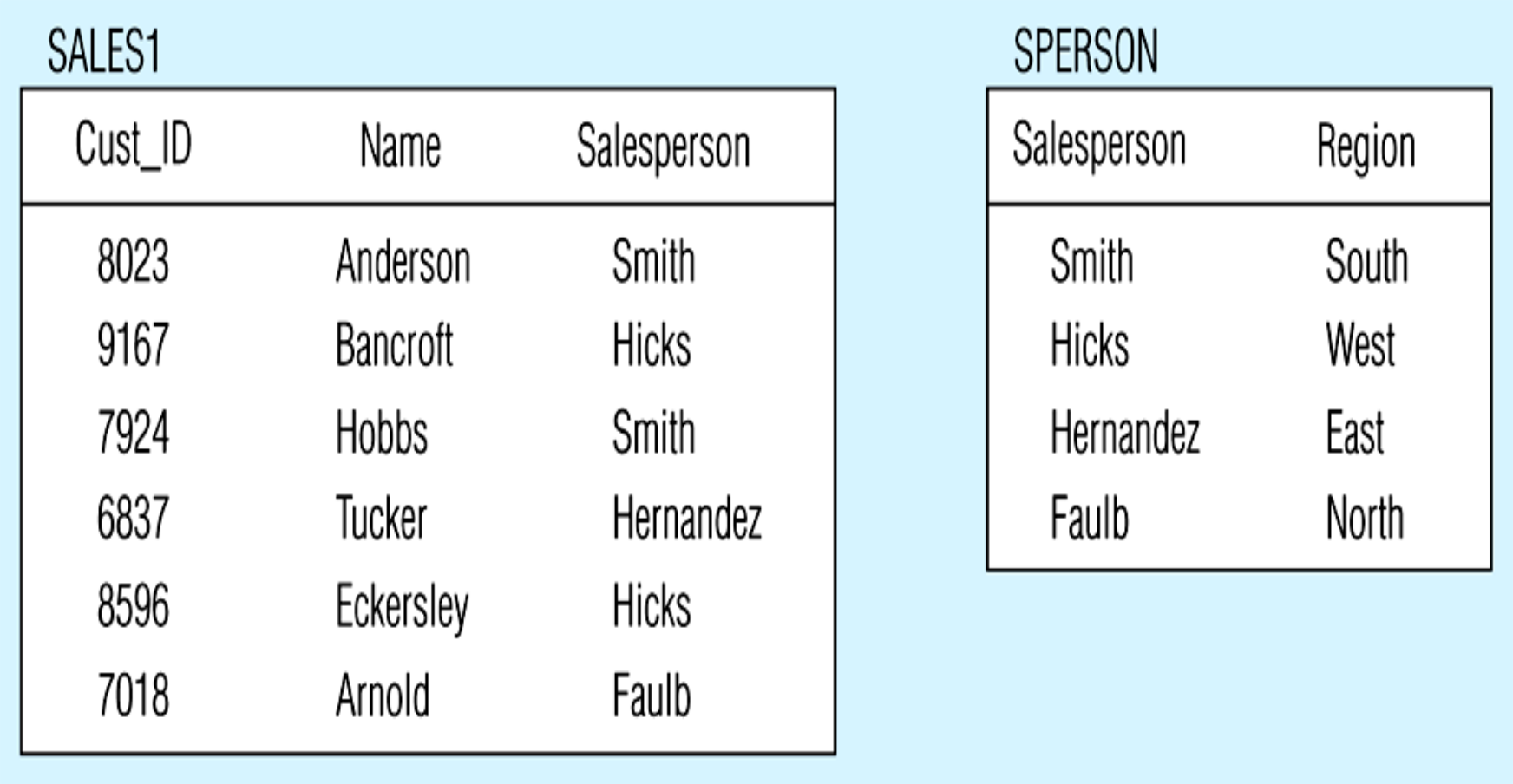
BUT

CustID → Salesperson → Region

Transitive dependency
(not 3rd NF)

