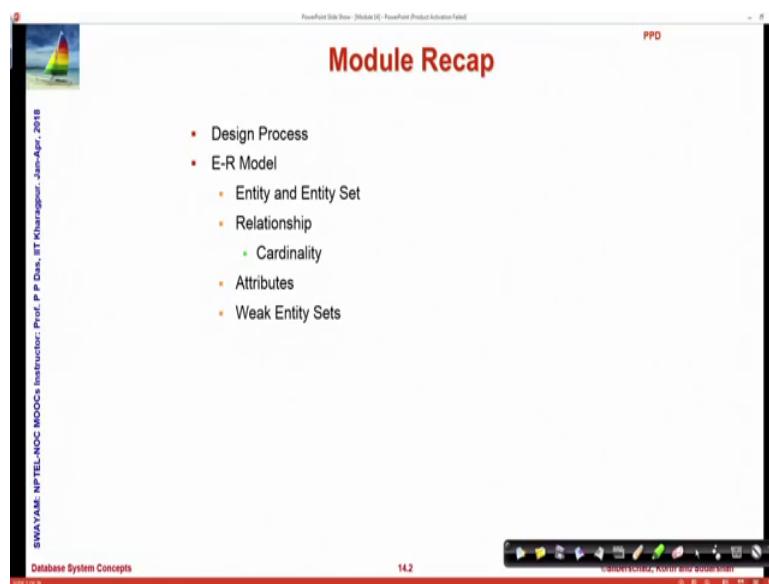


**Database Management System**  
**Prof. Partha Pratim Das**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 14**  
**Entity-Relationship Model/2**

Welcome to module 14 of database management systems. In the last module we will started our discussions on the entity relationship model.

(Refer Slide Time: 00:29)



The screenshot shows a PowerPoint slide with the title 'Module Recap' in red. The slide content is a bulleted list of topics:

- Design Process
- E-R Model
  - Entity and Entity Set
  - Relationship
    - Cardinality
  - Attributes
  - Weak Entity Sets

At the bottom left, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018'. At the bottom right, it says 'Database System Concepts' and '14.2'.

We will continue that in this module as well, and actually conclude it in the next module. So, these are the items that we had discussed in the last module.

(Refer Slide Time: 00:39)

The screenshot shows a PowerPoint slide with the title 'Module Objectives' in red at the top center. Below the title is a bulleted list of objectives:

- To illustrate E-R Diagram notation for E-R Models
- To explore translation of E-R Models to Relational Schemas

On the left side of the slide, there is vertical text: 'SWAYAM-NPTEL-MOOCs Instructor: Prof. P. P. Das, ST Kharegpur - Jan-Apr- 2018' and 'Database System Concepts'. At the bottom right, there is a video player showing a man speaking, with the number '143' next to it.

In the present one, we will first illustrate the entity relationship diagram notation; that graphical notation for E-R model, how nicely this can be shown in terms of certain diagrams. And then we will explore how E-R models can be translated to relational schemas, which is a basic step of the logical design.

(Refer Slide Time: 01:11)

The screenshot shows a PowerPoint slide with the title 'Module Outline' in red at the top center. Below the title is a bulleted list of topics:

- E-R Diagram
- E-R Model to Relational Schema

On the left side of the slide, there is vertical text: 'SWAYAM-NPTEL-MOOCs Instructor: Prof. P. P. Das, ST Kharegpur - Jan-Apr- 2018' and 'Database System Concepts'. At the bottom right, there is a video player showing a man speaking, with the number '144' next to it.

So, these are the topics. So, we start with the E-R diagram.

(Refer Slide Time: 01:16)

The slide is titled "Entity Sets" in red. It contains a bulleted list: "■ Entities can be represented graphically as follows:" followed by three points: "▪ Rectangles represent entity sets.", "▪ Attributes listed inside entity rectangle", and "▪ Underline indicates primary key attributes". Below the list are two rectangles representing entity sets. The first rectangle is labeled "instructor" at the top, with attributes "ID", "name", and "salary" listed below. The second rectangle is labeled "student" at the top, with attributes "ID", "name", and "tot\_cred" listed below. A video player interface is visible at the bottom right, showing a man speaking.

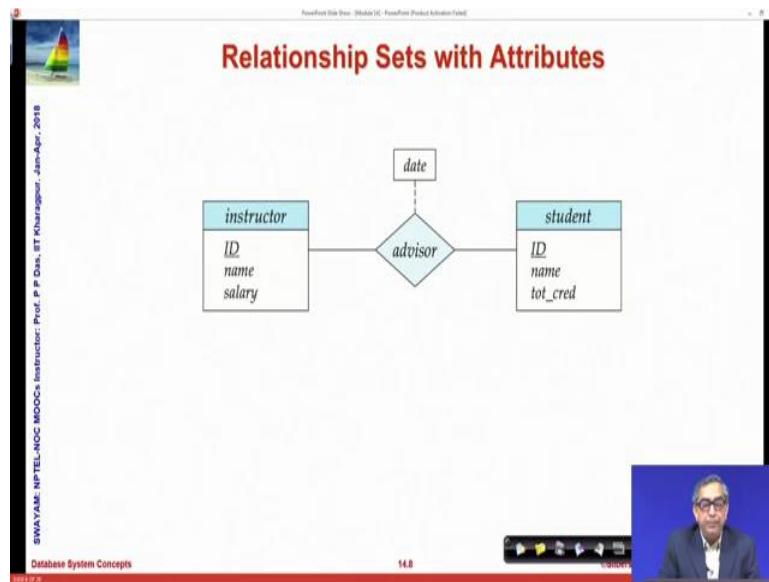
Naturally the first thing to represent in an E-R model is the entity set. Every entity set is represented by a rectangle. On the top, we write the name of that entity set so you can see examples here the instructor. And student are the two entity sets and below that we write the names of the attributes that are involved. And we underline the attribute or attributes that form the primary key of that entity set.

(Refer Slide Time: 01:49)

The slide is titled "Relationship Sets" in red. It contains a bulleted list: "■ Diamonds represent relationship sets." Below the list is a diagram showing a relationship between two entity sets. Two rectangles are connected by a diamond-shaped connector. The left rectangle is labeled "instructor" with attributes "ID", "name", and "salary". The right rectangle is labeled "student" with attributes "ID", "name", and "tot\_cred". The diamond connector is labeled "advisor". A video player interface is visible at the bottom right, showing a man speaking.

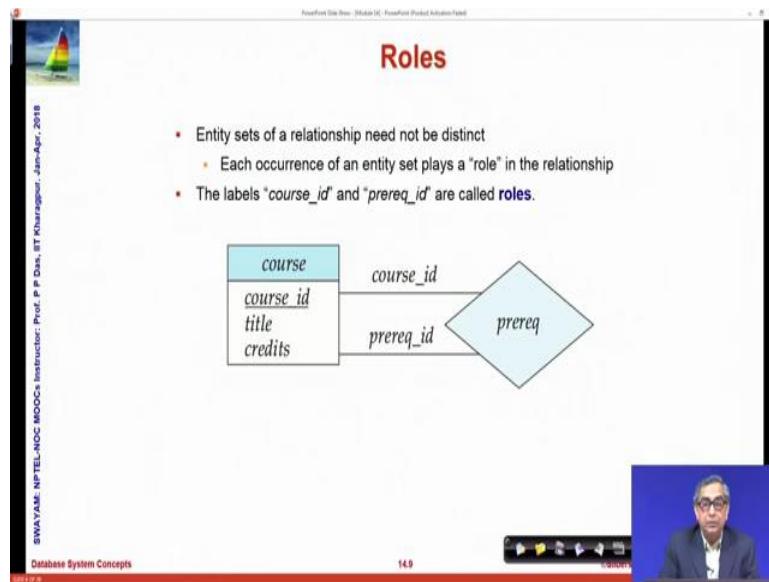
A relationship between two entity sets is represented by a diamond, and 2 connecting lines to the 2 entity sets. So, here it says that advisor is a relationship between entity set instructor, and entity set student. Trying to convey the real-world situation that students are advised by the instructors or students have instructors and so on.

(Refer Slide Time: 02:23)



As we had mentioned that relationships could also have attributes. So, if the advisor relationship has an attribute date then it will be tagged to the adviser relationship with the attribute coming as a within a rectangle and attached to the name of the relationship by a dotted line. So, this shows that advisor is the relationship between instructor. And student and the adviser relationship has an attribute date.

(Refer Slide Time: 02:58)



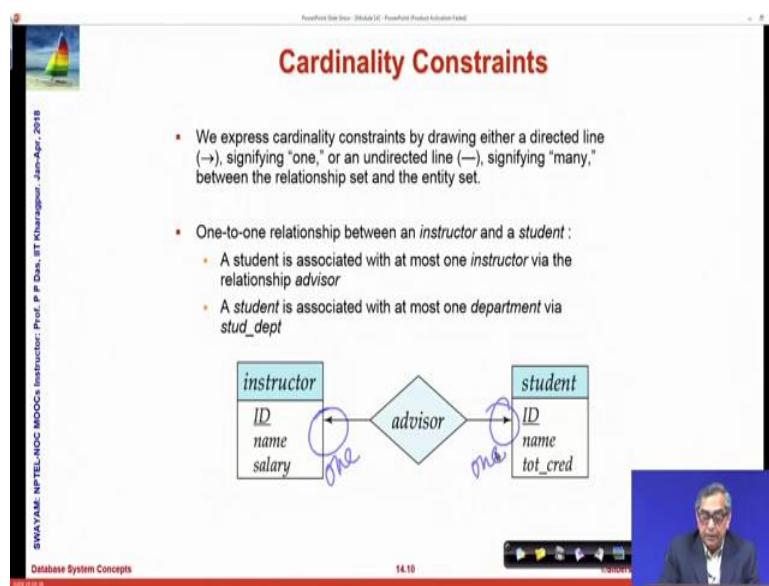
It is possible that the relationship that hold between 2 entity sets can be can use entity sets which are same. That is, it is possible that a set is related to itself.

So, as an example we show, the entity set course which has a relationship prereq, prerequisite which takes a course id, and relates it to another course id called the prerequisite id. Because obviously, if a course has a prerequisite. Then that prerequisite itself is another course id which must occur in this table itself.

So, in this case, if you can see that unlike the earlier case the prereq id is not actually a field of this relation course. So, we say these are rules. So, we say the rule that prereq relate from the course relation to itself are course id and prereq id; where in the actual table both of them relate to course id, but prereq will pair them to show which course has what prerequisite. And we often need this kind of; we have seen similar instances of this while we treated dealt with the recursive queries in databases.

The source destination of aligns problem that we discussed has similar kind of relationship structure. So, you can think about a relationship flies from the set of source to this set of destination, and basically this; these 2 sets the places source places in the destination place are necessarily the same set the same relation. There could be a constraint on the cardinality.

(Refer Slide Time: 05:07)



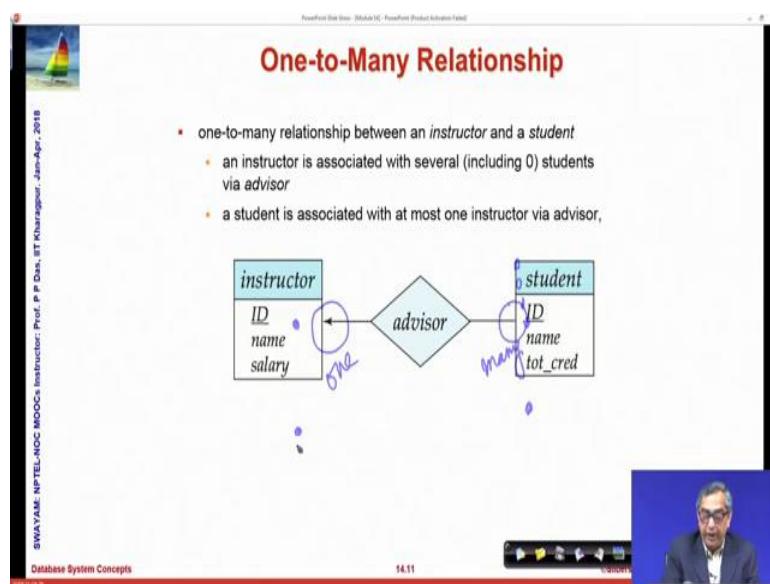
So, the line that links the relationship diamond with the rectangles of the relations rectangles of the entity sets. Those lines could have an arrow at the end or may not have an arrow at the end. So, if it has got an arrow, then it means that suppose it has an arrow it means 1. And if it does not have a arrow if it is simple then it means many. So, using

this rotation we can designate one to one to many these all different kinds of cardinalities that we had discussed.

So, for example, if we are showing this arrow on both hands, both ends of this relationship advisor then, it means that it is a one to one relationship because there is an arrow here. So, this is 1. There is an arrow here. So, this is 1. There is an arrow here. So, this is 1; which means that a student is associated at most with at most one instructor. And it also means that an instructor is associated with at most one student.

This may not be a reality this usually is not the reality, but this we are just showing this as an example. So, if the student instructor relationship advisor relationship is one to one, then this is how we will denote it.

(Refer Slide Time: 06:54)



If it is one to many; so, which side is 1, this side is 1 and this side is many. So, it is one to many from instructor to student; which says that every student has at most one instructor. And an instructor may have several students. It could be null also it could be none also.

So, for an instructor here there are many students, but for a student here there are only at most one instructor. So, it is one to many and this is how we designate.

(Refer Slide Time: 07:35)

Many-to-One Relationships

- many-to-one relationship between a *student* and a *instructor*,
  - an *instructor* is associated with at most one *student* via *advisor*,
  - and a *student* is associated with several (including 0) *instructors* via *advisor*

<i>instructor</i>		<i>student</i>
<i>ID</i>		<i>ID</i>
<i>name</i>		<i>name</i>
<i>salary</i>		<i>tot_cred</i>

advisor

14.12

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Datta, BT Kharagpur - Jan-Apr., 2018  
Database System Concepts

14.12

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Datta, BT Kharagpur - Jan-Apr., 2018  
Database System Concepts

A similar thing will happen, if I read the same relation in the other direction. So, instructor to student was one to many. So, student instructor also drawn in the same way, because this is this is the one side, this is the one side and this is the many side.

So, if the situation is the same. So, if we read from the student instructor, then it is also designates the many to one relationship.

(Refer Slide Time: 08:05)

Many-to-Many Relationship

- An *instructor* is associated with several (possibly 0) *students* via *advisor*
- A *student* is associated with several (possibly 0) *instructors* via *advisor*

<i>instructor</i>		<i>student</i>
<i>ID</i>		<i>ID</i>
<i>name</i>		<i>name</i>
<i>salary</i>		<i>tot_cred</i>

14.13

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Datta, BT Kharagpur - Jan-Apr., 2018  
Database System Concepts

14.13

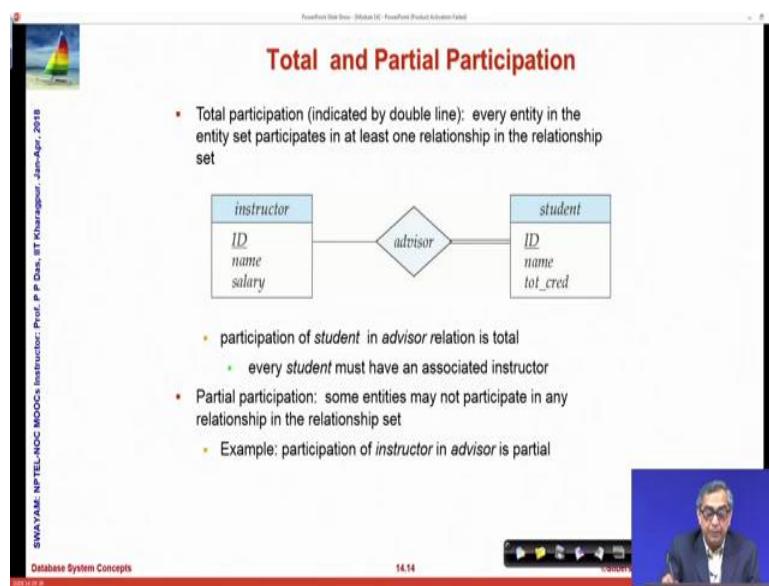
SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Datta, BT Kharagpur - Jan-Apr., 2018  
Database System Concepts

And finally, we can have a many to many relationship, where there is no arrow at either end which means that an instructor is associated with; several possibly none no student

why the advisor relation and the student may also have several instructors by the advisor relationship.

