

Database Management System
Prof. Partha Pratim Das
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture - 19
Relational Database Design (Contd.)

Welcome to module 19 of Database Management Systems; we have been discussing relational database design and this is the fourth part; fourth module in that series.

(Refer Slide Time: 00:38)

PPD

Module Recap

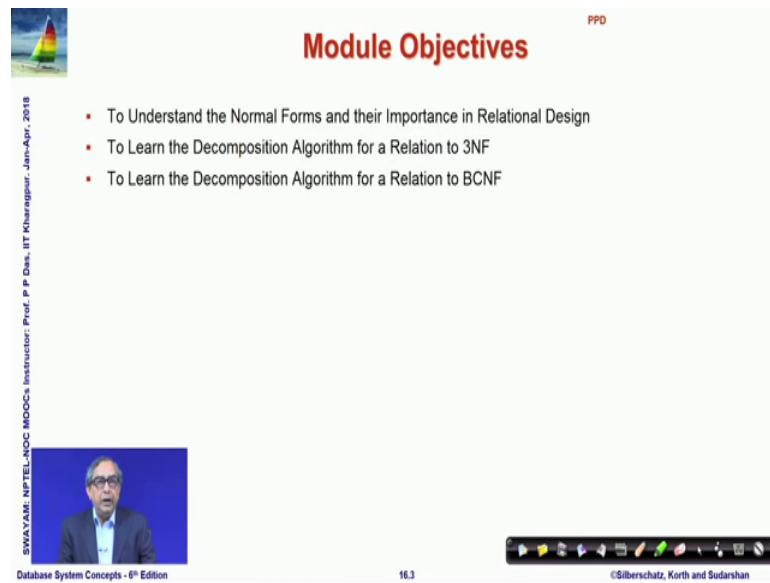
- Algorithms for Functional Dependencies
- Lossless Join Decomposition
- Dependency Preservation

SWAYAM: NPTEL-NOC MOOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 9th Edition 16.2 ©Silberschatz, Korth and Sudarshan

In the last module, we have discussed about algorithms for functional dependencies lossless joint decomposition and dependency preservation. So, based on this foundational algorithms and concepts.

(Refer Slide Time: 00:51)



The slide, titled "Module Objectives", features a small sailboat icon in the top left corner. The main content is a bulleted list of three objectives. On the left side, there is a vertical text string and a small video inset of a man speaking. The bottom of the slide contains a navigation bar with various icons and the slide number "16.3".

PPD

Module Objectives

- To Understand the Normal Forms and their Importance in Relational Design
- To Learn the Decomposition Algorithm for a Relation to 3NF
- To Learn the Decomposition Algorithm for a Relation to BCNF

SWAYAM: NPTEL-NOC MOOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

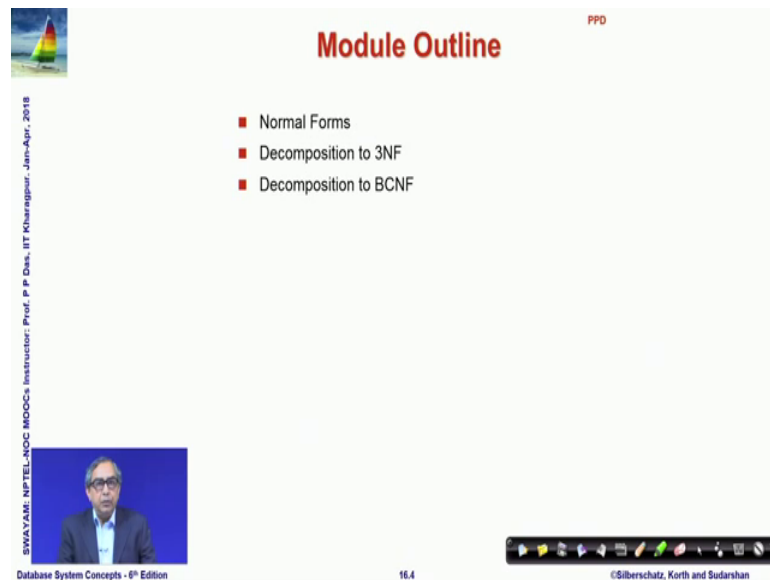
Database System Concepts - 8th Edition

16.3

©Silberschatz, Korth and Sudarshan

We will in today's module get into understanding the core design aspects of relational databases; that is a normal forms and how important they are in terms of the relational design. We would specifically learn about decomposition of a relational schema into the third normal form and into Boyce Codd BCNF form.

(Refer Slide Time: 01:20)



The slide, titled "Module Outline", features a small sailboat icon in the top left corner. The main content is a bulleted list of three topics. On the left side, there is a vertical text string and a small video inset of a man speaking. The bottom of the slide contains a navigation bar with various icons and the slide number "16.4".

PPD

Module Outline

- Normal Forms
- Decomposition to 3NF
- Decomposition to BCNF

SWAYAM: NPTEL-NOC MOOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

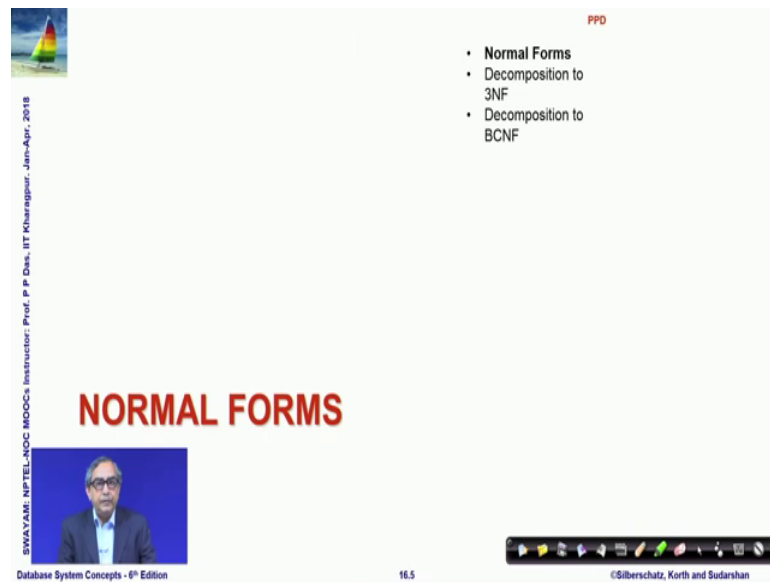
Database System Concepts - 8th Edition

16.4

©Silberschatz, Korth and Sudarshan

So, our topics will be the three normal forms decomposition of 3 NF and into BCNF.

(Refer Slide Time: 01:27)



PPD

- Normal Forms
- Decomposition to 3NF
- Decomposition to BCNF

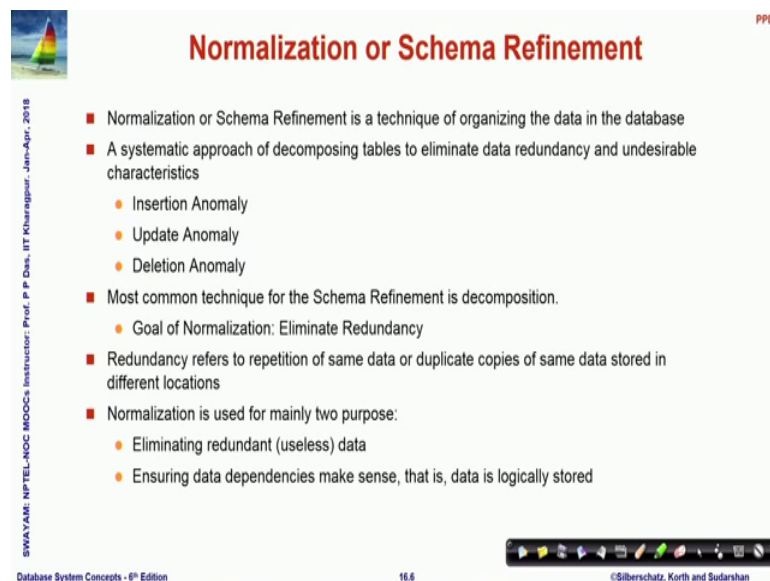
NORMAL FORMS

SWAYAM: NPTEL-NOC MOCs Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 8th Edition 16.5 ©Silberschatz, Korth and Sudarshan

So, starting with the normal forms.

(Refer Slide Time: 01:29)



PPD

Normalization or Schema Refinement

- Normalization or Schema Refinement is a technique of organizing the data in the database
- A systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics
 - Insertion Anomaly
 - Update Anomaly
 - Deletion Anomaly
- Most common technique for the Schema Refinement is decomposition.
 - Goal of Normalization: Eliminate Redundancy
- Redundancy refers to repetition of same data or duplicate copies of same data stored in different locations
- Normalization is used for mainly two purpose:
 - Eliminating redundant (useless) data
 - Ensuring data dependencies make sense, that is, data is logically stored

SWAYAM: NPTEL-NOC MOCs Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 8th Edition 16.6 ©Silberschatz, Korth and Sudarshan

So, normal forms or normalization of a schema is a technique of refinement to organize the data in the database. So, the question naturally arises as to why do we need to do this refinement after we have done a design based on possibly the E-R diagram based approach that we had talked of we had identified the entities and we had identified the attributes for the entities their relationships; then why do we need to normalize?

The answer to this question lies in the fact that a design for a relational schema may give rise to a variety of anomalies in terms of the data. These are typically three anomalies which concerns us most the insertion, the update and the deletion anomaly. So, the anomaly is happen when there is redundancy in the data in terms of the schema. And whether there will be redundant data and how much what kind of redundant data would be there depends on the design of the database schema depends on the design of the normal form that we are using for it.

But if we have redundancy then there is potential for anomalies and therefore, we want to reduce the redundancy and get rid of this anomaly.

(Refer Slide Time: 03:00)

Anomalies

- Update Anomaly:** Employee 519 is shown as having different addresses on different records
- Insertion Anomaly:** Until the new faculty member, Dr. Newsome, is assigned to teach at least one course, his details cannot be recorded
- Deletion Anomaly:** All information about Dr. Giddens is lost if he temporarily ceases to be assigned to any courses.

Employees' Skills

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

Resolution: Decompose the Schema

- Update: (ID, Address), (ID, Skill)
- Insert: (ID, Name, Hire Date), (ID, Code)
- Delete: (ID, Name, Hire Date), (ID, Code)

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201
424	Dr. Newsome	29-Mar-2007	?