Figure 5.25 -- Removing a transitive dependency

(a) Decomposing the SALES relation



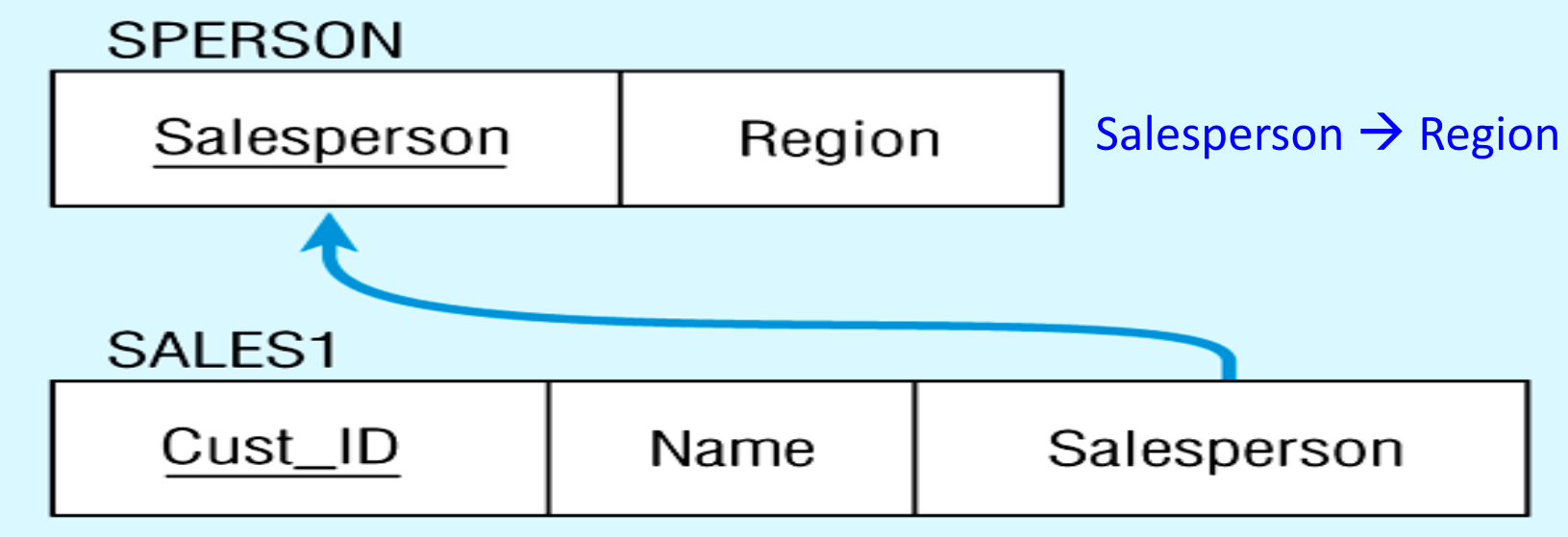
SALES1

Cust_ID	Name	Salesperson
8023	Anderson	Smith
9167	Bancroft	Hicks
7924	Hobbs	Smith
6837	Tucker	Hernandez
8596	Eckersley	Hicks
7018	Arnold	Faulb