Now, we can you can certainly figure out that in the particular case of student instructor scenario of providing advice, one to one as well as many to many are not the usual real world scenarios, but these are we have just using to show you how to model this usual scenario would be from instructor to student it is one to many relationship.

(Refer Slide Time: 08:50)



A relationship could be total or it could be partial. If you if one side of the relationship, or whichever side of the relationship is total, then we draw a double line. So, you can you can see here in the diagram we are drawing a double line, which means that in the advisor relationship the involvement of the student is total, which means that every student must feature in the advisor relationship. Or in other words every student must have an advisor. But it is partial on the instructor side because every instructor may not have a student.

So, this double life shows that reality. Some entities may not participate in any relationship is a partial.

(Refer Slide Time: 09:59)

**Notation for Expressing More Complex Constraints**

- A line may have an associated minimum and maximum cardinality, shown in the form  $l..h$ , where  $l$  is the minimum and  $h$  the maximum cardinality
  - A minimum value of 1 indicates total participation.
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of \* indicates no limit.

```

    graph LR
      instructor[instructor] -- "0..*" -->|advisor| student[student]
      student -- "1..1" -->|advisor| instructor
  
```

Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

SWAYAM-NPTEL-MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018  
Database System Concepts

14.15

Now, these constraints the cardinality constraints can be made more precise by actually using numbers. You can actually say on the 2 sides of the relationship, that at the minimum how many entities should relate, and at the maximum how many entities can relate. For example, if we are saying that we are on the right-hand side here, if you see we are saying that it is maximum minimum is one maximum is one which what does it say it says that every student the minimum is one. So, every student must feature in the advisor relationship.

So, in real world, every student must have a an advisor, must have an instructor. It says maximum is one; which says that every student can have at most one instructor. So, this one to one one, dot dot one says that every student must have at least one instructor, every student must have at most one instructor. So, together it says that every student must have exactly one instructor.

Whereas if I if we see on this side, it says that 0 dot, dot, star, star stands for no limit. It can be anything any number. So, the minimum is 0 which means that an instructor may not have a student. And star says that if the instructor can have any number of student. Naturally 0, 1, 2, 3, 4, 2, or 200. So, any instructor can advise any number of students.

So, these kind of precise number constraints can be put in addition to the one to many, or one to one, many to many kind of notations in the diagram. So, when we do that we have the precise cardinality of the complex relations that exist.

(Refer Slide Time: 12:10)

Notation to Express Entity with Complex Attributes

```

instructor
ID
name
first_name
middle_initial
last_name
address
street
street_number
street_name
apt_number
city
state
zip
{ phone_number }
date_of_birth
age()

```

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

Database System Concepts

14.16

Prof. P. P. Das

Next, we take a look into the handling of the complex attributes. The first you remember that, the first kind of complex attribute is one which is composite. Say, name which has first name middle name, initial middle initials and last name.

(Refer Slide Time: 12:26)

Expressing Weak Entity Sets

- In E-R diagrams, a weak entity set is depicted via a double rectangle
- We underline the discriminator of a weak entity set with a dashed line
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond
- Primary key for section – (course\_id, sec\_id, semester, year)

```

course
course_id
title
credits
sec_course
section
sec_id
semester
year

```

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

Database System Concepts

14.17

Prof. P. P. Das

So, when we have that, then the way we represent is at the actual name of the attribute is at the outermost level. And it is composites are written with certain shift on the left. So, these all say that these are composites of name. So, if this says that street city state zip are composites of address, and further indentation say, that these are composites of street. So, this is how graphically we show that how complex attributes feature.

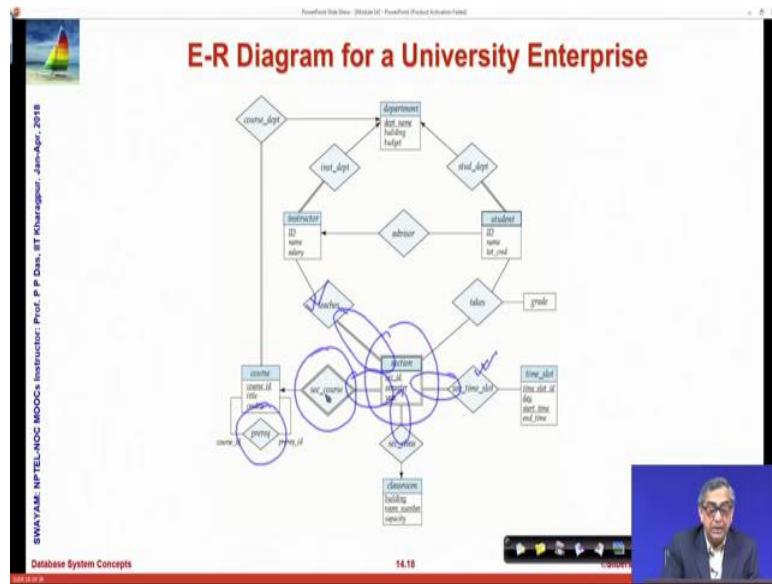
Now, let us go back to discussing the weak entity sets. In the E-R diagram, a weak entity set is represented by a double rectangle. You remember the section is a weak entity set. And why is it so? Because the same course may have 2 different, I am sorry, 2 different courses may have the same section id semester and year that is 2 courses 2 or more courses may run sections by the same name in the same semester and the year. So, a section cannot be uniquely identified by these 3 attributes. It needs a relationship with the identifying entity set course to be specific the course id so that the entities here in can be uniquely specified.

So, since this has happened. So, we designate that by putting this double rectangle around the weak entity set section. We underlined the discriminator of a weak entity set with dashed line. So, you remember these are the discriminators. Because given the identifying attribute in the identifying set. These are the attributes which distinguish different tuples of section. So, they are not shown with solid underline, they are shown as dotted underline, dashed underlined. So, that you can make out that this is a weak entity set and these are the discriminators.

The relationship set connecting the weak entity set to the identifying strong entity set, is also so, this the moment you have weak entity set. You know that there has to be a relationship to the strong entity set which identifies it. So, that relationship set course which say, course id against this minds that is designated with a double diamond. So, that you know that this is the identifying relationship between a weak entity set and the corresponding strong entity set. And once that happens in the primary key becomes the discriminators of section the weak entity set, and the primary key of the identifying a strong entity set the course. So, that forms our final primary key for this entity set section.

My new course id is not a part of this relation, but it actually plays the role through this section id as a key for the section relation without which the section entities in the section cannot be uniquely identified.

(Refer Slide Time: 16:28)



So, having said that this is the E-R diagram of the university enterprise. Some of the points that you could take a look at; this is the weak entity set we have just seen. This is the relationship to the identifying strong entity set. This is a prerequisite multi role relationship. This we can see is a total involvement. So, worry why is it a total involvement, because every section must have at least one teacher. So, there cannot be a section which does not feature in the teacher's relationship. Similarly, every section must get a timeslot where the classes for that section is held.

So, every section must feature in the sec timeslot. So, these are similarly, it must get a classroom. So, these are all different total involvements. That we total roles that we can see we can see some of that elsewhere as well. For example, you can see it here, we can see it here, because in between instructor and the department the inst department relationship. Certainly, every instructor must have a department. So, it is total, but it is not the same for the department, every department will not have instructors.

So, these is how if we can go through carefully. And for example, this is another which is total, which means that every course need a department you cannot run a course which does not have a department. So, this is how we can see that how the E-R diagram that the first conceptual level diagram of a very simple university enterprise is being designed following the notions and symbols of E-R model that we have already developed.

Next comes the part where, from this model which is primarily diagram based we have to really go to the relational schema, which is names of relations and attributes which is pretty much a straightforward job.

(Refer Slide Time: 18:41)

The slide has a title 'Reduction to Relation Schemas' in red. Below it is a bulleted list:

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database
- A database which conforms to an E-R diagram can be represented by a collection of schemas
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set
- Each schema has a number of columns (generally corresponding to attributes), which have unique names

At the bottom left, it says 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Datta, BT Kharagpur - June-April, 2018'. At the bottom center, it says 'Database System Concepts'. At the bottom right, it shows a timestamp '14:20' and some control icons. A small video window in the bottom right corner shows a man in a suit and glasses speaking.

So, entity sets and relationship sets have to be represented in terms of relational schema. What is the beauty of the E-R model? And the relational schema is that that, when you reduce the entity relationship model to relational schema both entity sets and relationships sets. Both of them turn out to be relational schemas. And so, that the database finally, can be represented simply as a set of schemas each one of which must have a set of identifying primary key.

(Refer Slide Time: 19:20)

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

## Representing Entity Sets

- A strong entity set reduces to a schema with the same attributes  
 $\text{student}(ID, name, tot\_cred)$
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set  
 $\text{section}(\text{course\_id}, \text{sec\_id}, \text{sem}, \text{year})$

```

    graph LR
        course[course  
course_id  
title  
credits] -- sec_course --> section[section  
sec_id  
semester  
year]
    
```

Database System Concepts 14.21

So, let us look into that. So, on the strong entity set, that reduces to schema with the same attribute as student. So, student has id name and total credit so, which we saw earlier. Now this gets converted to a schema with id being the primary key. The other case of weak entity set, section which had the three discriminators, and through set course relationship was identified from the strong entity set. Course borrows the primary key of the course to be defined in terms of this relational schema.

(Refer Slide Time: 20:05)

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

## Representing Relationship Sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

```

    graph LR
        instructor[instructor  
ID  
name  
salary] -- advisor --> student[student  
ID  
name  
tot_cred]
    
```

advisor =  $(s.id, i.id)$

Database System Concepts 14.22

One moment; this borrows the primary key from here and becomes.

(Refer Slide Time: 20:10)

PowerPoint User Guide - Module 1C - PowerPoint Product Activation Failed

## Representing Entity Sets

- A strong entity set reduces to a schema with the same attributes  
 $\text{student}(\underline{\text{id}}, \text{name}, \text{tot\_cred})$
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set  
 $\text{section}(\underline{\text{course\_id}}, \text{sec\_id}, \text{sem}, \text{year})$

course

course\_id  
title  
credits

see\_course

section

sec\_id  
semester  
year

Database System Concepts

14.21

Navigation icons: Back, Forward, Home, etc.

So, you can see that in the E-R model, the section did not have course id as an attribute, but while we reduce this to the relational schema through this sec course relationship, we have borrowed this primary key from course. The primary key of course, the course id and added that to section to make it a complete relational schema.

Next comes the representation of relationships. So, we are showing a relationship advisor so, which relates instructors to students. So, naturally every instructor is identified by id every student is identified by id. Since both of the attributes have the same name id. We are calling them as s underscore id and for the student and I underscore id for the instructor. So, the advisor relation is basically a pairing of these two ids which gives rise to a relationship which looks like this relationship schema which looks like this. So, it can in general say that if we have a relationship in the E-R model, which we want to represent in the schema then we will take, we will create a schema which has the primary key of both the sets, and put them together. And if the names clash we will just change the name with the name of the relation, and that will give us the schema for the relationship in this case the advisor.

So, you have seen how to represent entity sets, weak entity sets and relationships.

(Refer Slide Time: 22:03)

**Representation of Entity Sets with Composite Attributes**

The slide illustrates the representation of entity sets with composite attributes. It shows a schema for an 'instructor' entity set with attributes: ID, name, address, and phone\_number. The 'name' attribute is a composite attribute with components first\_name, middle\_initial, and last\_name. The 'address' attribute is a composite attribute with components street, street\_number, street\_name, apt\_number, city, state, and zip. The 'phone\_number' attribute is a multivalued attribute represented by a list of phone numbers.

- Composite attributes are flattened out by creating a separate attribute for each component attribute.
- Example: given entity set *instructor* with composite attribute *name* with component attributes *first\_name* and *last\_name* the schema corresponding to the entity set has two attributes *name\_first\_name* and *name\_last\_name*.
- Prefix omitted if there is no ambiguity (*name\_first\_name* could be *first\_name*)
- Ignoring multivalued attributes, extended *instructor* schema is:

```

instructor(ID,
           first_name, middle_initial, last_name,
           street_number, street_name,
           apt_number, city, state, zip_code,
           date_of_birth)
  
```

Ah let us look at how do we deal with composite attributes. Because so far, we had assumed that the relational schema has attributes, and every attribute has a domain. The type from where its values come. So, if I have a composite attribute, where every attribute has a set of components. Then the easiest way to handle this is to what is known as flatten the composite attribute.

So, flattening basically is for example, if I take a name it has 3 components. So, each one of them I can call by given new name, name underscore first name, name underscore middle initial name underscore last name. By prefixing with the attribute name, I make the names of these components necessarily unique. Now after I do the prefixing I might figure out that actually prefixing is not required first name itself is a unique. Because it does not occur anywhere else if it is then I can I may drop the prefix name. But in general, I can take the attribute name prefix on the component, and just flatten them out make them all attributes each separate attribute.

So, here when we flatten out we will have first name, middle, initial, last name as you can see these flattened out from here. Then we have street number, street name, apartment number, flattened out from the street. Subsequently, we have city state zip flattened out from here. So, all of them flattened out has become separate attributes. And flattening is a very straightforward mechanism by which you can convert complex composite attributes into the regular schema design getting to little bit of issue if you have a multi valued attribute.

(Refer Slide Time: 24:02)

The screenshot shows a PowerPoint slide with the title "Representation of Entity Sets with Multivalued Attributes". The slide content includes a bulleted list:

- A multivalued attribute  $M$  of an entity  $E$  is represented by a separate schema  $EM$
- Schema  $EM$  has attributes corresponding to the primary key of  $E$  and an attribute corresponding to multivalued attribute  $M$
- Example: Multivalued attribute  $phone\_number$  of  $instructor$  is represented by a schema:  
 $inst\_phone = (ID, phone\_number)$
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema  $EM$ 
  - For example, an  $instructor$  entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples: (22222, 456-7890) and (22222, 123-4567)

The video player at the bottom shows a man speaking, with the caption "Database system Concepts" and the time "14:24".

Multivalued attribute is one where one attribute may have multiple values at the same time. And the example we talked about is a phone number.

I may have multiple phone numbers. So, certainly against an attribute I can keep only one value. So, if my attribute is multiple value. Then the basic idea is to use a separate schema to maintain this multiple values. For example, if I have to maintain multiple phone numbers of an instructor. I may have a separate I may decide to have a separate relation which relates the key of the instructor relation, and the attribute that I want to maintain multiply.

So, in this relationship in this relation inst underscore phone against the same id I can have different phone numbers. So, there will be different records which match on the id, but do not match on the phone number which gives me the different values that the phone number can take. And then this inst phone in conjunction with the instructor relation will actually denote the multi valued phone number attribute. So, this is just an example showing that for one primary key of an instructor 1 to 2 2 2 2 and there are two phone numbers. So, this will basically mean you have two tuples in the new relation.

(Refer Slide Time: 25:37)

**Redundancy of Schemas**

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side
- Example: Instead of creating a schema for relationship set *inst\_dept*, add an attribute *dept\_name* to the schema arising from entity set *instructor*

So, with that we can handle multiple multivalued attributes also. Some of the relationships that we may have modeled, which we have done in doing the database E-R schema could have redundancy. For example, take a case here we have the instructor. And we have the student advisor relation is incidental here, and we have department. So, we want to say that the instructors belong to certain departments. Every instructor belongs to one department which is the totality of the relationship here.

Similarly, every student belongs to a department, totality of the relationship on this side. And inst dept inst\_dept in that context is a relationship which is between instructor and department. Similarly , so, we can there is certain redundancy in this, because we can get make this simpler if we just take the primary key of this relation and put in here. If we do that then basically this become redundant these are no more required. So, all that you are saying is the instructor has a dept name field which says which department does it belong to.

So, if there is a choice between whether you will keep such relationships or you will actually reduce the redundancy in the schema, and involve the primary key of the other relation into your primary table which is instructor or the student here. So, instead of creating a schema for relationship set inst department. You will simply add a department name. So, mind you this is at the at the E-R model level you did have a separate relationship, but while you reduce it to your relationship relational schema you are reducing that relationship by including the dept name as an attribute in instructor.

