NORMALIZATION: DATABASE DESIGN THEORY

Contents

- Informal design guidelines for relation schemas,
- Functional Dependencies,
- Normal Forms Based on Primary Keys,
- Second and Third Normal Forms,
- Boyce-Codd Normal Form,
- Multivalued Dependencies and Fourth Normal Form,
- Join Dependencies and Fifth Normal Form.
- **Normalization Algorithms:** Inference Rules,
- Equivalence, and Minimal cover,
- Properties of Relational Decompositions,
- Algorithms for Relational Database Schema Design.

Informal design guidelines for relation schemas

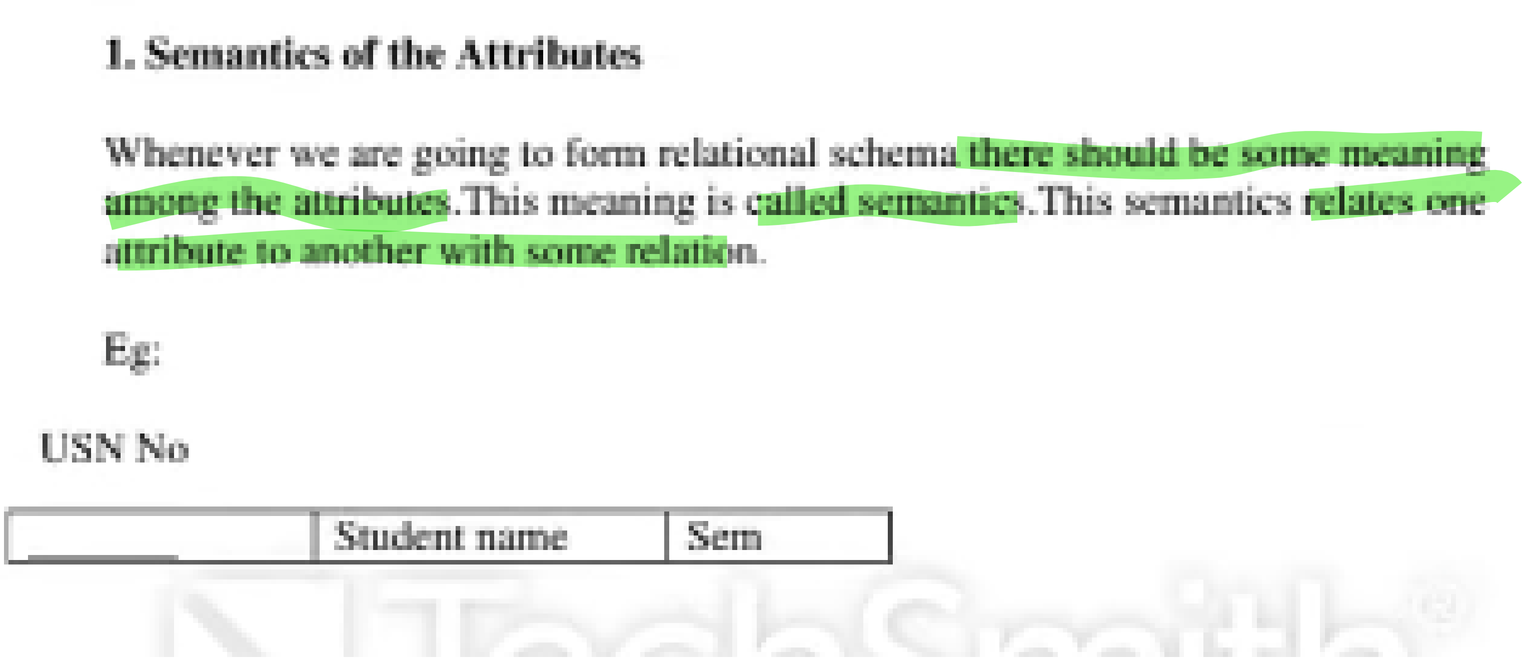
Discuss the informal design guidelines for relational schema in detail with an example for each.

Four informal guidelines that may be used as measures to determine the quality of relation schema design:

- Making sure that the semantics of the attributes is clear in the schema
- Reducing the redundant information in tuples
- Reducing the NULL values in tuples
- Disallowing the possibility of generating spurious tuples

Semantics of the Relation Attributes

Semantics, specifies how to interpret the attribute values stored in a tuple of the relation-in other words, how the attribute values in a tuple relate to one another.



1. Semantics of the Attributes

Whenever we are going to form relational schema there should be some meaning among the attributes.This meaning is called semantics.This semantics relates one attribute to another with some relation.

Eg:

USN No

	Student name	Sem

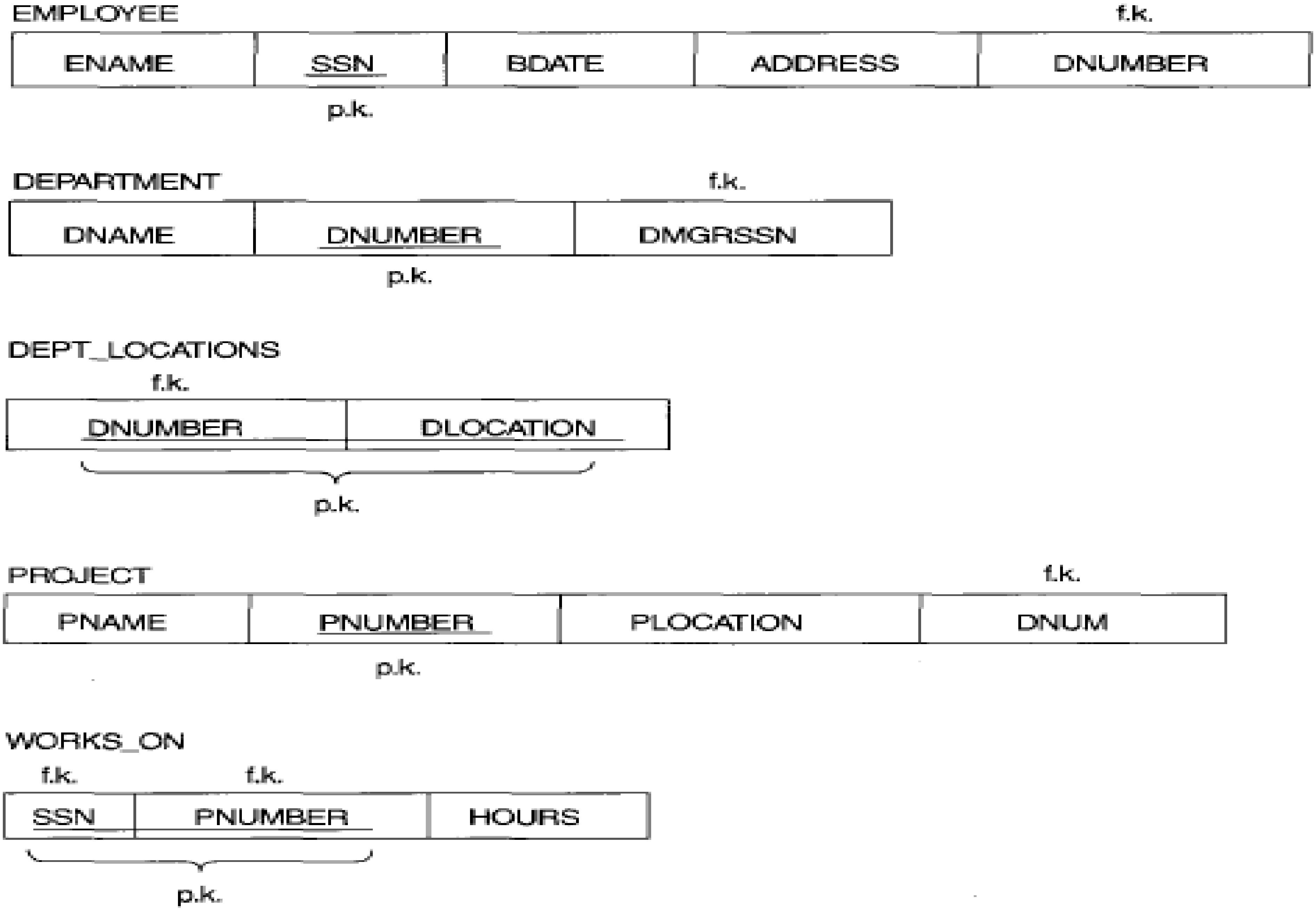
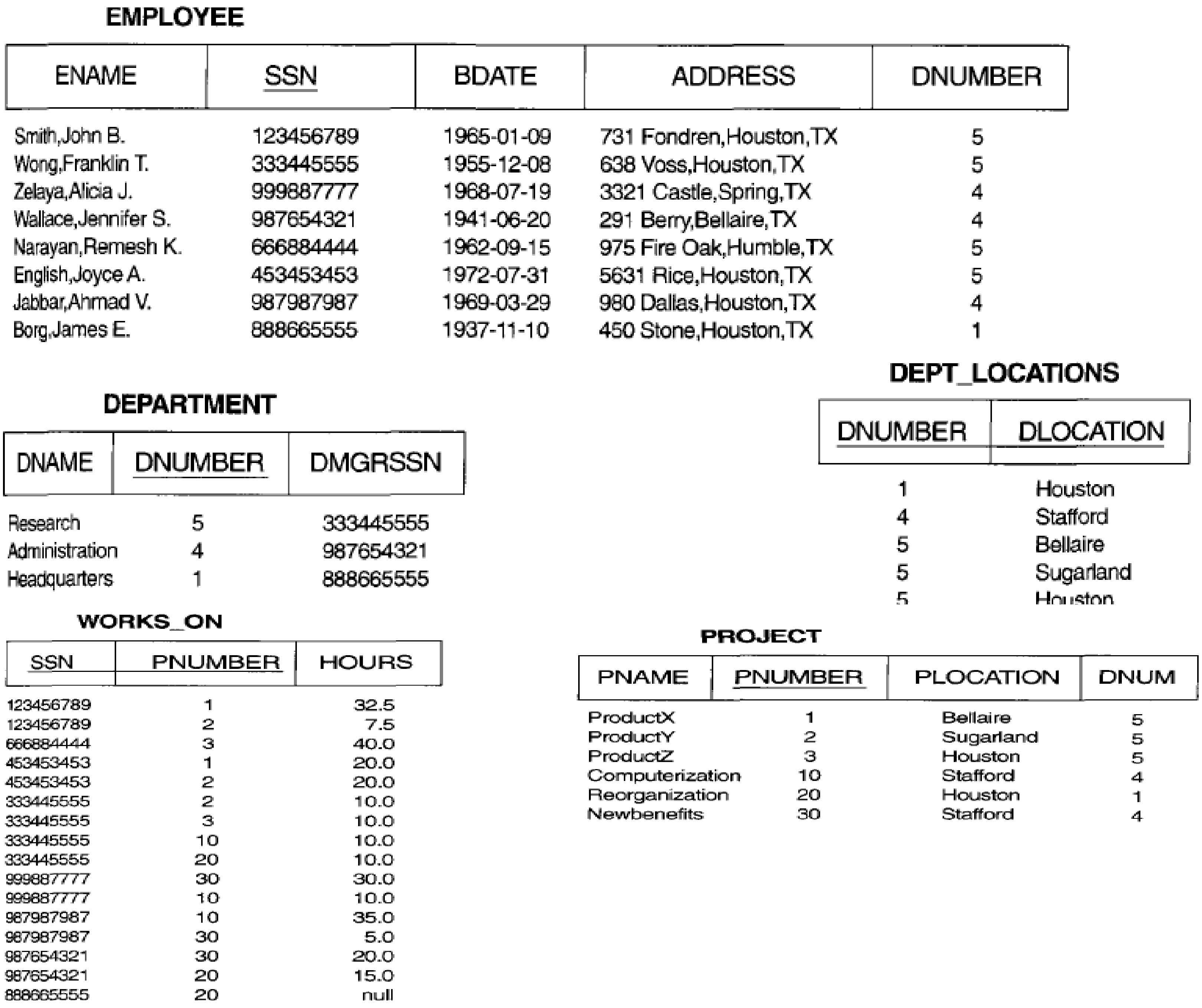


FIGURE 10.1 A simplified COMPANY relational database schema.



EMPLOYEE

ENAME	SSN	BDATE	ADDRESS	DNUMBER
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5
Zelaya,Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4
Narayan,Remesh K.	666884444	1962-09-15	975 Fire Oak,Humble,TX	5
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1

DEPARTMENT

DNAME	DNUMBER	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

SSN	PNUMBER	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	null

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

FIGURE **10.2** Example database state for the relational database schema of Figure 10.1.

- The DNUMBER attribute is a foreign key that represents an *implicit relationship between EMPLOYEE and* DEPARTMENT.
- The ease with which the meaning of a relation's attributes can be explained is an *informal measure of how well the relation is designed.*
- In DEPT_LOCATIONS and WORKS_ON, the schema DEPT_LOCATIONS represents a multi-valued attribute of DEPARTMENT, where as WORKS_ON represents an M:N relationship between EMPLOYEE and PROJ ECT
- Hence, all the relation schemas may be considered as easy to explain and hence good from the standpoint of having clear semantics. We can thus formulate the following informal design guideline.

GUIDELINE 1

- Design a relation schema so that it is easy to explain its meaning.
- Do not combine attributes from multiple entity types and relationship types into a single relation.