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201

Database System Concepts - 8th Edition 16.7 ©Silberschatz, Korth and Sudarshan

So, we will quickly take a look into the anomalies that are that we are talking of first one is called an update anomaly. So, we are showing you a snapshot of an instance of a database which has three attributes and you can look at the row having two entries the last two rows for employee code 519 and there are two different addresses in these two different rows. So, if we know that the employee will have a unique address or in other words if employee ID would determine the employee address functionally determine that employee address then this situation is not possible.

So, but when we try to update then it is for example, the employees address has changed. And while making that change this change will need to be incorporated in all the records having the same ID. And if because of some coding error or something we miss out to

update any of the address fields then we will have a difficulty and that difficulty is having inconsistent address data as in this case..

So, this is known as update anomaly similarly I could have an insertion anomaly which I am illustrating here in terms of another database schema which has four attributes. And we have faculty ID name the hiring date and the course name naturally given the faculty ID the faculty name and hire date should be unique. Now suppose a new faculty joins and as soon as the faculty joins he or she may not have an assigned course.

So if we want to enter that record here we will not be able to do that because we do not have any value for the course code. So, either we use a null value or we cannot actually enter this value; this kind of situation is known as a insertion anomaly. Similarly I could have a deletion anomaly in the in the same table we are showing that in the table the first highlighted row; the for faculty ID 389 if that faculty stops taking any course for the time being..

So, the association between 389 and the corresponding course code will be removed and once you remove that you remove this whole record in the process you actually lose the whole of the faculty information the ID, name and hire date. So, these are difficulties in these relational schemas and that lead to a whole lot of problems.

So, the resolution for this lie in terms of decomposing the schema that instead of having one relation, I will decompose this set of attributes into multiple different relations. So, for example, the update anomaly can be removed if we have two different tables; one that maintains ID with address and one that maintains ID with skill. So, in that case what will happen if the for every ID the address will not be repeated..

So, if the address is updated; it will be updated only at one place and it will not feature in the other table. Similarly to avoid insert or delete anomaly the other table schema can be split into ID name and hire date as one table and ID and code rows code as another table. And you can you can easily understand that if this is split in this way then you cannot have an insert anomaly because you can insert a new faculty without assigning a course to him because that will feature in as a separate record in a different table similarly in the same way the deletion anomaly also disappears.

So, these anomalies are resultant of the redundant data that we are having and can be removed by taking care of the process of decomposition.

(Refer Slide Time: 07:03)

Desirable Properties of Decomposition

- Lossless Join Decomposition Property
 - It should be possible to reconstruct the original table
- Dependency Preserving Property
 - No functional dependency (or other constraints) should get violated

ORIGINAL SCHEMA

1 NF

2 NF

3 NF

SWAYAM: NPTEL-NOC MOCs Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 9th Edition

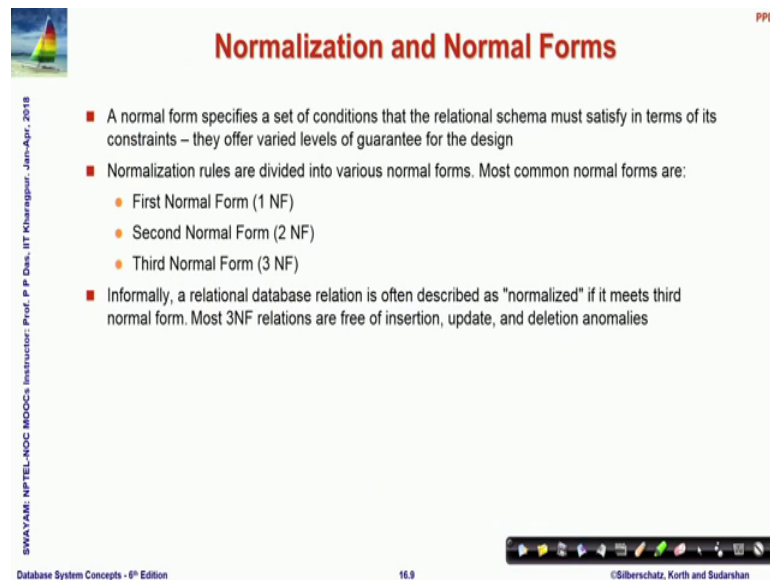
16.8

©Silberschatz, Korth and Sudarshan

Now, when we decompose then we would desire certain properties to be hold held and we talked about this loosely earlier as well. We would require the lossless join decomposition property that it should be possible to take any instance of the two or more decomposed relations and join them by natural join using common set of attributes and get back the original instance of the relation if that does not happen then the relationship is lossy we have discussed it at length in the last module. At the same time we would want that all functional dependencies that hold must be; can must be testable in the decomposed set of relation.

So, all functional dependencies when they are projected in terms of the decomposed set of relations; they must be testable within them. So, that to test for a dependency I do not need to carry out a join this is a point we discussed in the last module as well. So, based on that once you start with the original schema, you can check for what are the different possibilities or sources of redundancy define constraints based on that and step by step; you could convert a schema into a one normal form have more constraints put onto it convert it into two normal form have further constraints decompose it into third normal form and so, on.

(Refer Slide Time: 08:34)



PPD

Normalization and Normal Forms

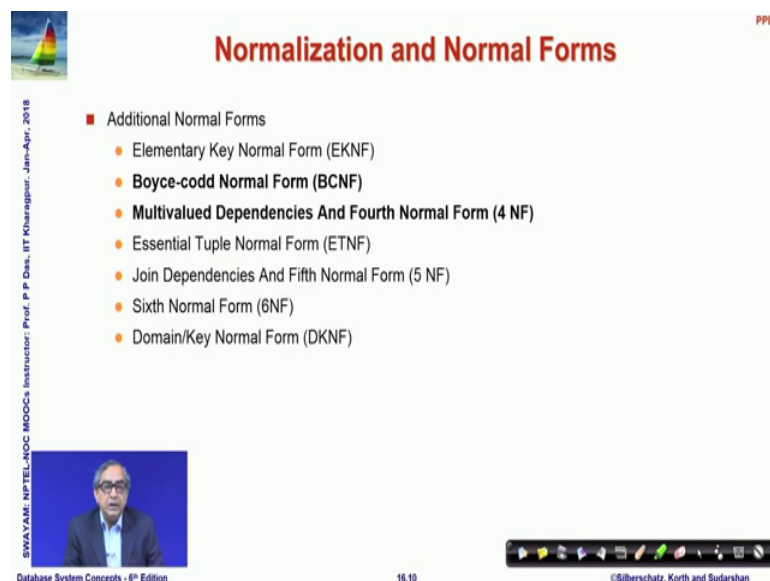
- A normal form specifies a set of conditions that the relational schema must satisfy in terms of its constraints – they offer varied levels of guarantee for the design
- Normalization rules are divided into various normal forms. Most common normal forms are:
 - First Normal Form (1 NF)
 - Second Normal Form (2 NF)
 - Third Normal Form (3 NF)
- Informally, a relational database relation is often described as "normalized" if it meets third normal form. Most 3NF relations are free of insertion, update, and deletion anomalies

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargapur, Jan-Apr, 2018

Database System Concepts - 6th Edition 16.9 ©Silberschatz, Korth and Sudarshan

So, normalization is a process through which we do this kind of decomposition and make sure that once a relational schema is expressed in terms of a normal form; it satisfies a given set of properties that that normal form should adhere to. And the common normal forms are 1 NF, 2 NF and 3 NF and loosely speaking when we say if a database schema is normalized; we normal usually mean that it is in the 3 NF form a third normal form. And most third number form relations are free of insert, delete or update anomalies. So, that they are a good positive in the design.

(Refer Slide Time: 09:12)



PPD

Normalization and Normal Forms

- Additional Normal Forms
 - Elementary Key Normal Form (EKNF)
 - Boyce-codd Normal Form (BCNF)
 - Multivalued Dependencies And Fourth Normal Form (4 NF)
 - Essential Tuple Normal Form (ETNF)
 - Join Dependencies And Fifth Normal Form (5 NF)
 - Sixth Normal Form (6NF)
 - Domain/Key Normal Form (DKNF)

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargapur, Jan-Apr, 2018

Database System Concepts - 6th Edition 16.10 ©Silberschatz, Korth and Sudarshan

Of course, these are not the only normal forms as you can see there is a whole lot of lists of variety of normal forms; we will not study all of them we will study further in the next module the other two highlighted ones.

(Refer Slide Time: 09:26)

First Normal Form (1 NF)

- A relation is in first Normal Form if and only if all underlying domains contain atomic values only
- In other words, a relation doesn't have multivalued attributes (MVA)
- Example:
 - *STUDENT*(Sid, Sname, Cname)

Students		
SID	Sname	Cname
S1	A	C,C++
S2	B	C++, DB
S3	A	DB
SID : Primary Key		

MVA exists → Not in 1NF

Students		
SID	Sname	Cname
S1	A	C
S1	A	C++
S2	B	C++
S2	B	DB
S3	A	DB
SID : Primary Key		

No MVA → In 1NF

Source: <http://www.edugrabs.com/normal-forms/1nf/>

Database System Concepts - 6th Edition 16.11 ©Silberschatz, Korth and Sudarshan

But first let us get started with the first normal form which we had talked about earlier as well; that first normal form is one where the multivalued attributes are not allowed. So, if you think about a relationship where you have a student relationship between student the her name and the courses taken by the student then since the students take multiple courses; the C name in this case can take multiple values. So, we do not allow that we expand them into different rows and that once we have done that we say that relation is in the one normal form.

(Refer Slide Time: 10:02)

PPD

First Normal Form (1 NF): Possible Redundancy

■ Example:

- Supplier(SID, Status, City, PID, Qty)

Supplier:

SID	Status	City	PID	Qty
S1	30	Delhi	P1	100
S1	30	Delhi	P2	125
S1	30	Delhi	P3	200
S1	30	Delhi	P4	130
S2	10	Karnal	P1	115
S2	10	Karnal	P2	250
S3	40	Rohtak	P1	245
S4	30	Delhi	P4	300
S4	30	Delhi	P5	315

Key : (SID, PID)

Drawbacks:

- Deletion Anomaly** – If we delete the tuple <S3,40,Rohtak,P1,245>, then we lose the information about S3 that S3 lives in Rohtak.
- Insertion Anomaly** – We cannot insert a Supplier S5 located in Karnal, until S5 supplies at least one part.
- Update Anomaly** – If Supplier S1 moves from Delhi to Kanpur, then it is difficult to update all the tuples containing (S1, Delhi) as SID and City respectively.

Normal Forms are the methods of reducing redundancy. However, Sometimes 1 NF increases redundancy. It does not make any efforts in order to decrease redundancy.