(Refer Slide Time: 27:53)

The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.

Example: The *section* schema already contains the attributes that would appear in the *sec\_course* schema

```
graph LR; course[course  
course_id  
title  
credits] --- sec_course((sec_course)); section[section  
sec_id  
semester  
year] --- sec_course;
```

So, that is called the reduction of schema which is often used. for one to one relationship. So, this is this, was this is good if you have many to one relation. Because this was possible, because every instructor has one department. So, if you just include the department name with the instructor. Or every student has one department ah. So, it is possible that way, but if you have a one to one relationship, then naturally you can do the similar reduction by I by choosing the either side as the many. You know, because because the the unique side has to come on the many side. The unique side here. This is the unique side here. Because every instructor has a unique department and this is the many side here.

So, the unique side has to attribute has to come in here. The unique side primary key has to come in here. So, instead of so, you can apply the same principle to a one to one relationship by treating any one of them as a many side, and add the extra attribute on the other side to get rid of this additional schema. The schema corresponding to a relationship set linking a weak entity set to it is underlying strong entity set is certainly redundant, we have already seen this. So, this (Refer Time: 29:13) is made redundant by including the primary key of the identifying relation, identifying entity set in the weak entity set. So, that is another reduction of schema that can be done.

(Refer Slide Time: 29:31)

The screenshot shows a Microsoft PowerPoint slide titled "Module Summary" in red font at the top center. Below the title, there is a bulleted list of two items:

- Illustrated E-R Diagram notation for E-R Models
- Discussed translation of E-R Models to Relational Schemas

On the left side of the slide, there is a vertical watermark or logo for "SWAYAM-NPTEL-MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018". At the bottom left, it says "Database System Concepts". At the bottom right, it shows the time as "14.28" and the slide number as "10 of 10". The bottom right corner also has a note: "Under construction, poster and about section". The overall background of the slide is white.

So, to summarize we have in this module illustrated the entity relationship diagram, which are very nice ways of graphically representing what we see in the real world. So, it has graphical representation of entity sets, attributes the key attributes primary key attributes the weak entity sets, and the relationships along with the cardinality information.

And then we have shown that using certain reduction rules how we can easily reduce this entity relationship diagram or entity relationship model into the traditional relational schema. And we have seen that both the entity sets as well as the relationship sets become relational schemas.

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

**Lecture – 15**  
**Entity-Relationship Model/3**

Welcome to module 15 of Database Management Systems. We have been discussing about entity relationship model and this is the third and the closing module on this topic.

(Refer Slide Time: 00:37)

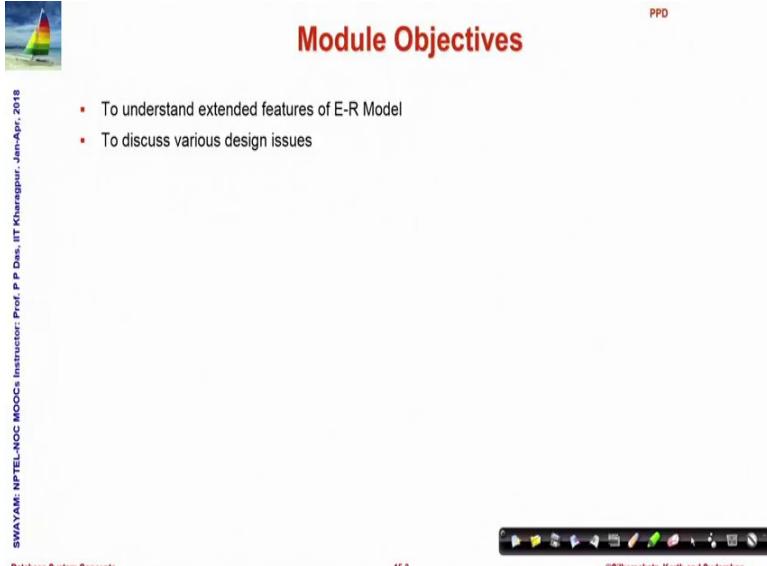
The slide has a white background with a decorative header featuring a sailboat icon and the text "Module Recap" in red. In the top right corner, there is a small "PPD" label. On the left side, there is vertical text: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur, Jan-Apr, 2018". At the bottom, there is a footer with icons for navigation, a copyright notice: "©Silberschatz, Korth and Sudarshan", and page numbers: "15.2" and "317".

**Module Recap**

- E-R Diagram
- E-R Model to Relational Schema

So, in the last module we have discussed about E-R diagram and we have also seen how E-R model can be converted to a relational schema.

(Refer Slide Time: 00:47)

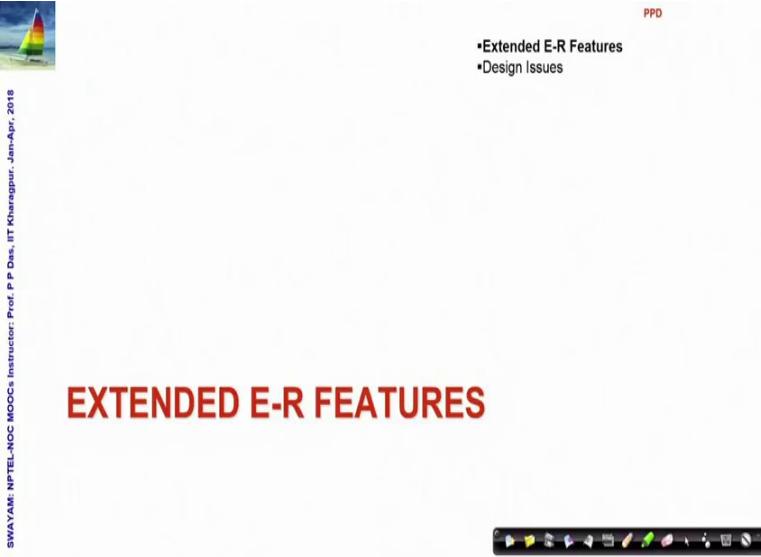


The slide title is "Module Objectives". It features a small sailboat icon in the top left corner. In the top right corner, there is a small red "PPD" logo. On the left side, vertical text reads: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr. 2018". The main content is a bulleted list: "To understand extended features of E-R Model" and "To discuss various design issues". At the bottom, it says "Database System Concepts" and "15.3". A navigation bar is at the bottom right, and a copyright notice "©Silberschatz, Korth and Sudarshan" is also present.

In this module we will try to go through a few extended features of E-R model, try to show some of the more complicated situations how they can be modeled in the E-R model and along with that we will discuss a variety of design issues that will follow.

So, these are the outline.

(Refer Slide Time: 01:19)



The slide title is "EXTENDED E-R FEATURES". It features a small sailboat icon in the top left corner. In the top right corner, there is a small red "PPD" logo. On the left side, vertical text reads: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr. 2018". The main content is a bulleted list: "Extended E-R Features" and "Design Issues". At the bottom, it says "Database System Concepts" and "15.5". A navigation bar is at the bottom right, and a copyright notice "©Silberschatz, Korth and Sudarshan" is also present.

So, we start with extended entity relationship features.

(Refer Slide Time: 01:23)

The slide has a header 'Non-binary Relationship Sets' and a small sailboat icon in the top left. On the left, vertical text reads 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018'. In the center is an E-R diagram showing three entity sets: 'instructor', 'project', and 'student'. The 'instructor' set has attributes 'ID', 'name', and 'salary'. The 'student' set has attributes 'ID', 'name', and 'tot\_cred'. A ternary relationship 'proj\_guide' connects the three sets. The 'project' set is shown above the relationship, and the 'instructor' and 'student' sets are below it. The relationship 'proj\_guide' is represented by a diamond shape. Below the diagram is a video feed of a man in a suit, identified as Prof. P. P. Das. The bottom of the slide features a toolbar and the text '©Silberschatz, Korth and Sudarshan'.

The first that we note is so, far in the entity relationship model we have talked about relationships between two entity sets. So, we have talked about the student attending courses or instructors advising students and so, on. So, such relationships are naturally called binary, but it is possible that more than two entity sets, let us say three entity sets could be involved in the same relation and we show an example here where we have three entity sets instructor, student and project.

So, the project entity set is a list of projects being done by the students or to be done by the students. So, the relationship project guide is a relationship between the guide who is an instructor, the student who will do the project and the project that has to be performed. So, all three together define this relationship; so, in such cases it is possible in E-R model that we can represent it conveniently as a non binary relationship.

Now this is an E-R diagram this is called a ternary relationship R.

(Refer Slide Time: 03:02)

The slide has a header 'Cardinality Constraints on Ternary Relationship' with a sailboat icon. On the left, vertical text reads: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018'. Below the header is a bulleted list: '• We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint'. To the right is a diagram showing two entity sets, E<sub>1</sub> and E<sub>2</sub>, represented by rectangles. A diamond-shaped relationship symbol connects them, with arrows pointing from E<sub>1</sub> to the diamond and from the diamond to E<sub>2</sub>. At the bottom left is a video frame of a professor, and at the bottom right are navigation icons.

Now we have talked about cardinality constraints on binary relationship one to one, many to one one to many and many to many. And we specified that if we have a binary relationship say this is one entity set and this is another entity set and we have a relationship. So, if we just connect them it means a many to many relation, but if we have an arrow on one side entity set then it means on the arrow side its one.

So, this is from entity set E<sub>1</sub> to E<sub>2</sub> this is one to many; we could have arrow at both ends and; that means, one to one. Now the question is how will that work out what will be the meaning of arrow in terms of a ternary relationship.

(Refer Slide Time: 03:55)

**Cardinality Constraints on Ternary Relationship**

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- For example, an arrow from *proj\_guide* to *instructor* indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning.
  - For example, a ternary relationship  $R$  between  $A$ ,  $B$  and  $C$  with arrows to  $B$  and  $C$  could mean
    - Each  $A$  entity is associated with a unique entity from  $B$  and  $C$  or
    - Each pair of entities from  $(A, B)$  is associated with a unique  $C$  entity, and each pair  $(A, C)$  is associated with a unique  $B$
  - Each alternative has been used in different formalisms
  - To avoid confusion we outlaw more than one arrow

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

15.7 ©Silberschatz, Korth and Sudarshan

Now, in the case of ternary relationship or this would generalize to relationships of higher degree whether where more than three entity sets may be involved. We put a restriction that we will allow at most one arrow out of the ternary relationship. So, we could have only if we if we look at if we look at say this relationship then we could have an arrow only at this end.

But multiple arrows are not allowed and the reason is certainly to keep the semantics of the cardinality meaningful. For example, if we have a ternary relationship between  $A$   $B$  and  $C$  let us say this is  $A$  this is  $B$  and this is  $C$  and we have a ternary relationship between them. And let us say if we have a more than one arrow; there for example, suppose then we have say an arrow to  $B$  and an arrow to  $C$  the question is how should we interpret that?

Should we interpret that an entity of entity set  $A$  is associated with unique entity from  $B$  and  $C$  together or should we associate, should we interpret that the entities formed by the pair  $A$   $B$  and the entity formed by the pair  $A$   $C$  are uniquely related. So, there is multiplicity of interpretation; if we allow more than one arrow in case of eternity or higher degree relationship. So, what will follow for simplicity in this course and that is what is followed often in practice; is in case of a ternary or higher order relationship only one arrow will be allowed in that in it in that relationship.

(Refer Slide Time: 06:20)

The slide has a title 'Specialization' in red at the top right. On the left, there is a small sailboat icon. To the right of the title is a bulleted list:

- Top-down design process; we designate sub-groupings within an entity set that are distinctive from other entities in the set

Below the list is a hand-drawn diagram showing a rectangle labeled 'A' with an arrowhead pointing to a rectangle labeled 'B'. Below this, the text 'B is A' is written. At the bottom of the slide, there is a navigation bar with icons for back, forward, search, and other presentation controls.

Now, let us talk about specialization those of you who have some background of object oriented systems would be familiar with the notion of specialization and generalization in object oriented system. So, we say that if we have a certain concept say we have a concept called person and then we say that a student is a person; what we mean is a student is a specialization of person. And person is a generalization of student and in that process student inherits all the attributes of person, but in addition the student may have some specialized attributes.

So, what it means that if you look from the perspective of such specialization; say if I draw like this; this is an entity set A and this entity set B and to mark the specialization we show an arrowhead with a blank triangle at that end. So, if we by this what we mean is B is a A. So, B inherits all the properties of A, but can have some more properties. So, if you look at all the entities that A may have you will find that a subgroup of the entities in the entity set A have some additional common properties.

So, if A is set of persons and B is a set of students then A may have entities which represent people who are not students; who are employees, who could be retired and so, on, but there will be a number of entities. So, who have the commonality of being student they are enrolled in certain course of certain university and so on.

So, in terms of the E-R diagram E-R model what we do is we try to look at the entity A and move top down. So, that whenever we find a group of entities which have certain

commonality; we move them into a lower separate, specialized entity set and relate these two entity sets to the specialization relation.

(Refer Slide Time: 08:54)

## Specialization

- Top-down design process; we designate sub-groupings within an entity set that are distinctive from other entities in the set
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set
- Depicted by a triangle component labeled ISA (e.g., *instructor* "is a" *person*)
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked

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

Database System Concepts

15.8

©Silberschatz, Korth and Sudarshan

So, these sub groups form lower level entity sets and as I said that it is designated in a certain way. And as in the object oriented system the lower level entity set inherits all the attributes and relationships of participation of the higher level entity set. So, here is an example.

(Refer Slide Time: 09:16)

## Specialization Example

- Overlapping – *employee* and *student*
- Disjoint – *instructor* and *secretary*
- Total and partial

```

graph TD
    person["person<br/>ID<br/>name<br/>street<br/>city"] --> employee["employee<br/>salary"]
    person --> student["student<br/>tot_credits"]
    employee --> instructor["instructor<br/>rank"]
    employee --> secretary["secretary<br/>hours_per_week"]
  
```

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

Database System Concepts

15.9

©Silberschatz, Korth and Sudarshan

Can say the person at the so, called root of this hierarchy of specialization; it has a set of properties ID, name, street, city and employee is another relationship; another entity set which is a specialization of person relation person entity set.

So, employee inherits all attributes ID name street and city, but in addition the commonality of the entities in the employee entity set is a fact that all of them have a salary attribute. A similar entity set student is a specialization of person where the again all attributes are inherited, but there is a common attribute called total credits which is common for all the students, but is not available or common for the persons in general.

And as you can see that it could be hierarchical it could go further down employee could be specialized into instructor and secretary. Again by the rule of specialization instructor will inherit all attributes of employee which means that it will inherit attributes of person; which imply has inherited plus the employee specific attributes salary. So, it will inherit all five of those attributes and then it adds another attribute which is specific for the instructor which says a rank which is another specific attribute that you have.

Now, when we specialize a certain entity set into two or more entity sets like we specialized person in employee and student; then there could be different situations that could exist for example, certain entity may be a member of both employee as well as student. If that happens then we say that their overlapping entity sets or they could be disjoint where no member of instructor would be a member of the secretary and vice versa.

So, we say that this disjointness tell us that no instructor can be a secretary and no secretary can be an instructor. Whereas, overlapping specialized sets denote that well an employee may or may not be a student, but it is possible that some employee is also a student and vice versa and that is how we will represent this. And we will see that when we specialized the specialized entity could be total or they could be partial.

(Refer Slide Time: 12:06)

The slide is titled "Representing Specialization via Schemas". It features a logo of a sailboat on the left and a sidebar with course details: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018". The main content area contains a table showing schema representation and a list of drawbacks.

schema	attributes
person	ID, name, street, city
student	ID, tot_cred
employee	ID, salary

**Method 1:**

- Form a schema for the higher-level entity
- Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

**Drawback:**

- Getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

We will talk about that totality and partiality little later; let us look at how do we represent this information in the relational schema because as we have seen that whenever we have a E-R diagram; it is important to find out what is the relational schema that will be corresponding to that E-R diagram or E-R model. So, here we could do this in two ways one that we are showing here is form a schema for the higher level entity. So, form a schema for the person as you can see here that person is described in terms of four attributes and this form a schema for each of the lower level entity set where you include the primary key of the higher level entity set.