There is nothing wrong logically with these two relations, they are considered poor designs because they violate Guideline 1 by mixing attributes from distinct real-world entities

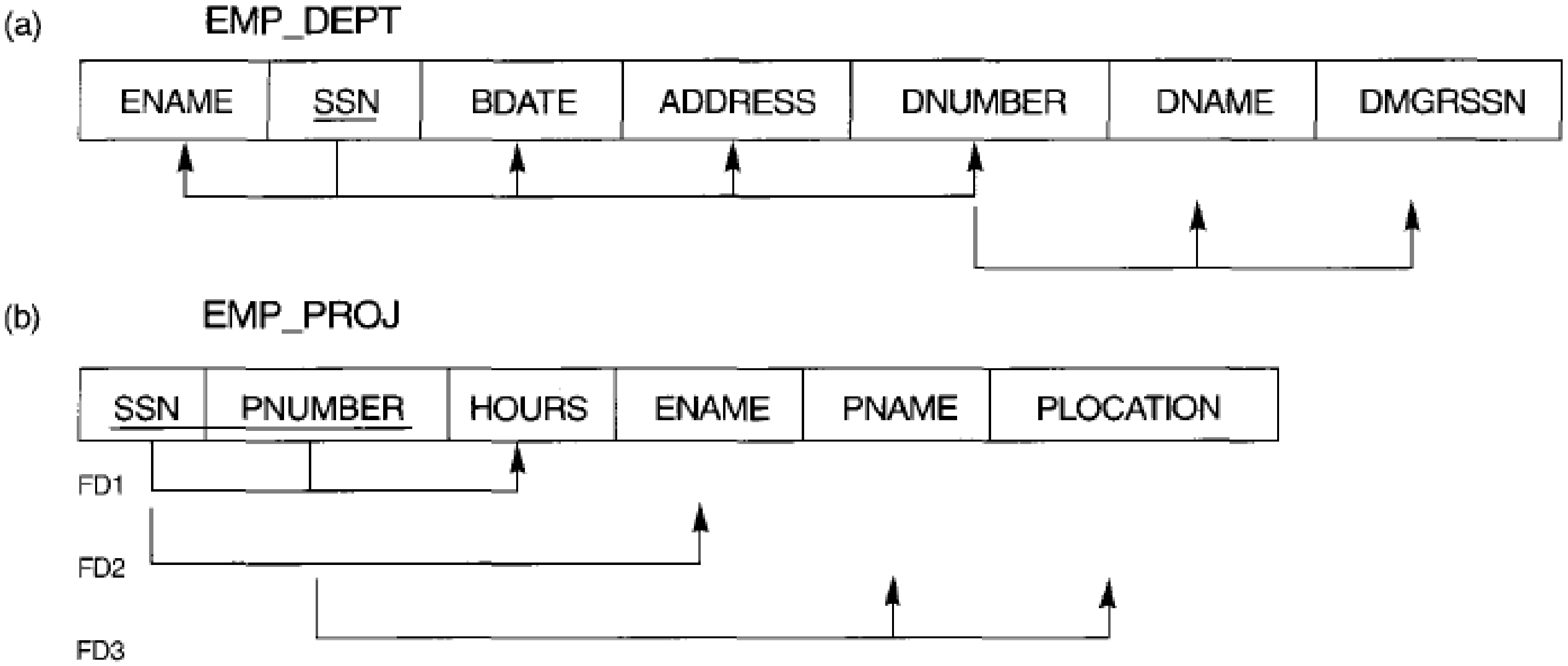
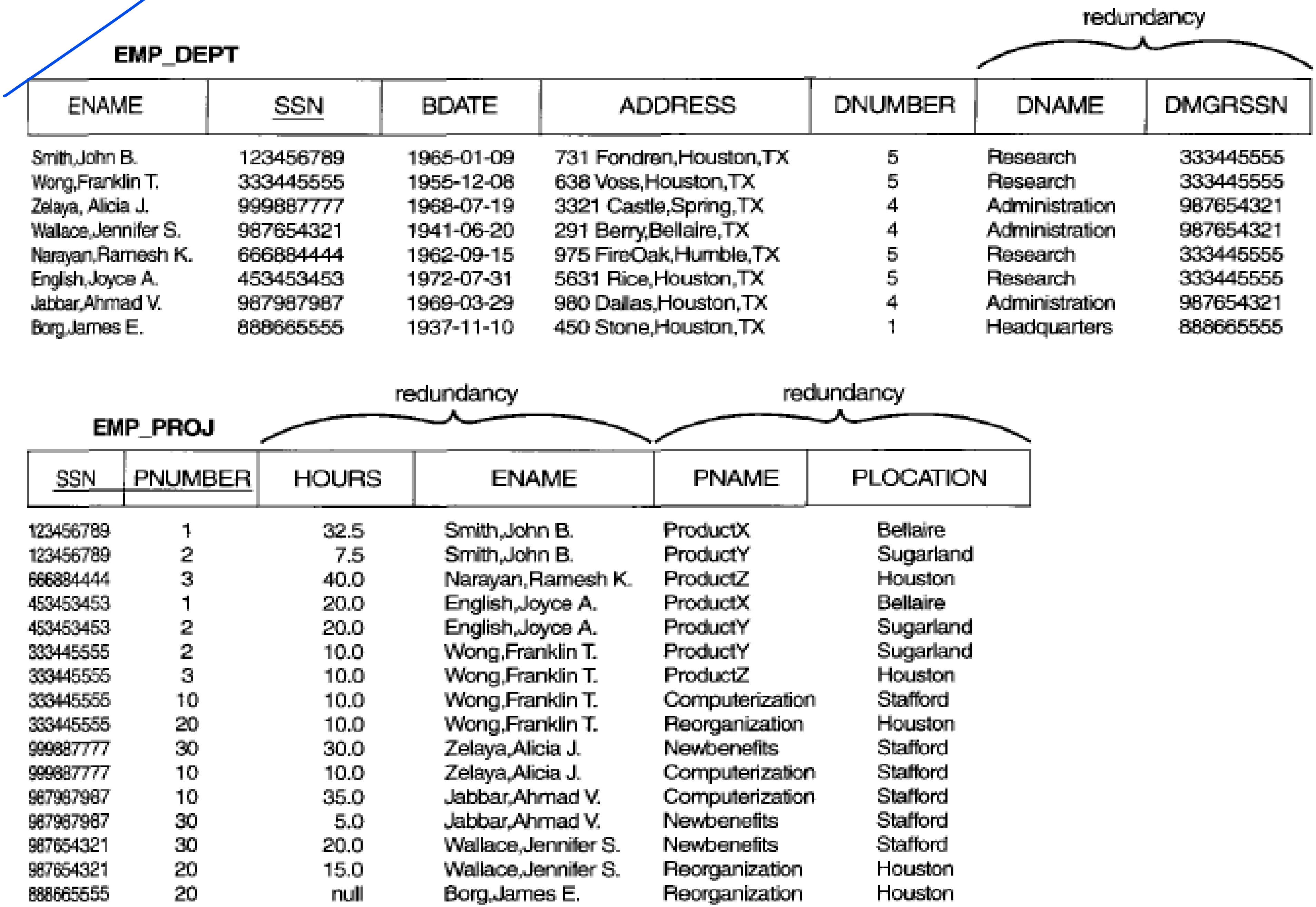


FIGURE 10.3 Two relation schemas suffering from update anomalies.

Redundant Information in Tuples and Update Anomalies

One goal of schema design is to minimize the storage space used by the base relations.



redundancy
redundancy
redundancy

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith,John B.	ProductX	Bellaire
123456789	2	7.5	Smith,John B.	ProductY	Sugarland
666884444	3	40.0	Narayan,Ramesh K.	ProductZ	Houston
453453453	1	20.0	English,Joyce A.	ProductX	Bellaire
453453453	2	20.0	English,Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong,Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong,Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong,Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong,Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya,Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya,Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar,Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar,Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace,Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace,Jennifer S.	Reorganization	Houston
888665555	20	null	Borg,James E.	Reorganization	Houston

FIGURE 10.4 Example states for EMP_DEPT and EMP_PROJ resulting from applying NATURAL JOIN to the relations in Figure 10.2. These may be stored as base relations for performance reasons.

- Another serious problem with using the relations in as base relations is the problem of update anomalies.
- These can be classified into insertion anomalies, deletion anomalies, and modification anomalies.

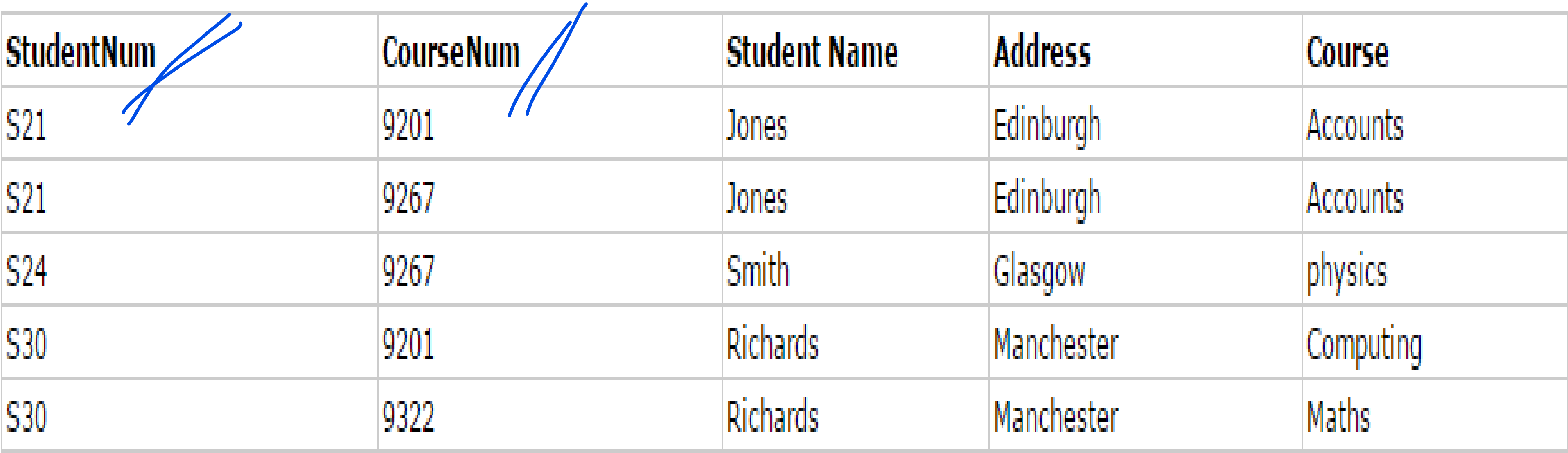
Insertion Anomalies:

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

Insertion anomalies can be differentiated into two types, illustrated by the following examples based on the EMP_DEPT relation.

1. To insert a new employee tuple into EMP_DEPT, we must include either the attribute values for the department that the employee works for, or nulls.
2. It is difficult to insert a new department that has no employees as yet in the EMP_DEPT relation.

we can't add a new course unless we have at least one student enrolled on the course.



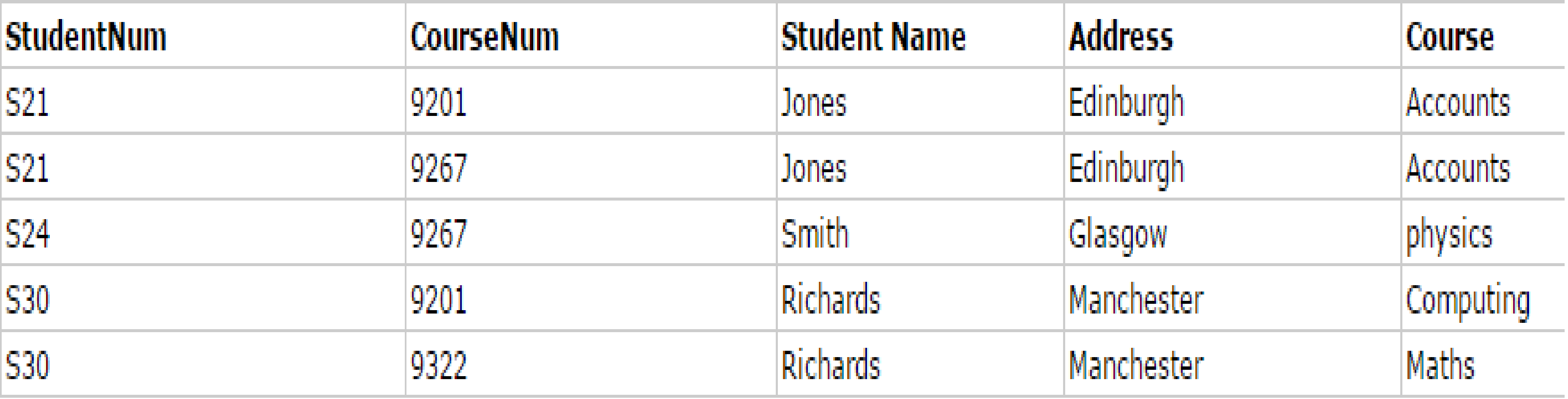
StudentNum	CourseNum	Student Name	Address	Course
S21	9201	Jones	Edinburgh	Accounts
S21	9267	Jones	Edinburgh	Accounts
S24	9267	Smith	Glasgow	physics
S30	9201	Richards	Manchester	Computing
S30	9322	Richards	Manchester	Maths

Deletion Anomalies:

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

- If we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database

Consider what happens if Student S30 is the last student to leave the course - All information about the course is lost.



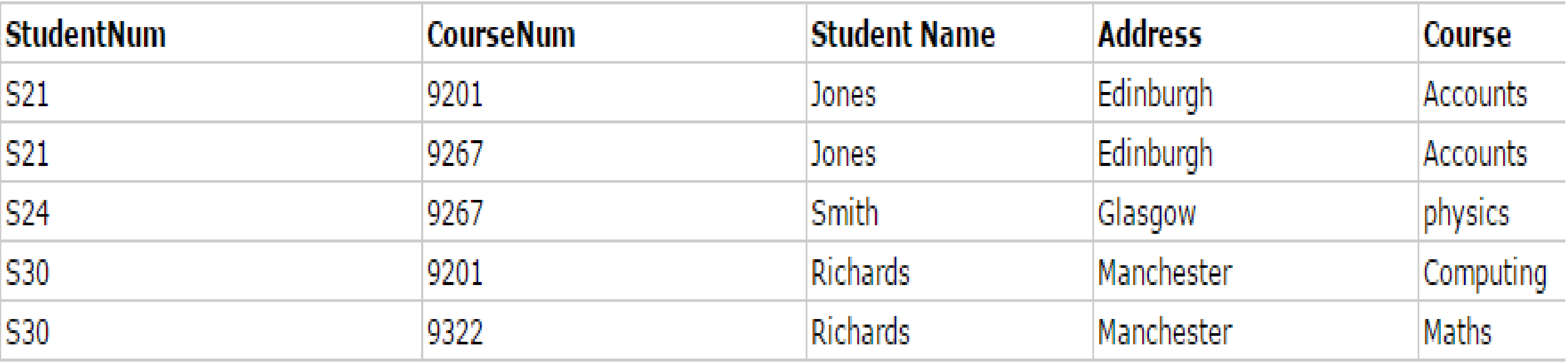
StudentNum	CourseNum	Student Name	Address	Course
S21	9201	Jones	Edinburgh	Accounts
S21	9267	Jones	Edinburgh	Accounts
S24	9267	Smith	Glasgow	physics
S30	9201	Richards	Manchester	Computing
S30	9322	Richards	Manchester	Maths

Modification Anomalies:

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

- In EMP_DEPT, if we change the value of one of the attributes of a particular department-say, the manager of department 5-we must update the tuples of all employees who work in that department; otherwise, the database will become inconsistent.