SWAYAM: NPTEL-NOC MOOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018
Source: <http://www.edugrabs.com/normal-forms/1nf/>

Database System Concepts - 9th Edition 16.12 ©Silberschatz, Korth and Sudarshan

But one normal form may give rise to a variety of different redundancies and therefore, anomalies. So, this is another instance; in fact, the earlier instances that you saw all of them were also in one normal form, but they had deletion insertion and update anomaly. So, here is another example where we are illustrating that.

(Refer Slide Time: 10:26)

PPD

First Normal Form (1 NF): Possible Redundancy

■ When LHS is not a Superkey :

- Let $X \rightarrow Y$ is a non trivial FD over R with X is not a superkey of R, then redundancy exist between X and Y attribute set.
- Hence in order to identify the redundancy, we need not to look at the actual data, it can be identified by given functional dependency.
- Example : $X \rightarrow Y$ and X is not a Candidate Key \Rightarrow X can duplicate \Rightarrow corresponding Y value would duplicate also.

X	Y
1	3
1	3
2	3
2	3
4	6

■ When LHS is a Superkey :

- If $X \rightarrow Y$ is a non trivial FD over R with X is a superkey of R, then redundancy does not exist between X and Y attribute set.
- Example : $X \rightarrow Y$ and X is a Candidate Key \Rightarrow X cannot duplicate \Rightarrow corresponding Y value may or may not duplicate.

X	Y
1	4
2	6
3	4

SWAYAM: NPTEL-NOC MOOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018
Source: <http://www.edugrabs.com/normal-forms/1nf/>

Database System Concepts - 9th Edition 16.13 ©Silberschatz, Korth and Sudarshan

So, it is a possible that if I have a functional dependency X determining Y which is nontrivial functional dependency over the set of attributes and X is not a super key; then there exists a redundancy between X and Y attribute set. So, on the left the we have

shown an instance of this relationship on only on the X and Y attributes and you can see since X is not a key; I can have two rows having the value one in X.

And since the value is 1 in X; the value Y will be same for these two rows and we have redundancy of that please all. Please remember that X is not a super key; so, there are other attributes which actually form the super key and therefore, such instances are possible.

Whereas if you look at the right column where the left hand side X is a super key then such instances will not happen.

(Refer Slide Time: 11:22)

Second Normal Form (2NF)

- Relation R is in Second Normal Form (2NF) only iff :
 - R should be in 1NF and
 - R should not contain any *Partial Dependency*

Partial Dependency:

Let R be a relational Schema and X, Y, A be the attribute sets over R where
 X : Any Candidate Key, Y : Proper Subset of Candidate Key, and A : Non Key Attribute

If $Y \rightarrow A$ exists in R , then R is not in 2NF.

$(Y \rightarrow A)$ is a Partial dependency only if

- Y : Proper subset of Candidate Key
- A : Non Prime Attribute

Source: <http://www.edugrabs.com/2nf-second-normal-form/>

Database System Concepts - 9th Edition 16.14 ©Silberschatz, Korth and Sudarshan

Moving on the second normal form which is obviously, a relation is in second normal form if it is in first normal form and it does not have any partial dependency. So, what is the partial dependency? I have given the definition here partial dependency why determining A if that that can hold in the set of functional dependency then if I have that Y is a proper subset of a candidate key and A is a nonprime attribute in nonprime attribute is one which one nonprime attribute we defined in the last module is an attribute which does not feature in any of the candidate keys.

So, if Y is a proper subset of a candidate key which functionally determines a nonprime attribute; then this is known as a partial dependency and if there is partial dependency

then the relationship is not in second normal form. So, second normal form will require that the relation is in 1 NF and there is no partial dependency.

(Refer Slide Time: 12:25)

Second Normal Form (2 NF)

Example:
 • *STUDENT(Sid, Sname, Cname)* (already in 1NF)

Students:

SID	Sname	Cname
S1	A	C
S1	A	C++
S2	B	C++
S2	B	DB
S3	A	DB

(SID, Cname): Primary Key

Functional Dependencies:
 {SID, Cname} → Sname
 SID → Sname

Partial Dependencies:
 SID → Sname (as SID is a Proper Subset of Candidate Key {SID, Cname})

Post Normalization

R1:

SID	Sname
S1	A
S2	B
S3	A

{SID}: Primary Key

R2:

SID	Cname
S1	C
S1	C++
S2	C++
S2	DB
S3	DB

{SID, Cname}: Primary Key

The above two relations R1 and R2 are
 1. Lossless Join
 2. 2NF
 3. Dependency Preserving

Database System Concepts - 9th Edition 16.15 ©Silberschatz, Korth and Sudarshan

So, here I were showing an example where on the left you can see that SID and C name together forms a key and SID determines S name. So, SID and C name together also determines S name naturally SID determining S name is a partial dependency because the left hand side SID is a proper subset of the candidate key SID C name. And S name is not featuring in any candidate key. So, S name is actually a nonprime attribute and the result of that as you can see in the first two rows or in the third and fourth row you can see that S name is repeated.

So, there is redundancy and therefore, consequently we will have anomalies that we have talked of, but we can normalize we can decompose this into two separate relations R1 and R2 as I am showing on the right; where you associate SID and S name in one table and SID and C name in other table. Naturally then the dependency that the partial dependency that you had disappears because SID determining S name in R1; now becomes is not a partial dependency because in that table SID becomes a primary key. So, it does not qualify as a partial dependency..

So, R1 and R2 both are in second normal form and you will get rid of the redundancy that you saw and this decomposition is ensures that it has a list lossless join incidentally;

this is we have not guaranteed that it is in second normal form and it has also the dependency preservation.

(Refer Slide Time: 14:04)

PPD

Second Normal Form (2 NF): Possible Redundancy

■ Example:
 ● **Supplier(SID, Status, City, PID, Qty)**

Supplier:

SID	Status	City	PID	Qty
S1	30	Delhi	P1	100
S1	30	Delhi	P2	125
S1	30	Delhi	P3	200
S1	30	Delhi	P4	130
S2	10	Karnal	P1	115
S2	10	Karnal	P2	250
S3	40	Rohtak	P1	245
S4	30	Delhi	P4	300
S4	30	Delhi	P5	315

Key : (SID, PID)

Partial Dependencies:
 SID → Status
 SID → City

Post Normalization

Sup_City :

SID	Status	City
S1	30	Delhi
S2	10	Karnal
S3	40	Rohtak
S4	30	Delhi

FDD of Sup_City :

Sup_Qty :

SID	PID	Qty
S1	P1	100
S1	P2	125
S1	P3	200
S1	P4	130
S2	P1	115
S2	P2	250
S3	P1	245
S4	P4	300
S4	P5	315

FDD of Sup_qty :

Drawbacks:

- **Deletion Anomaly** – If we delete a tuple in Sup_City, then we not only loose the information about a supplier, but also loose the status value of a particular city.
- **Insertion Anomaly** – We cannot insert a City and its status until a supplier supplies at least one part.
- **Update Anomaly** – If the status value for a city is changed, then we will face the problem of searching every tuple for that city.

Source: <http://www.edugrabs.com/2nf-second-normal-form/>

Database System Concepts - 9th Edition

16.16

©Silberschatz, Korth and Sudarshan

But it is possible again in second normal form a relation could be in second normal form yet it could have some possible redundancies. So, there is a design instance that I am showing with the supplier ID, SID the status key which are functionally determined by SID and the product and quantity values..

So, that in the table supplier SID and PID together form say key whereas, and as that happens you can clearly see that there is a lot of redundancy that you can see in terms of the status happening and which will cause you different anomalies to occur. So, if I normalize in the second normal form on the right then I will have a supplier city say with the three attributes SID status and city and another supplier quantity which has SID PID and quantity naturally in this there is no partial dependency anymore.

Earlier we had SID determining status as a partial dependency because SID is a proper was a proper subset of the primary key which is SID CID, but after I normalize this dependency does not exist, but yet there will be redundancy in this relationship and there the status will continue to be redundant.

(Refer Slide Time: 15:30)

Second Normal Form (2 NF): Possible Redundancy

- In the **Sup_City** relation :
 - City → Status**
 - Non Key Attribute → Non Key Attribute
- In the **STUDENT** relation:
 - SID → Cname**
 - Proper Subset of 1 CK → Proper Subset of other CK

Diagram (i) shows a dependency from a Non Key Attribute to another Non Key Attribute. Diagram (ii) shows a dependency from a Proper Subset of one Candidate Key to a Proper Subset of another Candidate Key.

Source: <http://www.edugrabs.com/2nf-second-normal-form>

Database System Concepts - 6th Edition 16.17 ©Silberschatz, Korth and Sudarshan

And for that reason we have to move on to the next type of normal form. So, this I am just explaining here as to what are the possible redundancy sources of possible redundancy that you can have in 2 NF.

(Refer Slide Time: 15:43)

Third Normal Form (3 NF)

Let **R** be the relational schema.

- [E. F. Codd, 1971] **R** is in 3NF only if:
 - R** should be in 2NF
 - R** should not contain transitive dependencies (OR, Every non-prime attribute of **R** is non-transitively dependent on every key of **R**)
- [Carlo Zaniolo, 1982] Alternately, **R** is in 3NF iff for each of its functional dependencies $X \rightarrow A$, at least one of the following conditions holds:
 - X** contains **A** (that is, **A** is a subset of **X**, meaning $X \rightarrow A$ is trivial functional dependency), or
 - X** is a superkey, or
 - Every element of **A-X**, the set difference between **A** and **X**, is a *prime attribute* (i.e., each attribute in **A - X** is contained in some candidate key)
- [Simple Statement] A relational schema **R** is in 3NF if for every FD $X \rightarrow A$ associated with **R** either
 - A** \subseteq **X** (i.e., the FD is trivial) or
 - X** is a superkey of **R** or
 - A** is part of some key (not just superkey!)

Handwritten note: } BCNF

Source: <http://www.edugrabs.com/3nf-third-normal-form>

Database System Concepts - 6th Edition 16.18 ©Silberschatz, Korth and Sudarshan

In the 3 NF; third normal form what you define is your relation first of all has to be in 2 NF. So, we are looking at the first definition these are there are three forms of definitions given all of them are actually equivalent, you do not have to worry about why and how they are equivalent slowly you will start understanding.

