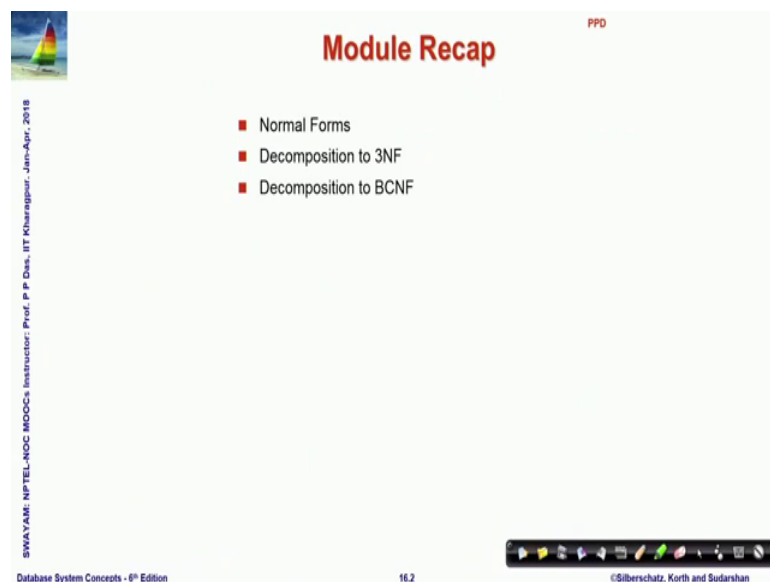


Database Management System
Prof. Partha Pratim Das
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 20
Relational Database Design (Contd.)

Welcome to module 20 of Database Management Systems. We have been discussing about relational database design, since the last 4 modules and this will be the concluding part of relational database design.

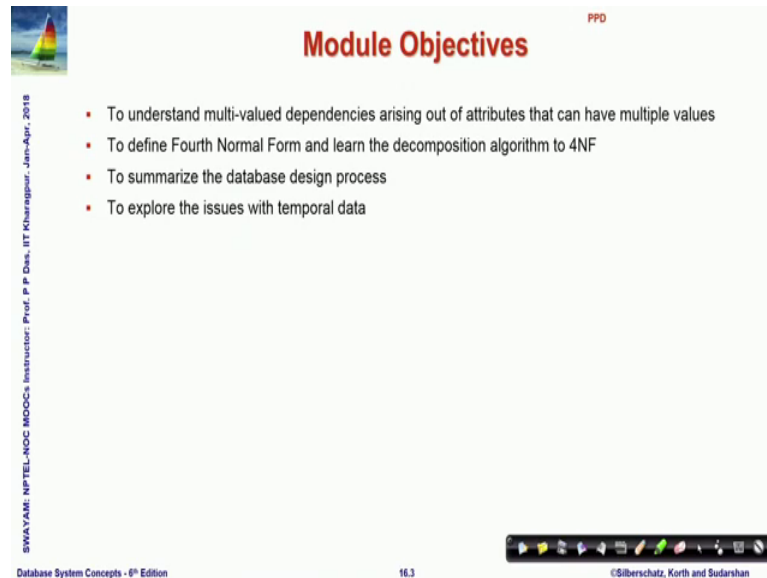
(Refer Slide Time: 00:33)



The slide is titled "Module Recap" in red text. It features a small image of a sailboat in the top left corner. The main content is a bulleted list of three items, each preceded by a red square bullet point: "Normal Forms", "Decomposition to 3NF", and "Decomposition to BCNF". The slide also includes a vertical text on the left side: "SWAYAM: NPTEL-NOC MCOCC Instructor: Prof. P P Das, IIT Kharagpur, Jan-Apr, 2018". At the bottom, there is a footer with "Database System Concepts - 6th Edition" on the left, "16.2" in the center, and "©Silberschatz, Korth and Sudarshan" on the right. A navigation bar with various icons is located at the bottom right.

In the last module, we have seen some very key concepts of relational design, that of normal forms third and Boyce Codd normal form specifically and how to decompose into them? And how do we get benefit in terms of doing this kind of decomposition? In removing the anomalies by reducing the redundancy in the design.

(Refer Slide Time: 00:59)



PPD

Module Objectives

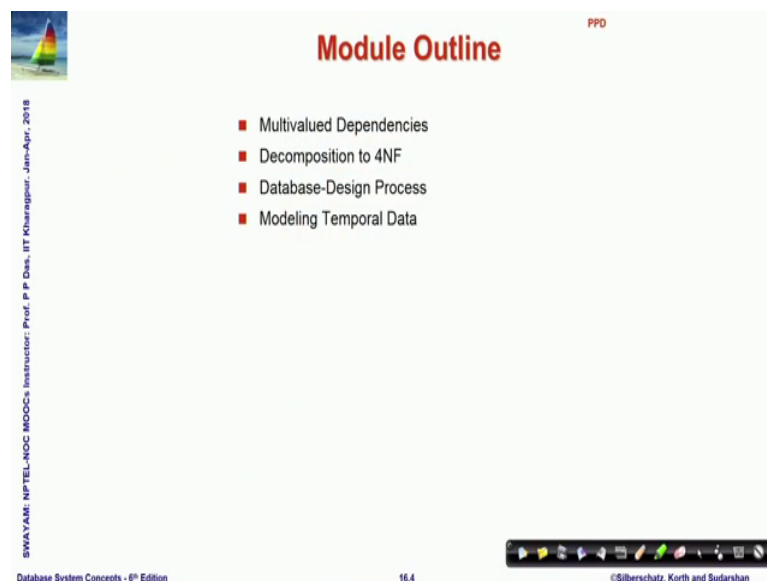
- To understand multi-valued dependencies arising out of attributes that can have multiple values
- To define Fourth Normal Form and learn the decomposition algorithm to 4NF
- To summarize the database design process
- To explore the issues with temporal data

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

Database System Concepts - 6th Edition 16.3 ©Silberschatz, Korth and Sudarshan

In view of that in the background of that, in this module we will try to understand a new kind of data dependency, an additional kind of data dependency, which is called multivalued dependency. Which can occur when an attribute can have multiple possible values, which we had eliminated in the first normal form together and based on that, we will define 4th normal form and decomposition into 4NF and then we will summarize this whole set of discussions of relational database design, and talk a little bit about what happens, when you have temporal data in your system.

(Refer Slide Time: 01:39)



PPD

Module Outline

- Multivalued Dependencies
- Decomposition to 4NF
- Database-Design Process
- Modeling Temporal Data

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

Database System Concepts - 6th Edition 16.4 ©Silberschatz, Korth and Sudarshan

(Refer Slide Time: 01:47)

Multivalued Dependency

■ **Persons(Man, Phones, Dog_Like)**

Person :			Meaning of the tuples
Man(M)	Phones(P)	Dogs_Like(D)	
			Man M have phones P, and likes the dogs D.
M1	P1/P2	D1/D2	M1 have phones P1 and P2, and likes the dogs D1 and D2.
M2	P3	D2	M2 have phones P3, and likes the dog D2.
Key : MPD			

There are no non trivial FDs because all attributes are combined forming Candidate Key i.e. MDP. In the above relation, two multivalued dependencies exists –

- Man \twoheadrightarrow Phones
- Man \twoheadrightarrow Dogs_Like

A man's phone are independent of the dogs they like. But after converting the above relation in Single Valued Attribute, each of a man's phones appears with each of the dogs they like in all combinations.

Post 1NF Normalization

Man(M)	Phones(P)	Dogs_Likes(D)
M1	P1	D1
M1	P2	D2
M2	P3	D2
M1	P1	D2
M1	P2	D1

Source: <http://www.edugrabs.com/multivalued-dependency-mvd/>

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

So, these are the outline points, based on our objectives and we start with the discussion of multivalued dependency. Consider a situation like this. So, here we are trying to represent an individual instead of persons with 3 attributes, man which is an may be id or name of that person, phones and dog like. So, the idea is that the here persons and they can have 1, 2, 3 any number of phones, which is true for all of us and then a person may have any number of dogs that he or she likes.

So, both of these phones and dog like D, P and D attributes can take multiple values and ah. So, if we if we if you look at the 1 NF normalized form here. So, in 1 NF what we do? We create separate rows for them. So, we have created separate rows for M 1 against P 1 and P 1 or P 2 in phones and similarly for D 1 and D 2. So, once we have done that then we have here, we can see I have highlighted with yellow, you can see the different redundancies that are arising.

Because, since I have phones and dog liking attributes. So, it is possible that if phone takes 2 values and dog like takes 2 values, then actually 4 different combinations of them are possible. But in reality, it may be in reality it may be actually these 2 are true, that M 1 has phone P 1 and likes dog D 1 M 1 has phone P 2 and likes dog D 2, but you could also have such redundant tuples coming in, because there they are now valid.