SPERSON

Salesperson	Region
Smith	South
Hicks	West
Hernandez	East
Faulb	North

Figure 5.25(b) Relations in 3NF



CustID → Name

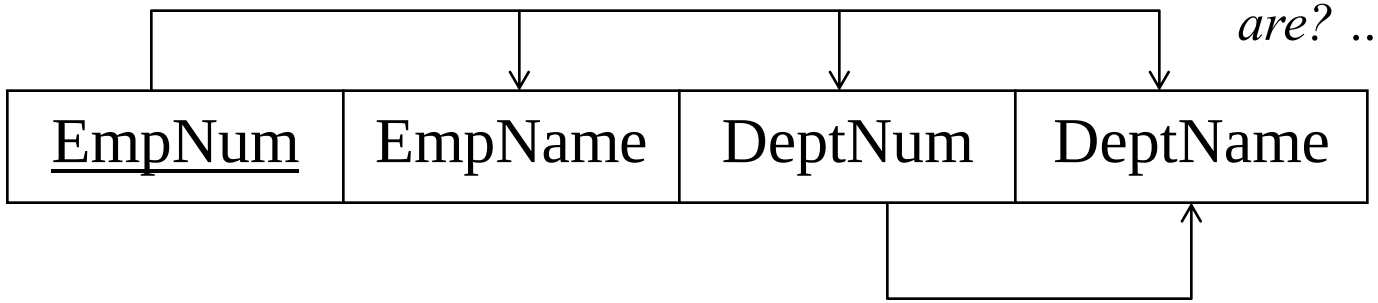
CustID → Salesperson

Now, there are no transitive dependencies…
Both relations are in 3rd NF

Third Normal Form

Consider this **Employee** relation

Candidate keys are? …

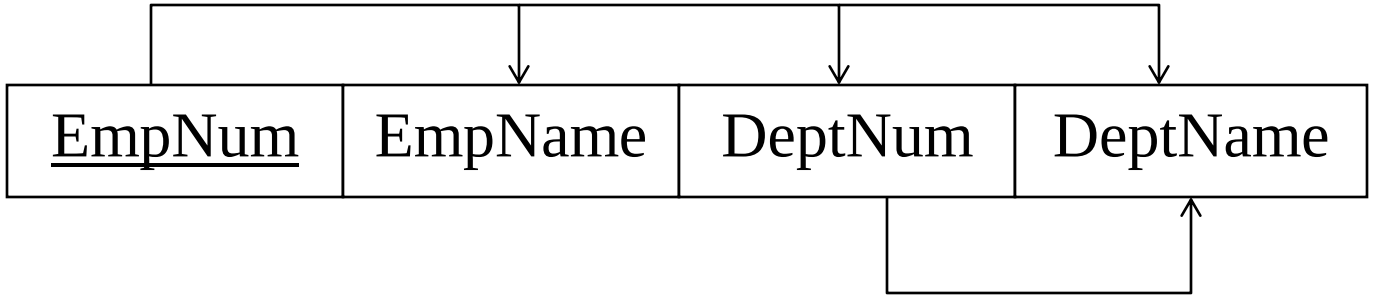


EmpName, DeptNum, and DeptName are non-key attributes.

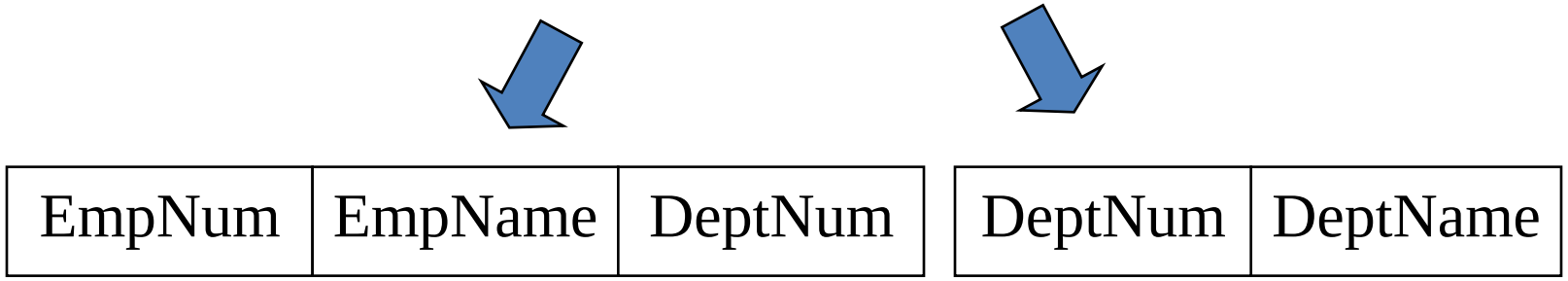
DeptNum determines DeptName, a non-key attribute, and DeptNum is not a candidate key.