Consider Jones moving address - you need to update all instances of Jones's address.



StudentNum	CourseNum	Student Name	Address	Course
S21	9201	Jones	Edinburgh	Accounts
S21	9267	Jones	Edinburgh	Accounts
S24	9267	Smith	Glasgow	physics
S30	9201	Richards	Manchester	Computing
S30	9322	Richards	Manchester	Maths

Based on the preceding three anomalies, we can state the guideline that follows

GUIDELINE 2

- Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations.
- If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.

Null Values in Tuples

- In some schema designs, we may group many attributes together into a "fat" relation.
- If many of the attributes do not apply to all tuples in the relation, we end up with many nulls in those tuples.
- Moreover, nulls can have multiple interpretations, such as the following:

1. The attribute *does not apply to this tuple.*
2. The attribute value for this tuple is *unknown.*
3. The value is *known but absent; that is, it has not been recorded yet.*

GUIDELINE 3:

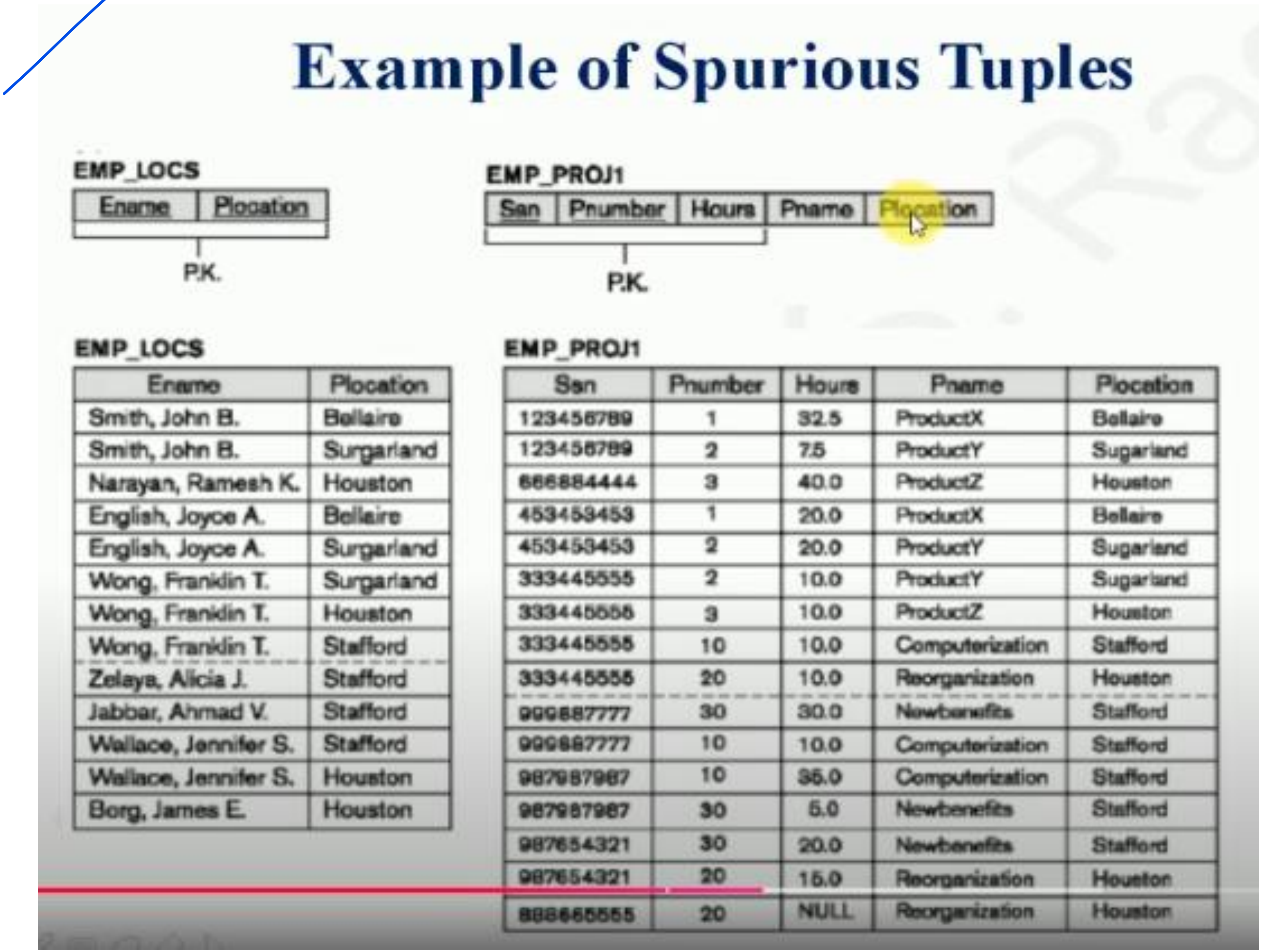
- As far as possible, avoid placing attributes in a base relation whose values may frequently be null.
- If nulls are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation.

Generation of Spurious Tuples

A spurious tuple is, basically, a record in a database that gets created when two tables are joined badly.

Let us consider two relation schema
Emp_Locs(ename, plocation)
Emp_proj1(eno, pnumber, hours, pname, plocation)

If we attempt a natural join operation on above relation schema, the result produces many more tuples than the original set of tuples. Additional tuples that were not there in Emp_proj are called spurious tuples because they represent wrong information which is not valid



Example of Spurious Tuples

EMP_LOCS

Ename	Plocation

P.K.

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation

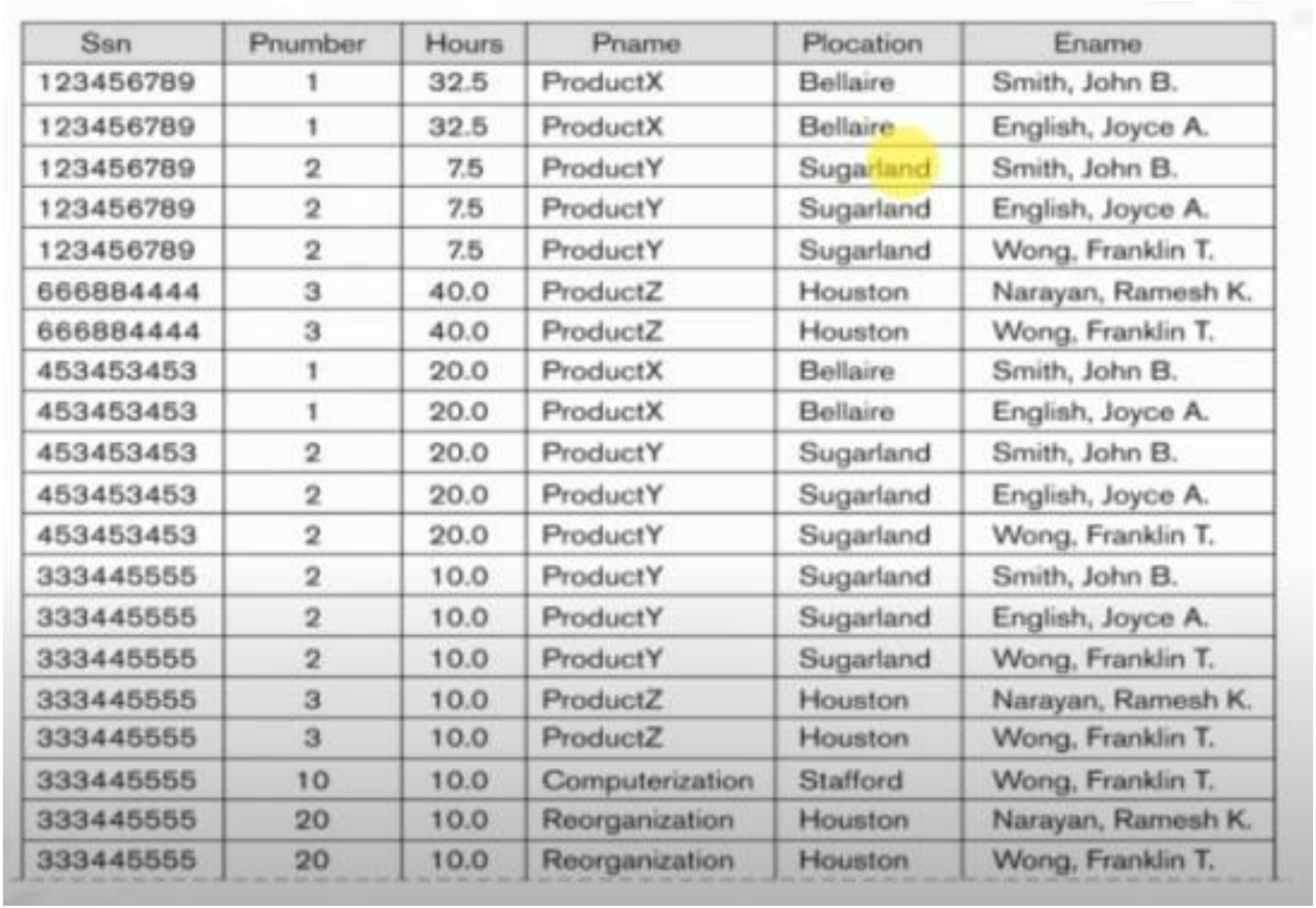
P.K.

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Surgarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Surgarland
Wong, Franklin T.	Surgarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston



Ssn	Pnumber	Hours	Pname	Plocation	Ename
123456789	1	32.5	ProductX	Bellaire	Smith, John B.
123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
123456789	2	7.5	ProductY	Sugarland	Smith, John B.
123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
453453453	1	20.0	ProductX	Bellaire	Smith, John B.
453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
453453453	2	20.0	ProductY	Sugarland	Smith, John B.
453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
333445555	2	10.0	ProductY	Sugarland	Smith, John B.
333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

Lost information

- A First Example of Lost Information
  - What is Lost in the Join of R and S?

R = (A, B, C)

A	B	C
a1	b2	c1
a2	b2	c1
a3	b4	c2

S = (D, C)

D	C
d1	c1
d2	c2
d4	c2
d5	c3

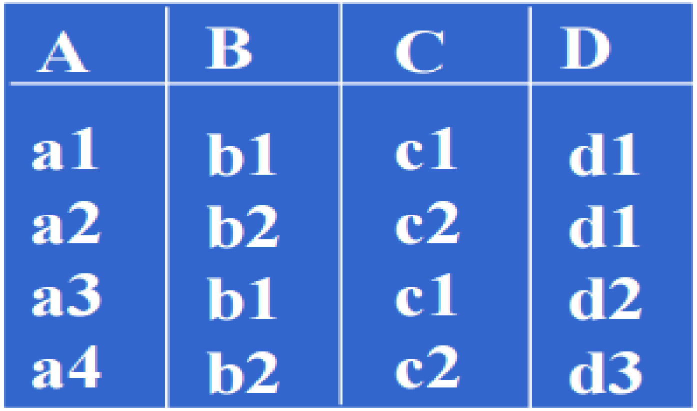
RS(A, B, C, D)

A	B	C	D
a1	b2	c1	d1
a2	b2	c1	d1
a3	b4	c2	d2
a3	b4	c2	d4

lost info of (d5, c3) after join R & S

Consider the relation R (A, B, C, D) having 4 tuples in Base relation

R(A, B, C, D)



A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d1
a3	b1	c1	d2
a4	b2	c2	d3

R1 (B, C) JOIN R2 (A, D)

A	B	C	D
A1	B1	C1	D1
A1	B2	C2	D1
A2	B2	C2	D1
A2	B1	C1	D1
A3	B1	C1	D2
A4	B2	C2	D3

GUIDELINE 4

- Design relation schemas so that they can be joined with equality conditions on attributes that are either primary keys or foreign keys in a way that guarantees that no spurious tuples are generated.
- Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations, because joining on such attributes may produce spurious tuples.