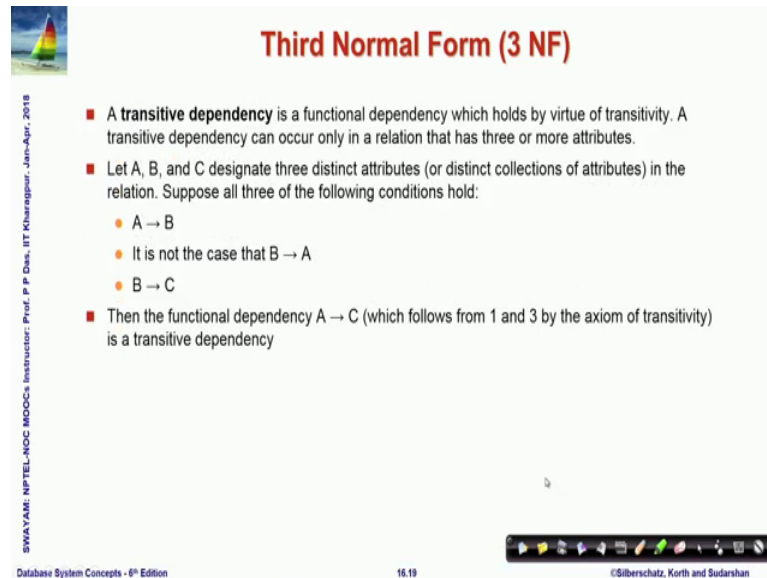
But we take it in three different forms because each form of the definition allow us to understand certain aspect of the three normal form. So, the first thing which is true for everything is it has to be in the second normal form and it should not contain any transitive dependency which means that if I have X determining Y and Y determining Z; then I should not have X determining Z which can be inferred transitively as you know through the transitive axiom.

Alternatively there was an alternate definition given later on by Zaniolo and I have stated a simpler simplified version of that at the bottom. So, we will say that a relational schema is in 3 NF if for every functional dependency X determining A that holds on this schema either it is a trivial dependency which is A is a subset of X or X is a super key.

So, this is kind of the condition also as you had seen earlier this also is a condition to be in Boyce Codd normal form. So, you can easily understand the 3 NF is a any relation which is in 3 NF is also in the Boyce Codd normal form, but we add a fourth third condition where you say that we will say this is in 3 NF; even if the first two conditions are not satisfied, but A is a part of some key just note the wording is a part of some key not just the super key..

So, if A is a part of some key then and the first two conditions are also not are not satisfied even then we will say that the relation is in third normal form. So, to check for a relation to be in third normal form; we will actually check for whether any one of the three conditions hold.

(Refer Slide Time: 18:00)



Third Normal Form (3 NF)

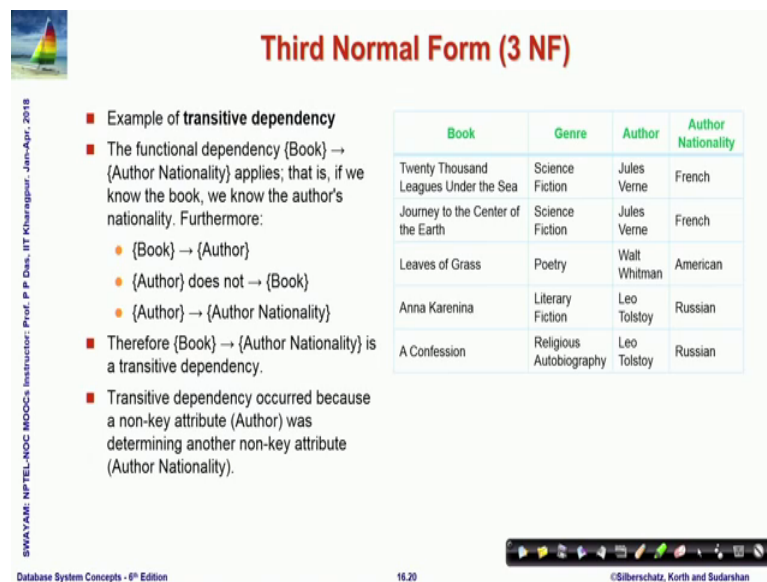
- A **transitive dependency** is a functional dependency which holds by virtue of transitivity. A transitive dependency can occur only in a relation that has three or more attributes.
- Let A, B, and C designate three distinct attributes (or distinct collections of attributes) in the relation. Suppose all three of the following conditions hold:
 - $A \rightarrow B$
 - It is not the case that $B \rightarrow A$
 - $B \rightarrow C$
- Then the functional dependency $A \rightarrow C$ (which follows from 1 and 3 by the axiom of transitivity) is a transitive dependency

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 8th Edition 16.19 ©Silberschatz, Korth and Sudarshan

So, this is a definition of transitive dependency which I have just loosely told you. So, I will skip over this.

(Refer Slide Time: 18:09)



Third Normal Form (3 NF)

- Example of **transitive dependency**
- The functional dependency $\{Book\} \rightarrow \{Author\}$ applies; that is, if we know the book, we know the author's nationality. Furthermore:
 - $\{Book\} \rightarrow \{Author\}$
 - $\{Author\}$ does not $\rightarrow \{Book\}$
 - $\{Author\} \rightarrow \{Author\ Nationality\}$
- Therefore $\{Book\} \rightarrow \{Author\ Nationality\}$ is a transitive dependency.
- Transitive dependency occurred because a non-key attribute (Author) was determining another non-key attribute (Author Nationality).

Book	Genre	Author	Author Nationality
Twenty Thousand Leagues Under the Sea	Science Fiction	Jules Verne	French
Journey to the Center of the Earth	Science Fiction	Jules Verne	French
Leaves of Grass	Poetry	Walt Whitman	American
Anna Karenina	Literary Fiction	Leo Tolstoy	Russian
A Confession	Religious Autobiography	Leo Tolstoy	Russian

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 8th Edition 16.20 ©Silberschatz, Korth and Sudarshan

There is given another example of a very different kind of a relationship book genre author and author nationality as you can understand. Given the book you know the author there is a functional dependency given the author do you know the author nationality and the, but author \rightarrow does not actually determine the book because the author may have written multiple books. But given that book determines author and author

determines author nationality we have that book determines author nationality and therefore, we have redundancy possibility of redundancy in here which is a transitive redundancy due to this transitive dependency that we have.

(Refer Slide Time: 18:48)

PPD

Third Normal Form (3 NF)

■ Example:

- **Sup_City(SID, Status, City)** (already in 2NF)

Sup_City:		
SID	Status	City
S1	30	Delhi
S2	10	Karnal
S3	40	Rohtak
S4	30	Delhi

SID: Primary Key

- Redundancy?
 - Status
- Anomaly?
 - Yes

Sup_City :

SID	Status	City
-----	--------	------

FDD of Sup_City :

Functional Dependencies:
SID → Status, SID → City
City → Status

Transitive Dependency :
SID → Status (As SID → City and City → Status)

Post Normalization

SC:		CS:	
SID	City	City	Status
S1	Delhi	Delhi	30
S2	Karnal	Karnal	10
S3	Rohtak	Rohtak	40
S4	Delhi		

SID: Primary Key

City: Primary Key

The above two relations SC and CS are

1. Lossless Join
2. 3NF
3. Dependency Preserving

So, here is a the earlier example where you can as you can see clearly in this diagram you can if you note this diagram you can see that SID determines city and city determines status. So, this is it this is the transitive dependency that SID determines status..

So, if that happens and status becomes redundant and therefore, there could be anomalies. And we can easily normalize by making them into SID and city and city and status. And in that naturally that that redundancy goes away because you have no more the transitive dependency in the relationship; you only have SID determining the city which is a primary key in S C and city determining status which is the primary key in the C S.

(Refer Slide Time: 19:50)

Third Normal Form (3 NF)

- Example
 - Relation *dept_advisor*.
 - *dept_advisor*(*s_ID*, *i_ID*, *dept_name*)
 - $F = \{s_ID, dept_name \rightarrow i_ID, i_ID \rightarrow dept_name\}$
 - Two candidate keys: *s_ID, dept_name*, and *i_ID, s_ID*
 - *R* is in 3NF
 - $s_ID, dept_name \rightarrow i_ID$
 - *s_ID, dept_name* is a superkey
 - $i_ID \rightarrow dept_name$
 - *dept_name* is contained in a candidate key

A relational schema *R* is in 3NF if for every FD $X \rightarrow A$ associated with *R* either

- $A \subseteq X$ (i.e., the FD is trivial) or
- *X* is a superkey of *R* or
- *A* is part of some key (not just superkey!)

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargapur, Jan-Apr, 2018
Database System Concepts - 6th Edition 18.22 ©Silberschatz, Korth and Sudarshan

So, there are these are other examples that that you can go through where we have I have taken the example of a student ID I ID and the department name and shown that what kind of problems, you might get into in this. In this case you can see that the relationship actually is in the there because there are two candidate keys and. So, this SID department name is a super key and this relationship is in the third normal form. Because IID determining department name is contained in a candidate key. So, that is the it is a it is in 3 NF due to the third condition that we have had shown.

(Refer Slide Time: 20:41)

Redundancy in 3NF

- **There is some redundancy in this schema**
- Example of problems due to redundancy in 3NF (*J*: *s_ID*, *L*: *i_ID*, *K*: *dept_name*)
 - $R = (J, L, K)$
 - $F = \{JK \rightarrow L, L \rightarrow K\}$

<i>J</i>	<i>L</i>	<i>K</i>
<i>j</i> ₁	<i>l</i> ₁	<i>k</i> ₁
<i>j</i> ₂	<i>l</i> ₁	<i>k</i> ₁
<i>j</i> ₃	<i>l</i> ₁	<i>k</i> ₁
null	<i>l</i> ₂	<i>k</i> ₂

- repetition of information (e.g., the relationship *l*₁, *k*₁)
 - (*i_ID*, *dept_name*)
- need to use null values (e.g., to represent the relationship *l*₂, *k*₂ where there is no corresponding value for *J*).
 - (*i_ID*, *dept_name*) if there is no separate relation mapping instructors to departments

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargapur, Jan-Apr, 2018
Database System Concepts - 6th Edition 18.23 ©Silberschatz, Korth and Sudarshan

So, when you, but this is a where you can there is some redundancy in this schema that you can observe. So, this is just constructed and you have been because of this redundancy you have been able to we have had to use null values in this case.

(Refer Slide Time: 21:02)

Third Normal Form (3 NF): Possible Redundancy

- A table is automatically in 3NF if one of the following hold :
 - (i) If relation consists of two attributes.
 - (ii) If 2NF table consists of only one non key attributes
- If $X \rightarrow A$ is a dependency, then the table is in the 3NF, if one of the following conditions exists:
 - If X is a superkey
 - If X is a part of superkey
- If $X \rightarrow A$ is a dependency, then the table is said to be NOT in 3NF if the following:
 - If X is a proper subset of some key (partial dependency)
 - If X is not a proper subset of key (non key)

(ii)

Source: <http://www.edugrabs.com/2nf-second-normal-form/>

Database System Concepts - 6th Edition 16.24 ©Silberschatz, Korth and Sudarshan

So, in a third normal form there is possible redundancy coming in and these are the different cases that we have to check through.