So, when you are forming the schema of person of student which is a specialization of person you include the ID which is the key of the higher level entity set person. And along with that you include the so, called local attributes or attributes which are specific to this lower level entity set in this case total credit similar thing happens with employee.

Now this representation is in a way optimized because you are representing the information only once when it is needed, but the drawback is if you have to find out information about say employee; then you will not only have to access the employee entity set or the corresponding relation in the relational schema, but you will also have to access the parent or higher level entity set to get the attribute values which are inherited. And if you have a multi level hierarchy as we have shown this could involve accessing multiple relations to find information about a single entity in an entity set.

So, this is an in terms of data representation this is an optimized representation, but it has the overhead of having to access multiple entity sets to get information about certain entities.

(Refer Slide Time: 14:22)

**Representing Specialization as Schemas (Cont.)**

Method 2:

- Form a schema for each entity set with all local and inherited attributes

schema	attributes
↗ person ↗ student ↗ employee	ID, name, street, city ID, name, street, city, tot_cred ID, name, street, city, salary

- Drawback: *name, street and city* may be stored redundantly for people who are both students and employees

**SVAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Dass, IIT Kharagpur - Jan-Apr. 2018**

**Database System Concepts**

15.11 ©Silberschatz, Korth and Sudarshan

An alternate scheme would be that based on the hierarchy of specialization, you assume all attributes as they are inherited and then represent every entity set in full. So, when you the representation of person does not change, but when you represent student now earlier you are just having ID and total credit; now in you include all entities that are inherited.

Similarly, you do the same thing; so, you have the all entities of the parent to all attributes of the parent entity set as well as the local attribute of that specific entity set. Now this naturally makes it easy to extract information from a for a single entity set, but at the same time, you are storing the same data redundantly for people who are having overlapped representation. So, if we have as we know student and employee overlapped. So, the same entity will happen in student as well as in employee; so, it will the information of the common attributes name street city etcetera they will occur in both these tables in the design.

So, these are two methods and now we have just given you the relative advantages and its advantages of the same and based on a particular situation you will have to choose what is a good method to represent.

(Refer Slide Time: 15:51)

**Generalization**

- A **bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.

```
graph TD; Student[Student] --> UGStd[UG Std.]; Student --> PGStd[PG Std.]
```

SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018  
Database System Concepts  
15.12  
©Silberschatz, Korth and Sudarshan

You can look at now you know from the object based system that if you have a specialization hierarchy you can think of it as a generalization hierarchy also.

The generalization hierarchy goes in a bottom up manner. So, instead of actually starting with an entity set and finding out subsets of entities which have greater commonality between them and putting them as specialized, you could actually group them in terms of finding out what they share and create a higher level entities. For example, the way I am saying is let us say that I have one entity set which say UG student and have another entity set in the same university which say PG student.

So, there are both of them are students naturally they are disjoint a person cannot be UG as well as PG student at the same time. And once you represent that you find that well there are a lot of information which are common between these two entity sets like the student roll number, name and so on so, forth. So, you could choose that well you instead of having them as two separate entity sets, you could extract out the common attributes and put them at a higher level entity. So, all that you are doing is instead of going top down you are going bottom up in the whole approach.

(Refer Slide Time: 17:21)

The slide features a title 'Generalization' in red at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list of points:

- A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way
- The terms specialization and generalization are used interchangeably

On the left margin, vertical text reads: SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018. At the bottom left is the text 'Database System Concepts'. The bottom right corner shows the copyright notice '©Silberschatz, Korth and Sudarshan'.

So, if you do that then there you can easily see that specialization and generalization are inverse of each other and they are used interchangeably in terms of the relational entity relationship design.

(Refer Slide Time: 17:35)

The slide features a title 'Design Constraints on a Specialization/Generalization' in red at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list of points:

- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization
  - **total**: an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong to one of the lower-level entity sets
- Partial generalization is the default. We can specify total generalization in an ER diagram by adding the keyword **total** in the diagram and drawing a dashed line from the keyword to the corresponding hollow arrow-head to which it applies (for a total generalization), or to the set of hollow arrow-heads to which it applies (for an overlapping generalization).
- The **student** generalization is total: All student entities must be either graduate or undergraduate. Because the higher-level entity set arrived at through generalization is generally composed of only those entities in the lower-level entity sets, the completeness constraint for a generalized higher-level entity set is usually total.

On the right side, there is an Entity-Relationship (ER) diagram illustrating the concepts. It shows a 'person' entity set at the top level, which generalizes into 'employee' and 'student'. The 'employee' entity set has attributes 'salary' and 'rank'. The 'student' entity set has attributes 'tot\_credits' and 'hours\_per\_week'. Below 'employee', there is another generalization step leading to 'instructor' and 'secretary' entity sets, each with their own attributes.

On the left margin, vertical text reads: SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018. At the bottom left is the text 'Database System Concepts'. The bottom right corner shows the copyright notice '©Silberschatz, Korth and Sudarshan'.

The other constraint that you can identify, you should identify is the constraint of completeness which say that if I have a entity set say a person. And then we have specializations of employee and student; the question is for a higher level entity set is it

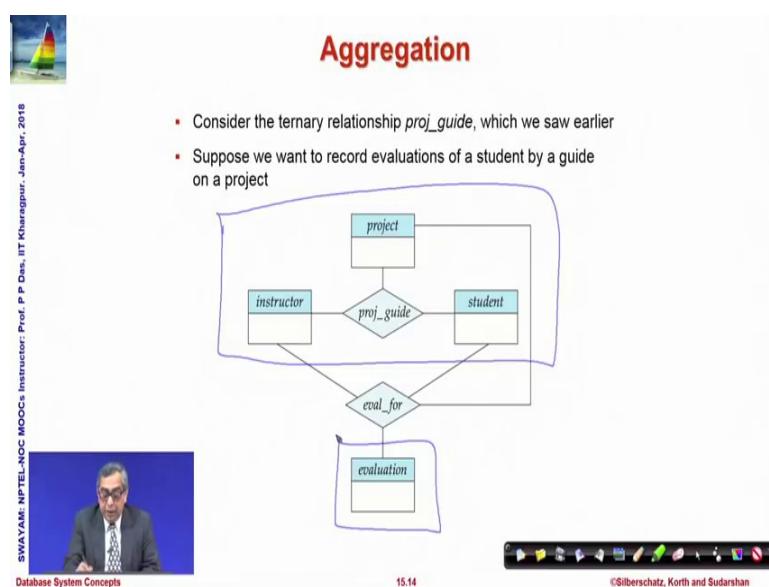
necessarily that every entity will be represented in one of the or more than one of the lower level entity sets.

If that is guaranteed that an entity must belong to one of the lower level entity set we say that it is a complete specialization. But if it is that a high level entity is may or may not be featuring in a entity set, which is at a lower level then we will say it is a partially specialized hierarchy. So, the both of these are possible depending on different situation that we have. So, by default we assume partial specialization and. So, if we want to say a certain specialization is total we will have to write the keyword total by the side of the arrow head that is representing the specialization hierarchy.

You can say that the example I talked of in uniting unite undergraduate and graduate or postgraduate students into the entity set of students, gives you a hierarchy which is total because every entity in the entity set student must be either a UG student or a PG student; it is not possible that I have a student who is neither a UG student nor a PG student. So, every high entity at the higher level entity set student must feature in one of these two specializations.

So, therefore they are necessarily total in the relationship. So, this is the completeness constraint that you can think of.

(Refer Slide Time: 19:52)



Moving on let us talk about another feature which is known as aggregation; the situation is like this we have already talked about this part of the diagram which is a ternary relationship which relates project instructor and student.

Now, let us say once the project progresses you would need to add evaluation to that. So, there is another entity set which represents evaluation and how we are grading or putting marks and so, on. So, naturally the evaluation of a student will be dependent on the project, the student and the supervisor and that will relate to the evaluation. So, evaluation eval for the relationship is necessarily a relationship between four entities or four entity sets so, to say.

(Refer Slide Time: 20:47)

**Aggregation (Cont.)**

- Relationship sets `eval_for` and `proj_guide` represent overlapping information
  - Every `eval_for` relationship corresponds to a `proj_guide` relationship
  - However, some `proj_guide` relationships may not correspond to any `eval_for` relationships
    - So we cannot discard the `proj_guide` relationship
- Eliminate this redundancy via aggregation
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

SWAYAM, NPTEL-NOC, MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jam-Apri- 2018  
Database System Concepts

15.15 ©Silberschatz, Korth and Sudarshan

Now, the question is how do we represent this information? The relationship sets eval for project guide the two that we saw if we just want to recall once more. The project guide involves three of the relation entity sets and the eval for relates to four of the entity sets. Now every eval for relationship corresponds to a project guide relationship; that is if I have an entity in eval for relationship, I will have a corresponding entity in the project guide relationship which specifies the student project and the instructor.

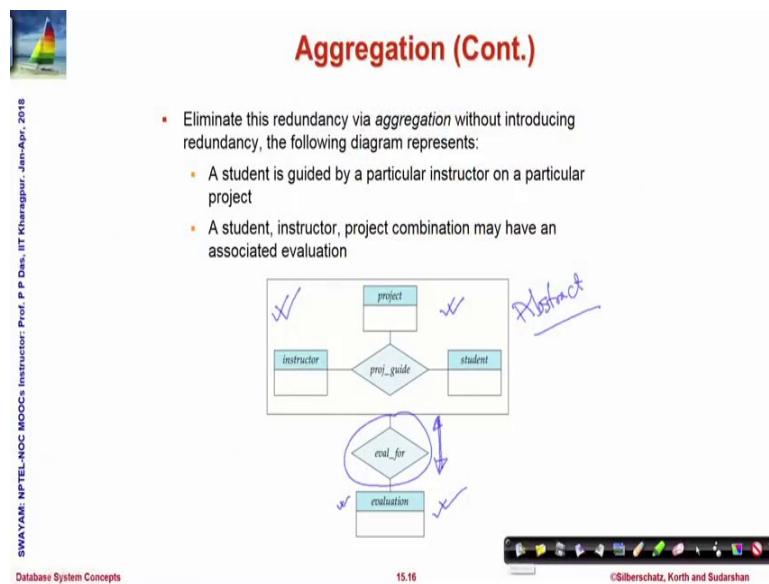
So, but it is other way it is possible that some project guide relationship may not correspond to any eval relationship; that is it is possible that there is a allotted project by a student with a particular instructor which has yet not been evaluated; the evaluation process is not complete or the time has not come.

So, if we have to represent the information only for the eval for relationship; we will get partial information because it is possible that some entities in eval for does not have the eval for information do not feature there, but need to be preserved because I need to remember the project guide, instructor, the student and the project. So, we need to keep both duplicating the information.

So, we can use aggregation to eliminate this duplication of information or redundancy of information. So, what we do is we treat the first relationship; the project get relationship as if it is an abstract entity and then you allow relationship between two relationships; this is something we did not do before relationship so, far has always been between entity sets.

So, what you can see that project guide relationship itself as if it is an virtual entity; it is an abstract entity and then you allow the relationship between the project guide and the eval for relationship which relates to the evaluation entity set this really this shows I mean I will just show you in the diagram.

(Refer Slide Time: 23:08)



So, this is how it will now work out to be. So, this is the abstract project guide entity set which is an abstract entity set because it is actually relationship. And that relates to eval for which on the other site has the evaluation. So, what will happen is a student is guided by a particular instructor in a particular project will feature in this abstract entity set; which relates the three entity sets project student and instructor. And if it has an

evaluation then the; this through this relationship it will be represented and the evaluation value will exist.

So, we know that if a project is evaluated then it certainly have a corresponding entity in the abstract entity set project guide, but the reverse may not be true I may have an entity in the project guide entity set which does not have an evaluation. So, by using this aggregation model; I can represent the information of this situation model this situation more accurately than I could do otherwise.

(Refer Slide Time: 24:30)

**Representing Aggregation via Schemas**

- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes
- In our example:
  - The schema eval\_for is:  
 $\text{eval\_for}(\text{s\_ID}, \text{project\_id}, \text{i\_ID}, \text{evaluation\_id})$
  - The schema proj\_guide is redundant

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018  
Database System Concepts

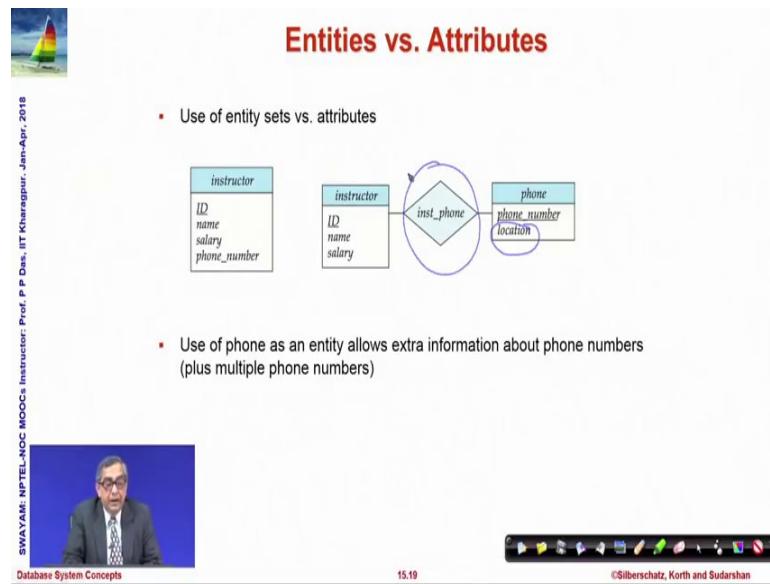
15.17 ©Silberschatz, Korth and Sudarshan

So, this can be represented again how to represent this in terms of the schema? So, what we do we represent the aggregation we create a schema containing the primary key of the aggregated relationship. The primary key of the associated entity set and all the other descriptive attributes and put them together.

So, in our example the schema would be eval four and that schema will have these are entities these are attributes of the aggregated or abstract entity set which is coming from the student project and the instructor entities and this is for the evaluation ID. So, we put this together; so, now, you can see that all of these are related representing who is the guide of which student in what project and if this exists then this gives you the evaluation.

So, naturally once this has been represented; the project guide schema by itself becomes redundant and therefore, it can be removed. So, this is a process through which we come to the decision of actually having this schema to represent all the required information. Naturally if the evaluation is not done then the evaluation ID for eval for will not exist and that will be a null showing that it is not present right now ok.

(Refer Slide Time: 26:25)



Now, given these basic features as well as the extended features let me talk about a few design issues which will be required to see what kind of information that that the different challenges that we have seen so, far. For example, we have seen the case of multivalued attributes.

So, and the way we can represent that is using that multivalued attribute as a separate entity set like the phone number which also has the advantage of having its own added information. For example, once we do this then not only I can have against the same instructor ID; I can have multiple phone numbers, but I can have location for each one of these phone numbers. And I make use of this relation relationship that I create which allow me to represent this multi valued attribute.

(Refer Slide Time: 27:30)

The slide is titled "Entities vs. Relationship sets". It features a diagram illustrating relationships between entities. At the top left is a small sailboat icon. On the left edge, vertical text reads: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018". The main diagram shows three entities: "section", "registration", and "student". "section" has attributes "sec\_id", "semester", and "year". "student" has attributes "ID", "name", and "tot\_cred". Two relationship sets, "section\_reg" and "student\_reg", connect "section" and "registration", and "registration" and "student" respectively. The "registration" entity is represented by a rectangle with three dashed lines inside, indicating it is a relationship set.

**Use of entity sets vs. relationship sets**  
Possible guideline is to designate a relationship set to describe an action that occurs between entities

```
graph LR; section[entity section] --> section_reg{relationship section_reg}; section_reg --- registration[relationship set registration]; registration --- student_reg{relationship student_reg}; student_reg --- student[entity student]
```

**Placement of relationship attributes**  
For example, attribute date as attribute of advisor or as attribute of student

So, this is a common technique that will be used frequently in such cases you can have entities versus relationship for example, if we have inform we need to keep information about registration how students register to different sections; then we could represent registration as an entity set and have different relationships of section registration which specify how registration is related to section and student reg; which specify how do the station is related to students to represent that kind of information.