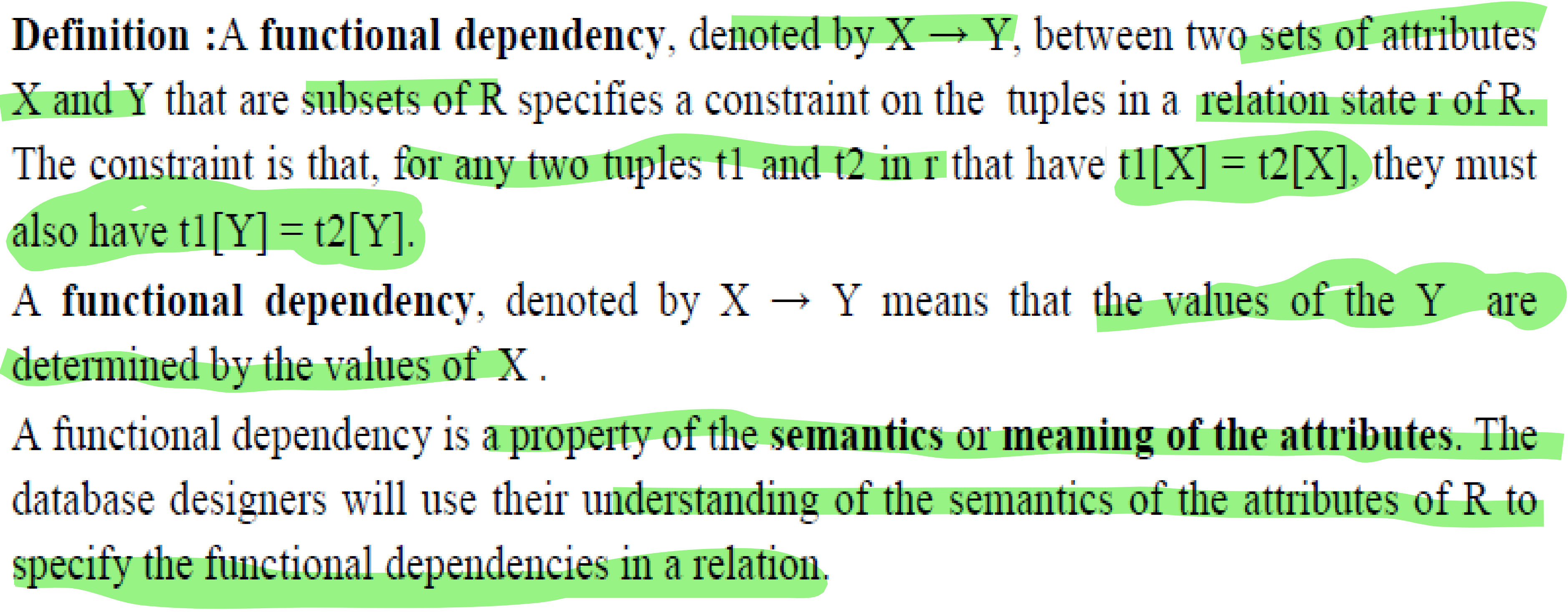
Summary and Discussion of Design Guidelines

The problems we pointed out, which can be detected without additional tools of analysis, are as follows:

1. Anomalies that cause redundant work to be done during insertion into and modification of a relation, and that may cause accidental loss of information during a deletion from a relation
2. Waste of storage space due to nulls and the difficulty of performing aggregation operations and joins due to null values
3. Generation of invalid and spurious data during joins on improperly related base relations

Give the mathematical definition of the functional dependency and explain with an example.

Functional Dependencies



Definition :A **functional dependency**, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a constraint on the tuples in a relation state r of R. The constraint is that, for any two tuples t1 and t2 in r that have t1[X] = t2[X], they must also have t1[Y] = t2[Y].

A **functional dependency**, denoted by $X \rightarrow Y$ means that the values of the Y are determined by the values of X .

A functional dependency is a property of the **semantics** or **meaning of the attributes**. The database designers will use their understanding of the semantics of the attributes of R to specify the functional dependencies in a relation.

Consider the relation schema EMP_PROJ from the semantics of the attributes and the relation, the following functional dependencies should hold:

a. Ssn→Ename

b. Pnumber →{Pname, Plocation}

c. {Ssn, Pnumber}→Hours

These functional dependencies specifies that

(a) the value of an employee's Social Security number (Ssn) uniquely determines the employee name (Ename),

(b) the value of a project's number (Pnumber) uniquely determines the project name (Pname) and location (Plocation), and

(c) Combination of Ssn and Pnumber values uniquely determines the number of hours the employee currently works on the project per week (Hours).

Types of functional dependency :

1. A functional dependency $X \rightarrow Y$ is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold any more;

   Ex: **{Ssn, Pnumber} → Hours** is a full dependency (neither Ssn → Hours nor Pnumber→Hours holds).

2. A functional dependency $X \rightarrow Y$ is a **partial functional dependency** if removal of any attribute A from X and the dependency still holds;

   Ex: **{Ssn, Pnumber}→Ename** is partial because Ssn→Ename holds.

3. A functional dependency $X \rightarrow Y$ in a relation schema R is a **transitive dependency** if there exists a set of attributes Z in R such that $X \rightarrow Z$ and $Z \rightarrow Y$ hold.

   Ex: The dependency Ssn→Dmgr_ssn is transitive in EMP_DEPT, because of the dependencies Ssn → Dnumber and Dnumber → Dmgr_ssn.

4. **Trivial Functional Dependency**. If a **functional dependency** (FD) X → Y holds, where Y is a subset of X, then it is called a **trivial** FD.

   Non-**trivial** − If an FD X → Y holds, where Y is not a subset of X, then it is called a non-**trivial** FD.

Normal Forms Based on Primary Keys

Normalization of Relations

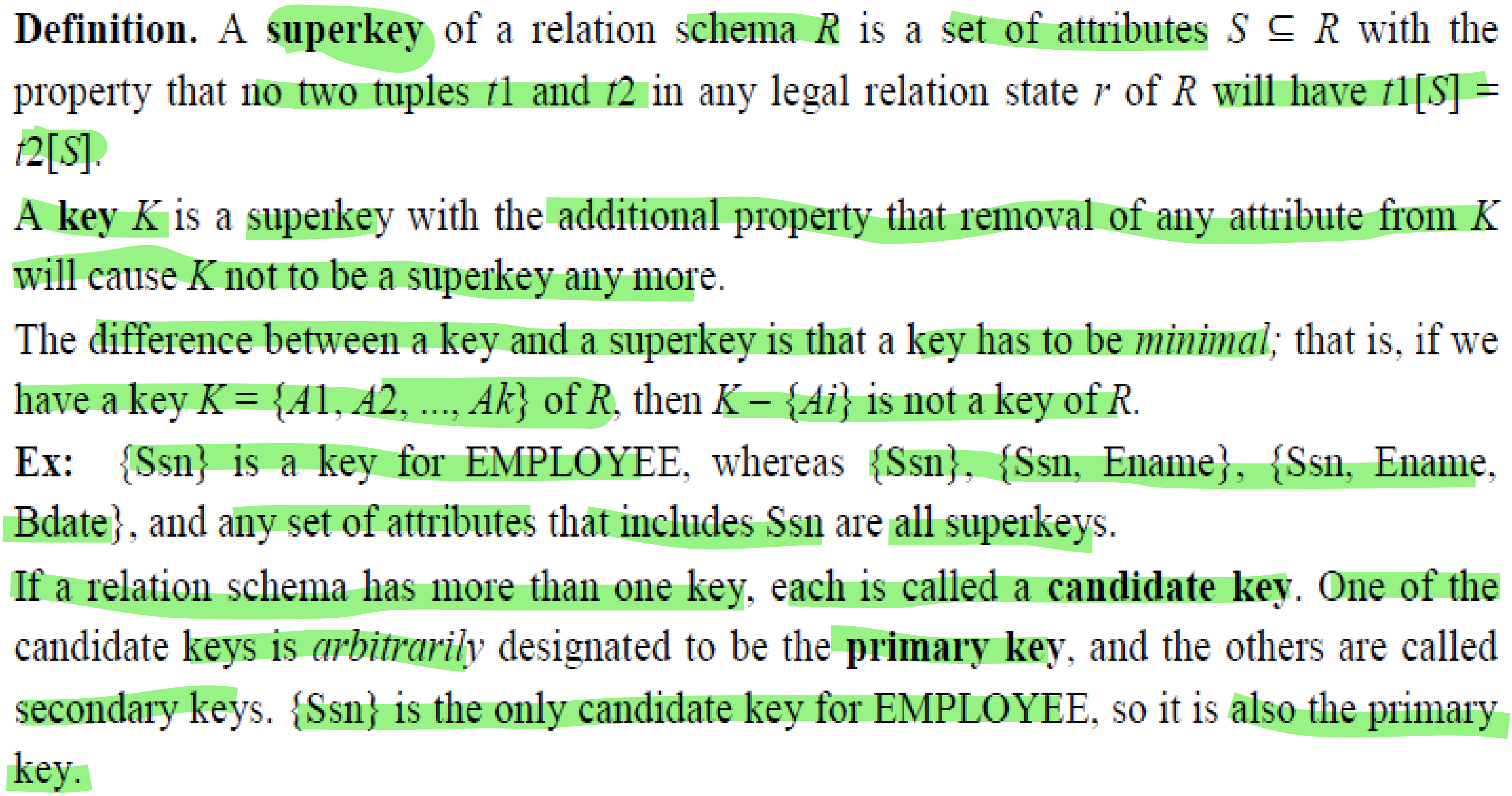
The normalization process, as first proposed by **Codd** (1972). Codd proposed three normal forms, which he called first, second, third normal form and Boyce-Codd normal form (BCNF). All these normal forms are based on **functional dependencies among the attributes** of a relation.

Later, a fourth normal form (4NF) and a fifth normal form (5NF) were proposed, based on the concepts of **multivalued dependencies and join dependencies, respectively;**

- **Normalization of data** can be considered a process of analyzing the given relation schemas based on their **FDs and primary keys** to achieve the desirable properties of (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anomalies.
- It can be considered as a “filtering” or “purification” process to make the design have successively better quality.

- **Definition.** The **normal form** of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized.

Definitions of Keys and Attributes Participating in Keys.



Definition. A **superkey** of a relation schema R is a set of attributes $S \subseteq R$ with the property that no two tuples $t1$ and $t2$ in any legal relation state r of R will have $t1[S] = t2[S]$.

A **key** K is a superkey with the additional property that removal of any attribute from K will cause K not to be a superkey any more.

The difference between a key and a superkey is that a key has to be *minimal*; that is, if we have a key $K = \{A1, A2, ..., Ak\}$ of R, then $K - \{Ai\}$ is not a key of R.

Ex: {Ssn} is a key for EMPLOYEE, whereas {Ssn}, {Ssn, Ename}, {Ssn, Ename, Bdate}, and any set of attributes that includes Ssn are all superkeys.

If a relation schema has more than one key, each is called a **candidate key**. One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called secondary keys. {Ssn} is the only candidate key for EMPLOYEE, so it is also the primary key.

Definition. An attribute of relation schema *R* is called a **prime attribute** of *R* if it is a member of some candidate key of *R*. An attribute is called **nonprime** if it is not a prime attribute—that is, if it is not a member of any candidate key.

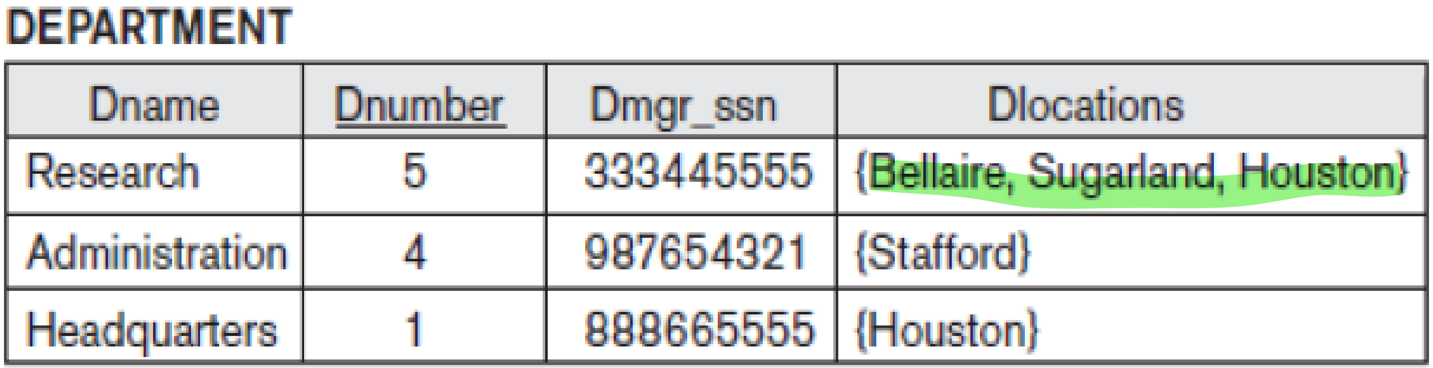
Ex: Ssn and Pnumber are prime attributes of WORKS_ON, whereas other attributes of WORKS_ON are nonprime.

1NF

First Normal Form (1NF)

First normal form (1NF) states that the domain of an attribute must include only *atomic* (simple, indivisible) *values* and that the value of any attribute in a tuple must be a *single value* from the domain of that attribute.

Consider the following DEPARTMENT relation, It is not in 1NF. Because the domain of Dlocations contains sets of values and hence is nonatomic.



DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

There are three main techniques to achieve first normal form for such a relation:

1. Remove the attribute Dlocations that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key Dnumber of DEPARTMENT. The primary key of this relation is the combination {Dnumber, Dlocation}.

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

2. Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT. In this case, the primary key becomes the combination {Dnumber, Dlocation}. This solution has the disadvantage of introducing *redundancy* in the relation.

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

3. If a maximum number of values is known for the attribute for example, if it is known that at most three locations can exist for a department—replace the Dlocations attribute by three atomic attributes: Dlocation1, Dlocation2, and Dlocation3. This solution has the disadvantage of introducing NULL values if most departments have fewer than three locations.

Second Normal Form (2NF)

Definition. A relation schema *R* is in 2NF if every nonprime attribute *A* in *R* is *fully functionally dependent* on the primary key of *R*.

Second normal form (2NF) is based on the concept of full functional dependency. A functional dependency X → Y is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold any more.

Example1: {Ssn, Pnumber} → Hours is a full dependency (neither Ssn → Hours nor Pnumber→Hours holds).

A functional dependency X → Y is a **partial functional dependency** if removal of any attribute A from X and the dependency still holds;

Example2: The dependency {Ssn, Pnumber}→Ename is partial because Ssn→Ename holds.