(Refer Slide Time: 21:13)

DECOMPOSITION TO 3NF

- Normal Forms
- Decomposition to 3NF
- Decomposition to BCNF

Database System Concepts - 6th Edition 16.25 ©Silberschatz, Korth and Sudarshan

So, next what ; so, we have seen the different normal forms first normal form no multivalued attribute then the second normal form no partial dependency then the third normal form where you do not have any transitive dependency. So, all these are cascading definitions. So, in third normal form you have low multivalued attribute, no partial dependency and no transitive dependency.

So, now what will take a look into is how if I am given a relational schema and if it is violating any one or more of this condition. So, that the schema is not in the three normal form third normal form then how can we decompose it into the third normal form?

(Refer Slide Time: 21:55)

The slide is titled "Third Normal Form: Motivation" in red text. It features a small image of a sailboat in the top left corner. The main content is a bulleted list:

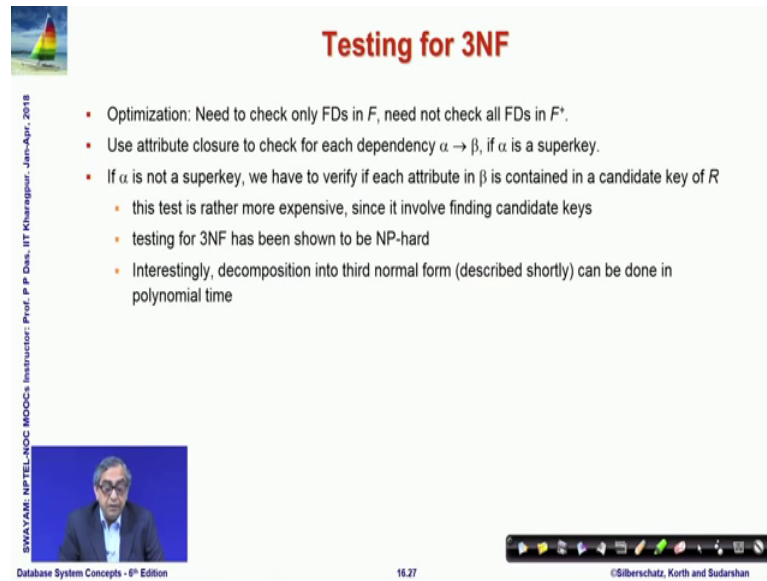
- There are some situations where
 - BCNF is not dependency preserving, and
 - Efficient checking for FD violation on updates is important
- Solution: define a weaker normal form, called Third Normal Form (3NF)
 - Allows some redundancy (with resultant problems; as seen above)
 - But functional dependencies can be checked on individual relations without computing a join
 - **There is always a lossless-join, dependency-preserving decomposition into 3NF**

At the bottom left, there is a small video inset showing a man speaking. The slide footer includes the text "Database System Concepts - 9th Edition" on the left, "19.26" in the center, and "©Silberschatz, Korth and Sudarshan" on the right.

So, the question naturally is certainly is can it always be done is the basic question that can I always decompose a schema into third normal form the answer is yes you can and that is always a lossless join and dependency preserving decomposition into third normal form which is of great value.

Because that is we said is that desirable properties of our decomposition and if you recall our discussions in the earlier part of the relational design modules, then you would recall that Boyce Codd normal form also we had discussed at the early stages. And that gives you a decomposition which is lossless join, but it does not guarantee preservation of the dependencies with third normal form does that.

(Refer Slide Time: 22:49)



Testing for 3NF

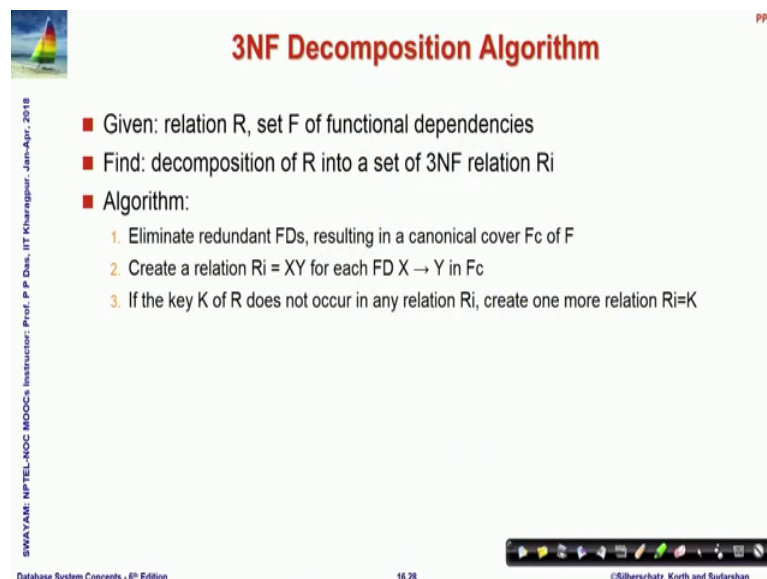
- Optimization: Need to check only FDs in F , need not check all FDs in F^+ .
- Use attribute closure to check for each dependency $\alpha \rightarrow \beta$, if α is a superkey.
- If α is not a superkey, we have to verify if each attribute in β is contained in a candidate key of R
 - this test is rather more expensive, since it involves finding candidate keys
 - testing for 3NF has been shown to be NP-hard
 - Interestingly, decomposition into third normal form (described shortly) can be done in polynomial time

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargapur, Jan-Apr, 2018

Database System Concepts - 6th Edition 16.27 ©Silberschatz, Korth and Sudarshan

So, naturally there are different algorithms first the question is can you test if a relationship is in third normal form; I will not go into the details of that and the computer science result here is testing for third normal form is an NP hard problem. So, there is no known polynomial time algorithm for that, but the interesting thing is the actually that decomposition can be done in very simply in polynomial time.

(Refer Slide Time: 23:17)



3NF Decomposition Algorithm

- Given: relation R , set F of functional dependencies
- Find: decomposition of R into a set of 3NF relation R_i
- Algorithm:
 1. Eliminate redundant FDs, resulting in a canonical cover F_c of F
 2. Create a relation $R_i = XY$ for each FD $X \rightarrow Y$ in F_c
 3. If the key K of R does not occur in any relation R_i , create one more relation $R_i = K$

PPD

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargapur, Jan-Apr, 2018

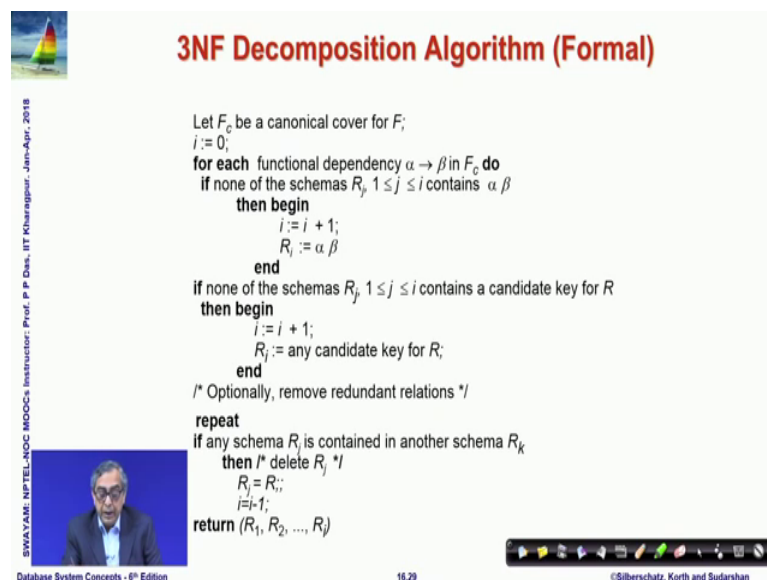
Database System Concepts - 6th Edition 16.28 ©Silberschatz, Korth and Sudarshan

So, what do you have what is the decomposition algorithm very written in very simple terms you want to you have given a relation R and a set of functional dependencies that

hold on you. So, you first compute a canonical cover you know what is a canonical cover. So, you compute a canonical covers you eliminate extraneous attributes eliminate redundant FDs and you have the canonical cover F_c from F then you create for every functional dependency X determining Y that exists in the canonical cover.

You compute you make a relation say the i th relation taking union of X and Y . So, you call it the relation $X Y$ and you do that for all the functional dependencies in the cover. And after that if you find that the key does not occur in any one of these decomposed relations as generated, then you generate one separate relation to represent the key.

(Refer Slide Time: 24:19)



3NF Decomposition Algorithm (Formal)

```

Let  $F_c$  be a canonical cover for  $F$ ;
 $i := 0$ ;
for each functional dependency  $\alpha \rightarrow \beta$  in  $F_c$  do
  if none of the schemas  $R_j$ ,  $1 \leq j \leq i$  contains  $\alpha, \beta$ 
  then begin
     $i := i + 1$ ;
     $R_i := \alpha, \beta$ 
  end
if none of the schemas  $R_j$ ,  $1 \leq j \leq i$  contains a candidate key for  $R$ 
then begin
   $i := i + 1$ ;
   $R_i :=$  any candidate key for  $R$ ;
end
/* Optionally, remove redundant relations */
repeat
if any schema  $R_i$  is contained in another schema  $R_k$ 
then /* delete  $R_i$  */
   $R_i = R_k$ ;
   $i := i - 1$ ;
return  $(R_1, R_2, \dots, R_i)$ 

```

Database System Concepts - 6th Edition 19.29 ©Silberschatz, Korth and Sudarshan

That is a very simple algorithm and I just wrote it in simple hand. So, that you can understand it easily, but here is the formal algorithm. So, if you are interested to rigor I mean in the in the rigor of how 3 NF decomposition will happen here is the algorithm, but I will not go through these in steps.

(Refer Slide Time: 24:37)

3NF Decomposition Algorithm

- Above algorithm ensures:
 - Each relation schema R_i is in 3NF
 - Decomposition is d
 - Dependency preserving and
 - Lossless-join

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 9th Edition 16.30 ©Silberschatz, Korth and Sudarshan

So, that ensures that each relation R_i that I have decomposed and generated is actually in third normal form and this decomposition is dependency preserving and is lossless join we are not proving that but we are just using that result.