So, this is the situation which we try to try to capture, in terms of what you see here multivalued dependencies, where which is row shown in terms of double arrows as you

can see here. So, man I say determines multi determines phones man multi determines dog likes. So, there are 2 different multi valued dependencies in this case. So, this multi valued dependency adds a new source of redundancy in our data, and that is very real in various models of our system.

(Refer Slide Time: 04:19)

Multivalued Dependency

- If two or more independent relations are kept in a single relation, then Multivalued Dependency is possible. For example, Let there are two relations :
 - *Student(SID, Sname) where (SID → Sname)*
 - *Course(CID, Cname) where (CID → Cname)*
- There is no relation defined between Student and Course. If we kept them in a single relation named *Student_Course*, then MVD will exists because of *m:n Cardinality*
- If two or more MVDs exist in a relation, then while converting into SVAs, MVD exists.

Student:		Course:		SID	Sname	CID	Cname
SID	Sname	CID	Cname				
S1	A	C1	C	S1	A	C1	C
S2	B	C2	B	S1	A	C2	B
				S2	B	C1	C
				S2	B	C2	B

2 MVDs exist:
 1. SID →→ CID
 2. SID →→ Cname

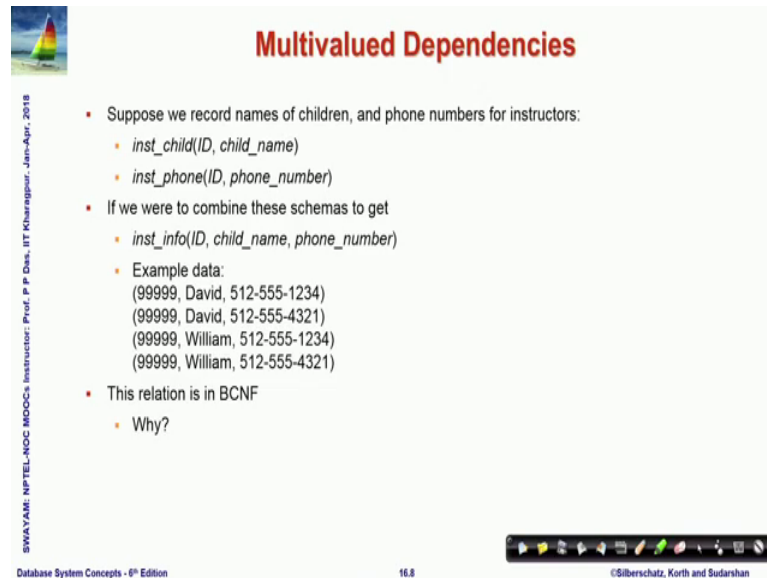
Source: <http://www.edugrabs.com/multivalued-dependency-mvd/>

Database System Concepts - 6th Edition 16.7 ©Silberschatz, Korth and Sudarshan

So, let us move on. So, this is just another example, we have 2 different relations student give the student id and name, courses giving course id and name and the corresponding functional dependencies in them, you can see 2 instances of that. But if we as such, there is no relationship between student and course, but if we choose to keep them in a single relation, say student course which I have shown on bottom right here. Then there will be you can see lot of redundancies coming in, because since I they can be in terms of all different combinations. So, S 1 has in name A may be taking course C 1 having name C, but again the S 1 having name A could be taking course C 2 having name B, you just do not know which one is correct.

So, you can see that here again you have 2 multiple value dependencies, one where SID multi determines CID and SID multi determines C name. So, these are the 2 different multiple values that you can, find against the SID and this is ah. So, if 2 or more multi valued dependencies exist in a relation, then while we convert the we convert multivalued attributes into single valued attributes, then the multi value dependency will show up. So, that is the basic problem that we would like to address.

(Refer Slide Time: 05:59)



Multivalued Dependencies

- Suppose we record names of children, and phone numbers for instructors:
 - inst_child*(ID, child_name)
 - inst_phone*(ID, phone_number)
- If we were to combine these schemas to get
 - inst_info*(ID, child_name, phone_number)
 - Example data:
 - (99999, David, 512-555-1234)
 - (99999, David, 512-555-4321)
 - (99999, William, 512-555-1234)
 - (99999, William, 512-555-4321)
- This relation is in BCNF
 - Why?

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

Database System Concepts - 6th Edition 16.8 ©Silberschatz, Korth and Sudarshan

This is another example of 2 relations, where the id and child are together, when id and phone number are together. So, naturally if I combine them into a single relation, you have a set of possibilities of multiple different tuples. Because given an id there could be multiple children, there given an id there could be multiple phone numbers. Mind you, this relation of isn't info is still in Boyce Codd normal form, because there is no dependence there is no functional dependency that holds on this relation. So, the key of this relation is the union of all the 3 attributes and therefore, that being the key and no functional dependency holding on it, naturally vacuously makes it Boyce Codd normal form, but you can still see that there are redundancy in that is data.

(Refer Slide Time: 06:48)

Multivalued Dependencies (MVDs)

Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The **multivalued dependency**

$$\alpha \twoheadrightarrow \beta$$

holds on R if in any legal relation $r(R)$, for all pairs for tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that:

$$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$$

$$t_3[\beta] = t_1[\beta]$$

$$t_4[\beta] = t_2[\beta]$$

Example: A relation of university courses, the books recommended for the course, and the lecturers who will be teaching the course:

- $\text{course} \twoheadrightarrow \text{book}$
- $\text{course} \twoheadrightarrow \text{lecturer}$

Course	Book	Lecturer	Tuples
AHA	Silberschatz	John D	t1
AHA	Nederpelt	William M	t2
AHA	Silberschatz	William M	t3
AHA	Nederpelt	John D	t4
AHA	Silberschatz	Christian G	
AHA	Nederpelt	Christian G	
OSO	Silberschatz	John D	
OSO	Silberschatz	William M	

So now, let us define multivalued dependency in a formal way and. So, we say that alpha multi determines beta, naturally alpha and beta both have to be subsets of the given set of attributes. When we say that? When there are for all pairs of tuples t 1 and t 2 such that they match on the fields of alpha, this till this point it looks like functional dependencies. There exists 2 more tuples t 3 and t 4 such that this condition hold, what are the conditions? Look, carefully here we say that all of them match on the alpha attributes which is fine, then you say that t 3 matches with t 1 in the beta attributes and t 3 matches on the remaining attributes with t 2. Similarly, t 4 matches with t 2 in the beta attributes and t 4 matches with t 1 on the remaining attributes.

So, let us look at an example, gets confusing. So, here is course book and lecturer ah. So, it is a relationship of university courses known naturally, every course has multiple recommended books and every course has been taken by multiple different lecturers from time to time. So, course can have multiple books. So, there is a multivalued dependency here, it can be taught by multiple lectures.

So, there is a multivalued dependency here and therefore, I can have an instance of this particular relation and I am just showing you, how to test for the multivalued dependency course multi determines book. So, these are the 2, 4 tuples I have marked t 1, t 2, t 3, t 4 if you look into the first condition. So, this is your alpha I am checking for. So, this is alpha this is beta. So, this is beta and this is ah. So, to say R minus beta minus alpha ok.

So, the first condition that all these tuples will have to match on alpha yes, they do, all 4 of them have AHA here. So, that is fine take at the second condition t 3 on beta is Silberschatz and t 1 on beta is also Silberschatz. So, they match and t 3 on the remaining attributes remaining attributes are, if I take out beta if I take out book it is AHA it is course and the lecturer that is remaining. Now it already matches on the course. So, I do not have to check for that, but. So, I can just check for whether it matches on lecturer, between some checking for this rule, whether t 3 and t 2 match yes t and t 2 match, they have the same name for the lecturer.

Look at the next one which is t 4 and t 2 match on beta, t 4 and t 2 match on beta yes, they have the same name of the book, and whether t 4 and t 1 match on the lecturer, this rule t 4 and t 1 match on the lecturer this rule. So, it also satisfies. So, I can say that this relation has holds the multivalued dependency course multi determining book. In a similar way you can you can mark your t 1, t 2, t 3, t 4 on this and check for course multi determining the lecturer, actually we will we will soon state that; if course multi determines book, then it is trivial that course will also multi determine lecturer.