Is the relation in 3NF? … no Is the relation in BCNF? … no

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.

Other Normal Forms

- Boyce-Codd NF
  - All determinants are candidate keys…there is no determinant that is not a unique identifier
- 4th NF
  - No multivalued dependencies
- 5th NF
  - No “lossless joins”
- Domain-key NF
  - The “ultimate” NF…perfect elimination of all possible anomalies

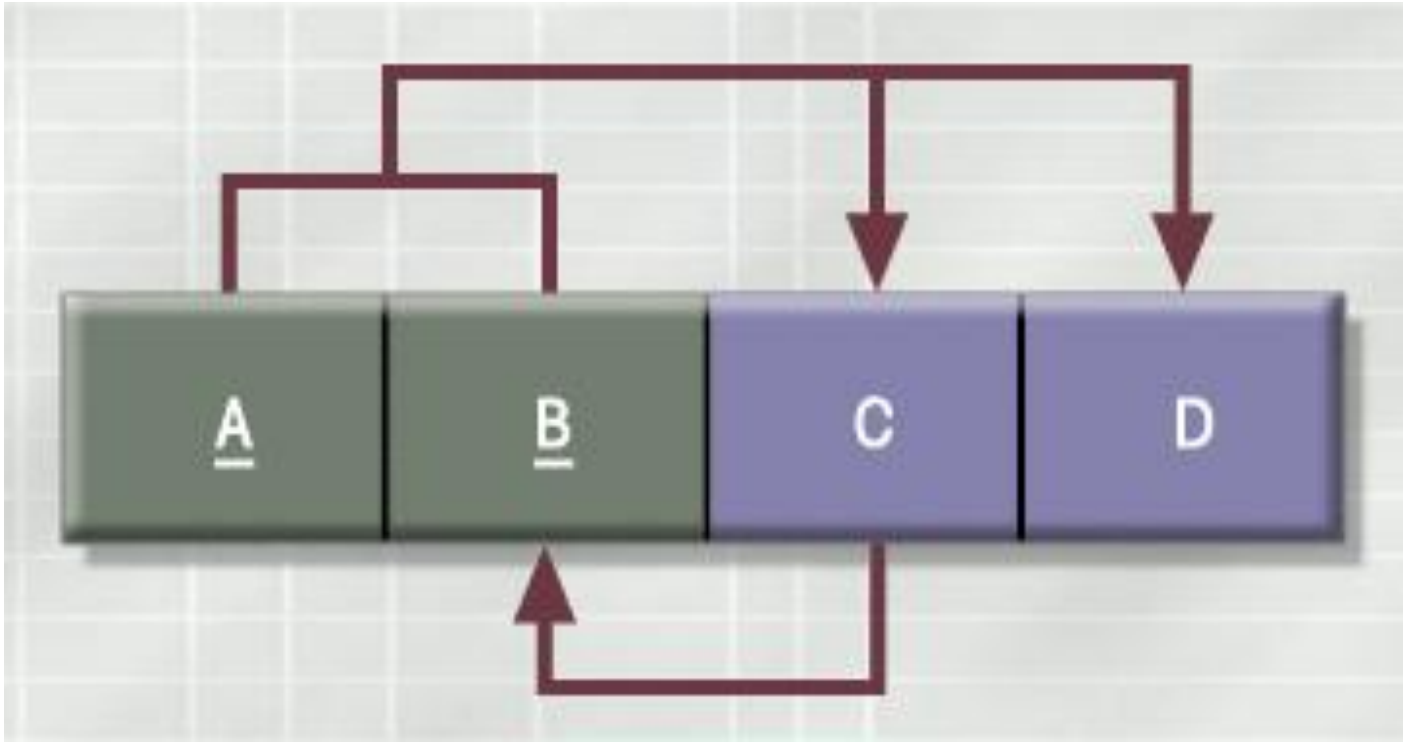
Boyce-Codd Normal Form (BCNF)

- **A table is in Boyce-Codd normal form (BCNF) if every determinant in the table is a candidate key.**