(Refer Slide Time: 24:54)

Example of 3NF Decomposition

- Relation schema:
 $cust_banker_branch = (customer_id, employee_id, branch_name, type)$

Functional Dependencies:

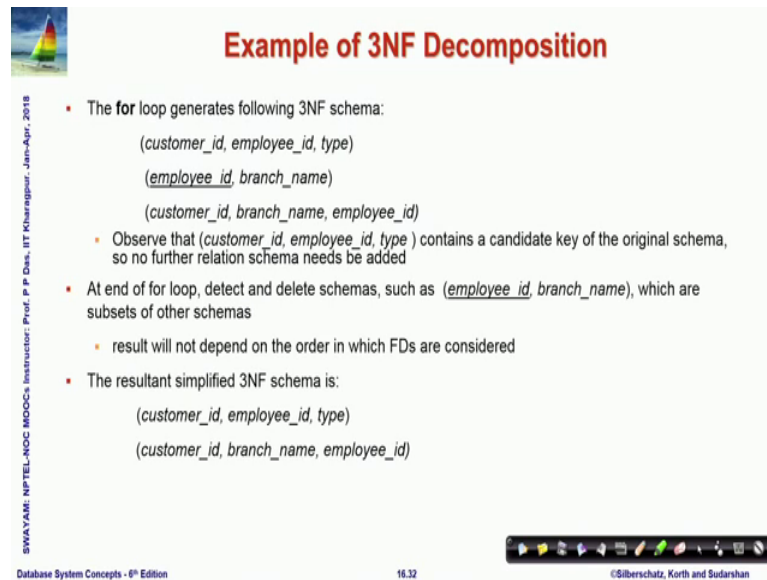
- $customer_id \rightarrow employee_id$
- $customer_id \rightarrow branch_name$
- $customer_id \rightarrow type$
- $employee_id \rightarrow type$
- $branch_name \rightarrow type$

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

So, here is an example of a schema; so, we have a customer banker branch. So, these are the four attributes and these are the different functional dependencies that exist. Now naturally given this first thing you will have to do is first thing you have to do is to look at the different to look at taking the canonical cover the minimal cover.

So, if you compute try to compute the minimal cover; you will find that branch name actually is extraneous in the first dependency. So, you can remove that and there is nothing else.

(Refer Slide Time: 25:36)



Example of 3NF Decomposition

- The for loop generates following 3NF schema:
 - $(customer_id, employee_id, type)$
 - $(employee_id, branch_name)$
 - $(customer_id, branch_name, employee_id)$
- Observe that $(customer_id, employee_id, type)$ contains a candidate key of the original schema, so no further relation schema needs be added
- At end of for loop, detect and delete schemas, such as $(employee_id, branch_name)$, which are subsets of other schemas
 - result will not depend on the order in which FDs are considered
- The resultant simplified 3NF schema is:
 - $(customer_id, employee_id, type)$
 - $(customer_id, branch_name, employee_id)$

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018
Database System Concepts - 9th Edition 16.32 ©Silberschatz, Korth and Sudarshan

So, your canonical cover turns out to be this set of dependencies and then you go over and for each one of them. So, you take each one the first one is customer ID employee ID determines type. So, for that you generate a schema customer ID, employee ID and type again you take the second functional dependency employee ID determines branch name. So, create employee ID and branch name as a different schema and in this way you will generate three decomposed schema in the third normal form.

Now, once you have done that then you find that your if you look into the original key it was customer ID and employee ID and you find that here in the third second and the third you already have that. So, you do not need to add a separate relation for accommodating the key and also the third relation. So, we can now declare that no further key needs to be added and we have the final 3 NF decomposition..

So, at the end of the fault detect and delete. So, this is this is a stated in terms of the detailed algorithm, but this is you can say that the employee ID and branch name the second relation in the decomposition is actually a subset of the third relation. So, you can remove that as well. So, you will be left with only two relations in this decompose

schema which both of which are in third normal form and this decomposition is guaranteed you lossless join and dependency preservation.

(Refer Slide Time: 27:17)

Practice Problem for 3NF Decomposition: 1

- $R = ABCDEFGH$
- $FDs = \{A \rightarrow B, ABCD \rightarrow E, EF \rightarrow GH, ACDF \rightarrow EG\}$

Solution is given in the next slide (hidden from presentation – check after you have solved)

SWAYAM: NPTEL-NOC MOOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 9th Edition 16.33 ©Silberschatz, Korth and Sudarshan

So, I have given some practice problems for you I have also given the solution, but the solution is not in the current run of the presentation; you will get see them in the presentation as hidden slides. So, you first try solving them and once you have solved them then you look at the solution in the slide.

(Refer Slide Time: 27:37)

Practice Problem for 3NF Decomposition: 2

- $R = CSJDPQV$
- $FDs = \{C \rightarrow CSJDPQV, SD \rightarrow P, JP \rightarrow C, J \rightarrow S\}$

Solution is given in the next slide (hidden from presentation – check after you have solved)

SWAYAM: NPTEL-NOC MOOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 9th Edition 16.35 ©Silberschatz, Korth and Sudarshan

So, there are two problems; so, this is a second one and you can solve them in that way.

(Refer Slide Time: 27:40)

PPD

- Normal Forms
- Decomposition to 3NF
- Decomposition to BCNF

DECOMPOSITION TO BCNF

SWAYAM: NPTEL-NOC MOCs Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 6th Edition 16.37 ©Silberschatz, Korth and Sudarshan

Next is the we will quickly recap on the decomposition of BCNF Boyce Codd normal form which we had seen earlier.

(Refer Slide Time: 27:49)

Testing for BCNF

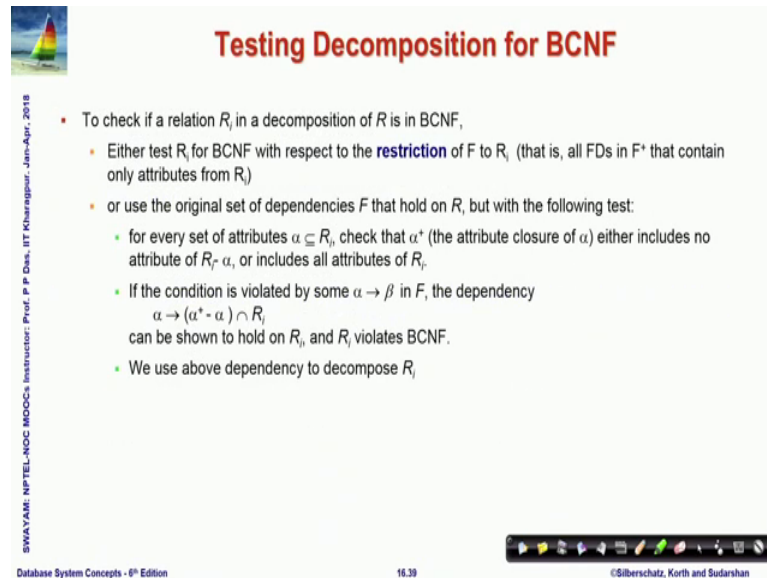
- To check if a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF
 1. compute α^+ (the attribute closure of α), and
 2. verify that it includes all attributes of R , that is, it is a superkey of R .
- **Simplified test:** To check if a relation schema R is in BCNF, it suffices to check only the dependencies in the given set F for violation of BCNF, rather than checking all dependencies in F^+ .
 - If none of the dependencies in F causes a violation of BCNF, then none of the dependencies in F^+ will cause a violation of BCNF either.
- However, **simplified test using only F is incorrect when testing a relation in a decomposition of R**
 - Consider $R = (A, B, C, D, E)$, with $F = \{A \rightarrow B, BC \rightarrow D\}$
 - Decompose R into $R_1 = (A, B)$ and $R_2 = (A, C, D, E)$
 - Neither of the dependencies in F contain only attributes from (A, C, D, E) so we might be misled into thinking R_2 satisfies BCNF.
 - In fact, dependency $AC \rightarrow D$ in F^+ shows R_2 is not in BCNF.

SWAYAM: NPTEL-NOC MOCs Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 6th Edition 16.38 ©Silberschatz, Korth and Sudarshan

And we know that the Boyce Codd normal form guarantees that there will have be every dependency that exists must be either trivial or the left hand side must be a super key. So, using the algorithms, you can test for the Boyce Codd normal form which is described here I am not going through in steps.

(Refer Slide Time: 28:08)



Testing Decomposition for BCNF

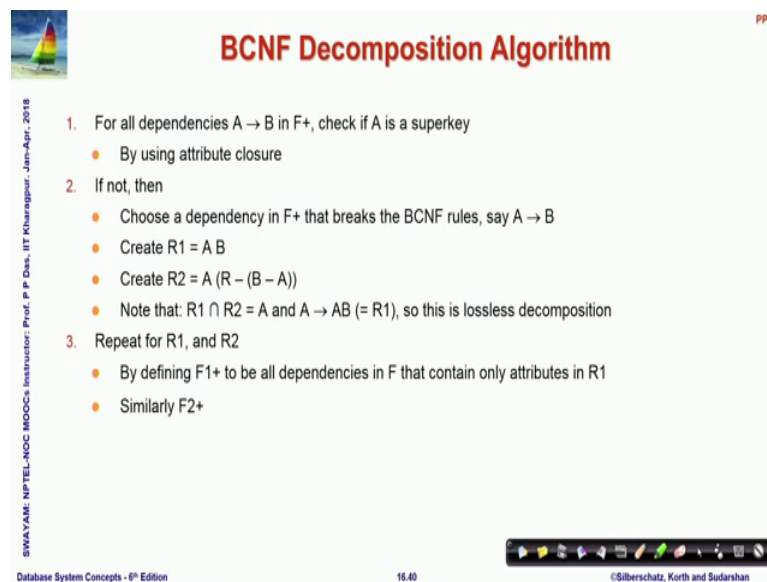
- To check if a relation R_i in a decomposition of R is in BCNF,
 - Either test R_i for BCNF with respect to the **restriction** of F to R_i (that is, all FDs in F^+ that contain only attributes from R_i)
 - or use the original set of dependencies F that hold on R , but with the following test:
 - for every set of attributes $\alpha \subseteq R_i$, check that α^+ (the attribute closure of α) either includes no attribute of $R_i - \alpha$, or includes all attributes of R_i .
 - If the condition is violated by some $\alpha \rightarrow \beta$ in F , the dependency $\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i$ can be shown to hold on R_i , and R_i violates BCNF.
 - We use above dependency to decompose R_i

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P P Das, IIT Khargapur, Jan-Apr, 2018

Database System Concepts - 6th Edition 16.39 ©Silberschatz, Korth and Sudarshan

And here is the more detailed formal algorithm to find determine whether a Boyce Codd normal form is in a decomposed form is in Boyce Codd.