(Refer Slide Time: 10:36)

Example

- Let R be a relation schema with a set of attributes that are partitioned into 3 nonempty subsets.
 Y, Z, W
- We say that $Y \twoheadrightarrow Z$ (Y **multidetermines** Z) if and only if for all possible relations $r(R)$
 $\langle y_1, z_1, w_1 \rangle \in r$ and $\langle y_1, z_2, w_2 \rangle \in r$
then
 $\langle y_1, z_1, w_2 \rangle \in r$ and $\langle y_1, z_2, w_1 \rangle \in r$
- Note that since the behavior of Z and W are identical it follows that
 $Y \twoheadrightarrow Z$ if $Y \twoheadrightarrow W$

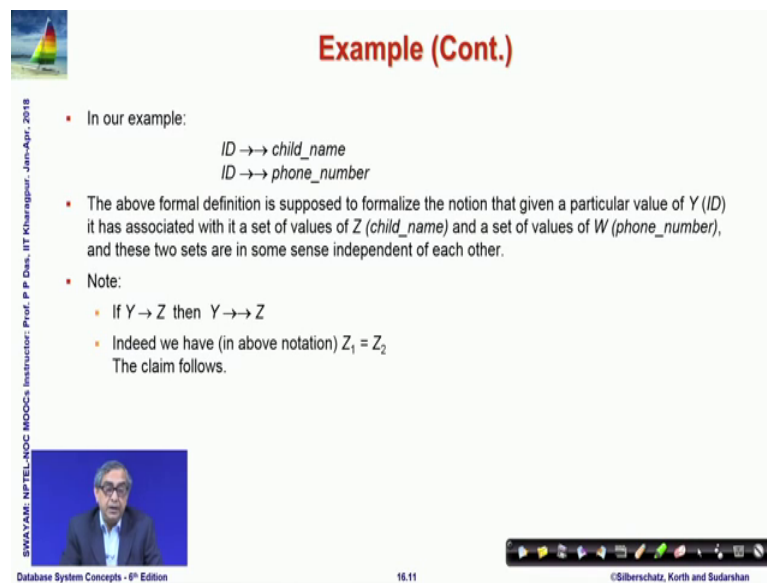
Database System Concepts - 6th Edition
16.10
©Silberschatz, Korth and Sudarshan

So, this is just to tell you if you have 3 non-empty sets of attributes Y, Z and W and then we say, Y multi determine Z , if and only if there are these are the possible relations. That I can have Y_1 and Z_1, W_1 in a relation and Y_1 and Z_2, W_2 in the relation, then I can

have Y_1, Z_1 with W_2 and Y_1, Z_2 is W_1 , that is you can basically take the cross of these to other 2 attributes and those are r tuples, possible tuples in your relation and ah.

So, you can you can naturally if you read it in little in a different way, then you can observe that since the behavior of Z and W are identical they are switchable. So, if Y multi determine Z , then you can you have Z Y multi determining W and vice versa. So, this is; what is a core observation in terms of the multi value dependencies?

(Refer Slide Time: 11:40)



Example (Cont.)

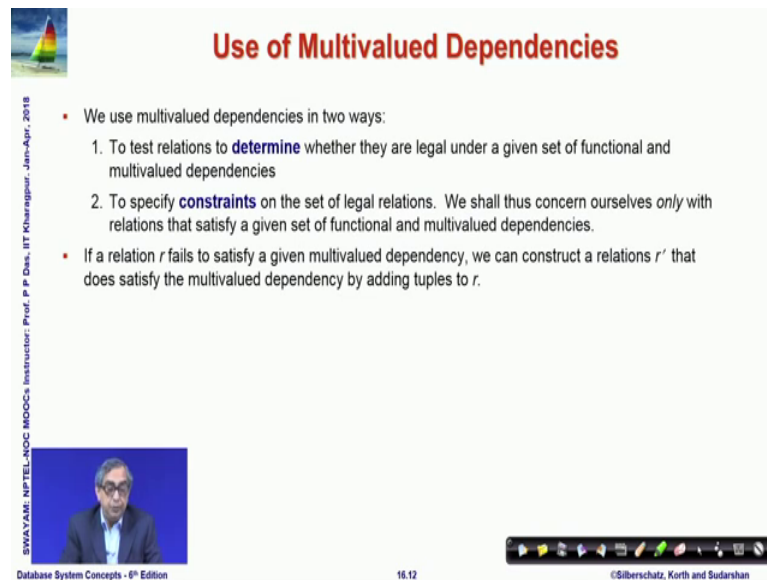
- In our example:
 $ID \twoheadrightarrow child_name$
 $ID \twoheadrightarrow phone_number$
- The above formal definition is supposed to formalize the notion that given a particular value of Y (ID) it has associated with it a set of values of Z ($child_name$) and a set of values of W ($phone_number$), and these two sets are in some sense independent of each other.
- Note:
 - If $Y \rightarrow Z$ then $Y \twoheadrightarrow Z$
 - Indeed we have (in above notation) $Z_1 = Z_2$
The claim follows.

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

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

So, this is in terms of our example, you can now clearly understand that id multi determines $child$ name and id multi determines $phone$ number, in the earlier example that we took and ah. So, we can also note that if there is a functional dependency, Y functionally determines Z then; obviously, Y will multi determine Z , that is that is just quite obvious.

(Refer Slide Time: 12:06)



Use of Multivalued Dependencies

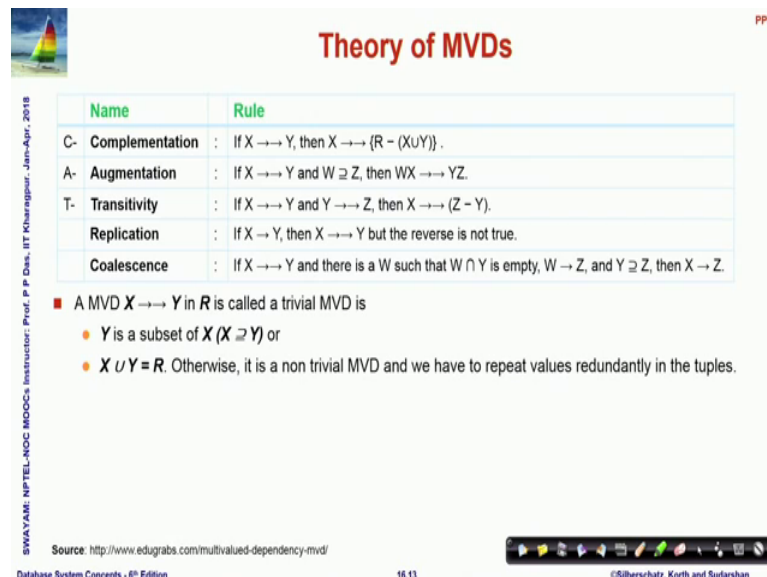
- We use multivalued dependencies in two ways:
 - To test relations to **determine** whether they are legal under a given set of functional and multivalued dependencies
 - To specify **constraints** on the set of legal relations. We shall thus concern ourselves *only* with relations that satisfy a given set of functional and multivalued dependencies.
- If a relation r fails to satisfy a given multivalued dependency, we can construct a relations r' that does satisfy the multivalued dependency by adding tuples to r .

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

Database System Concepts - 8th Edition 16.12 ©Silberschatz, Korth and Sudarshan

So, we have to we can make use of multi value dependency to specify, further constraints to remove redundancies and defining what is legal in a relation. And if a relation fails to satisfy a given multivalued dependency, then we can construct a relation r primed, that does satisfy the multi valued dependency by adding tuples to that r right?

(Refer Slide Time: 12:32)



Theory of MVDs

Name	Rule
C- Complementation	: If $X \twoheadrightarrow Y$, then $X \twoheadrightarrow \{R - (XY)\}$.
A- Augmentation	: If $X \twoheadrightarrow Y$ and $W \supseteq Z$, then $WX \twoheadrightarrow YZ$.
T- Transitivity	: If $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$, then $X \twoheadrightarrow (Z - Y)$.
Replication	: If $X \rightarrow Y$, then $X \twoheadrightarrow Y$ but the reverse is not true.
Coalescence	: If $X \twoheadrightarrow Y$ and there is a W such that $W \cap Y$ is empty, $W \rightarrow Z$, and $Y \supseteq Z$, then $X \rightarrow Z$.

- A MVD $X \twoheadrightarrow Y$ in R is called a trivial MVD is
 - Y is a subset of X ($X \supseteq Y$) or
 - $X \cup Y = R$. Otherwise, it is a non trivial MVD and we have to repeat values redundantly in the tuples.

Source: <http://www.edugrabs.com/multivalued-dependency-mvd/>

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

Database System Concepts - 8th Edition 16.13 ©Silberschatz, Korth and Sudarshan

Now, once having defined the notion of multi valued dependency, we next proceed to check, how do we reason about that. So, I would remind you about functional

dependencies, and the different rules of functional dependencies Armstrong's rules, that we had introduced the all of these of augmentation transitivity and all that.

So, in terms of functional dependencies we have 3 rules, commonly called the cat rules. Which purely involve the functional dependencies, first is a complementation which is a kind which we have just discussed shown, that if X multi determines Y , then X multi determines $R \text{ minus } X \text{ union } Y$ with multi determines the remaining set of attributes.