If a relation schema is not in 2NF, it can be *second normalized by* decomposing EMP_PROJ into the three relation schemas EP1, EP2, and EP3 shown in Figure 15.11(a), each of which is in 2NF.

separate all FD's into each table

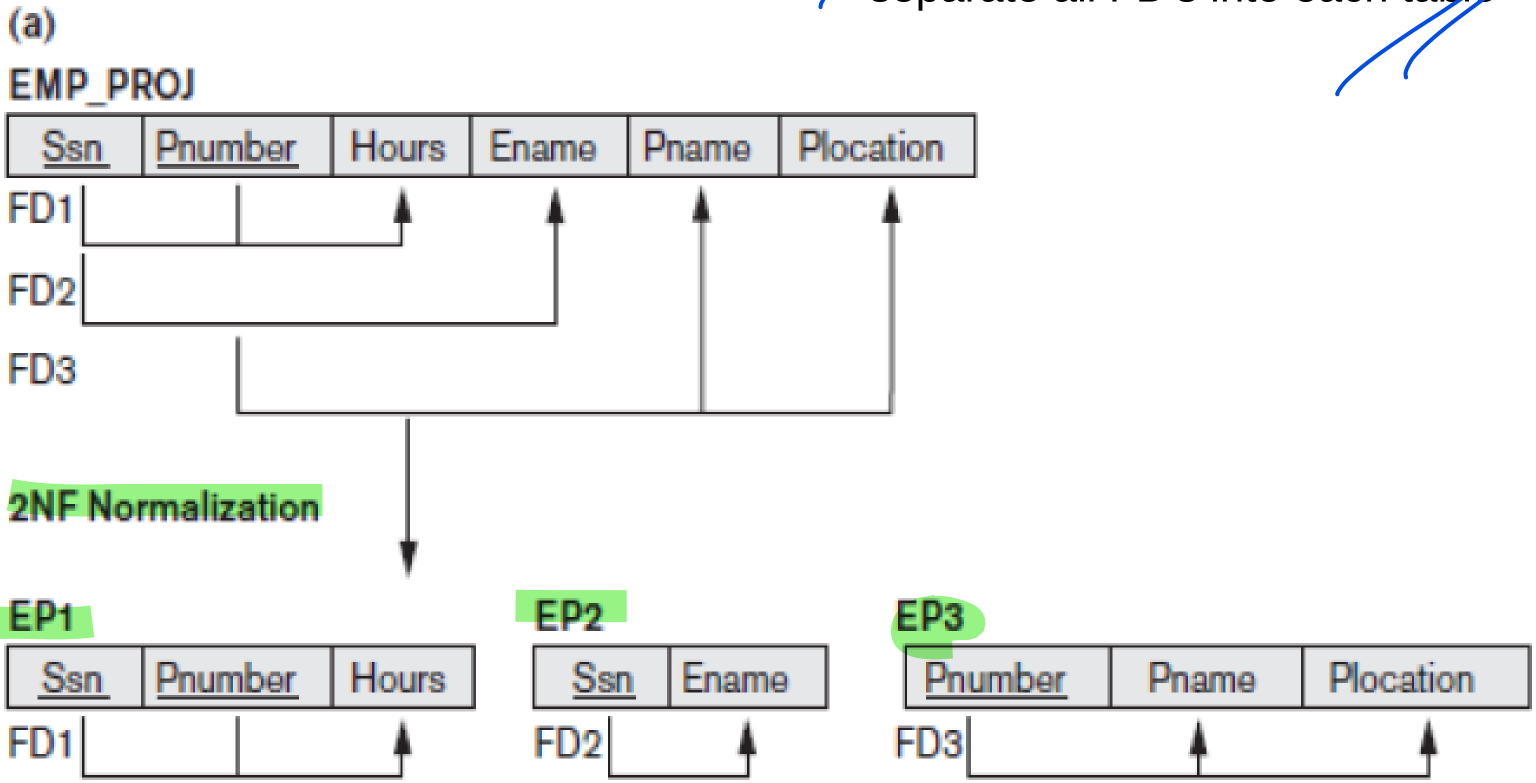


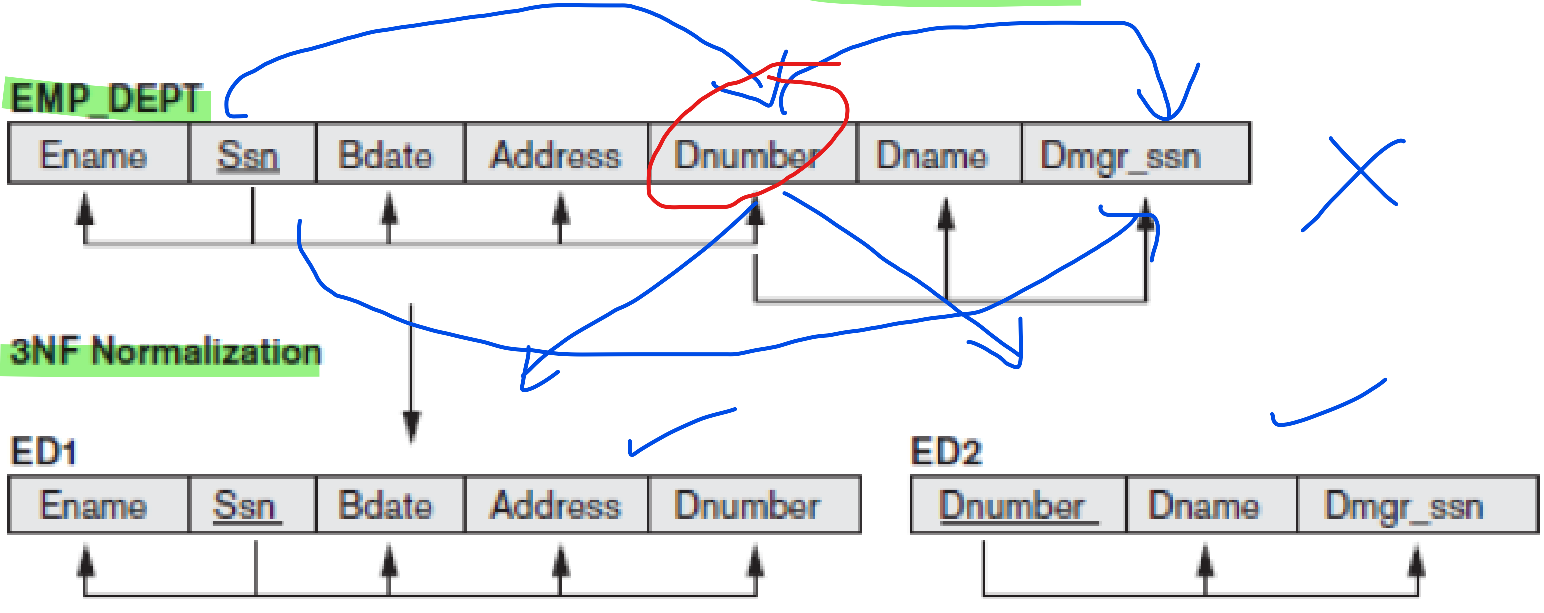
Figure 15.11- Normalizing into 2NF and 3NF.
(a) Normalizing EMP_PROJ into 2NF relations.

Third Normal Form (3NF)

Definition: A relation schema *R* is in 3NF if it satisfies 2NF *and* no nonprime attribute of *R* is transitively dependent on the primary key.

Third normal form (3NF) is based on the concept of transitive dependency. A functional dependency *X*→*Y* in a relation schema *R* is a **transitive dependency** if there exists a set of attributes *Z* in *R* such that *X*→*Z* and *Z*→*Y* hold.

Ex: The dependency Ssn→Dmgr_ssn is transitive in EMP_DEPT, because of the dependencies Ssn → Dnumber and Dnumber → Dmgr_ssn.



EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

3NF Normalization

ED1

Ename	Ssn	Bdate	Address	Dnumber

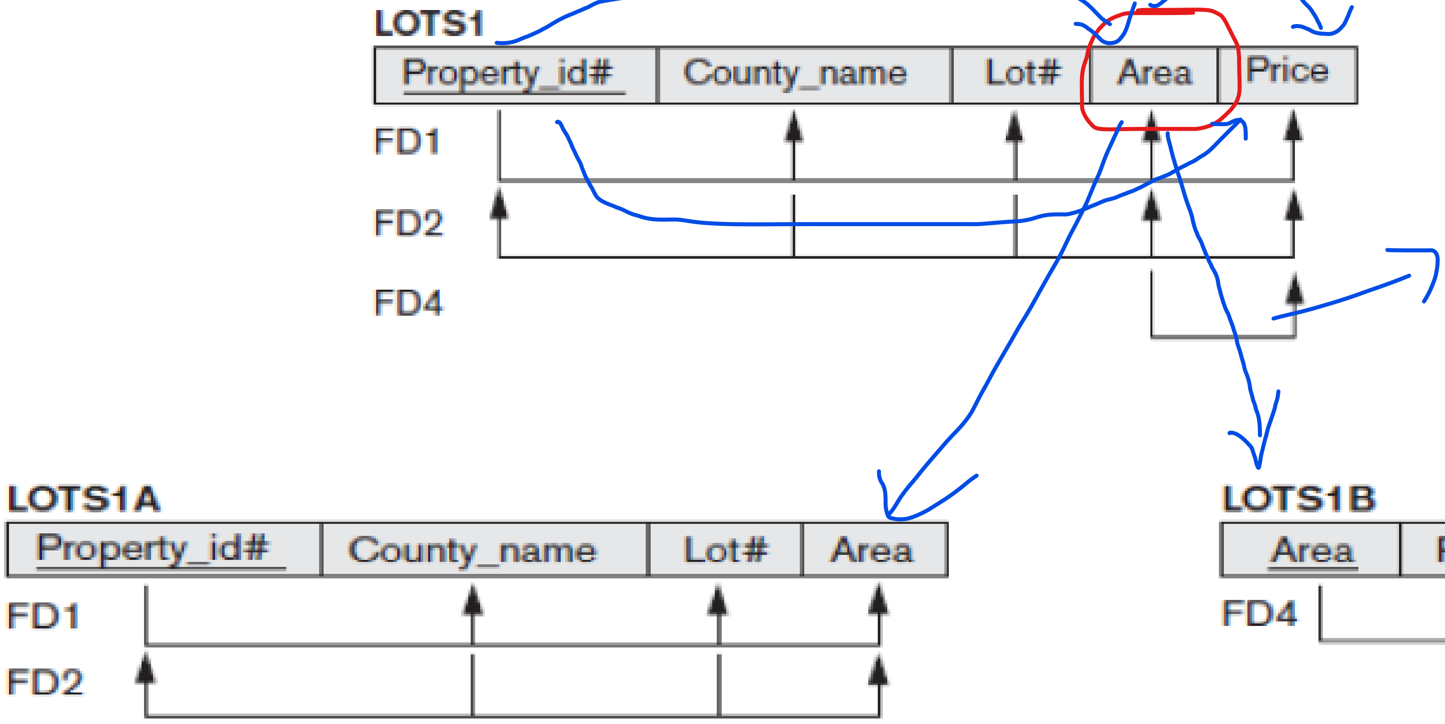
ED2

Dnumber	Dname	Dmgr_ssn

Normalizing EMP_DEPT into 3NF relations.

FD4 in LOTS1 violates 3NF because Area is not a superkey and Price is not a prime attribute in LOTS1.

To normalize LOTS1 into 3NF, we decompose it into the relation schemas LOTS1A and LOTS1B shown in Figure 15.12(c).We construct LOTS1A by removing the attribute Price that violates 3NF from LOTS1 and placing it with Area (the lefthand side of FD4 that causes the transitive dependency) into another relation LOTS1B. Both LOTS1A and LOTS1B are in 3NF.



not fully FD,
are is not
prime attribute

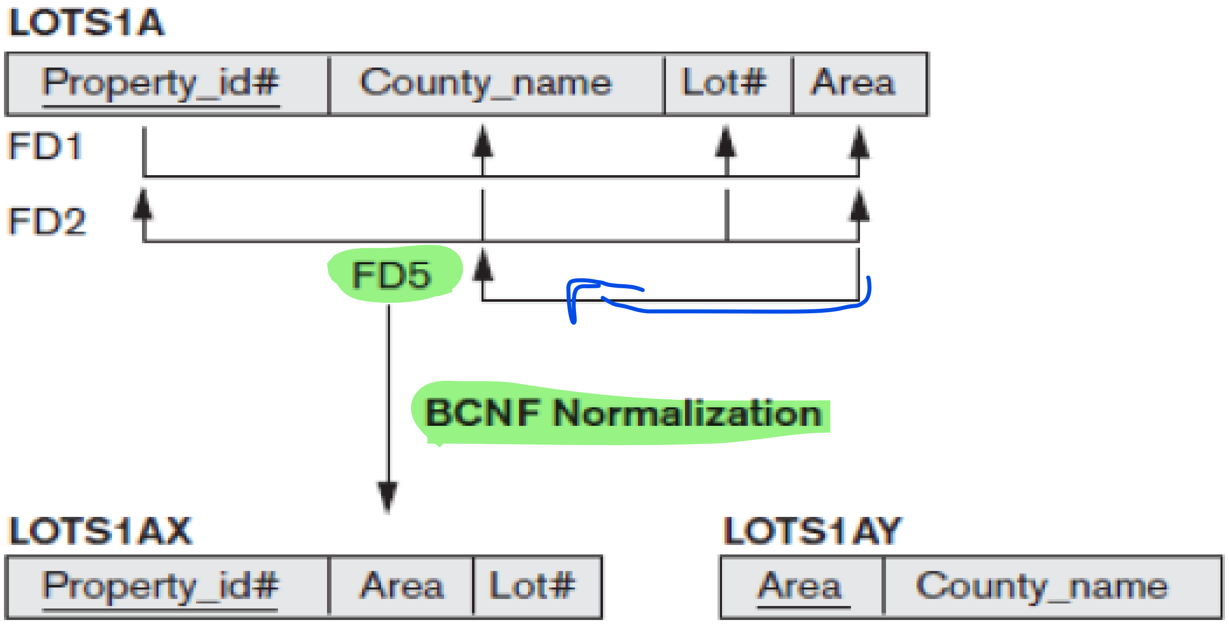
Figure 15.12(c). Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B.

Boyce-Codd Normal Form.

Definition. A relation schema *R* is in BCNF if whenever a *nontrivial* functional dependency *X*→*A* holds in *R*, then *X* is a superkey of *R*.

The BCNF is based on the concept non trivial dependency. If an FD X → Y holds, where Y is not a subset of X, then it is called a non-**trivial** FD.