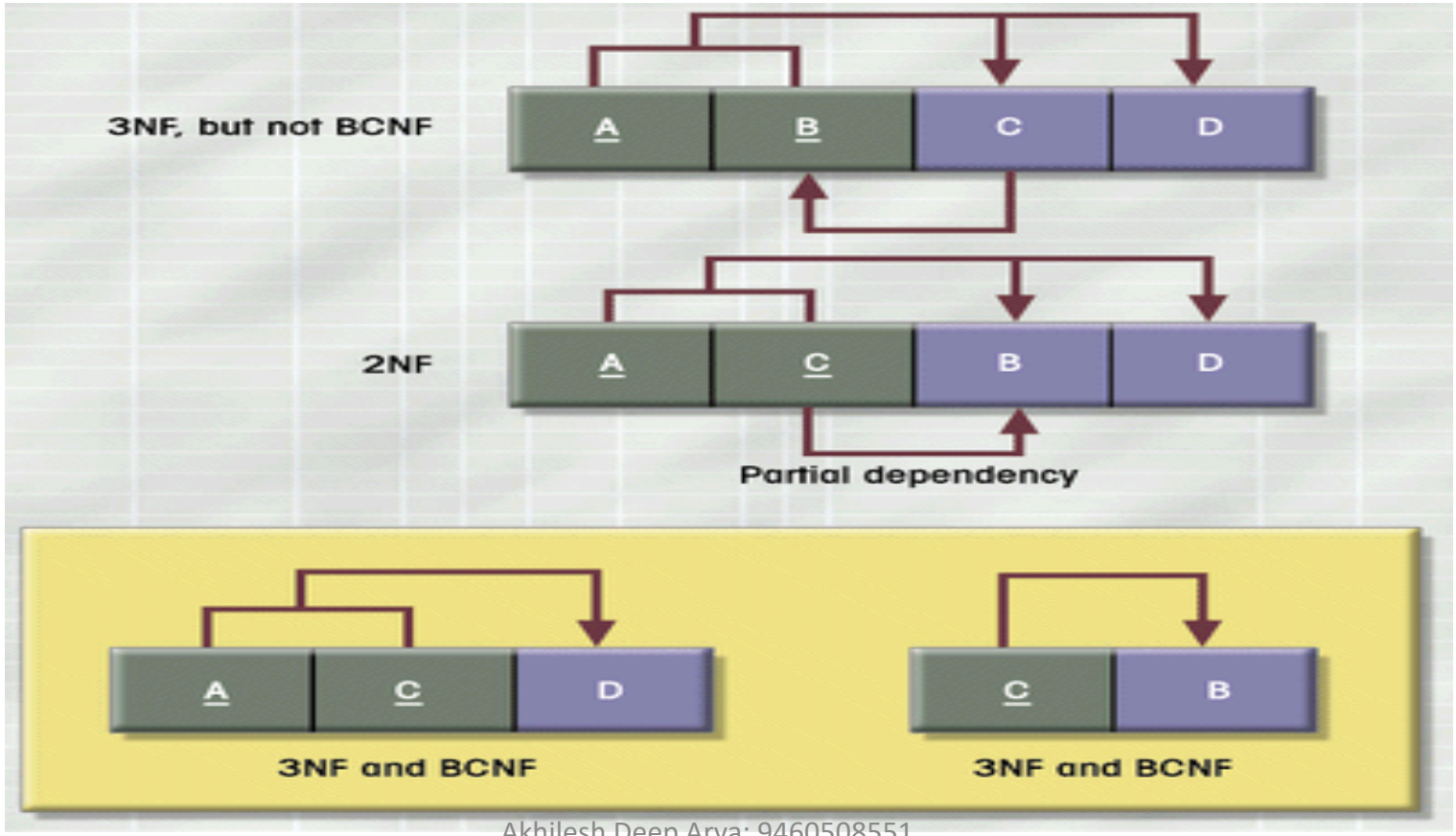
  (A determinant is any attribute whose value determines other values with a row.)