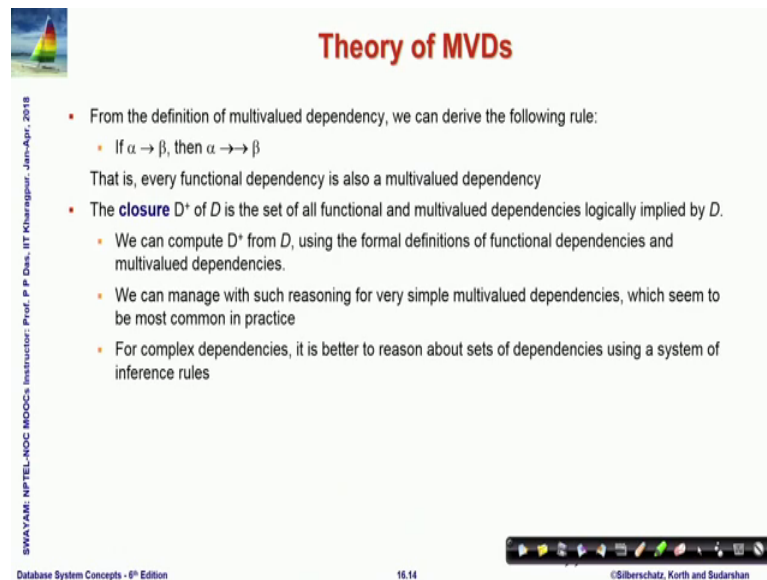
Augmentation that is I can augment any multivalued dependency with left and putting attributes on the left and right-hand side, as long as I put all attributes that I put on the right-hand side, I put them on the left-hand side. I may put more attributes on the left-hand side, but all attributes that I put on the right-hand side here Z must be a subset of the attributes that I put on the left-hand side, that augmentation is possible. Transitivity is manifesting in a little different way, if X multi determines Y and Y multi determines Z then, X multi determines $Z \text{ minus } Y$.

So, these are the these are the 3 rules which are basically these 3 are rules that, involve only multi valued dependencies and the other 2 rules, actually involve the relationship between multi value dependency the replication rule and the coalescence rule, which are between the multi value dependency and the functional dependency. We are not going deeper into that further, or trying to take specific examples and show how they work.

I just want you to know that such rules exist through which, you can define similar algorithms for multivalued dependency also, as we did for functional dependency like as you can understand the most critical algorithm to define would be the algorithm of closure, which can again be used in the situation where I have functional as well as multivalued dependency.

So, just we will keep that, in little bit advance space of this course. So, just know that such things exist, but we are not going into the details of that. Finally, for a multivalued dependency where, X determines Y we call that MVD to be trivial. If either y is a subset of X which is the notion we used for functional dependencies or there is a second condition here, that the union of the X and Y that left hand right hand side gives you the whole set of attributes, otherwise a it is a non-trivial multivalued dependency and we have to repeat the values. So, these are the 2 conditions, if they satisfy then we know that we have a trivial multi value dependency and we do not want to deal with that.

(Refer Slide Time: 15:40)



Theory of MVDs

- From the definition of multivalued dependency, we can derive the following rule:
 - If $\alpha \twoheadrightarrow \beta$, then $\alpha \twoheadrightarrow \beta$

That is, every functional dependency is also a multivalued dependency

- The **closure** D^* of D is the set of all functional and multivalued dependencies logically implied by D .
 - We can compute D^* from D , using the formal definitions of functional dependencies and multivalued dependencies.
 - We can manage with such reasoning for very simple multivalued dependencies, which seem to be most common in practice
 - For complex dependencies, it is better to reason about sets of dependencies using a system of inference rules

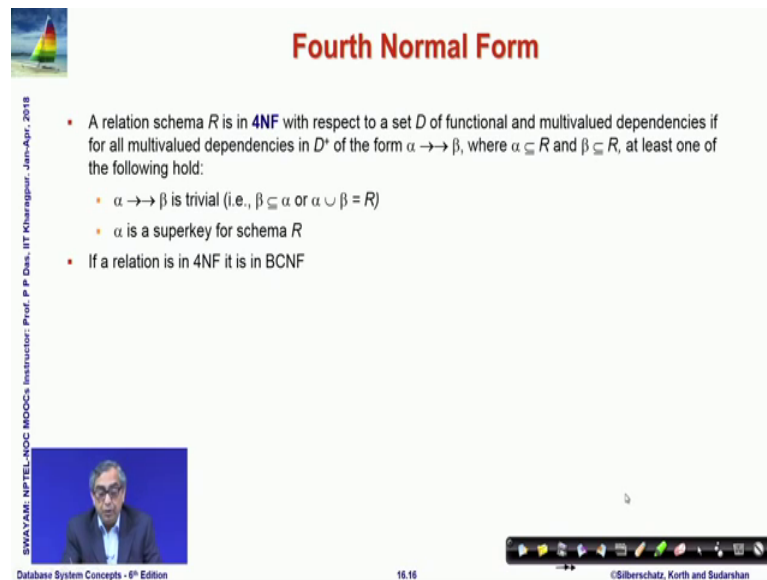
SWAYAM: NPTEL-NOC MDOCS Instructor: Prof. P. P. Das, IIT Khargpur, Jan-April, 2018

Database System Concepts - 8th Edition 16.14 ©Silberschatz, Korth and Sudarshan

So, there is little bit of references to the theory given here, I have mentioned that there are closure algorithms ah. So, that given a set of dependencies I will now generalize and set dependencies, which means that there could be functional dependencies, always lies multivalued dependencies. Even a set of dependencies you can define a closure of all of these functional and multivalued dependencies together, that are implied by the given set and we can have all those parallel definitions of closure of the dependencies, the minimal cover canonical cover and so on.

So, I just want you to note that these things have been defined and the existent theory, but will be beyond the current course that we are pursuing. So, it is now that we have is we have seen a another additional source of redundancy in our data, in terms of multiple values and in terms of the multi value dependency that hold.

(Refer Slide Time: 16:50)



Fourth Normal Form

- A relation schema R is in **4NF** with respect to a set D of functional and multivalued dependencies if for all multivalued dependencies in D^+ of the form $\alpha \twoheadrightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following hold:
 - $\alpha \twoheadrightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
 - α is a superkey for schema R
- If a relation is in 4NF it is in BCNF

SWAYAM: NPTEL-NOC MBOCS Instructor: Prof. P.P. Das, IIT Khargpur, Jan-April, 2018

Database System Concepts - 8th Edition

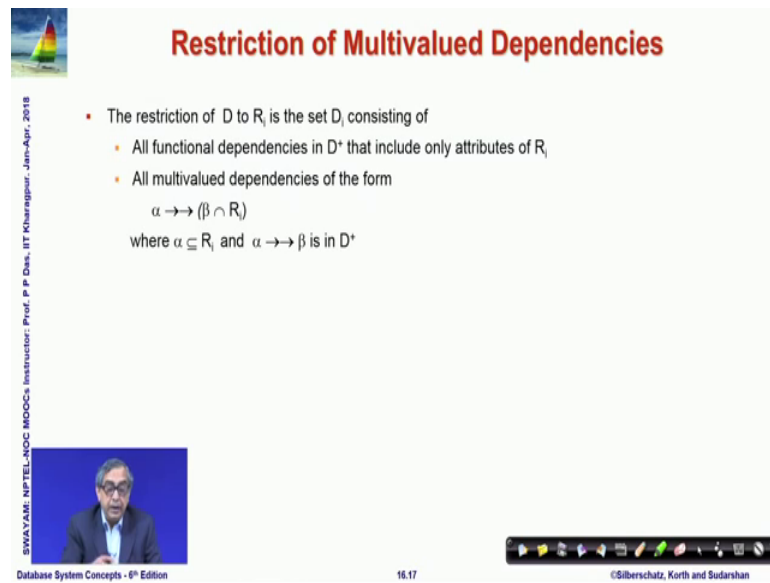
16.16

©Silberschatz, Korth and Sudarshan

So, we would now like to look into if such dependencies exist, then how do you decompose a relation to satisfy that the redundancy caused by such dependencies are not affecting us. So, such a normal form it is beyond the third normal form is called to be said to be a 4th normal form or 4 NF. Where you say that a relation is in 4 m NF if, every multi value dependency alpha multi determining beta, in the closure of the set of dependencies is either trivial, trivial means that left hand side is a subset of the right-hand side or the union of the left and right-hand side gives you the whole set of attributes.

So, it is either trivial every dependency is either trivial, or alpha left-hand side is a superset of the schema R, you can very well relate that this is just a little twist on the definition of the Boyce Codd normal form, where the second condition was identical and only thing in the first condition instead of MVD, you had a functional dependency. So, when we have this, we say we are a relation would be in the in the 4th normal form. Naturally, if a relation is in 4th normal form, it is trivial that it will be in the Boyce Codd normal form, but the reverse will not be necessarily true.