FD5 violates BCNF in LOTS1A because AREA is not a superkey of LOTS1A. decompose LOTS1A into two BCNF relations LOTS1AX and LOTS1AY, shown in Figure 15.13(a).



LOTS1A

Property_id#	County_name	Lot#	Area

FD1

FD2

FD5

BCNF Normalization

LOTS1AX

Property_id#	Area	Lot#

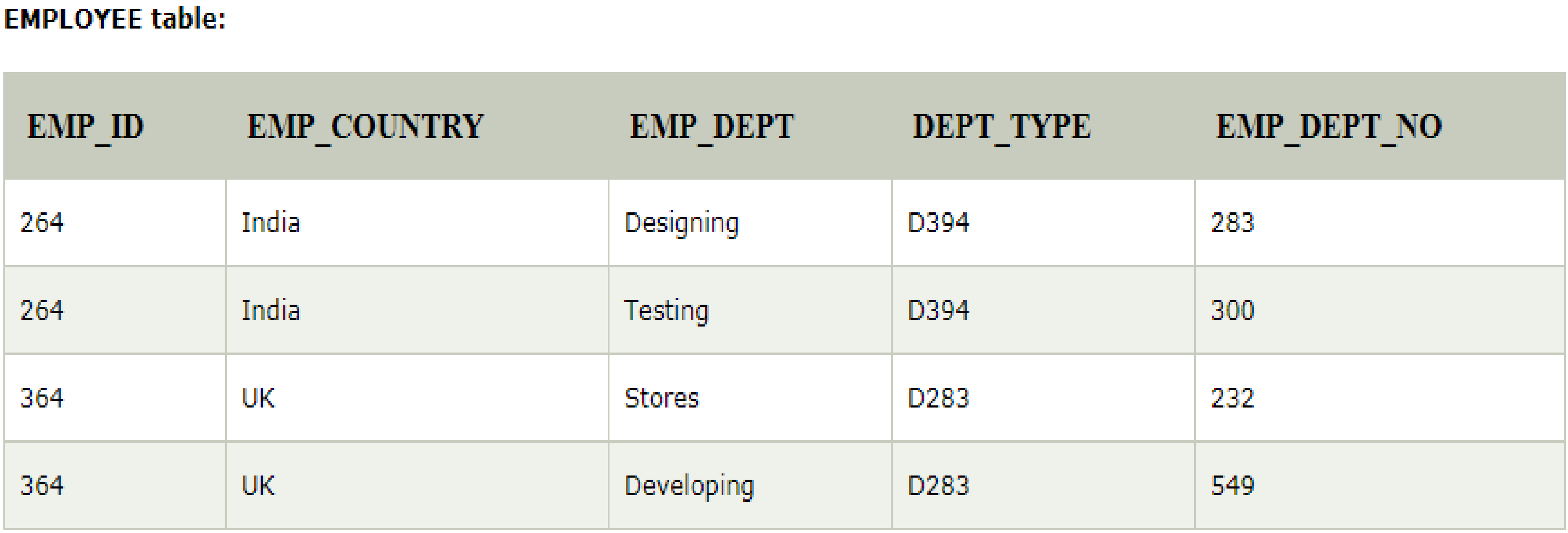
LOTS1AY

Area	County_name

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

Example: 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

```
EMP_ID  →  EMP_COUNTRY
EMP_DEPT  →  {DEPT_TYPE, EMP_DEPT_NO}
```

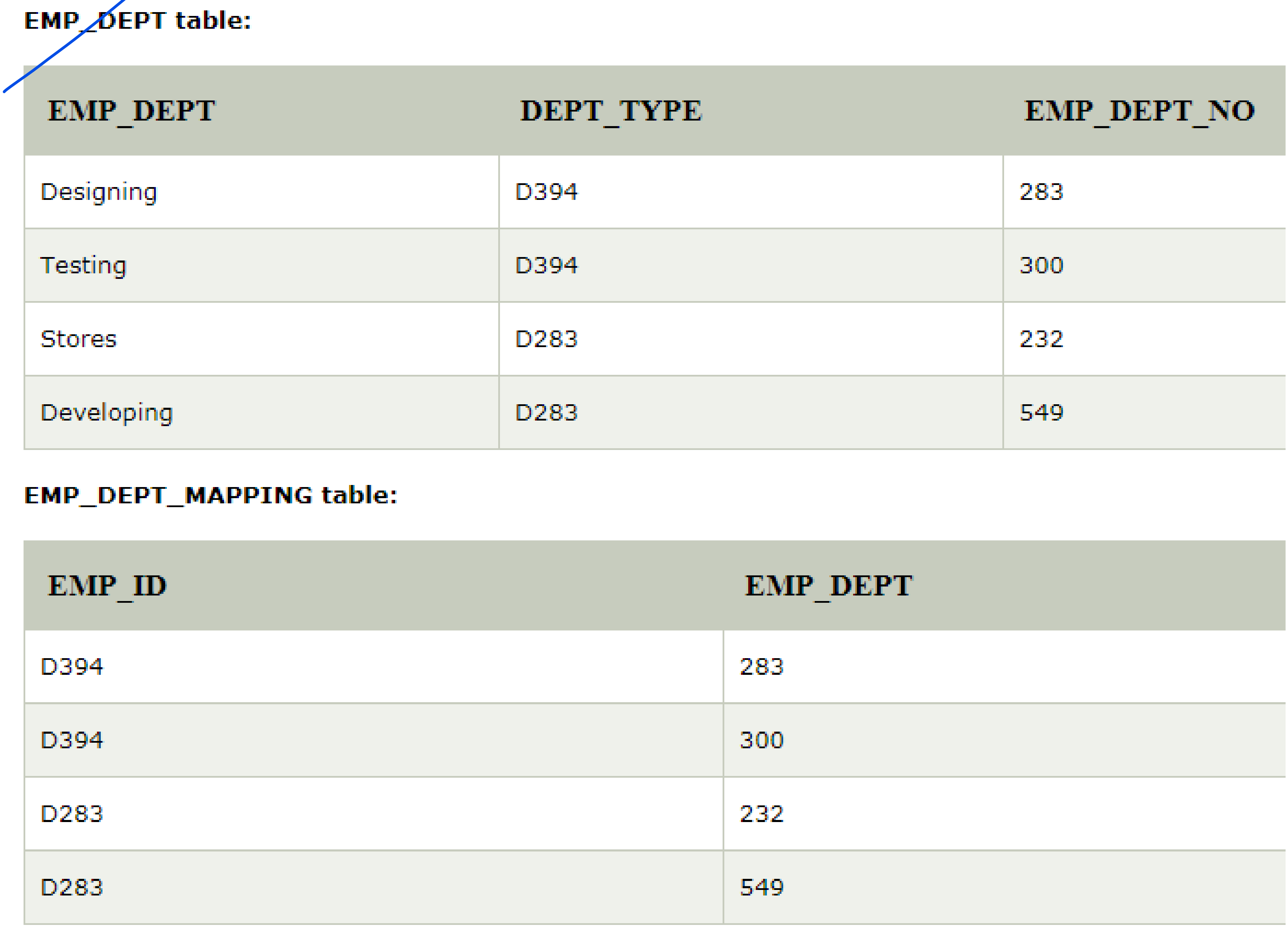
Candidate key: {EMP-ID, EMP-DEPT}

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

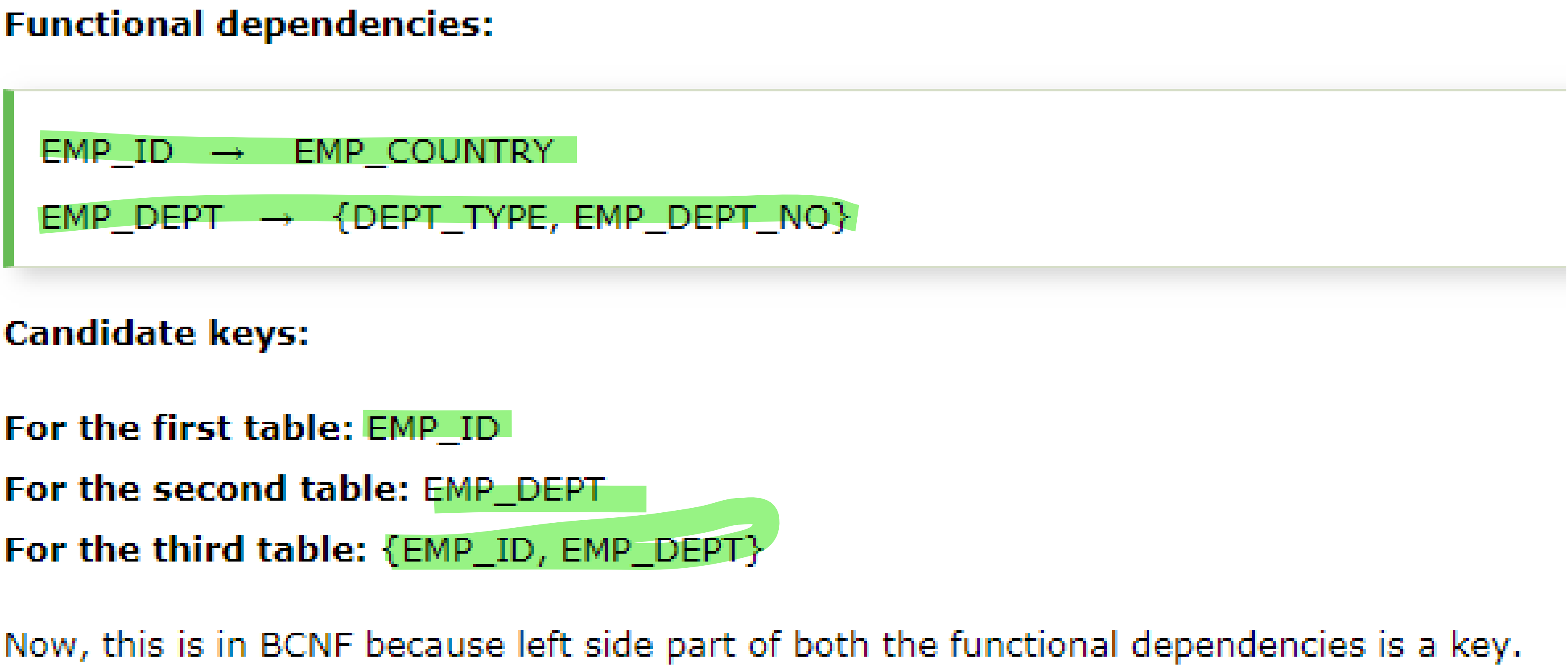


EMP_DEPT table:

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}

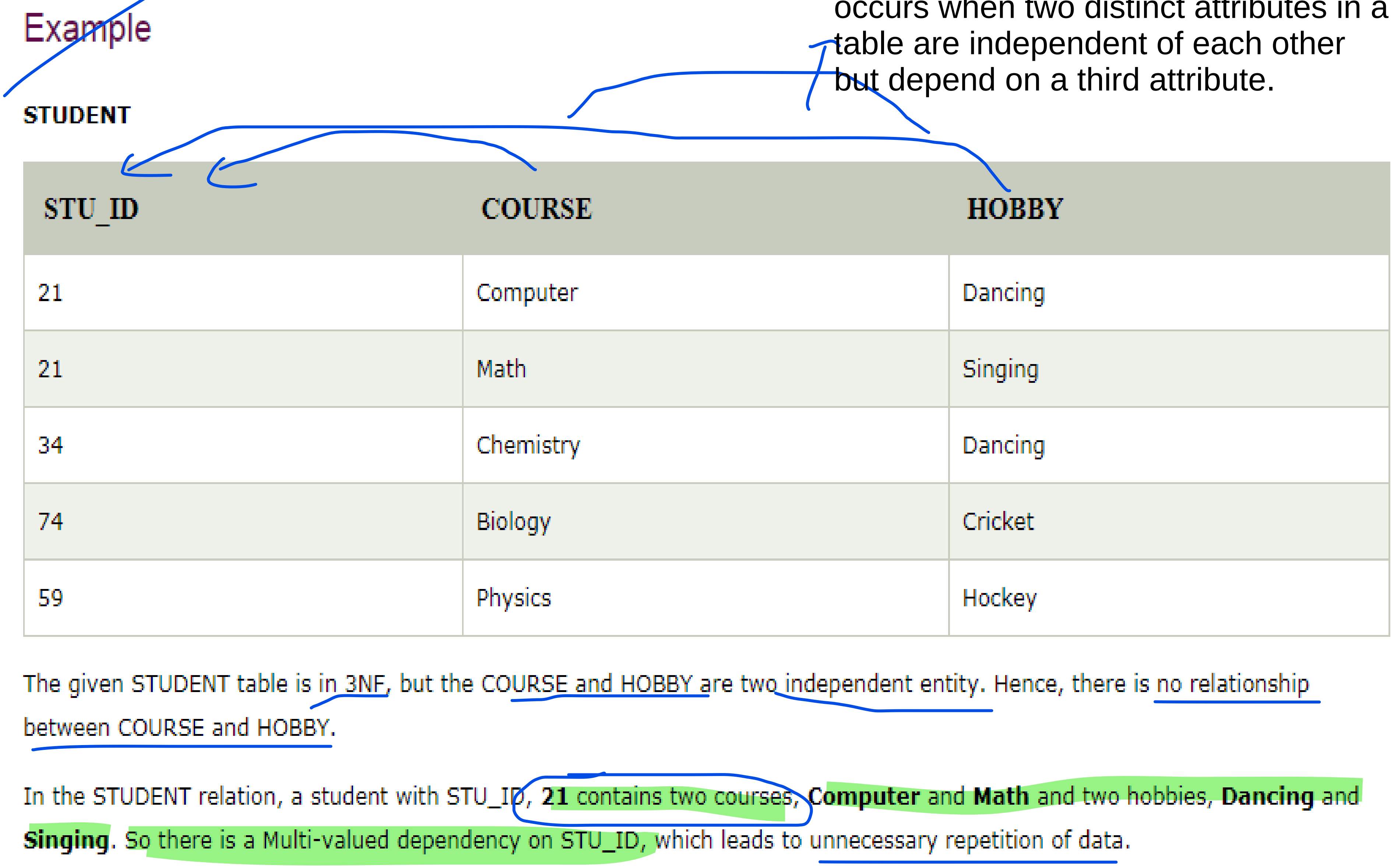
Now, this is in BCNF because left side part of both the functional dependencies is a key.

Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- 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.

Multi-valued dependency occurs when two distinct attributes in a table are independent of each other but depend on a third attribute.

occurs when two distinct attributes in a table are independent of each other but depend on a third attribute.



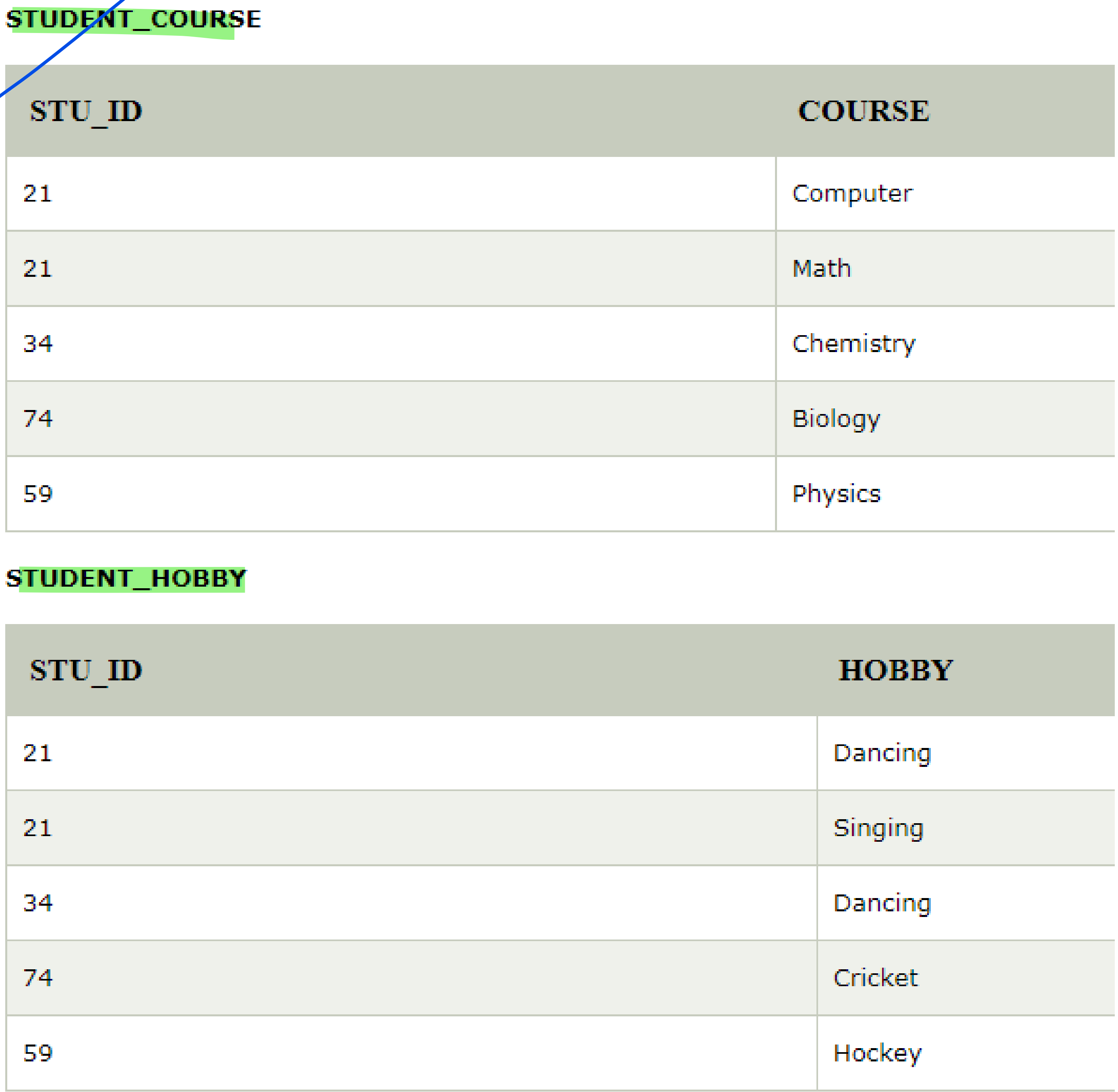
Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.



STUDENT_COURSE

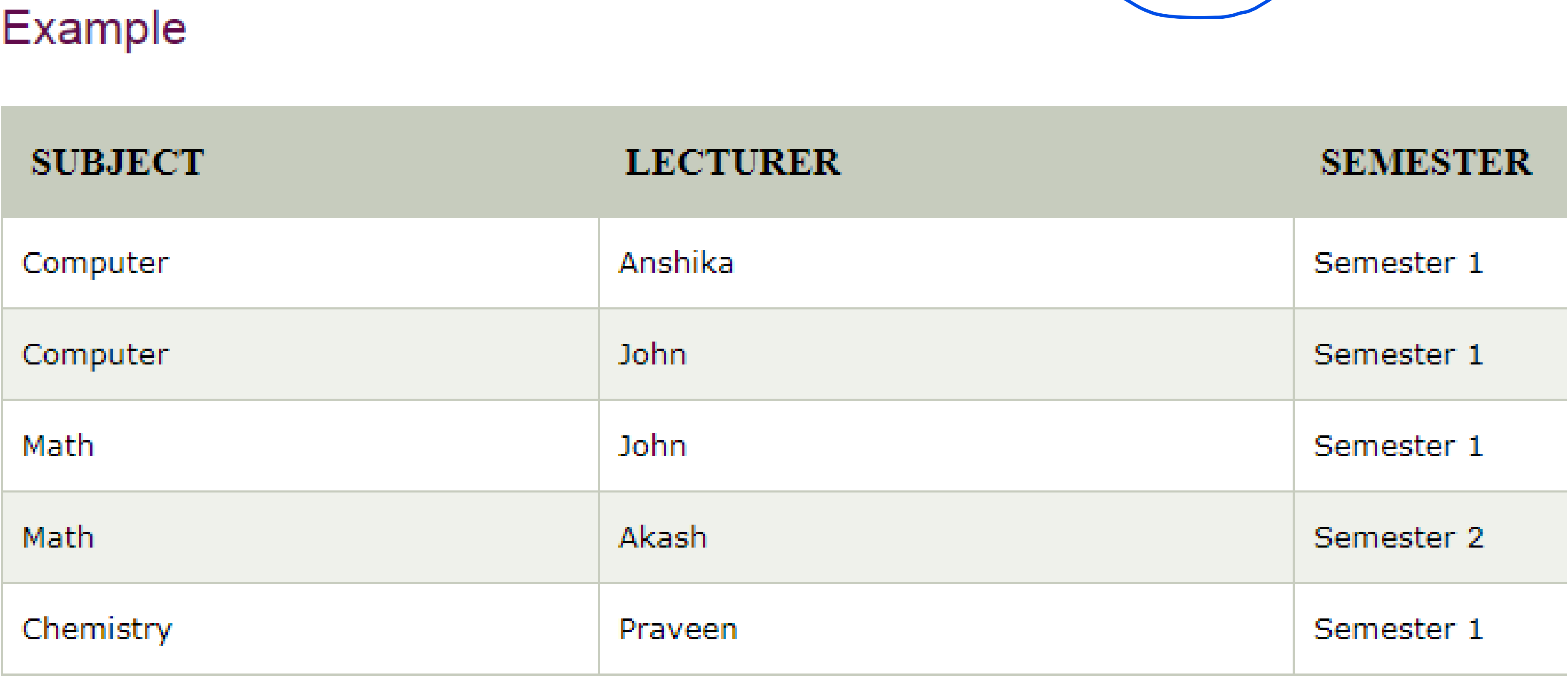
STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Fifth normal form (5NF)

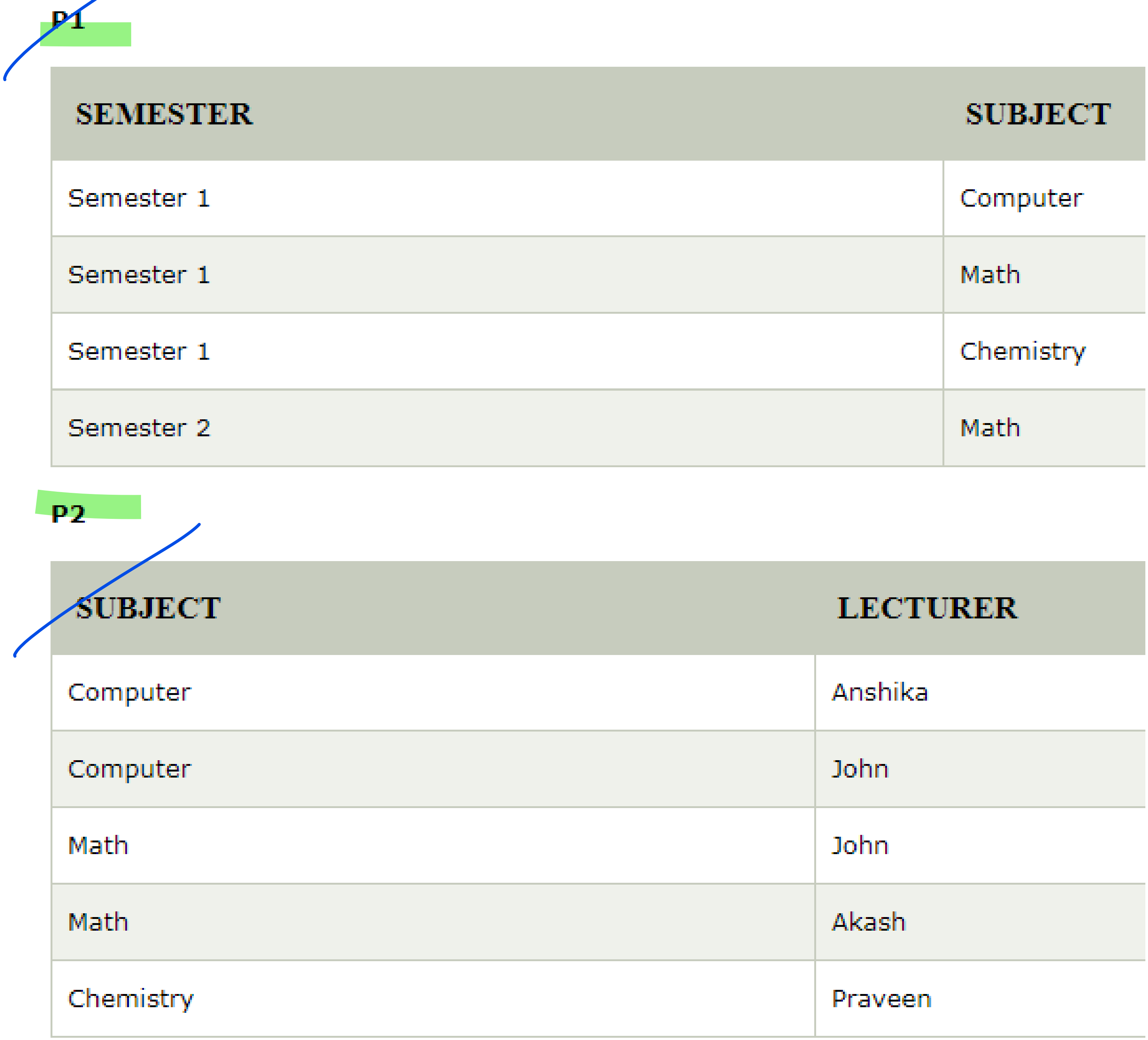
- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).



Example

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

- In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.
- Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.
- So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:



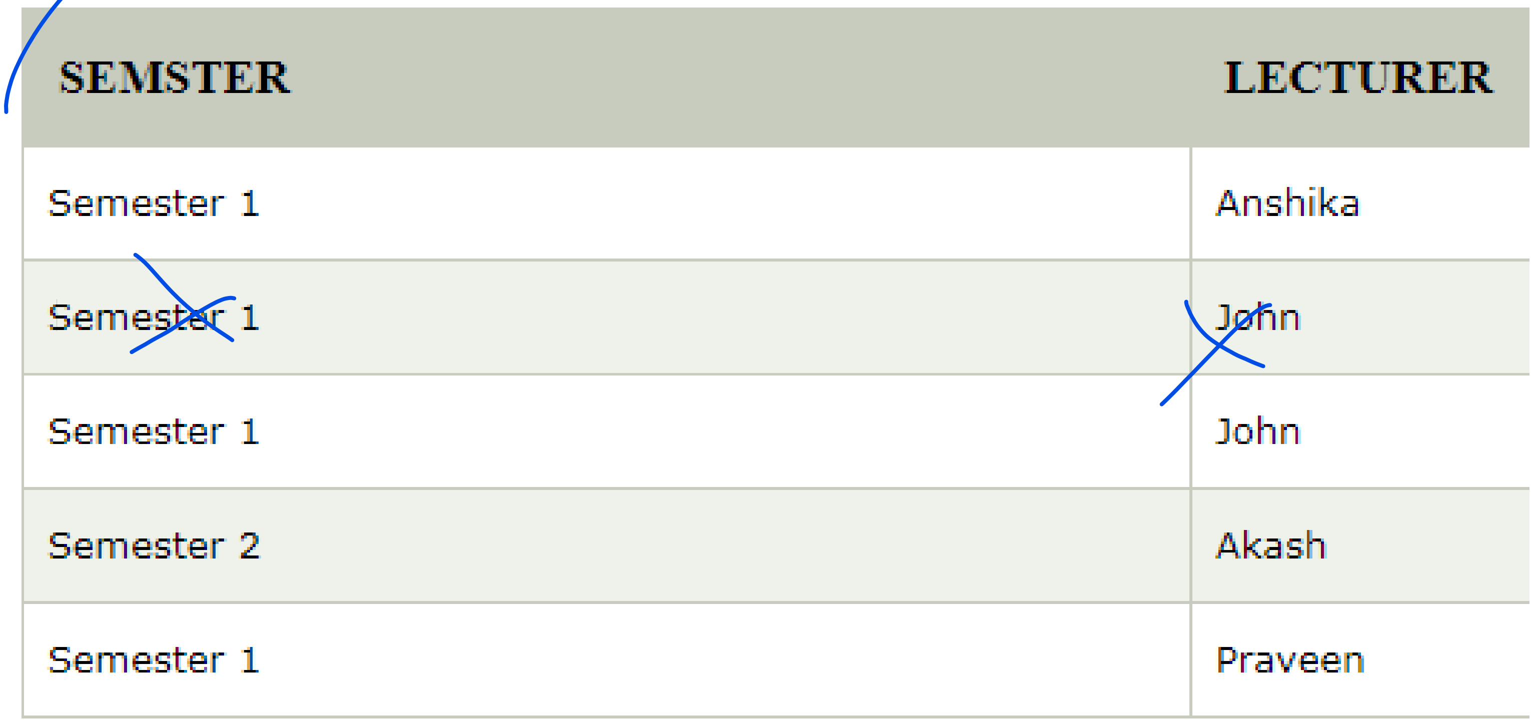
P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3



SEMSTER	LECTURER
Semester 1	Anshika
~~Semester 1~~	~~John~~
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

Normalization Algorithms: Inference Rules, Equivalence, and Minimal cover,
Properties of Relational Decompositions,
Algorithms for Relational Database Schema Design.