- **If a table contains only one candidate key, the 3NF and the BCNF are equivalent.**
- **BCNF is a special case of 3NF.**

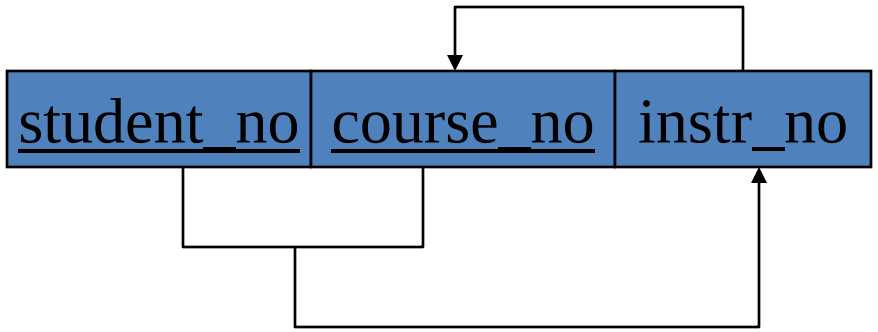
A Table That Is In 3NF But Not In BCNF



The Decomposition of a Table Structure to Meet BCNF Requirements



In 3NF, but not in BCNF:

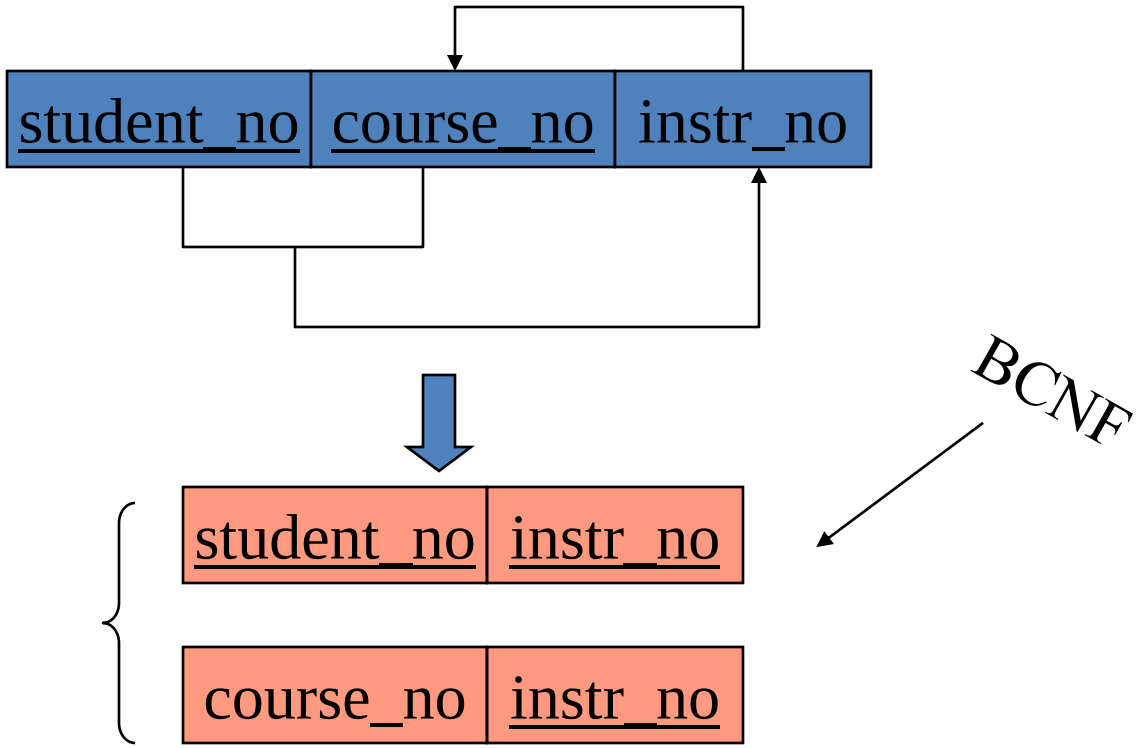


Instructor teaches one course only.