(Refer Slide Time: 18:19)



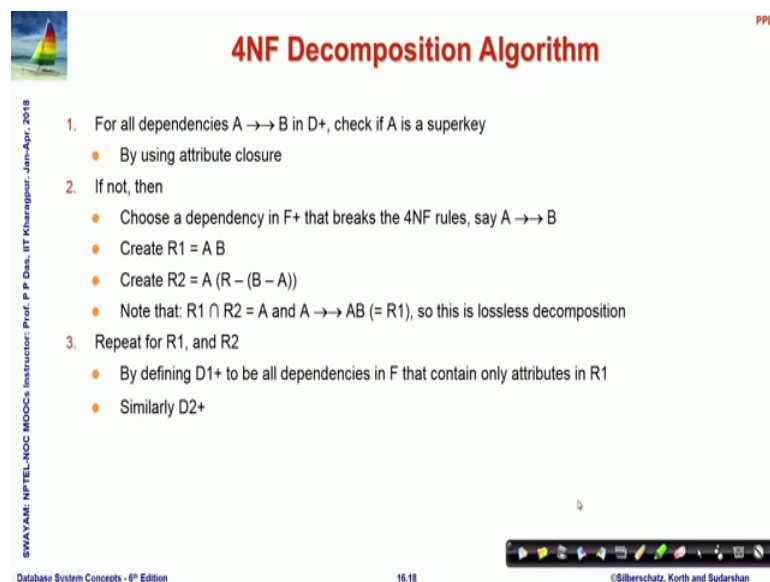
Restriction of Multivalued Dependencies

- The restriction of D to R_i is the set D_i consisting of
 - All functional dependencies in D^+ that include only attributes of R_i
 - All multivalued dependencies of the form $\alpha \twoheadrightarrow (\beta \cap R_i)$ where $\alpha \subseteq R_i$ and $\alpha \twoheadrightarrow \beta$ is in D^+

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

So, again the same set of concepts that, if I have a set of dependencies and you have a decomposed relation then smaller relation, then I can project that set of dependencies, in terms of a particular subset of the attributes and here is the condition that is given.

(Refer Slide Time: 18:43)



4NF Decomposition Algorithm

1. For all dependencies $A \twoheadrightarrow B$ in D^+ , check if A is a superkey
 - By using attribute closure
2. If not, then
 - Choose a dependency in F^+ that breaks the 4NF rules, say $A \twoheadrightarrow B$
 - Create $R_1 = A \text{ B}$
 - Create $R_2 = A (R - (B - A))$
 - Note that: $R_1 \cap R_2 = A$ and $A \twoheadrightarrow AB (= R_1)$, so this is lossless decomposition
3. Repeat for R_1 , and R_2
 - By defining D_{1+} to be all dependencies in F that contain only attributes in R_1
 - Similarly D_{2+}

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

So, the decomposition algorithm into 4NF is exactly like the decomposition algorithm of the Boyce Codd, normal form BC NF. Only difference being that, now you may be doing this crucial step of to a decomposition, for every multivalued dependency also earlier we were doing this only for the functional dependency.

So, now if there is any offending multivalued dependency, which is not satisfying the phone in a form? We can decompose the relation in terms of R 1 and R 2, as in here which is exactly like the Boyce Codd normal form and then the rest of it is simple. If ah if it is you know by this another important point, that you that you must note is in this process you actually guarantee lossless join.

So, this also continues to be in lossless join, with every decomposition and then you keep on repeating till all dependencies in f, in your set has been dealt with the attributes in R 1 and have converted them into the 4 NF form. So, you have a total 4 NF decomposition happening.

(Refer Slide Time: 20:04)

4NF Decomposition Algorithm

```
result := {R};
done := false;
compute D*;
Let Di denote the restriction of D* to Ri
while (not done)
  if (there is a schema Ri in result that is not in 4NF) then
    begin
      let α → β be a nontrivial multivalued dependency that holds
      on Ri such that α → Ri is not in Di, and α ∩ β = ∅;
      result := (result - Ri) ∪ (Ri - β) ∪ (α, β);
    end
  else done := true;
```

Note: each R_i is in 4NF, and decomposition is lossless-join

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

(Refer Slide Time: 20:13)

Example of 4NF Decomposition

Example:

- Person_Modify(Man(M), Phones(P), Dog_Likes(D), Address(A))
- FDs:
 - FD1 : Man \twoheadrightarrow Phones
 - FD2 : Man \twoheadrightarrow Dogs_Like
 - FD3 : Man \rightarrow Address
- Key = MPD
- All dependencies violate 4NF

Post Normalization

Person_Phones

Man | Phones

FD1

SuperKey : Man,Phones

Person_DogsLikes

Man | Dogs_Likes

FD2

SuperKey : Man,Dog-Likes

Person_Address

Man | Address

FD3

Key : Man

Man(M)	Phones(P)	Dogs_Likes(D)	Address(A)
M1	P1	D1	49-ABC,Bhiwani(HR.)
M1	P2	D2	49-ABC,Bhiwani(HR.)
M2	P3	D2	36-XYZ,Rohtak(HR.)
M1	P1	D2	49-ABC,Bhiwani(HR.)
M1	P2	D1	49-ABC,Bhiwani(HR.)

In the above relations for both the MVD's - 'X' is Man, which is again not the super key, but as $X \cup Y = R$ i.e. (Man & Phones) together make the relation. So, the above MVD's are trivial and in FD 3, Address is functionally dependent on Man, where Man is the key in Person_Address, hence all the three relations are in 4NF.

Database System Concepts - 6th Edition 18.20 ©Silberschatz, Korth and Sudarshan

Ah let us take a this here, is the like before here is a formal algorithm for those who would be interested, to formally study the steps. Ah here I am just showing examples of 4 NF decomposition. So, we started this discussion with a person relational scheme, having man phone and dog likes MPD, I have added I have just modified and I have added another attribute address. So, that in addition to the multi value dependencies, I can also have a functional dependency. So, we have 2 multivalued dependencies like before, man multi determining phones and man multi determining dog like, but now we have a functional dependency man determining address the key continues to be MPD.

So, all of these dependencies will violate the 4 NF, because none of them satisfy the either of the condition, that none of them are trivial and on for none of them left hand side is a super key because a key is MPD. So, you can see that in on instances of this, relational schema you will have multiple redundant records, in the actual instance. So, on the right we normalize we normalize by taking f D 1; take the union of man and phones that gives you the first relation and then the rest of it. Then again you split based on f D 2, you have the second relation in the decomposition man and dog like and the third one gets generated as a byproduct of that, which is man and address.

And you have 3 relations now, which together represent the original relation each one of them is in 4th normal form. Actually, what happens is, when you when you have decomposed then, f D 1 in this has become a relation where, the multivalued dependency man multi determining phones can be checked in terms of a functional dependency itself, and that that is what gives you the multi value dependency. And since it is multivalued

so, man and phones together continues to form the key, similarly in the second one the man and dog like is the key. Because you just have the multivalued dependency and given the same man, you will have multiple dogs whom he or she likes, but in the third one in the person address where you have man and address you have only man as the key, because man is a functional dependency that holds.

(Refer Slide Time: 23:01)

Example of 4NF Decomposition

- $R = (A, B, C, G, H, I)$
- $F = \{ A \twoheadrightarrow B, B \twoheadrightarrow HI, CG \twoheadrightarrow H \}$
- R is not in 4NF since $A \twoheadrightarrow B$ and A is not a superkey for R
- Decomposition
 - a) $R_1 = (A, B)$ (R_1 is in 4NF)
 - b) $R_2 = (A, C, G, H, I)$ (R_2 is not in 4NF, decompose into R_3 and R_4)
 - c) $R_3 = (C, G, H)$ (R_3 is in 4NF)
 - d) $R_4 = (A, C, G, I)$ (R_4 is not in 4NF, decompose into R_5 and R_6)
 - $A \twoheadrightarrow B$ and $B \twoheadrightarrow HI \rightarrow A \twoheadrightarrow HI$, (MVD transitivity), and
 - and hence $A \twoheadrightarrow I$ (MVD restriction to R_4)
 - e) $R_5 = (A, I)$ (R_5 is in 4NF)
 - f) $R_6 = (A, C, G)$ (R_6 is in 4NF)