We can have placement of relationship attributes also attribute date we have talked about as an attribute of adviser to designate as when that particular instructor became adviser of a student is a common situation that we have already seen.

(Refer Slide Time: 28:25)

**Binary Vs. Non-Binary Relationships**

- Although it is possible to replace any non-binary ( $n$ -ary, for  $n > 2$ ) relationship set by a number of distinct binary relationship sets, a  $n$ -ary relationship set shows more clearly that several entities participate in a single relationship
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - For example, a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being known)
  - But there are some relationships that are naturally non-binary
    - Example: *proj\_guide*

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

15.21 ©Silberschatz, Korth and Sudarshan

There is also question of the choice being made between the binary and non binary relationship ternary or higher degree. Now as it turns out that it is possible that you could represent ternary relationships directly or you can decompose that. For example, a ternary relationship can be decomposed in terms of two binary relationship; for example, let us say if we talk about persons then person every person has parents; so, he or she has a father and a mother.

Now, if we represent this as a ternary relationship then the one difficulty that we have that a person must have both the father and mother to be represented there. For example, if we can come to a situation where only the mother is known, the father is not known I will not be able to represent that because it will always have to come as a triplet of three persons thel the person under consideration her father and her mother, but if I represent the person and the father in one relationship; the person and the mother in another relationship, then I can take care of the situation where when one of the parents are known; I can still represent this.

So, there are certain tradeoffs which can be done between the choice of binary and non binary relationships, but; obviously, there are certain relationships which are inherently non binary for example, the project guide example we have seen. The project guide information cannot be decomposed without certain loss of information to be represented

by say the instructor and the project and another relationship between the student and the project it really that does not represent the same information.

(Refer Slide Time: 30:20)

**Converting Non-Binary Relationships to Binary Form**

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
- Replace  $R$  between entity sets  $A$ ,  $B$  and  $C$  by an entity set  $E$ , and three relationship sets:
  - $R_A$ , relating  $E$  and  $A$
  - $R_B$ , relating  $E$  and  $B$
  - $R_C$ , relating  $E$  and  $C$
- Create an identifying attribute for  $E$  and add any attributes of  $R$  to  $E$
- For each relationship  $(a_i, b_i, c_i)$  in  $R$ , create
  - a new entity  $e_i$  in the entity set  $E$
  - add  $(e_i, a_i)$  to  $R_A$
  - add  $(e_i, b_i)$  to  $R_B$
  - add  $(e_i, c_i)$  to  $R_C$

The diagram (a) shows a ternary relationship  $R$  connecting entities  $A$ ,  $B$ , and  $C$ . Diagram (b) shows the decomposition into binary relationships:  $R_A$  connects  $E$  and  $A$ ;  $R_B$  connects  $E$  and  $B$ ; and  $R_C$  connects  $E$  and  $C$ .

So, in general you can convert a non binary relationship by in the binary form by doing this. So, this is a ternary relationship being shown and for doing that these are the three entity sets involving the ternary relationship and to make decompose into a ternary relationship; what we do is into binary relationships we inject a new entity artificial entity set  $E$  and then we define three different relations between them.

So, which individually relates to the entity sets  $A$ ,  $B$  and  $C$ . So, this is a standard decomposition and you can easily understand that  $A$ ,  $B$  and  $C$  in our earlier example could all be persons and  $R_A$  could mean that father of  $R_B$  could mean mother of and so, on. So, I can do it in decompose it in this manner and represent that.

(Refer Slide Time: 31:24)

The slide has a header 'Converting Non-Binary Relationships (Cont.)' with a small sailboat icon. The main content is a bulleted list:

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of  $R$ 
    - Exercise: add constraints to the relationships  $R_A$ ,  $R_B$  and  $R_C$  to ensure that a newly created entity corresponds to exactly one entity in each of entity sets  $A$ ,  $B$  and  $C$
  - We can avoid creating an identifying attribute by making  $E$  a weak entity set (described shortly) identified by the three relationship sets

On the left margin, vertical text reads: SWAYAM, NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018 Database System Concepts. At the bottom right are navigation icons and the text ©Silberschatz, Korth and Sudarshan.

Now while we do this decomposition we will also have to remember in that; we need to translate all constraints that are present for the ternary relationship. And often times it may become difficult to translate all constraints, it may not be possible and there may be instances in the translated schema that cannot correspond to an instance of the original relationship.

So, we will have to avoid we can we will have to take care of this situation by identifying attributes. And making use of the weak entity sets which we have already seen in our earlier discussions.

(Refer Slide Time: 32:09)

The slide has a header 'E-R Design Decisions' in red. On the left, there is a small sailboat icon and some vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018'. The main content is a bulleted list of design decisions:

- The use of an attribute or entity set to represent an object
- Whether a real-world concept is best expressed by an entity set or a relationship set
- The use of a ternary relationship versus a pair of binary relationships
- The use of a strong or weak entity set
- The use of specialization/generalization – contributes to modularity in the design
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure

At the bottom, there is a video frame showing a man speaking, the text 'Database System Concepts', the number '15.24', and a navigation bar.

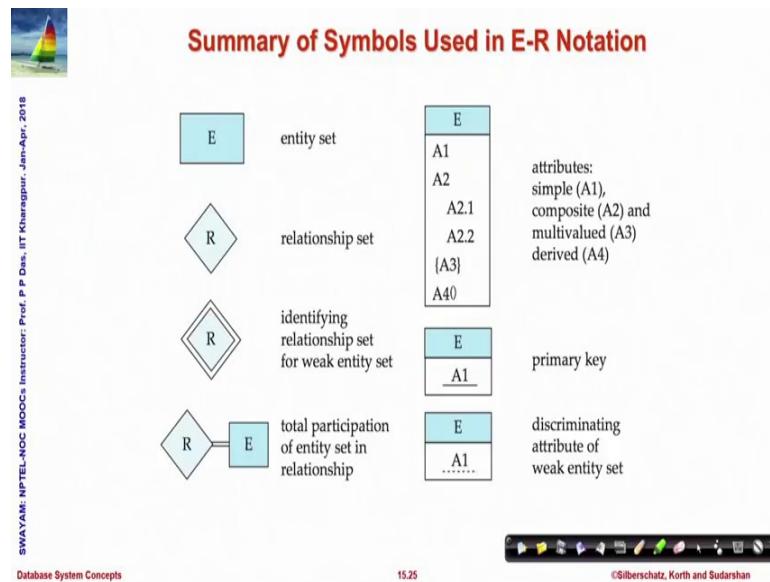
So, if we summarize the discussions on the E-R design decisions; we see that the first decision that we need to take in case of design is the use of an attribute or entity set to represent the object.

So, that is the first modeling that what is the concept and what are the attributes or what is the representing entity set for the object that we are trying to deal with instructor, student project and so, on. And we will also have to see whether in the real world this actually is an entity set or it is a relationship set that it is not a concept by itself, but is a concept which relates two or more entity sets and thereby becomes a set of representation.

The use of ternary relationship versus a pair of binary relationship; this trade off will have to be weighed as a design consideration; we have to look into the use of strong or weak entity set. So, we will have to identify the weak entity sets and see if they should be represented through the identifying relation as against a strong entity set. We have to identify the specialization generalization situation where so, that we can get more specific information and create appropriate modularity in the design.

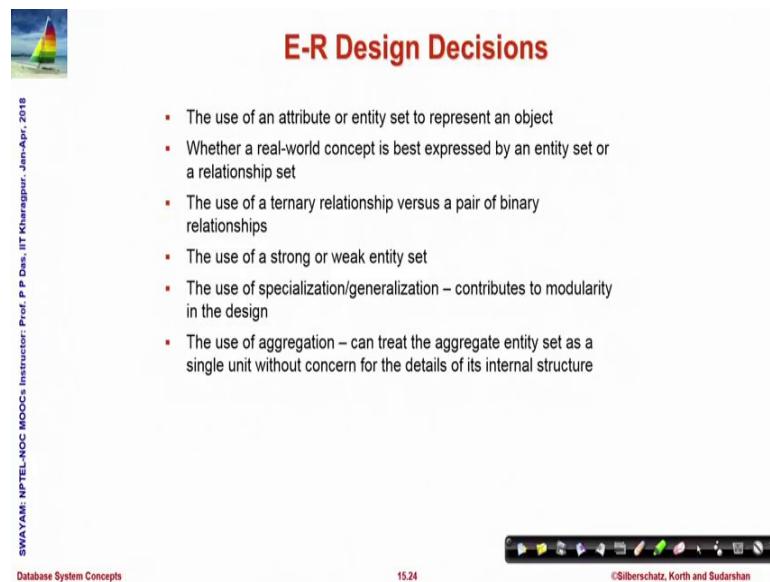
We have to look at aggregation which where we can aggregate entity sets bound by a relationship and create an abstract single unit which can play a role of an independent entity set in the whole design.

(Refer Slide Time: 34:00)



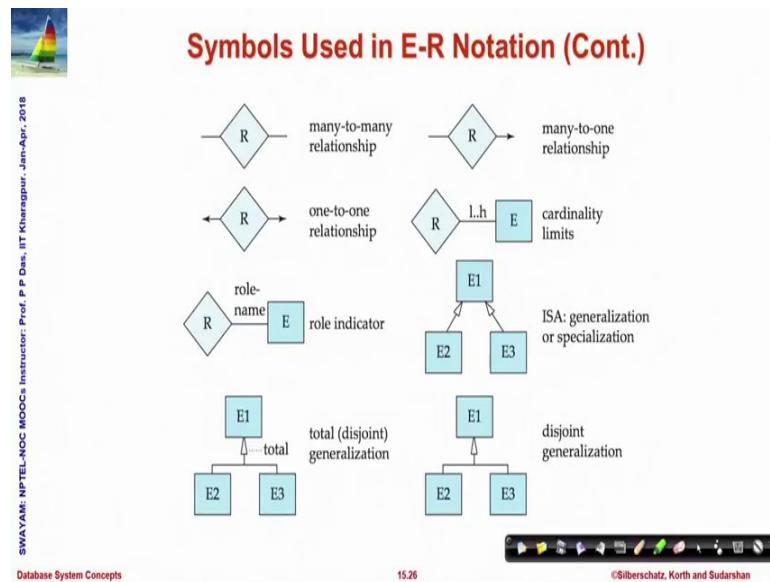
So, this is the basic. So, these are the basic design decisions that you need to make.

(Refer Slide Time: 34:02)



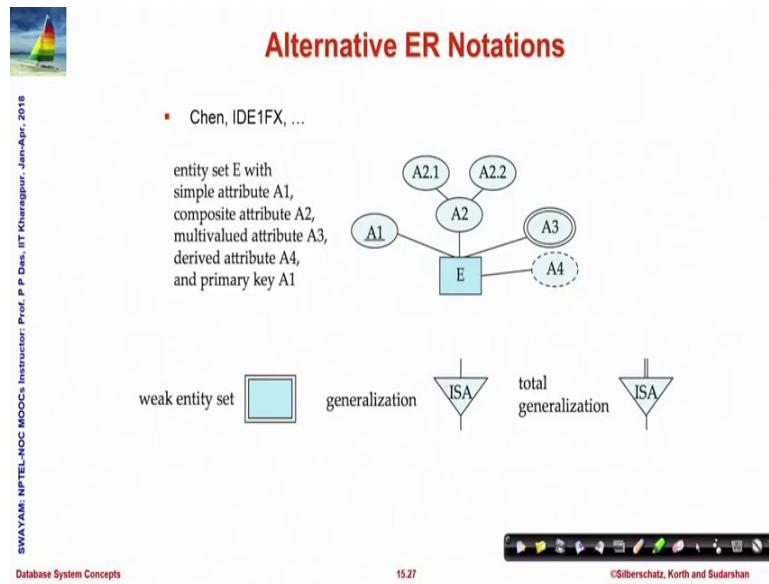
And we will certainly come up with lot more of design decisions as we go along. And before I close in the presentation I have summarized the different symbols that are used in the E-R notation. So, I will not these have already been discussed in depth. So, I will not go through them one by one, but I have put them as a list in the couple of slides.

(Refer Slide Time: 34:30)



This is a next slide in that which will be a quick reference for you while you are initially doing the E-R diagram so, that you know exactly which symbol to pick up for what situation.

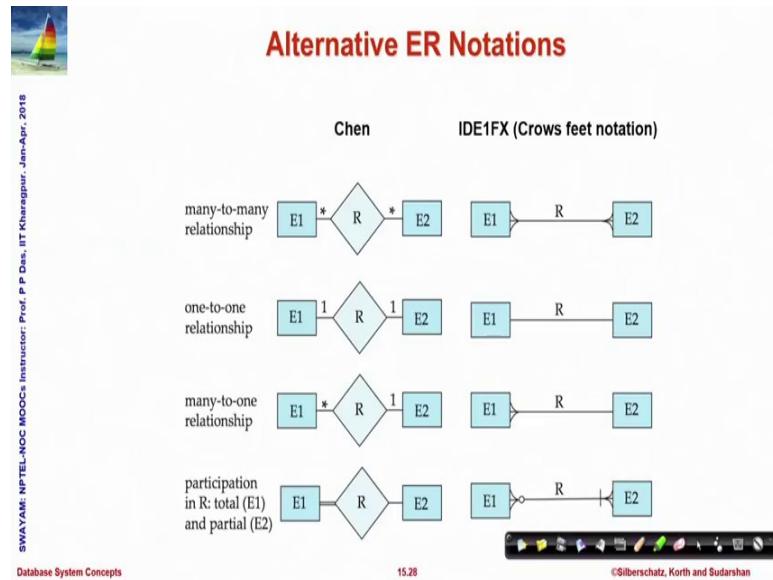
(Refer Slide Time: 34:42)



And at the end also there are few slides which show you that the E-R notation itself is not a unique one.

There are multiple ways to represent similar things for example, this is one which is showing you different composite attributes, the generalization, relationship is shown differently.

(Refer Slide Time: 35:04)



So, there are; these are all different styles of showing the constraints that that apply to a particular relationship. And we will I mean we have included this not because we will use these alternate notations, but I have put them because it is possible that you come across some E-R diagram where these notations are used and if you come across and you are not able to identify then please refer to this slides.

(Refer Slide Time: 35:36)

**Module Summary**

- Discussed the extended features of E-R Model
- Deliberated on various design issues

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

Database System Concepts

15.29

©Silberschatz, Korth and Sudarshan

And you will be able to recognize what is what is corresponding symbol that you already know.

So, in this module we have discussed the extended features of E-R model and we have deliberated on certain design issues. And we will close our discussion on the entity relationship model here and move on to discuss the actual relational design.

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

**Lecture - 16**  
**Relational Database Design**

Welcome to Module 16 of Database Management Systems till the last module which closed with the third week.

(Refer Slide Time: 00:31)

The slide has a header "Week 03 Recap" in red, a small sailboat icon, and a "PPD" watermark. It lists topics covered in Week 3 across four modules:

- Module 11: Advanced SQL**
  - Accessing SQL From a Programming Language
  - Functions and Procedural Constructs
  - Triggers
- Module 12: Formal Relational Query Languages**
  - Relational Algebra
  - Tuple Relational Calculus (Overview only)
  - Domain Relational Calculus (Overview only)
  - Equivalence of Algebra and Calculus
- Module 13: Entity-Relationship Model/1**
  - Design Process
  - E-R Model
- Module 14: Entity-Relationship Model/2**
  - E-R Diagram
  - E-R Model to Relational Schema
- Module 15: Entity-Relationship Model/3**
  - Extended E-R Features
  - Design Issues

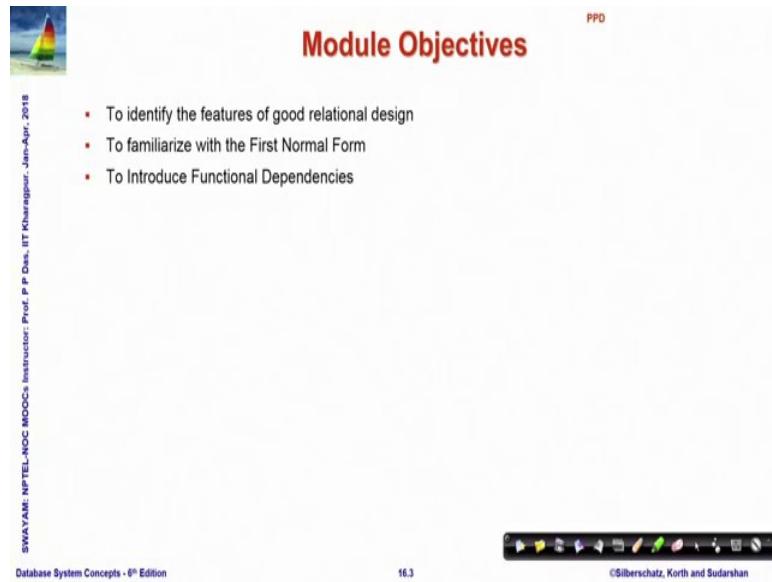
Vertical text on the left: SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr., 2018

Bottom footer: Database System Concepts - 8<sup>th</sup> Edition, 16.2, ©Silberschatz, Korth and Sudarshan

Specifically in the third week, we talked about certain advanced features of SQL and the formal query language in terms of relational and algebra and calculi and then, we talked in a depth in terms of the entity relationship model, the first basic conceptual level representation of the real world that we can do in terms of designing a system.