Inference Rules for Functional Dependencies

- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.

The Functional dependency has 6 types of inference rule:

1. Reflexive Rule (IR_1)
2. Augmentation Rule (IR_2)
3. Transitive Rule (IR_3)
4. Union Rule (IR_4)
5. Decomposition Rule (IR_5)
6. Pseudo transitive Rule (IR_6)

R A T U D P

1. Reflexive Rule (IR_1)

In the reflexive rule, if Y is a subset of X, then X determines Y. If X ⊇ Y then X → Y

Ex: **{fname, lname} → {fname}**

2. Augmentation Rule (IR_2)

The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z. If X → Y then XZ → YZ

Ex: **If {SSN} → {fname} then: {SSN, DName} → {fname, DName}**

3. Transitive Rule (IR_3)

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

If X → Y and Y → Z then X → Z

Ex: **If: {SSN} → {DNO}; {DNO} → {DName}**

Then: {SSN} → {DName}

4. Union Rule (IR_4)

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

Proof:

1. $X \rightarrow Y$ (given)
2. $X \rightarrow Z$ (given)
3. $X \rightarrow XY$ (using IR_2 on 1 by augmentation with X. Where $XX = X$)
4. $XY \rightarrow YZ$ (using IR_2 on 2 by augmentation with Y)
5. $X \rightarrow YZ$ (using IR_3 on 3 and 4)

5. Decomposition Rule (IR_5)

Decomposition rule is also known as project rule. It is the reverse of union rule.

This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Ex: If $\{SSN\} \rightarrow \{fname, DNO\}$

Then $\{SSN\} \rightarrow \{fname\}$

$\{SSN\} \rightarrow \{ DNO\}$

Proof:

1. $X \rightarrow YZ$ (given)
2. $YZ \rightarrow Y$ (using IR_1 Rule)
3. $X \rightarrow Y$ (using IR_3 on 1 and 2)

6. Pseudo transitive Rule (IR_6)

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$

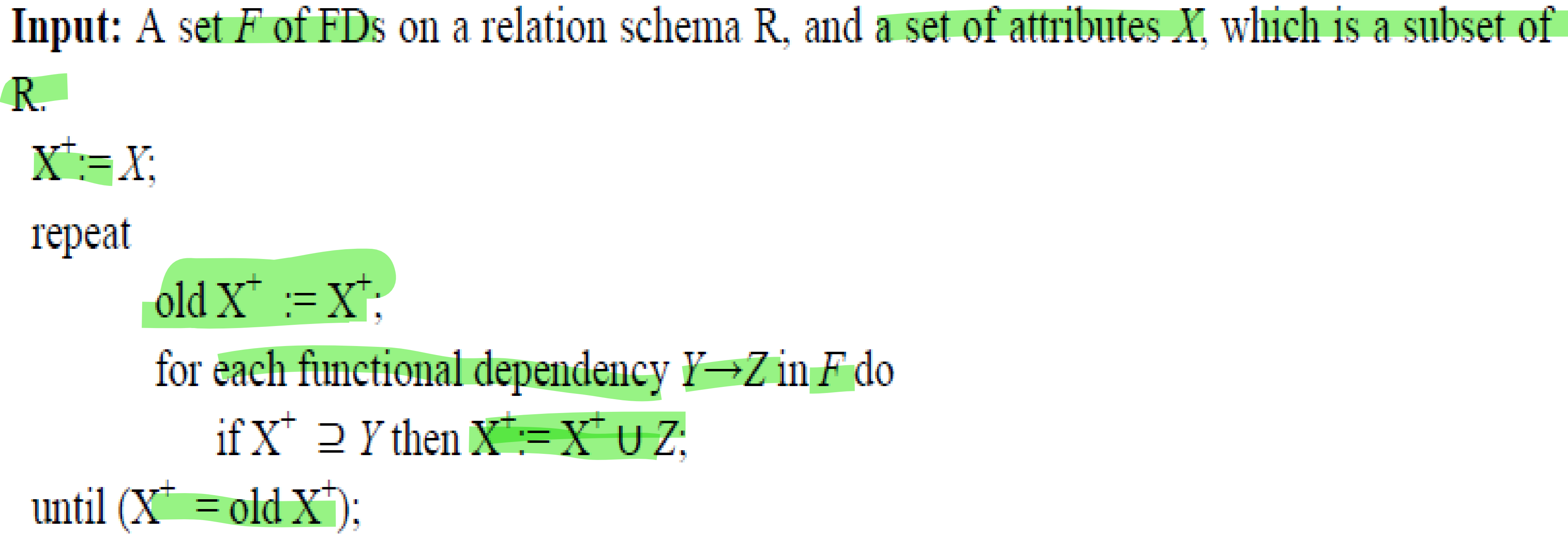
Proof:

1. $X \rightarrow Y$ (given)
2. $WY \rightarrow Z$ (given)
3. $WX \rightarrow WY$ (using IR_2 on 1 by augmenting with W)
4. $WX \rightarrow Z$ (using IR_3 on 3 and 2)

A systematic way to determine these additional functional dependencies is first to determine each set of attributes *X* that appears as a left-hand side of some functional dependency in *F* and then to determine the set of *all attributes* that are dependent on *X*.

Definition. For each such set of attributes *X*, we determine the set *X*+ of attributes that are functionally determined by *X* based on *F*; *X*+ is called the **closure of X under F.**

Algorithm. Determining *X*+, the Closure of *X* under *F*



Input: A set F of FDs on a relation schema R, and a set of attributes X, which is a subset of R.

```
$X^+ := X$;
repeat
        old $X^+ := X^+$;
        for each functional dependency $Y \to Z$ in $F$ do
            if $X^+ \supseteq Y$ then $X^+ := X^+ \cup Z$;
until ($X^+ = \text{old } X^+$);
```

Example 1, consider the relation schema EMP_PROJ from the semantics of the attributes, the following set *F* of functional dependencies are identified.

F = {

Ssn → Ename,

Pnumber → {Pname, Plocation},

{Ssn, Pnumber} → Hours

}

Using above algorithm, we calculate the following closure sets with respect to

F: {Ssn} + = {Ssn, Ename}

{Pnumber}+ = {Pnumber, Pname, Plocation}

{Ssn, Pnumber}+ = {Ssn, Pnumber, Ename, Pname, Plocation, Hours}

Example 2. Consider the following relation schema.

CLASS(Classid,Course#,Instr_name,Credit_hrs,Text,Publisher,Classroom , Capacity).

Let F, the set of functional dependencies for the above relation

FD1: Classid → Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity;

FD2: Course# → Credit_hrs;

FD3: {Course#, Instr_name} → Text, Classroom;

FD4: Text → Publisher

FD5: Classroom → Capacity

Using the inference rules about the FDs and applying the definition of closure, we can define the following closures:

{Classid}$^{+}$ ={Classid,Course#,Instr_name, Credit_hrs, Text, Publisher,Classroom, Capacity}

{Course#}$^{+}$ = {Course#, Credit_hrs}

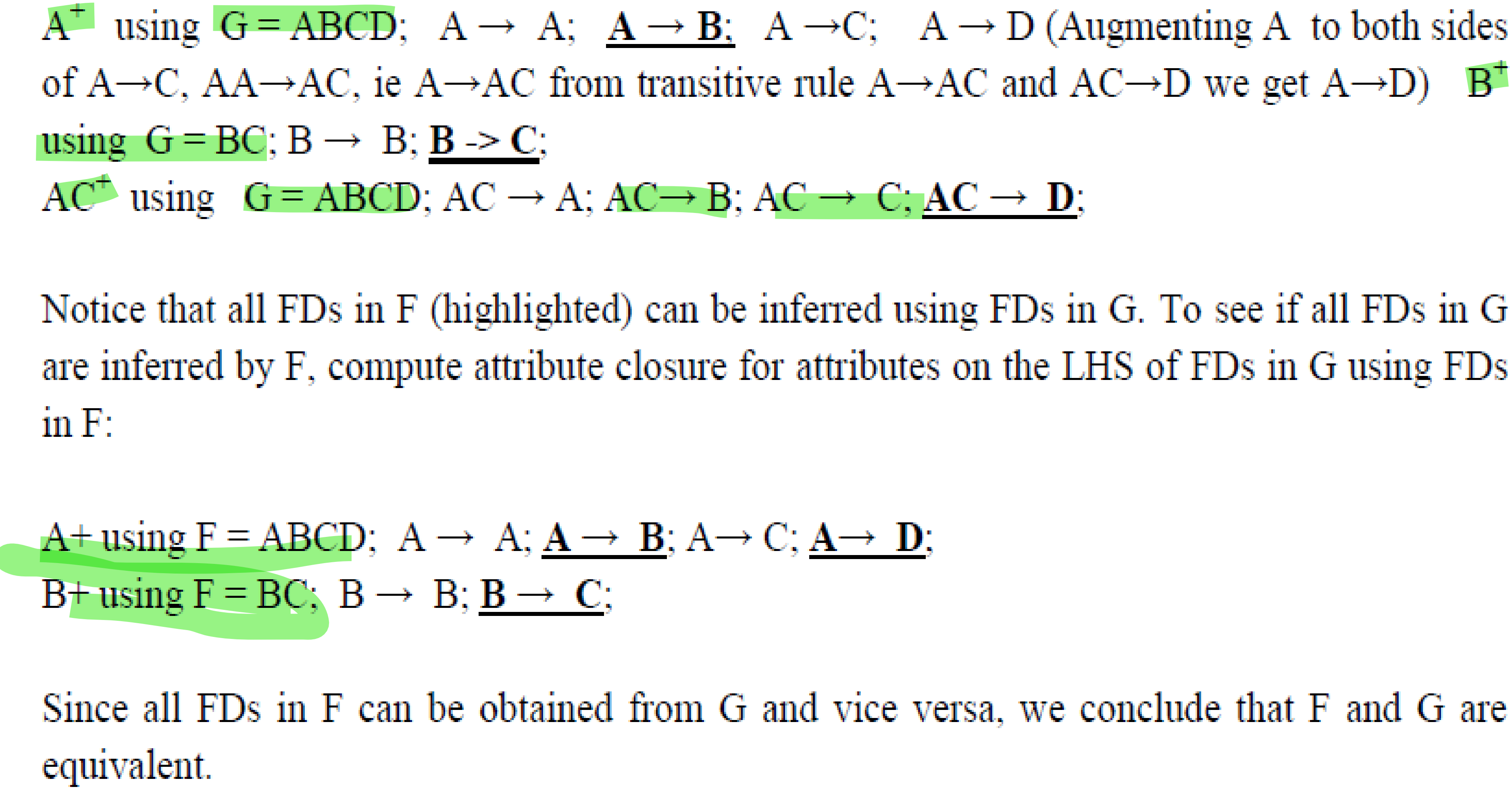
{Course#, Instr_name}$^{+}$ = {Course#, Credit_hrs, Text, Publisher, Classroom, Capacity}

Equivalence of Sets of Functional Dependencies:

- A set of functional dependencies F is said to cover another set of functional dependencies E if every FD in E is also in F+; that is, if every dependency in E can be inferred from F; alternatively, we can say that E is covered by F.
- Two sets of functional dependencies E and F are equivalent if E+ = F+. Therefore, equivalence means that every FD in E can be inferred from F, and every FD in F can be inferred from E; that is, E is equivalent to F if both the conditions E covers F and F covers E hold.
- We can determine whether F covers E by calculating X+ with respect to F for each FD X → Y in E, and then checking whether this X+ includes the attributes in Y. If this is the case for every FD in E, then F covers E.

[redacted]

Consider two sets of FDs, F and G, **F = {A → B, B → C, AC→D}** and **G = {A→B, B→C, A→D}** Are F and G equivalent? Take the attributes from the LHS of FDs in F and compute attribute closure for each using FDs in G:



A^+ using G = ABCD; A → A; **A → B**; A →C; A → D (Augmenting A to both sides of A→C, AA→AC, ie A→AC from transitive rule A→AC and AC→D we get A→D) B^+ using G = BC; B → B; **B -> C**;

AC^+ using G = ABCD; AC → A; AC→ B; AC → C; **AC → D**;

Notice that all FDs in F (highlighted) can be inferred using FDs in G. To see if all FDs in G are inferred by F, compute attribute closure for attributes on the LHS of FDs in G using FDs in F:

A+ using F = ABCD; A → A; **A → B**; A→ C; **A→ D**;

B+ using F = BC; B → B; **B → C**;

Since all FDs in F can be obtained from G and vice versa, we conclude that F and G are equivalent.