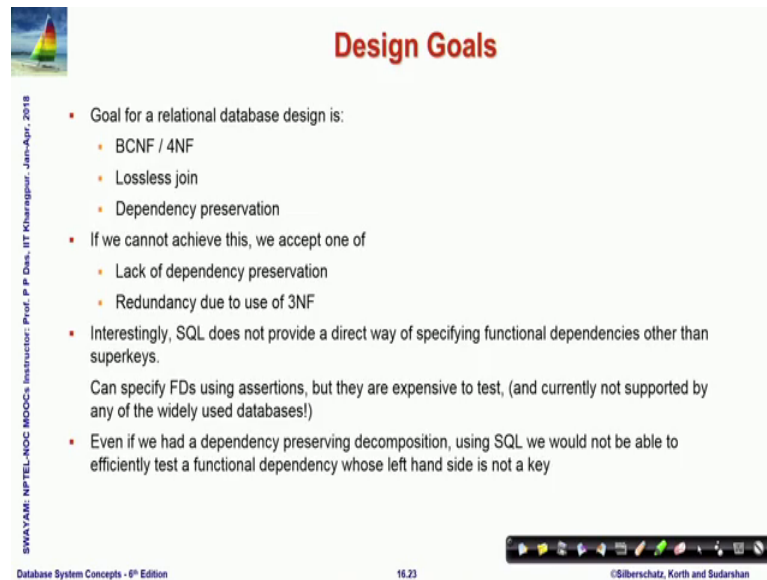
SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018
Database System Concepts - 6th Edition 16.21 ©Silberschatz, Korth and Sudarshan

So, this is a simple illustration of decomposition into 4 NF, here is a little more elaborate one, again this is a we have I have worked through the steps. So, there are 3 multivalued dependencies and you can see that, A multi determining B is not does not is not who does not hold the condition of 4 NF. So, you have to decompose, you decompose get R 1 which is in 4 NF and the remaining R 2 which is also not in 4 NF. So, decompose in R 3 which is in 4 NF and R 4, we can R 4 is not in 4 NF you decomposER 4 into R 5 and R 6 and work through that, and you will be able to see that R 5 is in 4 NF and R 6 also is in 4 NF, which gives a complete multivalued decomposition of this whole set.

Naturally with that, we will conclude our discussion on the decomposition process, there are there would be some more aspects to look at and there is lot of more normal forms that exist. But this is for all practical purposes; a database is normalized, when it is represented in terms of the third normal form. And I have discussed still I have discussed the 4th normal form, because in some places people prefer to represent also in 4th normal form.

So, that they guarantee that they have even less redundancy in the data, but leaving that, let us quickly take a round in terms of the what we have done so far and what is a basic overall design process that we should be following.

(Refer Slide Time: 24:43)



Design Goals

- Goal for a relational database design is:
 - BCNF / 4NF
 - Lossless join
 - Dependency preservation
- If we cannot achieve this, we accept one of
 - Lack of dependency preservation
 - Redundancy due to use of 3NF
- Interestingly, SQL does not provide a direct way of specifying functional dependencies other than superkeys.
Can specify FDs using assertions, but they are expensive to test, (and currently not supported by any of the widely used databases!)
- Even if we had a dependency preserving decomposition, using SQL we would not be able to efficiently test a functional dependency whose left hand side is not a key

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

Database System Concepts - 9th Edition 16.23 ©Silberschatz, Korth and Sudarshan

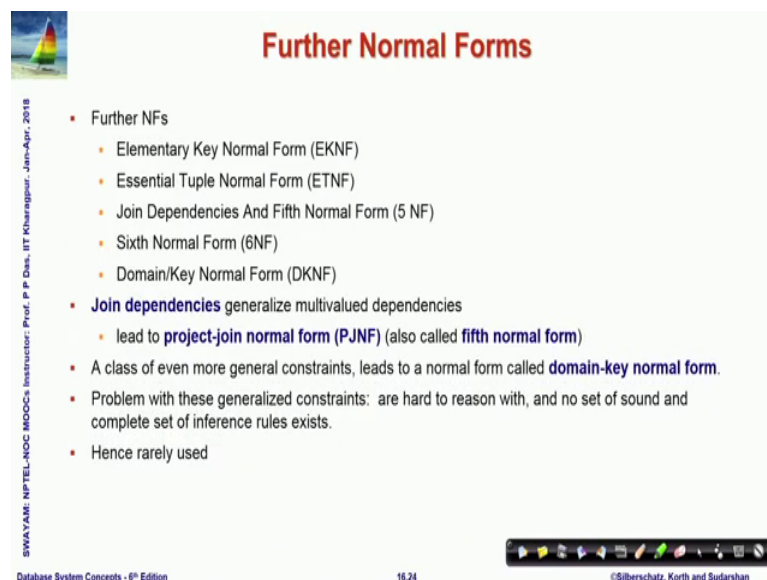
So, again to remind you the goal for our design is to have a relational database which is in BC NF or 3 NF has a lossless join due to the decomposition, and dependency preservation. If we cannot achieve that, I am I am sorry earlier what I meant is BC NF or 4 NF not BC NF and 3 NF. So, the idea would be I have a decomposition in BC NF and 4 NF lossless join and dependency preservation which may not be achievable. If I cannot achieve that then I go I have to sacrifice either, the lack of dependency preservation. So, dependencies will have to be checked using natural joint or, I will allow a little bit of redundancy and use the third normal form where I have the guarantee.

Now, at this point you must wonder and note that SQL, the language in which we are doing the creation and update and the query processing. That SQL does not provide any directory of specifying or checking any dependency, other than the functional other than the functional dependency that checks the super key. Super key is the only functional dependency that SQL would check, no other functional dependency or multivalued dependency and other type dependencies are can be specified or checked in SQL. You can do that using assertions, in the in the while discussing SQL I were talked about assertions we can do that using assertions, but that too is very expensive to test. So, it is

not usually supported by any of the databases, which are widely used because that slows down your every process very, very much.

So, you can understand that in terms of your design goals, you have to do a very good job to make sure that, your functional and multivalued dependencies are accurately expressed in the design and accordingly the schemas are normalized in the proper ways satisfying BC NF for 4 NF or 3 NF normal forms. But because, while you will actually have instances there will not be a practical way, to see if you are violating any one or more of these ah rules of dependencies that you have set.

(Refer Slide Time: 27:09)



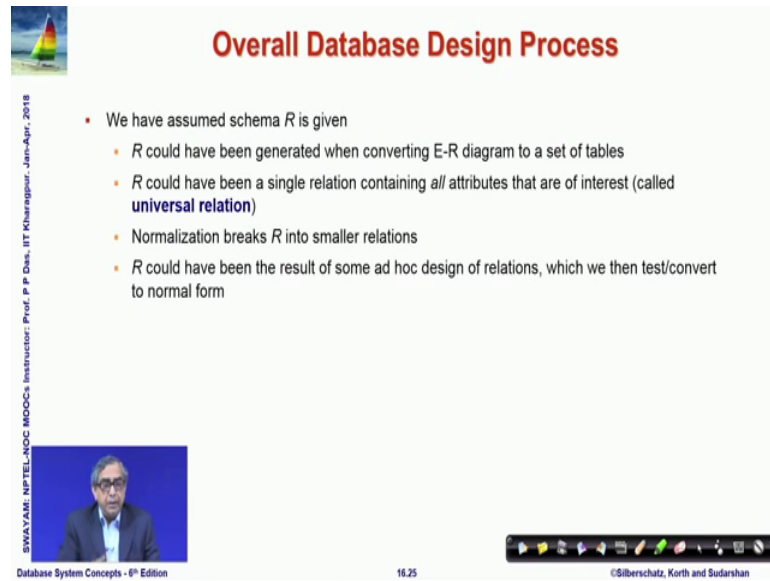
The slide, titled "Further Normal Forms", lists several advanced normal forms. It includes a small image of a sailboat in the top left corner. The text on the slide is as follows:

- Further NFs
 - Elementary Key Normal Form (EKNF)
 - Essential Tuple Normal Form (ETNF)
 - Join Dependencies And Fifth Normal Form (5 NF)
 - Sixth Normal Form (6NF)
 - Domain/Key Normal Form (DKNF)
- **Join dependencies** generalize multivalued dependencies
 - lead to **project-join normal form (PJNF)** (also called **fifth normal form**)
- A class of even more general constraints, leads to a normal form called **domain-key normal form**.
- Problem with these generalized constraints: are hard to reason with, and no set of sound and complete set of inference rules exists.
- Hence rarely used

At the bottom of the slide, there is a footer with the text "Database System Concepts - 9th Edition" on the left, "16.24" in the center, and "©Silberschatz, Korth and Sudarshan" on the right. A vertical text on the left edge reads "SWAYAM: NPTEL-NOC MOCs Instructor: Prof. P. Das, IIT Kharagpur, Jan-Apr, 2018".

So, as I mentioned there are actually these are not the only forms, there are various other normal forms as well and fifth normal form 6 normal form and so on, but it is very rarely these are very rarely used. It is not easy to decompose into these normal forms and by this decomposition does not give you enough returns in terms of the reduction of redundancy and removal of anomalies, that people often would have motivation to do them, but you should know that such normal forms exist.