Now, our next task would be to take it to more proper complete relational database design and this will have a lot of theory at different levels that we need to understand. We will slowly develop that and this discussion will span five modules that is we will take the whole week to complete.

(Refer Slide Time: 01:25)



**Module Objectives**

PPD

- To identify the features of good relational design
- To familiarize with the First Normal Form
- To Introduce Functional Dependencies

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Doshi, IIT Kharagpur - Jan-Apr - 2018

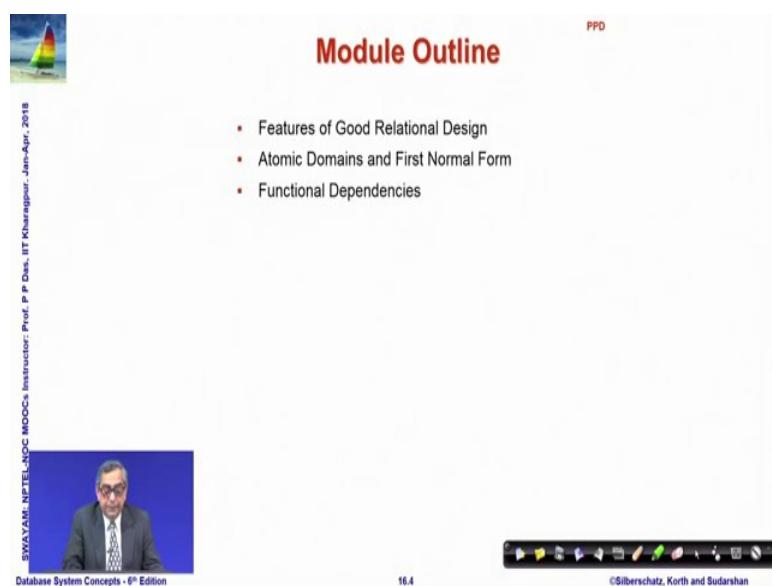
Database System Concepts - 8<sup>th</sup> Edition

16.3

©Silberschatz, Korth and Sudarshan

So, the objective of the current module, the first of the Relational Design Module is to identify features of good relational design having done the ER model. We have ER model we do the ER model, we have the entity sets relationships, we convert them to schema. We have seen how to do that and immediately we have some design, but the question is, is it a good design. So, we will discuss about what are the features of a good design and then, we will introduce the formal definition of what is First Normal Form and we will introduce a very critical concept of relational database design, the Functional Dependencies.

(Refer Slide Time: 02:07)



**Module Outline**

PPD

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Functional Dependencies

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Doshi, IIT Kharagpur - Jan-Apr - 2018



Database System Concepts - 8<sup>th</sup> Edition

16.4

©Silberschatz, Korth and Sudarshan

These are the module outline for that.

(Refer Slide Time: 02:10)

The slide features a small sailboat icon in the top left corner. In the top right, the text 'PPD' is written above a bulleted list: '•Features of Good Relational Design', '•Atomic Domains and First Normal Form', and '•Functional Dependencies'. The main title 'FEATURES OF GOOD RELATIONAL DESIGN' is centered in large red capital letters. Below the title, the text 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr., 2018' is visible. At the bottom, there is a navigation bar with icons and the text 'Database System Concepts - 8<sup>th</sup> Edition', '16.5', and '©Silberschatz, Korth and Sudarshan'.

So, to start with the features of good relational design, let us take an example.

(Refer Slide Time: 02:13)

The slide shows a table with columns: ID, name, salary, dept\_name, building, and budget. Handwritten annotations include a blue arrow pointing to the 'dept\_name' column, a blue circle around the 'dept\_name' column header, and a blue bracket spanning the 'dept\_name' and 'building' columns. To the right of the table, handwritten notes say 'Redundancy' with an arrow pointing to the 'dept\_name' column, 'Anomaly' with an arrow pointing to the 'building' column, and 'Update insertion deletion' with an arrow pointing to the 'dept\_name' column. The slide also includes the title 'Combine Schemas?' in red, the text 'Suppose we combine instructor and department into inst\_dept', '(No connection to relationship set inst\_dept)', and the footer information 'Database System Concepts - 8<sup>th</sup> Edition', '16.6', and '©Silberschatz, Korth and Sudarshan'.

Suppose we have seen the instructor relation, instructor entity set as a relation. You have seen the department relation. Now, let us consider that if these two were not two separate relations, if they were all kept in a common relation that is all the attributes are kept in the common relation, so earlier if you recall that your instructor relation was this and your department relation was this much. So, if we keep everything together, of course we

are calling it `inst_dept`, but please keep in mind this is not the same `inst_dept` that we discussed in terms of the ER model. This is just putting these two together.

Now, the question is if you look into this data carefully, for example if you look into this particular row, if you look into this particular row and if you look into this particular row, these are rows of instructors who all belong to computer science. Now, earlier we were representing the information of instructor only in this part. So, we just knew that it is computer science and we represent the information of department in this part. So, given a department name say computer science, we knew, where is it located, the building and what budget it has. Now, when we are combined, we will see that naturally since computer science is located in the Taylor building, we know that it has a budget of say 100,000. So, all of these records will have this information repeated.

So, this is not a very good situation. This is not a good situation because this kind of situation is typically in database known as redundancy, that is you have the same data in multiple places. So, what is the consequence of redundancy? For example, there could be different kinds of anomaly when you have redundancy. What is an anomaly? An anomaly is the possibility of certain data getting inconsistent. For example, let us say Computer Science department moves from Taylor building to Painter building. Now, what will happen if it moves to painter building? Then, I will need to remove this, make it a painter, make this value painter. I have to also do this, make this painter. I have to also do this, make this painter. So, if I have a change, then I will have to make the change at multiple entries. Think about the earlier situation where I just had these three in my department relation, then naturally computer science had only one row and therefore, this change, this update could be done at only one place.

So, it is not only that if while doing this in case of this redundancy, I have to do this multiple times. It also has the difficulty that if I forget to update any one of them or more of them, then I have inconsistent data. Similarly, if I want to insert a new value, I will have to do that for all this redundant information. If I have to delete say for some reason let say the university decides to wind up the Physics department, then I have to delete all these rows which have physics as an entry and the consequence of that is the department is deleted, but as a consequence of that I will delete the whole row and therefore, I will not only remove the department, but I will also remove the corresponding instructor who was enrolled for that department.

So, this kind of redundancy can lead to different kinds of anomalies in a database design. On the other hand, if you look at, well why am I complicating the whole situation? We have already had a good design in terms of where these anomalies were, not their departments were separate instructor was separate. In that case, the situation is that to answer some of the queries, I may have to do a very expensive join operation. For example, if I want to know if Einstein wants to know what is the budget of his department that cannot be found out from the earlier instructor database, instructor relation which had only these fields.

So, I have to pick up Einstein from here, do a join based on the department name, dept\_name with the department table department relation and then only, I will be able to find out that an Einstein belongs to Physics. Physics has a budget of 70000. So, Einstein's department has a budget 70000. So, there is a tradeoff between how much data information if you make redundant and lead to different anomalous situations or how much data you optimize in the representation, but get into the possible situation of having a higher cost in terms of answering your queries.

(Refer Slide Time: 07:49)

**Combine Schemas?**

- Suppose we combine *instructor* and *department* into *inst\_dept*
  - (No connection to relationship set *inst\_dept*)
- Result is possible repetition of information (*building* and *budget* against *dept\_name*)

ID	name	salary	dept_name	building	budget
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

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

Database System Concepts - 8<sup>th</sup> Edition      16.6      ©Silberschatz, Korth and Sudarshan

So, this is one of the core design issues that we will start with. So, let us look into some more.

(Refer Slide Time: 07:54)



## A Combined Schema Without Repetition

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

- Consider combining relations
  - $\text{sec\_class(sec\_id, building, room\_number)}$  and
  - $\text{section(course\_id, sec\_id, semester, year)}$into one relation
  - $\text{section(course\_id, sec\_id, semester, year, building, room\_number)}$
- No repetition in this case

Database System Concepts - 8<sup>th</sup> Edition

16.7

©Silberschatz, Korth and Sudarshan



Of these examples, let us say we look into another combined combination of schema. Suppose section is a relation which have the sections of a course which give the section id, semester, year and say section class is another relation which tell me for a section id, what is the building and room number where it is located. So, if we have this kind of relations combined into a common relation, then I have all of these coming from the section and this and these coming from the section class, but we can see that there is no repetition or redundant information in this case.

So, it is note that the combining schemas is necessarily always bad in terms of repetition or in terms of redundancy. So, different situations will have to be assessed.

(Refer Slide Time: 08:57)



## What About Smaller Schemas?

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

- Suppose we had started with  $\text{inst\_dept}$ . How would we know to split up (**decompose**) it into  $\text{instructor}$  and  $\text{department}$ ?
- Write a rule "if there were a schema  $(\text{dept\_name}, \text{building}, \text{budget})$ , then  $\text{dept\_name}$  would be a candidate key"
- Denote as a **functional dependency**:  
$$\text{dept\_name} \rightarrow \text{building, budget}$$
- In  $\text{inst\_dept}$ , because  $\text{dept\_name}$  is not a candidate key, the building and budget of a department may have to be repeated.
  - This indicates the need to decompose  $\text{inst\_dept}$
- Not all decompositions are good. Suppose we decompose  $\text{employee}(ID, name, street, city, salary)$  into
  - $\text{employee1}(ID, name)$
  - $\text{employee2}(name, street, city, salary)$
- The next slide shows how we lose information -- we cannot reconstruct the original  $\text{employee}$  relation -- and so, this is a **lossy decomposition**.

Database System Concepts - 8<sup>th</sup> Edition

16.8

©Silberschatz, Korth and Sudarshan



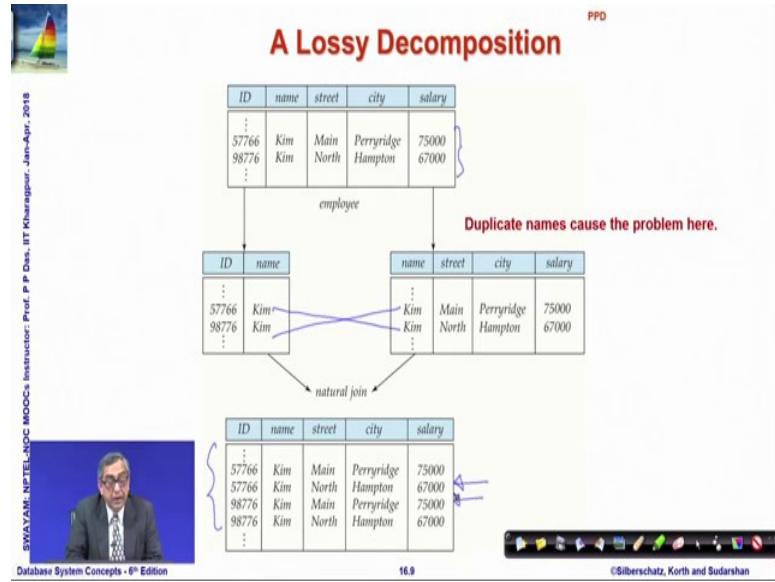
So, if we want to look at the other side that if we just as I said that if we make the schema smaller, so that we avoid redundancy and then, what we see that 12 from the combined `inst_dept` relationship that we saw. So, let me just show you once more. So, this is if we look at the `inst_dept`, then in this we can we know that from the earlier information about the department relationship that department name is a key, is a primary key of the relation which has department name, building and budget. What is the consequence of being a primary key? If it is a primary key, then no two records can match on the department name and be different in terms of the building and the budget.

If two records are there which have the same department name, they must be identical. So, they are distinguishable completely by that. So, let us see what is the consequence of this. So, we are saying that we write it as a rule that if there is a schema department, name, building, budget, then department name would be a candidate key and we write this observation that if two records match on the department name, they must match on the building and budget and very loosely, we will come to the formal definition. Very loosely we call this the functional dependency. We say that the building and budget is functionally dependent on the department name and that is a situation where we can split this `inst_dept` and create a smaller relationship because department name is not a candidate key in the `inst_dept`. It does not decide the records of `inst_dept` uniquely.

So, since it does not, so when the values of this key, this attribute department name is duplicated or triplicated, the values of the building and budget are repeated and we have the redundancy. So, this is a situation, very common situation which is indicative of the fact that we need a decomposition into smaller, but at the same time we can also observe, I mean let us take a different example. If we are thinking that decomposition is the panacea of solving these kind of redundancy and related problems, then let us try to see a different relationship `employee` which has id, name, street, city, salary and we want to make it smaller and want to make two relations id and name and name, city street, salary.

So, if we do that, then how do we get the salary for a particular id? We will naturally have to join these two relations in terms of the common attribute name. We have seen that in the query and the question is when I do this join, do I get back the original information or I lose some information.

(Refer Slide Time: 12:42)



Look at an example. So, here is an example of the combined instance and I have two different ids, but incidentally the names are same. The names of these two distinct employees are same. So, when I decompose, I get this relation which shows id and name. I get this relation which against the name shows this, but when I try to join them by natural join, I not only get the combination of this with this which is what I need, but I also get this combination. So, if I say this is what I get as well in terms of natural join, this is what I get as well in terms of the natural join which are really not there in the original relation.

So, you can see that in the natural join, I get four records, I get four rows whereas, in the original one I had only two rows. So, I get some entries which are actually erroneous. These are not there in the database. So, this is when this happens. We say that we have loss of information and such joins are said to be lossy joins. So, when we decompose, we need to make sure that our joins are lossless in nature; otherwise that is not a good design.

(Refer Slide Time: 14:08)



### Example of Lossless-Join Decomposition

SWAYAM NPTEL-NC MOOCs Instructor: Prof. P. Desai, IIT Kharagpur - Jan-Apr. 2018

- Lossless join decomposition
- Decomposition of  $R = (A, B, C)$   
 $R_1 = (A, B)$      $R_2 = (B, C)$

A	B	C
$\alpha$	1	A
$\beta$	2	B

$r$

A	B
$\alpha$	1
$\beta$	2

$\Pi_{A,B}(r)$

B	C
1	A
2	B

$\Pi_{B,C}(r)$

$\Pi_A(r) \bowtie \Pi_B(r)$

A	B	C
$\alpha$	1	A
$\beta$	2	B



So, you can see this is again a hypothetical example which shows three attributes in relation having three attributes. You have decomposed it into two relations having two attributes each and we have shown an instance and in this case, it shows that when I take the join, the original information I am sorry, wait.

When I take the join, the original information is completely retrieved. I get back the same table and when that happens, I say that the join is lossless. So, what we need to understand is on one side there is a need to decompose relations into smaller relations to reduce redundancy and while we do that, we will also have to keep this in mind that the smaller relations must be composable through certain natural join procedure to the original relation, and I must get back that original relation, otherwise I have a lossy join which is not acceptable. Also, the decomposition will have the costs of doing natural join every time I want to answer those queries.

(Refer Slide Time: 15:35)



PPD

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Functional Dependencies

## ATOMIC DOMAINS AND FIRST NORMAL FORM

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition

16.11

©Silberschatz, Korth and Sudarshan

The next that we look at is the way the relationships are categorized as First Normal Form.