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.



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

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

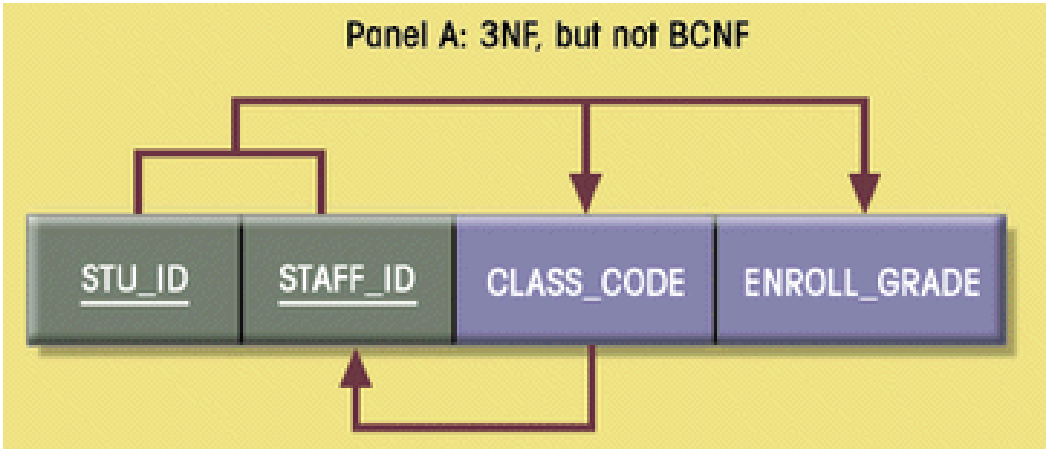
$instr\_no \rightarrow course\_no$

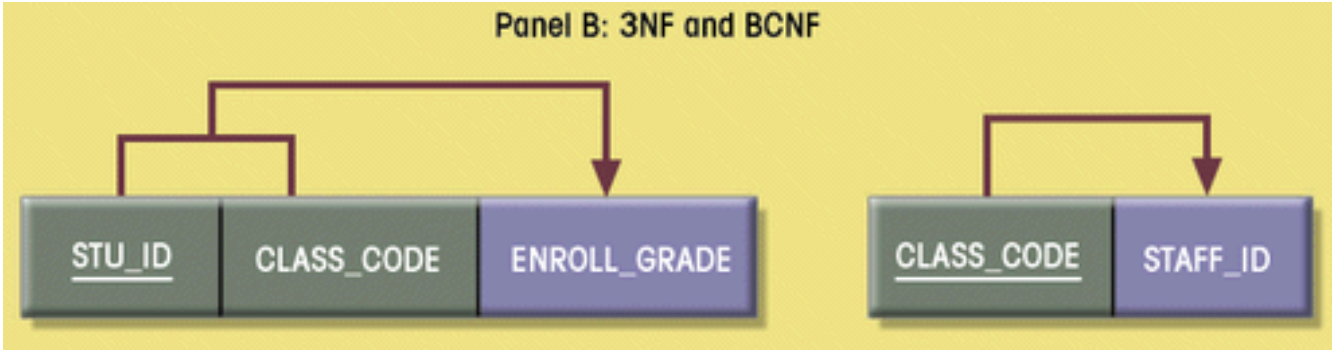
Sample Data for a BCNF Conversion

TABLE 5.2 SAMPLE DATA FOR A BCNF CONVERSION

STU_ID	STAFF_ID	CLASS_CODE	ENROLL_GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	27563	C
144	20	32456	B

Decomposition into BCNF





Denormalization

- **Normalization is only one of many database design goals.**
- **Normalized (decomposed) tables require additional processing, reducing system speed.**
- **Normalization purity is often difficult to sustain in the modern database environment. The conflict between design efficiency, information requirements, and processing speed are often resolved through compromises that include denormalization.**