(Refer Slide Time: 27:44)



Overall Database Design Process

- We have assumed schema R is given
 - R could have been generated when converting E-R diagram to a set of tables
 - R could have been a single relation containing *all* attributes that are of interest (called **universal relation**)
 - Normalization breaks R into smaller relations
 - R could have been the result of some ad hoc design of relations, which we then test/convert to normal form

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Khargpur, Jan-April, 2018

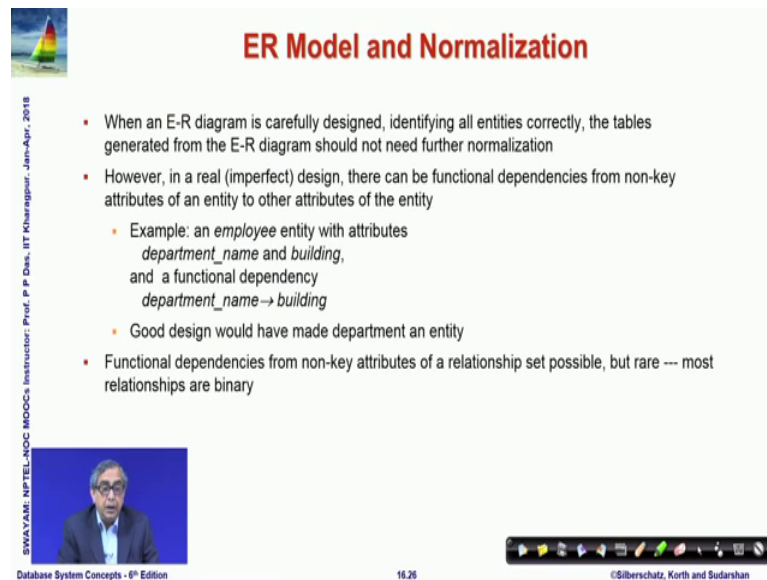
Database System Concepts - 8th Edition

16.25

©Silberschatz, Korth and Sudarshan

So, in the overall process if we look, at I mean what we have been doing is there are several tracks that we could be taking one possible thing is, the whole set of attributes have been generated while we have converted or relation has been generated. When you have converted the entity relationship diagram, the UML or the ER diagram into a set of tables, that is how we got our set of attributes or the relational schema R , it is also possible that we just started with a single relation containing all attributes, which is called the universal relation? And then normalization will break them into smaller relations. It could have been or could have been the result of some adds of design of relations also, and then you convert them.

(Refer Slide Time: 28:33)



ER Model and Normalization

- When an E-R diagram is carefully designed, identifying all entities correctly, the tables generated from the E-R diagram should not need further normalization
- However, in a real (imperfect) design, there can be functional dependencies from non-key attributes of an entity to other attributes of the entity
 - Example: an *employee* entity with attributes *department_name* and *building*, and a functional dependency $department_name \rightarrow building$
 - Good design would have made department an entity
- Functional dependencies from non-key attributes of a relationship set possible, but rare --- most relationships are binary

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

Database System Concepts - 9th Edition

16.26

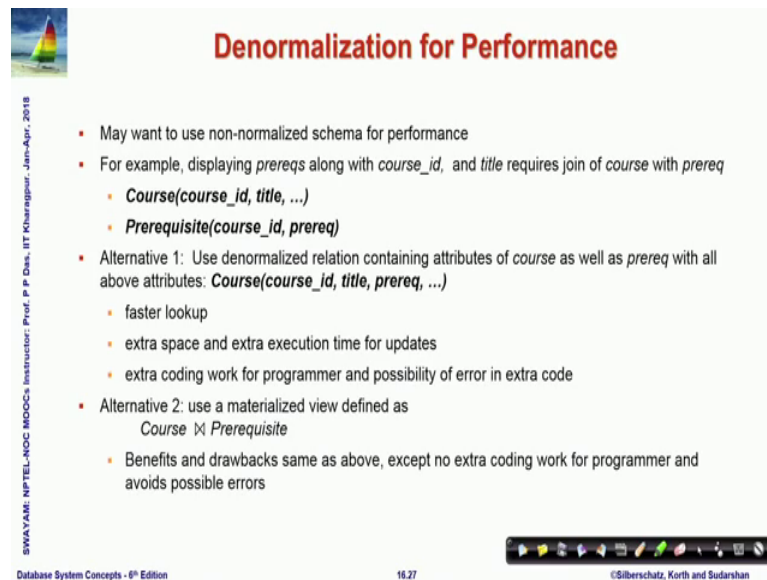
©Silberschatz, Korth and Sudarshan

So, there are possible all different possible tracks that can happen. So, if we have taken the ER model track, then frankly speaking if the ER model is carefully designed, then every entity defined in that ER model will have only the dependency; which are the determining super key.

So, just recall the employee department building kind of situation we discussed earlier. So, an employee entity has attributes department name and building, and there is a functional dependency from department name 2 building. So, what it means that in the entity relationship diagram itself we didn't do a good job. If we had done a good job then we would have identified that the department itself is an entity and therefore, would not feature as an attribute on the employee. So, it would have been I mean right there, we would have if we had called it as a separate entity, then that is equivalent of what we are doing now taking the relation and then breaking it down through decomposition.

So, functional dependencies from non-key attributes of a relationship are possible ah, but are rare. So, mostly the relationships are binary, and if you do a careful design of the ER model then many of these deep exercise of normalization you will not have to go through.

(Refer Slide Time: 29:55)



Denormalization for Performance

- May want to use non-normalized schema for performance
- For example, displaying *prereqs* along with *course_id*, and *title* requires join of *course* with *prereq*
 - **Course(course_id, title, ...)**
 - **Prerequisite(course_id, prereq)**
- Alternative 1: Use denormalized relation containing attributes of *course* as well as *prereq* with all above attributes: **Course(course_id, title, prereq, ...)**
 - faster lookup
 - extra space and extra execution time for updates
 - extra coding work for programmer and possibility of error in extra code
- Alternative 2: use a materialized view defined as
Course ⋈ *Prerequisite*
 - Benefits and drawbacks same as above, except no extra coding work for programmer and avoids possible errors

SWAYAM: NPTEL-NOC MDC03 Instructor: Prof. P. P. Das, IIT Khargpur, Jan-Apr, 2018

Database System Concepts - 8th Edition 16.27 ©Silberschatz, Korth and Sudarshan

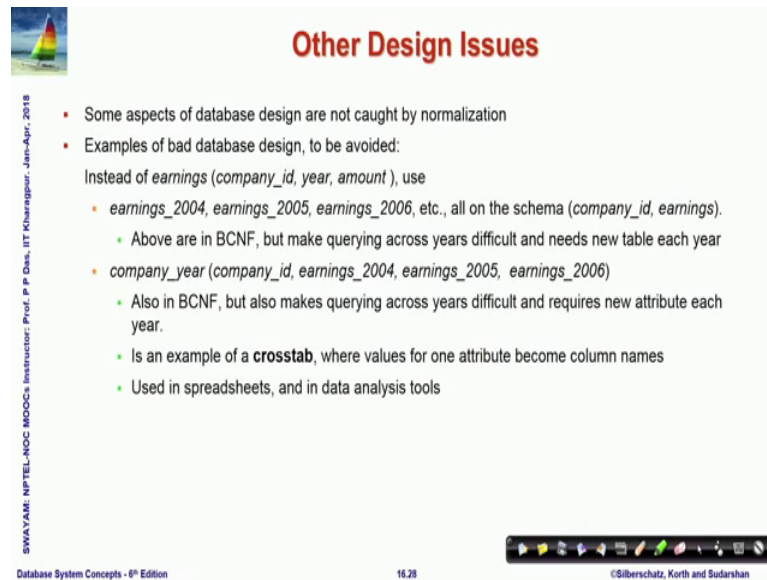
It should also be kept in your view, that at their times when you want to de normalize want to use denormalized relations, because if you have normalized and the only way to get back the original view is to perform join. So, if we of course, if you have prerequisites and if you want to say, view or print prerequisites with that title and course id naturally you will have to take a join with the course, which is expensive.

So, one option could be first alternative could be that, you use a de normalized relation, where the course prerequisite is actually included in the course and you know that will have you know violations of some of the normal forms. Because, there are there are functional dependencies between them, but that will certainly lead you to first a look up, because you have them in the same table you do not need to perform join. But you need extra space exact extra execution time for update, because you have redundant data you have redundancy while programming on that coding on that, because of this redundancy there could be possibility of error, because any of these anomalies can happen and your code will have to now take care of that.