(Refer Slide Time: 15:45)



### First Normal Form (1NF)

PPD

- Domain is **atomic** if its elements are considered to be indivisible units
  - Examples of non-atomic domains:
    - Set of names, composite attributes
    - Identification numbers like CS101 that can be broken up into parts
- A relational schema R is in **first normal form** if
  - the domains of all attributes of R are atomic
  - the value of each attribute contains only a single value from that domain
- Non-atomic values complicate storage and encourage redundant (repeated) storage of data
  - Example: Set of accounts stored with each customer, and set of owners stored with each account
  - We assume all relations are in first normal form

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition

16.12

©Silberschatz, Korth and Sudarshan

We consider that the domains of attributes are atomic if they are indivisible. So, anything that is a number, string and so on is considered to be atomic and we say a relational schema is in its First Normal Form if the domains of all attributes are atomic and all attributes single valued, there is no multi valued attribute. If these conditions are satisfied, then we will say that every relate that relational schema is in its First Normal Form. So, we will slowly understand the purpose of defining such normal forms, but let us initially understand the definition. So, if we have attributes which are composite in

nature, naturally my relationship, my relational schema is not in First Normal Form if we have attributes which are multiple valued, it is not so.

(Refer Slide Time: 16:44)

The slide has a decorative header with a sailboat icon and the title 'First Normal Form (Cont'd)' in red. On the left, there is vertical text: 'SWAYAM-NETELNOC-MOOCs', 'Instructor: Prof. P P Das', 'Prof. P P Das, IIT Kharagpur - Jan-Apr. 2018', and 'Database System Concepts - 8th Edition'. The main content is a bulleted list:

- Atomicity is actually a property of how the elements of the domain are used
  - Example: Strings would normally be considered indivisible
  - Suppose that students are given roll numbers which are strings of the form CS0012 or EE1127
  - If the first two characters are extracted to find the department, the domain of roll numbers is not atomic
  - Doing so is a bad idea: leads to encoding of information in application program rather than in the database

At the bottom, there is a navigation bar with icons for back, forward, search, and other presentation controls. The page number '16.13' is at the bottom center, and the copyright notice '©Silberschatz, Korth and Sudarshan' is at the bottom right.

So, if we say that we have possible values are like this, then if we just treat them as strings, then the corresponding relational schema is in First Normal Form, but if we say that from this string we can extract the first two characters which is CS which tells me what is a department. The next four characters gives me a number, the serial number of the particular student in the role. Then I am not actually using an atomic domain because my domain needs to be interpreted separately than just being a value. So, these are not parts of what can be a First Normal Form.

(Refer Slide Time: 17:28)



## First Normal Form (Cont'd)

PPD

- The following is not in 1NF

Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025 192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53; 182-929-2929
789	John	Doe	555-808-9633

- A telephone number is composite
- Telephone number is multi-valued

Source: [https://en.wikipedia.org/wiki/First\\_normal\\_form](https://en.wikipedia.org/wiki/First_normal_form)

Database System Concepts - 8<sup>th</sup> Edition

16.14

©Silberschatz, Korth and Sudarshan



So, I have given some examples of what is not and what is First Normal Form. So, this is an example where at the telephone number field exists and there can be multiple telephone numbers. So, this is not in First Normal Form because the telephone number itself is composite because it has different components and also, you can have multiple telephone number. So, this relation is not in the First Normal Form.

(Refer Slide Time: 17:54)



## First Normal Form (Cont'd)

PPD

- Consider:

Customer

Customer ID	First Name	Surname	Telephone Number1	Telephone Number2
123	Pooja	Singh	555-861-2025	192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53	182-929-2929
789	John	Doe	555-808-9633	

- Is in 1NF if telephone number is not considered composite
- However, conceptually, we have two attributes for the same concept
  - Arbitrary and meaningless ordering of attributes
  - How to search telephone numbers
  - Why only two numbers?

Source: [https://en.wikipedia.org/wiki/First\\_normal\\_form](https://en.wikipedia.org/wiki/First_normal_form)

Database System Concepts - 8<sup>th</sup> Edition

16.15

©Silberschatz, Korth and Sudarshan



What you can do? You can separate out these phone numbers into two different attributes; Telephone number 1 and 2. Even then it is not exactly in First Normal Form because you do not know in which order they should be handled. If you have to search for a telephone number, then you will have to search multiple attributes which are

conceptually same and then, the question is why only two attributes. Cannot anybody have 3 phone numbers, 7 phone numbers and so on. So, this is really not a good option.

(Refer Slide Time: 18:26)

**First Normal Form (Cont'd)**

- Is the following in 1NF?

Customer			
Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025
123	Pooja	Singh	192-122-1111
456	San	Zhang	182-929-2929
456	San	Zhang	(555) 403-1659 Ext. 53
789	John	Doe	555-808-9633

- Duplicated information
- ID is no more the key. Key is (ID, Telephone Number)

Source: [https://en.wikipedia.org/wiki/First\\_normal\\_form](https://en.wikipedia.org/wiki/First_normal_form)

Database System Concepts - 8<sup>th</sup> Edition

16.16

©Silberschatz, Korth and Sudarshan

So, the other way could be that for every telephone number, you introduce a separate row. Once you do that you already know you have redundancy and you have possibilities of varied kinds of anomalies that could happen.

(Refer Slide Time: 18:40)

**First Normal Form (Cont'd)**

- Better to have 2 relations:

Customer Name			Customer Telephone Number	
Customer ID	First Name	Surname	Customer ID	Telephone Number
123	Pooja	Singh	123	555-861-2025
456	San	Zhang	123	192-122-1111
789	John	Doe	456	(555) 403-1659 Ext. 53
			456	182-929-2929
			789	555-808-9633

- One-to-Many relationship between parent and child relations
- Incidentally, satisfies 2NF and 3NF

Source: [https://en.wikipedia.org/wiki/First\\_normal\\_form](https://en.wikipedia.org/wiki/First_normal_form)

Database System Concepts - 8<sup>th</sup> Edition

16.17

©Silberschatz, Korth and Sudarshan

So, one way it could be achieved is we follow the principle that we had seen in ER modelling that this multivalued dependency can be represented in terms of a separate

relation where against the customer id we just keep the telephone number. So, we can keep multiple of them and we take that out from the customer name. So, one to many relationship between the parent and the child, between the customer name and telephone number, every customer may have more than one telephone number is possible and that makes it 1 NF relation, First Normal Form relation and we will later on see that it also is 2 NF and 3 NF, but that is a future story.

(Refer Slide Time: 19:29)

The slide features a sailboat icon in the top left corner. In the top right corner, the letters 'PPD' are written in red. Below the title, there is a bulleted list of topics:

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Functional Dependencies

At the bottom of the slide, there is a navigation bar with icons for back, forward, search, and other presentation controls. The footer contains the text 'SWAYAM: NPTEL-NOC MOOCs', 'Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr. 2018', 'Database System Concepts - 6<sup>th</sup> Edition', '16.18', and '©Silberschatz, Korth and Sudarshan'.

## FUNCTIONAL DEPENDENCIES

Now, finally we come to the core of what the mathematical formulation which dictates much of the data base, relational database design is known as functional dependencies.

(Refer Slide Time: 19:47)



## Goal — Devise a Theory for the Following

SWAYAM NPTEL-NCX MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018

- Decide whether a particular relation  $R$  is in "good" form.
- In the case that a relation  $R$  is not in "good" form, decompose it into a set of relations  $\{R_1, R_2, \dots, R_n\}$  such that
  - each relation is in good form ✓
  - the decomposition is a lossless-join decomposition

$$\begin{aligned} R_i &= \text{set of attributes} \\ R &= \bigcup_i R_i \end{aligned}$$

Database System Concepts - 8<sup>th</sup> Edition

16.19

©Silberschatz, Korth and Sudarshan



I just talked about little bit of that while talking about department name building and budget. Now, to decide whether a particular relation is good or rather a particular relational scheme is good, we need to check against certain measures and if it is not good, we need to decompose it into a set of relations such that these conditions satisfy that every, each one of these,  $\{R_1, R_2, \dots, R_n\}$ . So, I mean if you have you now got rusted, then it is basically  $R_i$  is a set of attributes because it is a relational schema. A relational schema is a set of attributes.

So, naturally  $R$  will be the union of all of these,  $R_i$  the total set of attributes. So, instead of keeping all the information into one relation in one table, we are basically decomposing it into  $n$  different schemas.

(Refer Slide Time: 20:42)



## Goal — Devise a Theory for the Following

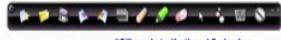
SWAYAM-NPTEL-NOOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018

- Decide whether a particular relation  $R$  is in "good" form.
- In the case that a relation  $R$  is not in "good" form, decompose it into a set of relations ( $R_1, R_2, \dots, R_n$ ) such that
  - each relation is in good form
  - the decomposition is a lossless-join decomposition
- Our theory is based on:
  - functional dependencies
  - multivalued dependencies

Database System Concepts - 8<sup>th</sup> Edition

16.19

©Silberschatz, Korth and Sudarshan



So, what we need to guarantee is each one of these relation  $R_1, R_2, \dots, R_n$  is in good form. How do I get back the original relation? Original relation that was represented by all that attributes enough is to take a lossless join. This would take a join and that this decomposition must give me a lossless join. So, to ensure that; we make use of two key ideas more foundationally; functional dependencies and then, multivalued dependencies.

(Refer Slide Time: 21:20)



## Functional Dependencies

SWAYAM-NPTEL-NOOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018

- Constraints on the set of legal relations
- Require that the value for a certain set of attributes determines uniquely the value for another set of attributes
- A functional dependency is a generalization of the notion of a key



16.20

©Silberschatz, Korth and Sudarshan



A functional dependency is a constraint on the set of legal relation. So, mind you it is a constraint on the schema and once that constraint is defined, it must hold for all relations that the schema satisfied. So, here we need that the value of certain set of attributes uniquely determine the value of another set of attributes. So, I know the value of three

attributes, I should be able to say that the values of the other four attributes would be fixed. So, you have already seen this notion in terms of key or super key. You have seen that similar type of concept exists where we said a key is a set of attributes, so that if the values of two rows are identical over these set of attributes, then the two peoples, the two rows must be totally identical.

So, key is something which does a similar thing as a functional dependency, but is more specific. Functional dependencies are generalization.

(Refer Slide Time: 22:30)

The slide has a title 'Functional Dependencies (Cont.)' in red at the top right. On the left, there is a small logo of a sailboat on water. The main content area contains the following text and a handwritten note:

- Let  $R$  be a relation schema
- $\alpha \subseteq R$  and  $\beta \subseteq R$
- The **functional dependency**

$\alpha \rightarrow \beta$

**holds on  $R$**  if and only if for any legal relations  $r(R)$ , whenever any two tuples  $t_1$  and  $t_2$  of  $r$  agree on the attributes  $\alpha$ , they also agree on the attributes  $\beta$ . That is,

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

A handwritten note in blue ink shows a diagram with two sets of brackets, one above the other, with arrows pointing from the first bracket to the second, illustrating the mapping between attributes  $\alpha$  and  $\beta$ .

At the bottom, there is footer text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr., 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '16.21', and '©Silberschatz, Korth and Sudarshan'.

So, let us formally define that let  $R$  be a relational schema which means that it is a set of attributes and let us say  $\alpha, \beta \subseteq R$ , then we write this and note this notation.  $\alpha$  is a set of attributes;  $\beta$  is another set of attributes. Both are subset of the same  $R$  and we say  $\alpha \rightarrow \beta$  that is if I know the value of a tuple over the attributes of  $\alpha$ , then the values of that tuple over the attributes of  $\beta$  would be fixed or in other words, they say that if I have two tuples  $t_1$  and  $t_2$  and their values over the set of  $\alpha$  attributes are same, then necessarily their values over the set of  $\beta$  attributes must be same and mind you this is something which is a design constraint. It is not just an incidental property. It is not just the fact that a particular instance of a schema satisfies this, but when you say this is a functional dependency, we need all possible past, present and future instances of the schema must satisfy this.

(Refer Slide Time: 24:03)



## Functional Dependencies (Cont.)

- Let  $R$  be a relation schema  
 $\alpha \subseteq R$  and  $\beta \subseteq R$
- The **functional dependency**  
 $\alpha \rightarrow \beta$   
holds on  $R$  if and only if for any legal relations  $r(R)$ , whenever any two tuples  $t_1$  and  $t_2$  of  $r$  agree on the attributes  $\alpha$ , they also agree on the attributes  $\beta$ . That is,  
 $t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$
- Example: Consider  $r(A,B)$  with the following instance of  $r$ .

1	4
1	5
3	7

- On this instance,  $A \rightarrow B$  does NOT hold, but  $B \rightarrow A$  does hold.



So, consider this if you take a relation, a schema with an instance as given here between two attributes **A** and **B**, then we can say at least given this instance not we still do not know what happens in the whole schema for all instances, but on this instance we can say that  $A \rightarrow B$  does not hold because between the first and the second record, the value of **A** is same one, but the value of **B** are different 4 and 5, but we can certainly say that on this instance at least  $B \rightarrow A$  holds because whenever the value if we take any two tuples, their value over **B** does not at all match. If they does not match, then naturally there is no question of what happens to the value of the tuple over the set of attributes **A**. So, we will say that  $B \rightarrow A$  holds in this instance.

(Refer Slide Time: 25:01)



## Functional Dependencies (Cont.)

- $K$  is a superkey for relation schema  $R$  if and only if  $K \rightarrow R$
- $K$  is a candidate key for  $R$  if and only if
  - $K \rightarrow R$ , and
  - for no  $\alpha \subset K$ ,  $\alpha \rightarrow R$
- Functional dependencies allow us to express constraints that cannot be expressed using superkeys. Consider the schema:  
 $inst\_dept (ID, name, salary, dept\_name, building, budget)$ .

We expect these functional dependencies to hold:

$$dept\_name \rightarrow building$$

and  $ID \rightarrow building$

but would not expect the following to hold:

$$dept\_name \rightarrow salary$$



So, given this definition of functional dependency, now we can have a formal definition of what the super key is. Super key is naturally a subset of attributes which → the whole set and a candidate key is a super key which is minimal which means that  $k$  is a candidate key. If the two conditions have to satisfy this condition say there is a super key that it → all the attributes and the other condition says minimality that there is no subset  $\alpha \subset k$ , such that  $\alpha \rightarrow R$  if there exists a subset  $\alpha \subset k$ , the proper subset  $\alpha \subset k$ . So, that  $\alpha \rightarrow R$ , then  $k$  would not be a candidate key. We will have to check for  $\alpha$ . So, these two; what we had stated earlier in qualitative terms and now mathematically established. So, we can say that there are different functional dependencies. For example, inst\_dept combined relation if we look at, then we know that **department name** → **building** functionally.

So, these are functional dependencies that must hold, but certainly we would not expect department name to functionally determine salary. That would be too much, right. So, functional dependencies are facts about the real world that we try to understand from the real world and then, represent in terms of the functional dependency formulation in the database.

(Refer Slide Time: 26:41)

The slide has a title 'Use of Functional Dependencies' in red at the top center. To the left is a small logo of a sailboat on water. On the right is a decorative footer bar with icons for navigation and search. The main content area contains a bulleted list of points:

- We use functional dependencies to:
  - test relations to see if they are legal under a given set of functional dependencies.
    - If a relation  $r$  is legal under a set  $F$  of functional dependencies, we say that  $r$  **satisfies**  $F$
  - specify constraints on the set of legal relations
    - We say that  $F$  **holds on**  $R$  if all legal relations on  $R$  satisfy the set of functional dependencies  $F$
- Note: A specific instance of a relation schema may satisfy a functional dependency even if the functional dependency does not hold on all legal instances
  - For example, a specific instance of *instructor* may, by chance, satisfy  $name \rightarrow ID$

At the bottom left, vertical text reads: SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Date, IIT Kharagpur - Jan-Apr., 2018. At the bottom center: Database System Concepts - 8<sup>th</sup> Edition 16.23 ©Silberschatz, Korth and Sudarshan.