(Refer Slide Time: 28:18)



BCNF Decomposition Algorithm

1. For all dependencies $A \rightarrow B$ in F^+ , check if A is a superkey
 - By using attribute closure
2. If not, then
 - Choose a dependency in F^+ that breaks the BCNF rules, say $A \rightarrow B$
 - Create $R_1 = A B$
 - Create $R_2 = A (R - (B - A))$
 - Note that: $R_1 \cap R_2 = A$ and $A \rightarrow AB (= R_1)$, so this is lossless decomposition
3. Repeat for R_1 , and R_2
 - By defining F_1^+ to be all dependencies in F that contain only attributes in R_1
 - Similarly F_2^+

PPD

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P P Das, IIT Khargapur, Jan-Apr, 2018

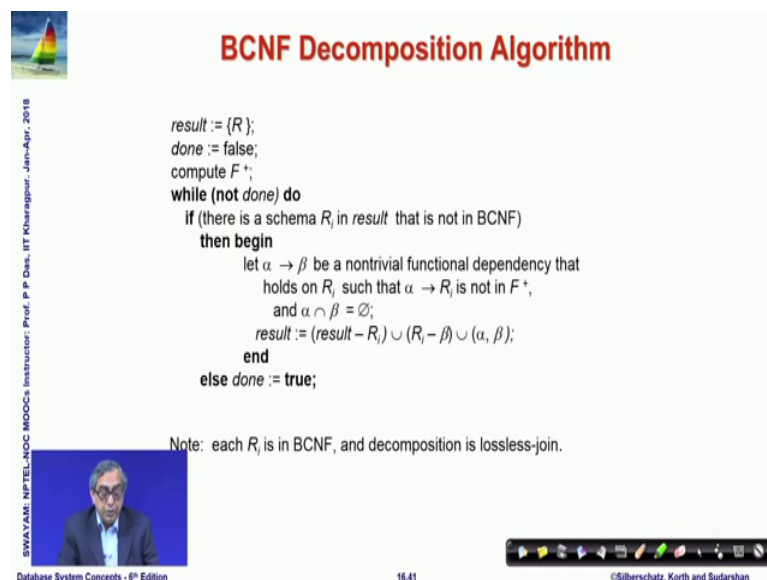
Database System Concepts - 6th Edition 16.40 ©Silberschatz, Korth and Sudarshan

So, I will just quickly recap on the algorithm to do that naturally for all dependencies you first determine the super key and check if A determining B is a super key or not if it and that you can easily do using attribute cover. If it is not a super key then you choose a dependency A determining B which violates and you form by Boyce Codd goes in every step it decomposes one relation into two separate relation..

So, one that you take by taking union of the attributes of A and B and the other where you take out B minus A; these attributes this difference attributes you take out from R and then you add A and make the other relationship. Naturally in between these two A is a common attribute and since and that will determine A B because A determines B..

So, A will determine A B that is whole of R1. So, naturally the lossless join is guaranteed and you repeat that keep on doing that for the resultant relations that you have got. keep on decomposing them till you finally, close and you have no more violating dependency and you will have a decomposition into Boyce Codd normal form.

(Refer Slide Time: 29:39)



BCNF Decomposition Algorithm

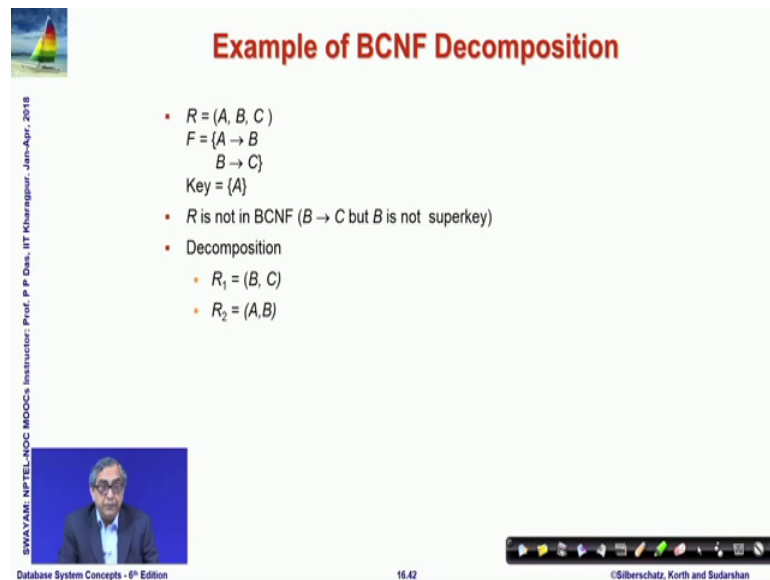
```
result := {R};
done := false;
compute F+;
while (not done) do
  if (there is a schema Ri in result that is not in BCNF)
    then begin
      let α → β be a nontrivial functional dependency that
        holds on Ri such that α → Ri is not in F+,
        and α ∩ β = ∅;
      result := (result - Ri) ∪ (Ri - β) ∪ (α, β);
    end
  else done := true;
```

Note: each R_i is in BCNF, and decomposition is lossless-join.

Database System Concepts - 9th Edition 16.41 ©Silberschatz, Korth and Sudarshan

Here is the formal algorithm again for you to go by steps if you are interested.

(Refer Slide Time: 29:46)



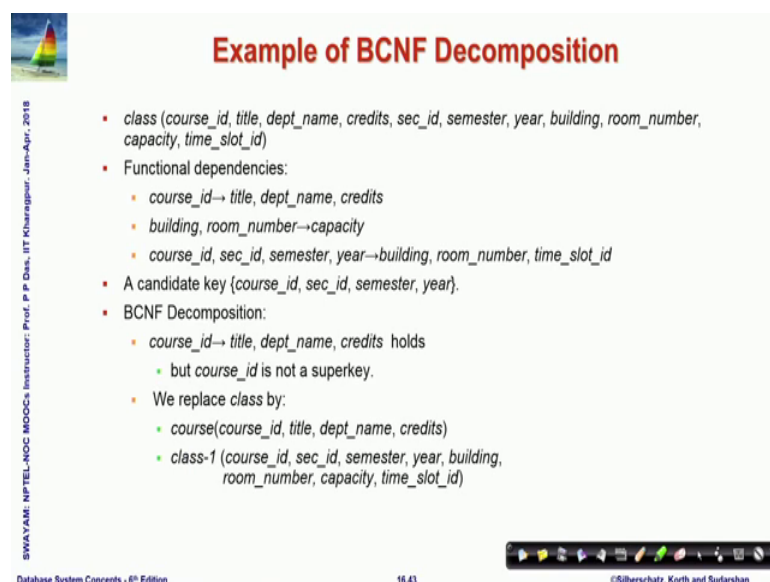
Example of BCNF Decomposition

- $R = (A, B, C)$
 $F = \{A \rightarrow B, B \rightarrow C\}$
Key = $\{A\}$
- R is not in BCNF ($B \rightarrow C$ but B is not superkey)
- Decomposition
 - $R_1 = (B, C)$
 - $R_2 = (A, B)$

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khairagpur, Jan-Apr, 2018
Database System Concepts - 6th Edition 16.42 ©Silberschatz, Korth and Sudarshan

Otherwise you know how to do this; again I have shown another example here which is showing that how to decompose in BCNF. So, you should practice this that is why I have work them out in steps here. So, here A determines B; B determine C naturally A is the key R is not in BCNF because B determining C is a functional dependency where B is not a super key.

(Refer Slide Time: 30:15)



Example of BCNF Decomposition

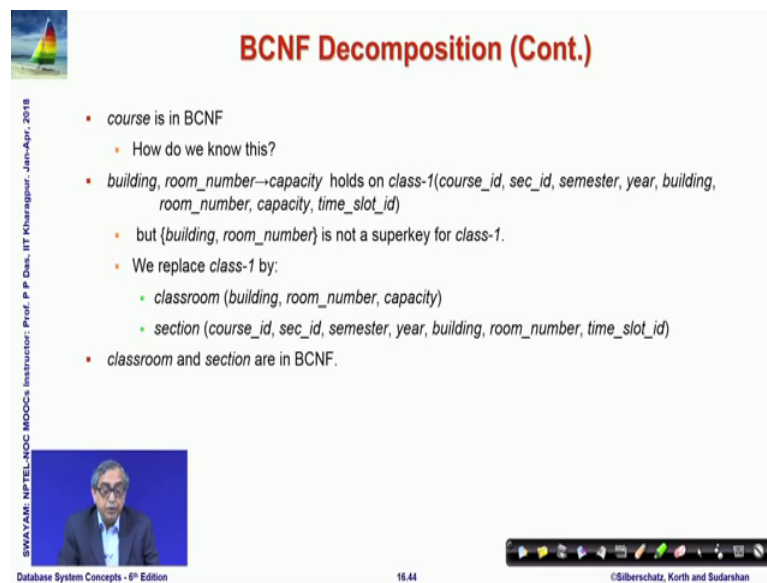
- *class* (*course_id*, *title*, *dept_name*, *credits*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)
- Functional dependencies:
 - $course_id \rightarrow title, dept_name, credits$
 - $building, room_number \rightarrow capacity$
 - $course_id, sec_id, semester, year \rightarrow building, room_number, time_slot_id$
- A candidate key $\{course_id, sec_id, semester, year\}$.
- BCNF Decomposition:
 - $course_id \rightarrow title, dept_name, credits$ holds
 - but $course_id$ is not a superkey.
 - We replace *class* by:
 - *course*(*course_id*, *title*, *dept_name*, *credits*)
 - *class-1* (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khairagpur, Jan-Apr, 2018
Database System Concepts - 6th Edition 16.43 ©Silberschatz, Korth and Sudarshan

So, you can decompose them in terms of. So, you can decompose in terms of B C as one relation and A B as another relation. Here is another example a more detailed one of a

class relationship which has a whole set of attributes and these functional dependencies and based on that the candidate key is course ID, section ID, semester and year and you can proceed with the BCNF decomposition; taking the first functional dependency that holds, but the left hand side the course ID is not a super key. So, you will replace it by a one relation; which is say new course relation and a new class relation which is the remaining attributes.

(Refer Slide Time: 31:06)



BCNF Decomposition (Cont.)