So, it does help in certain way in terms of getting a better efficiency, but it there is a there is a cost to pay in a different way also the other alternative could be you can have a materialized view, which is actually the joint and course of prerequisite. In terms of performance it has a same benefit or the costs as you say, but only thing is you will not need to do that extra coding. So, it is better from that perspective.

So, always keep the issue of de normalization in view, and we do a careful design that if it is very frequent that, you will have to compute a join then, you might want to sacrifice some of the redundancy some of the you know possibilities of having anomaly, and still have a you know de normalized design in your database.

(Refer Slide Time: 31:59)



Other Design Issues

- Some aspects of database design are not caught by normalization
- Examples of bad database design, to be avoided:
 - Instead of *earnings* (*company_id*, *year*, *amount*), use
 - earnings_2004*, *earnings_2005*, *earnings_2006*, etc., all on the schema (*company_id*, *earnings*).
 - Above are in BCNF, but make querying across years difficult and needs new table each year
 - company_year* (*company_id*, *earnings_2004*, *earnings_2005*, *earnings_2006*)
 - Also in BCNF, but also makes querying across years difficult and requires new attribute each year.
 - Is an example of a **crosstab**, where values for one attribute become column names
 - Used in spreadsheets, and in data analysis tools

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

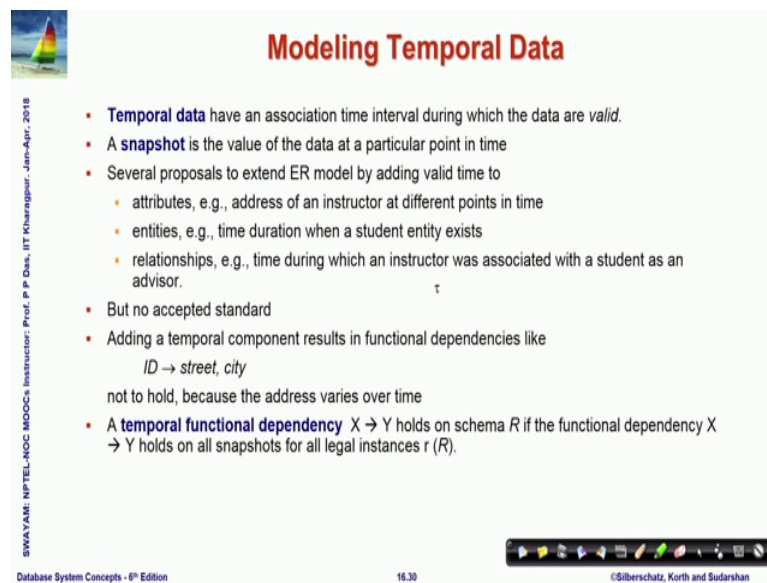
There are several other issues of design, which do not get captured in what we have designed. For example, let say very regularly we are we have returns, income tax returns to submit and we will be maintain income tax return tax your sales tax return and on all that, and you maintain your accounts book of transactions debit credit accounted and so on. Now naturally, these are all bound in terms of one-year effectivity.

So, when they come when in the next year comes, then you need a separate you know set of records to be done for that year. So, how do you. So, if you if you have such a table where you along with the company idea of year and amount and then how do you take care of this situation, because one way could be that you have all of these you take out year, from the attribute and you have separate table in every year. So, you will have to create new table and remember their name. So, if queries which run across year will be difficult to do.

The other way could be that, you every New Year you start renaming you know you do a year where your earnings from different years are shown on different columns. So, you are basically every year you have the result in terms of a different attribute. So, that also

is not a very good solution for a database it is something which is with the spreadsheets will often use, but in terms of data which has a certain format and needs to be you know redefined from scratch, at a at a different time frame in a different way, then you will come across these issues.

(Refer Slide Time: 33:56)



Modeling Temporal Data

- **Temporal data** have an association time interval during which the data are *valid*.
- A **snapshot** is the value of the data at a particular point in time
- Several proposals to extend ER model by adding valid time to
 - attributes, e.g., address of an instructor at different points in time
 - entities, e.g., time duration when a student entity exists
 - relationships, e.g., time during which an instructor was associated with a student as an advisor.
- But no accepted standard
- Adding a temporal component results in functional dependencies like
$$ID \rightarrow street, city$$
not to hold, because the address varies over time
- A **temporal functional dependency** $X \rightarrow Y$ holds on schema R if the functional dependency $X \rightarrow Y$ holds on all snapshots for all legal instances $r(R)$.

Database System Concepts - 6th Edition 16.30 ©Silberschatz, Korth and Sudarshan

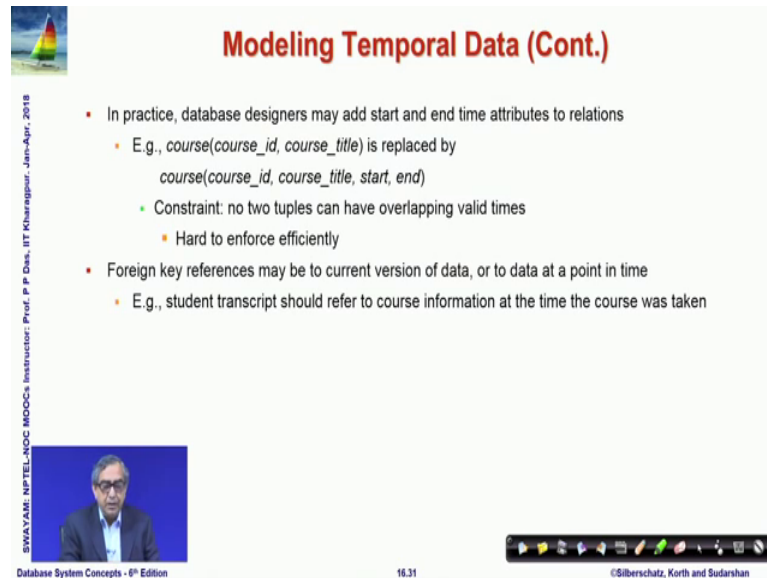
Ah let me close with just pointing out that, if we have a one kind of data that we have not looked at which are temporally in nature, that is all that we have said is the attributes and their values. So, if we put at value to an attribute then that value is taken to be the truth for now, and for the past and for the features. So, if that value changes, then you completely erase that in the database.

So, for example, today I stay at a certain address, tomorrow I may take up a different quarter my address has changed. So, in our design if there is against my employee id there is an address given, then once I change my quarter my address will change it will not be possible to recollect, what address I resided in say 2017.

So, temporal data of such kind, temporal data I am sorry just of such kind have an association with an interval. So, a snapshot often does not solve the problem. So, you have you have to decide, how you do that? Whether you can you would like to put some attributes, which specify the timestamp or you would like to I mean really have multivalued attributes, denoting the different time frames where they are they may have taken effect, there is no accepted standard. And the fact that, if you if you know that it

keeps on changing with time then your original dependencies might get affected they will they will change as well. So, these are the things that you will have to take care ah.

(Refer Slide Time: 35:30)



Modeling Temporal Data (Cont.)

- In practice, database designers may add start and end time attributes to relations
 - E.g., *course(course_id, course_title)* is replaced by *course(course_id, course_title, start, end)*
 - Constraint: no two tuples can have overlapping valid times
 - Hard to enforce efficiently
 - Foreign key references may be to current version of data, or to data at a point in time
 - E.g., student transcript should refer to course information at the time the course was taken

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

Database System Concepts - 9th Edition 16.31 ©Silberschatz, Korth and Sudarshan

This is another style, that many a times when you have to say that ok. This course with this title existed from this semester, a different semester the title may have changed. So, you can put a start and end attribute with which specifies what is the time for which the remaining attributes made sense a good design, but these also have issues because, if you do this kind of temporal intervals, then how do you make sure that between 2 records the intervals are not overlapped. So, you are not saying that at the same time this course had them X this of course, also had name Y. So, they have to be disjoint. So, how do you check for this ah this consistency of data, there is no easy way to do that.

So, the foreign key references and all those. So, handling of temporal data is another aspect which will have to be looked into very carefully, in the design and you will need to do some kind of design compromise and implementation has to take care of those issues.

(Refer Slide Time: 36:33)

Module Summary

- Understood multi-valued dependencies to handle attributes that can have multiple values
- Learnt Fourth Normal Form and decomposition to 4NF
- Discussed aspects of the database design process
- Studied the issues with temporal data

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

Database System Concepts - 8th Edition

16.32

©Silberschatz, Korth and Sudarshan

So, to summarize we have ah taken a full look into the multivalued dependencies and tried to understand what happens, when your attributes get multiple values. Learn the 4th normal form for that and the decomposition into that, and most importantly we have tried to summarize the core database design process. That we have been discussing for the last 4 modules, this is the 5th one including this and we have understood that and we have talked a little bit about the temporal data.

And with this we close our discussions on the relational database design, and from the next module we will move on to other aspects of the database systems.