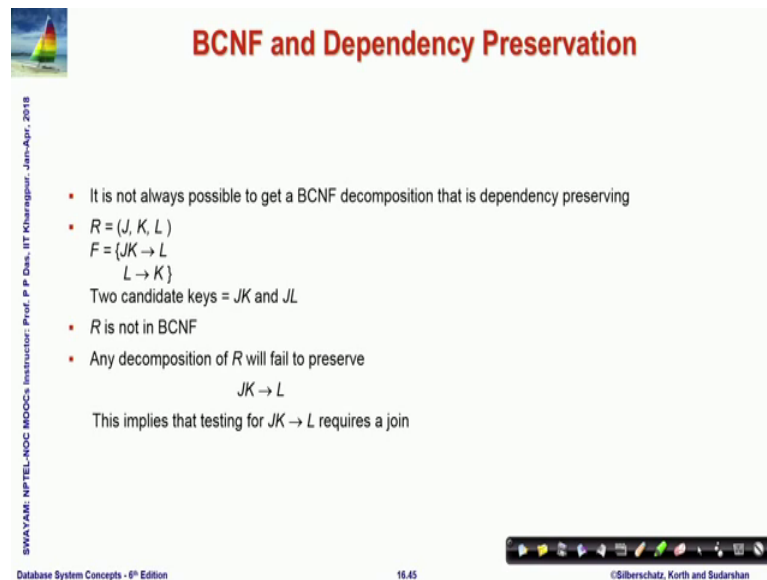
- *course* is in BCNF
 - How do we know this?
- *building, room_number* → *capacity* holds on *class-1*(*course_id, sec_id, semester, year, building, room_number, capacity, time_slot_id*)
 - but {*building, room_number*} is not a superkey for *class-1*.
 - We replace *class-1* by:
 - *classroom* (*building, room_number, capacity*)
 - *section* (*course_id, sec_id, semester, year, building, room_number, time_slot_id*)
- *classroom* and *section* are in BCNF.

SWAYAM: NPTEL-NOC MDOCC Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 6th Edition 16.44 ©Silberschatz, Korth and Sudarshan

And then you get convinced that *course* is in BCNF, but the other one the *class* is not because *building* and *room number* determines *capacity* where *building room number* together is not a super key. So, you split it again and you replace *class 1* in terms of 2 new relations *class room* and *section* and both of them are in BCNF and you are done with this.

(Refer Slide Time: 31:31)



BCNF and Dependency Preservation

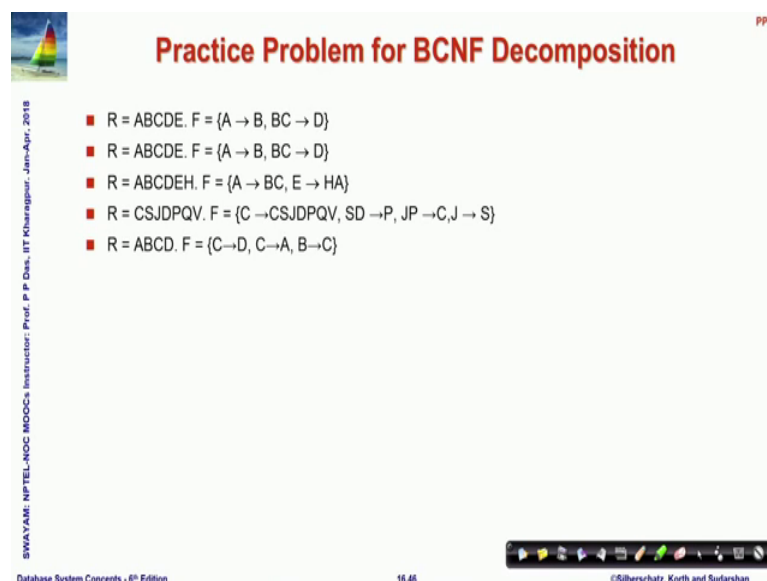
- It is not always possible to get a BCNF decomposition that is dependency preserving
- $R = (J, K, L)$
 $F = \{JK \rightarrow L, L \rightarrow K\}$
Two candidate keys = JK and JL
- R is not in BCNF
- Any decomposition of R will fail to preserve
 $JK \rightarrow L$
This implies that testing for $JK \rightarrow L$ requires a join

SWAYAM: NPTEL-NOC MOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 6th Edition 16.45 ©Silberschatz, Korth and Sudarshan

But BCNF as I would again warning you BCNF does not preserve dependence it gives you lossless join, but it does not preserve the dependencies. So, it is not always possible to decompose into BCNF with dependency preservation. So, here is an example which we saw little earlier and there are two candidate keys R is not in BCNF, you can clearly see and any decomposition will fail JK determining L and that will require a join. So, this will not preserve the dependencies in terms of the decomposition.

(Refer Slide Time: 32:06)



Practice Problem for BCNF Decomposition

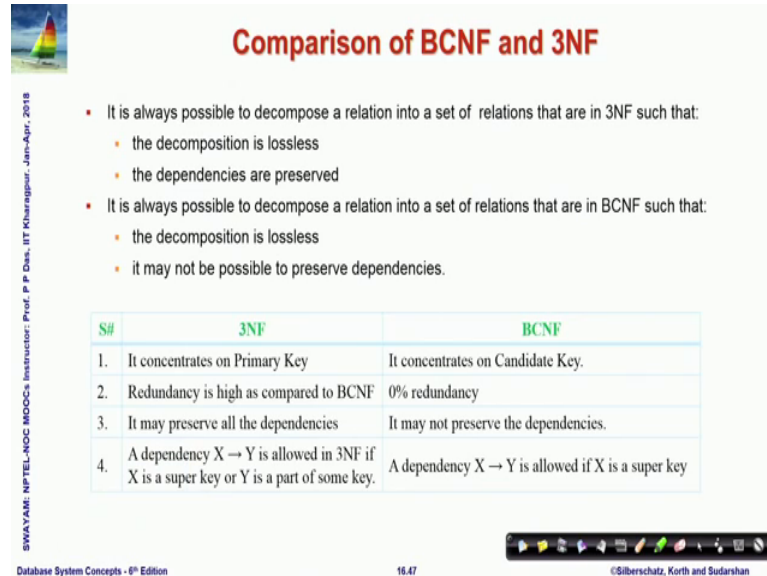
- $R = ABCDE, F = \{A \rightarrow B, BC \rightarrow D\}$
- $R = ABCDE, F = \{A \rightarrow B, BC \rightarrow D\}$
- $R = ABCDEH, F = \{A \rightarrow BC, E \rightarrow HA\}$
- $R = CSJDPQV, F = \{C \rightarrow CSJDPQV, SD \rightarrow P, JP \rightarrow C, J \rightarrow S\}$
- $R = ABCD, F = \{C \rightarrow D, C \rightarrow A, B \rightarrow C\}$

SWAYAM: NPTEL-NOC MOC's Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 6th Edition 16.46 ©Silberschatz, Korth and Sudarshan

Again I have given a set of practice problems here which we you should try and get confident in terms of the Boyce Codd from normal form normalization.

(Refer Slide Time: 32:19)



Comparison of BCNF and 3NF

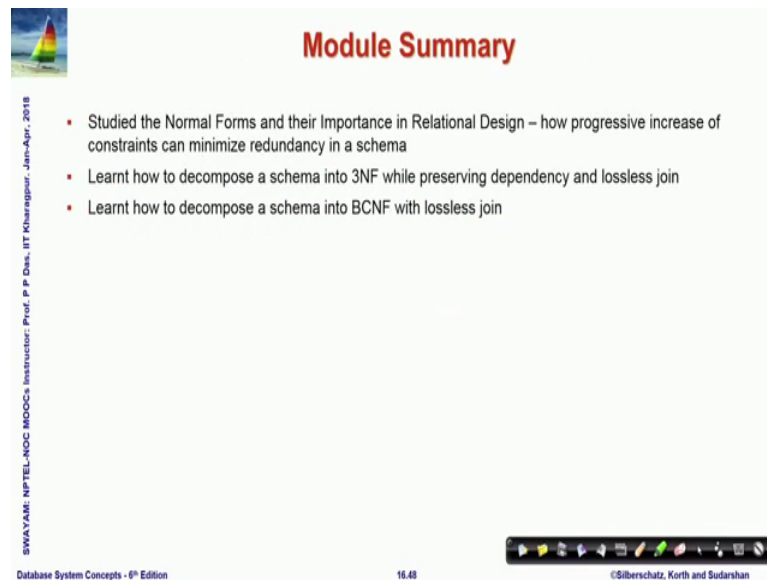
- It is always possible to decompose a relation into a set of relations that are in 3NF such that:
 - the decomposition is lossless
 - the dependencies are preserved
- It is always possible to decompose a relation into a set of relations that are in BCNF such that:
 - the decomposition is lossless
 - it may not be possible to preserve dependencies.

S#	3NF	BCNF
1.	It concentrates on Primary Key	It concentrates on Candidate Key.
2.	Redundancy is high as compared to BCNF	0% redundancy
3.	It may preserve all the dependencies	It may not preserve the dependencies.
4.	A dependency $X \rightarrow Y$ is allowed in 3NF if X is a super key or Y is a part of some key.	A dependency $X \rightarrow Y$ is allowed if X is a super key

SWAYAM: NPTEL-NOC MOCs- Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018
 Database System Concepts - 9th Edition 16.47 ©Silberschatz, Korth and Sudarshan

Now, it is always possible to decompose a relation into a set of relation in 3 NF; if the decomposition is lossless and the dependencies are preserved. Whereas, in case of BCNF it is not possible; so, here is a table which summarizes the relative comparison between Boyce Codd and third normal form because they are the common once Boyce Codd naturally is more strict it gives you lesser dependent lesser redundancies, but it cannot guarantee that your dependencies will be preserved. So, more often we will accept 3 NF as an acceptable normalized decomposition with some redundancy still existing it is possible and we cannot get rid of them.

(Refer Slide Time: 33:09)



The slide is titled "Module Summary" in a bold, red font. It features a small image of a sailboat in the top left corner. The main content is a bulleted list of three items. On the left side, there is vertical text providing course information. At the bottom, there is a footer with the course name, a slide number, and a copyright notice.

Module Summary

- Studied the Normal Forms and their Importance in Relational Design – how progressive increase of constraints can minimize redundancy in a schema
- Learnt how to decompose a schema into 3NF while preserving dependency and lossless join
- Learnt how to decompose a schema into BCNF with lossless join

SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargapur, Jan-Apr, 2018

Database System Concepts - 8th Edition 16.48 ©Silberschatz, Korth and Sudarshan

So, we have studied about the normal forms and their importance and how progressively we can increase the constraints to minimize redundancy in the schema and learned how to decompose a schema into third normal form and also in the Boyce Codd normal form.