So, we can use functional dependencies to test relations if they are valid under the set of functional dependencies. So, there could be multiple functional dependencies in the set and if a relation we are using  $r$  here just to remind you that a relation means that a

particular instance is legal under a set of functional dependencies. We will say that  $r$  satisfies that and if we have that it holds  $F$  will be satisfied by all possible instances of a relational schema  $R$ , then we say  $F$  holds on  $R$ .

So, a relation satisfies a functional set of functional dependencies and a relational schema for a relational schema, the functional dependency set of functional dependencies holds on that schema which means that for all possible past, present and future instances relations, the relations will satisfy the functional dependencies. So, we have for example  $\text{id}$ . We know  $\text{id} \rightarrow \text{name}$  that if the id is distinct, then the name has to be distinct, but we may find that instance where  $\text{name} \rightarrow \text{id}$ . So, we can say that  $\text{name} \rightarrow \text{id}$  is satisfied by a particular instance where it so happens that there is no two rows where the name is identical, but we cannot, may not be able to infer that as this dependency holding on the relational scheme as a whole because tomorrow we can get another entry, so that two rows might match on the name, but could still be distinct entries not matching on  $\text{id}$ .

(Refer Slide Time: 28:46)

The slide features a title 'Functional Dependencies (Cont.)' in red at the top right. On the left is a small logo of a sailboat on water. The main content area contains a bulleted list of points about trivial functional dependencies:

- A functional dependency is **trivial** if it is satisfied by all instances of a relation
  - Example:
    - $ID, name \rightarrow ID$
    - $name \rightarrow name$
  - In general,  $\alpha \rightarrow \beta$  is trivial if  $\beta \subseteq \alpha$

At the bottom of the slide, there is footer text: 'SWAYAM: NPTEL-NOCO Instructor: Prof. P. Das, IIT Kharagpur - Jam-Apr- 2018', 'Database System Concepts - 6<sup>th</sup> Edition', '10.24', and '©Silberschatz, Korth and Sudarshan'.

So, that is how this will have to be looked at in specificity. We say that a functional dependency is trivial if the left hand side is a superset of the right hand side. So, if I have a bigger set of attributes on the left hand side  $ID$  and  $name$ , then obviously  $\text{ID, name} \rightarrow \text{ID}$ ,  $\text{ID, name} \rightarrow \text{name}$ ,  $\text{name} \rightarrow \text{name}$ . So, if you just think about because in a functional dependency the left hand side attributes that tuples have to match on the left hand side attribute and if they do, then they must match on the right hand side attribute.

So, if the right hand side set of attributes is a subset of the left hand side, then obviously the functional dependency will be vacuously true and these are called trivial dependencies.

(Refer Slide Time: 29:33)

**Functional Dependencies (Cont.)**

- Functional dependencies are:

StudentID	Semester	Lecture	TA
1234	6	Numerical Methods	John
1221	4	Numerical Methods	Smith
1234	6	Visual Computing	Bob
1201	2	Numerical Methods	Peter
1201	2	Physics II	Simon

- $\text{StudentID} \rightarrow \text{Semester}$
- $\{\text{StudentID}, \text{Lecture}\} \rightarrow \text{TA}$
- $\{\text{StudentID}, \text{Lecture}\} \rightarrow \{\text{TA}, \text{Semester}\}$

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

Database System Concepts - 8<sup>th</sup> Edition

16.25

©Silberschatz, Korth and Sudarshan

So, in the next couple of slides, I have shown few examples of functional dependencies of different tables. Here **StudentID → Semester** which mean that we are trying to model that a student cannot be at the same time in two semesters, then **(Student ID, lecture) → TA** and so on and you can see for this particular relation **(Student ID, lecture)** also happens to be the candidate key.

(Refer Slide Time: 30:05)

The slide title is "Functional Dependencies (Cont.)". It features a small sailboat icon in the top left corner and the letters "PPD" in the top right corner. The slide content includes a table of employee data:

Employee ID	Employee Name	Department ID	Department Name
0001	John Doe	1	Human Resources
0002	Jane Doe	2	Marketing
0003	John Smith	1	Human Resources
0004	Jane Goodall	3	Sales

Below the table, three functional dependencies are listed:

- $\text{Employee ID} \rightarrow \text{Employee Name}$
- $\text{Employee ID} \rightarrow \text{Department ID}$
- $\text{Department ID} \rightarrow \text{Department Name}$

On the left side of the slide, there is vertical text: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr., 2018". At the bottom, it says "Database System Concepts - 8<sup>th</sup> Edition". On the right, it shows slide number 16.26 and copyright information: "©Silberschatz, Korth and Sudarshan".

These are another example. So, these are just go through them, try to convince yourself that these functional dependencies are very genuinely real world situations that can be modeled in this way.

(Refer Slide Time: 30:19)

The slide title is "Closure of a Set of Functional Dependencies". It features a small sailboat icon in the top left corner. The slide content is a list of points:

- Given a set  $F$  of functional dependencies, there are certain other functional dependencies that are logically implied by  $F$ 
  - For example: If  $A \rightarrow B$  and  $B \rightarrow C$ , then we can infer that  $A \rightarrow C$
- The set of all functional dependencies logically implied by  $F$  is the **closure** of  $F$
- We denote the closure of  $F$  by  $F^+$
- $F^+$  is a superset of  $F$ 
  - $F = \{A \rightarrow B, B \rightarrow C\}$
  - $F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

On the left side of the slide, there is vertical text: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr., 2018". At the bottom, it says "Database System Concepts - 8<sup>th</sup> Edition". On the right, it shows slide number 16.27 and copyright information: "©Silberschatz, Korth and Sudarshan".

Given a set of functional dependencies, we can actually compute a closure. For example, if  $A \rightarrow B$  and  $B \rightarrow C$ , then we can infer that  $A$  functionally determined because if two peoples match on,  $A \rightarrow B$  says that they match on B. Now, if  $B \rightarrow C$  also holds, then if they match on B, they match on C. So, if the match on A, then necessarily they may have

to match on C. So, this is called the logical implication of a set of functional dependencies and we will see more of this later, but if we take all functional dependencies of a given set F, that are logically implied from this set F. We said that is a closure set and we represent that by  $F^+$ .

So,  $F^+$  necessarily is a superset of F. So, here in that above example, this is F and this is  $F^+$ .

(Refer Slide Time: 31:00)

The slide is titled "Module Summary" in red. To the left of the title is a small image of a sailboat on water. Below the title is a bulleted list of three items:

- Identified the features of good relational design
- Familiarized with the First Normal Form
- Introduced the notion of Functional Dependencies

On the far left, there is vertical text that reads: "SWAYAM: NPTEL-NOC's MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr. 2018". At the bottom left, it says "Database System Concepts - 8th Edition". In the center bottom, it shows the time "16.28". On the right side, there is a copyright notice "©Silberschatz, Korth and Sudarshan".

So, we will continue more on the theory of functional dependencies, but let us conclude this module by summarizing that we have identified the features of good relational designs tradeoff between decomposition and lossless join properties that we need. We are familiarized with the First Normal Form and atomic domains and we have introduced the notion of functional dependencies on which we will build up more and try to get zeroing very concrete strategies for good results.

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

**Lecture - 17**  
**Relational Database Design (Contd.)**

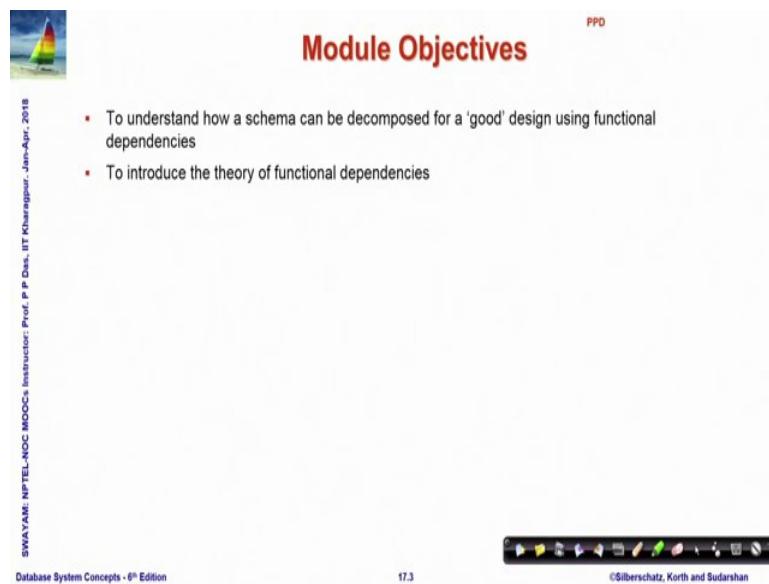
Welcome to the Module 17 of Database Management Systems. From the last module, we are discussing Relational Database Design. So, this is second in the series of five modules which we will discuss this.

(Refer Slide Time: 00:31)

The slide has a header 'Module Recap' in red. On the left, there is a small image of a sailboat on water. On the right, there is a small logo with the letters 'PPD'. The main content area contains a bulleted list of three items: 'Features of Good Relational Design', 'Atomic Domains and First Normal Form', and 'Functional Dependencies'. At the bottom, there is a footer with the text 'SWAYAM: NPTEL-NOCO's MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur', 'Database System Concepts - 8<sup>th</sup> Edition', '17.2', and '©Silberschatz, Korth and Sudarshan'.

We have already seen basic features of good relational design. We have studied about first normal form, atomic domains and got introduced to functional dependencies.

(Refer Slide Time: 00:43)



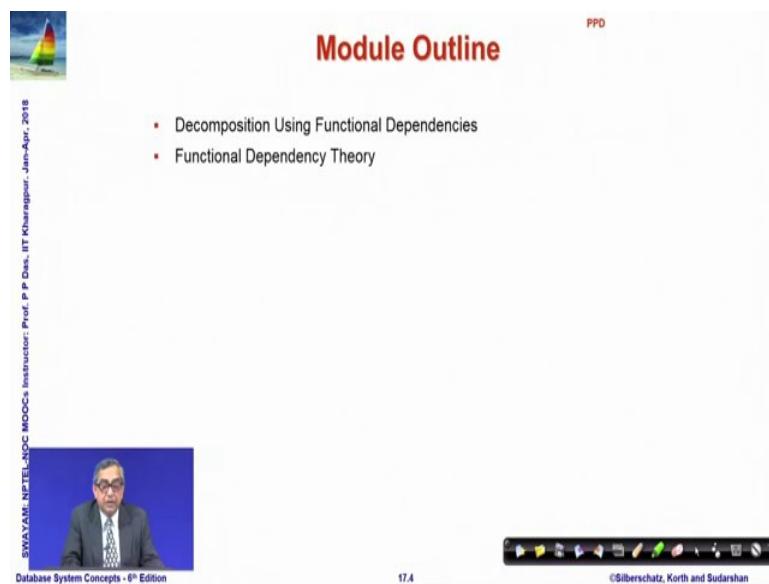
This slide is titled "Module Objectives" in red at the top right. It features a small sailboat icon in the top left corner. On the far left, there is vertical text: "SWAYAM: NPTEL-NOC MOOCs", "Instructor: Prof. P. P. Deshpande, IIT Kharagpur", and "Date: Apr. 2018". At the bottom left, it says "Database System Concepts - 8<sup>th</sup> Edition". The slide contains a bulleted list of objectives:

- To understand how a schema can be decomposed for a 'good' design using functional dependencies
- To introduce the theory of functional dependencies

The bottom right corner includes the copyright notice "©Silberschatz, Korth and Sudarshan" and a set of standard presentation navigation icons.

So, we will develop further on that to see how decompositions into good design can be done by making use of the notion of functional dependencies and we will more formally introduce the theory of functional dependencies.

(Refer Slide Time: 00:57)



This slide is titled "Module Outline" in red at the top right. It features a small sailboat icon in the top left corner. On the far left, there is vertical text: "SWAYAM: NPTEL-NOC MOOCs", "Instructor: Prof. P. P. Deshpande, IIT Kharagpur", and "Date: Apr. 2018". At the bottom left, it says "Database System Concepts - 8<sup>th</sup> Edition". The slide contains a bulleted list of topics:

- Decomposition Using Functional Dependencies
- Functional Dependency Theory

The bottom right corner includes the copyright notice "©Silberschatz, Korth and Sudarshan" and a set of standard presentation navigation icons. A video frame showing a man speaking is positioned in the lower-left area of the slide.

So, that is all that we discuss in this.

(Refer Slide Time: 01:00)

The slide features a small sailboat icon in the top left corner. In the top right, the text 'PPD' is written above a bulleted list: '•Decomposition Using Functional Dependencies •Functional Dependency Theory'. The main title 'DECOMPOSITION USING FUNCTIONAL DEPENDENCIES' is centered in large red capital letters. Below the title is a standard presentation navigation bar with icons for back, forward, search, and other controls.

SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P.P. Desai, IIT Kharagpur - Jan-Apr., 2018

Database System Concepts - 8<sup>th</sup> Edition

17.5

©Silberschatz, Korth and Sudarshan

So, decomposition using functional dependencies is the first thing that we look at.

(Refer Slide Time: 01:04)

The slide includes a sailboat icon and a video frame of a professor speaking. The main title 'Boyce-Codd Normal Form' is in red. Below it, a definition states: 'A relation schema R is in BCNF with respect to a set F of functional dependencies if for all functional dependencies in F+ of the form  $\alpha \rightarrow \beta$ , where  $\alpha \subseteq R$  and  $\beta \subseteq R$ , at least one of the following holds:'. Two bullet points follow: '•  $\alpha \rightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$ )' and '•  $\alpha$  is a superkey for R'. To the right is a diagram showing two boxes labeled E<sub>1</sub> and E<sub>2</sub> with arrows indicating a relationship between them. The bottom of the slide shows a navigation bar and copyright information.

SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P.P. Desai, IIT Kharagpur - Jan-Apr., 2018

Database System Concepts - 8<sup>th</sup> Edition

17.6

©Silberschatz, Korth and Sudarshan

The first normal form of relations which were studied, we look at its Boyce-Codd Normal Form. So, normal forms are kind of set of properties which is satisfied by a relational schema and if they are satisfied, then we have certain guarantees in terms of what can or cannot happen in that relational schema design. So, Boyce-Codd is a simplest kind of beyond 1NF is a simplest kind of normal form and a relational schema is said to be in Boyce-Codd normal form if with respect to a set of functional

dependencies, all functional dependencies in the closure. So, in respect of F, we compute  $F^+$  which is a closure and if I have a dependency  $\alpha \rightarrow \beta$ , then naturally  $\alpha \subseteq R, \beta \subseteq R$ , but what is important is every functional dependency in the closure set must either be trivial that is right hand side  $\subseteq$  the left hand side or the left hand side set  $\alpha$  must be super key. So, only those kind of functional dependencies are possible. No other functional dependencies are possible. If that is satisfied by the relational schema R, then it is said to be in the Boyce-Codd normal form.

(Refer Slide Time: 02:39)

**Boyce-Codd Normal Form**

A relation schema  $R$  is in BCNF with respect to a set  $F$  of functional dependencies if for all functional dependencies in  $F^+$  of the form

$$\alpha \rightarrow \beta$$

where  $\alpha \subseteq R$  and  $\beta \subseteq R$ , at least one of the following holds:

- $\alpha \rightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$ )
- $\alpha$  is a superkey for  $R$

Example schema *not* in BCNF:

*instr\_dept* (ID, name, salary, dept\_name, building, budget)

because dept\_name → building, budget holds on *instr\_dept*, but dept\_name is not a superkey

Navigation icons: back, forward, search, etc.

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

Database System Concepts - 8<sup>th</sup> Edition

17.6

©Silberschatz, Korth and Sudarshan

So, if we look at *inst\_dept* schema of the combined relations we saw last time, then we will know that certainly this is not in Boyce-Codd Normal Form because this functional dependency holds in this schema where it is neither a trivial dependency and nor department name is a super key. So, this is not in BCNF.

(Refer Slide Time: 03:10)

**Decomposing a Schema into BCNF**

- Suppose we have a schema  $R$  and a non-trivial dependency  $\alpha \rightarrow \beta$  causes a violation of BCNF
- We decompose  $R$  into:
  - $(\alpha \cup \beta)$
  - $(R - (\beta - \alpha))$
- In our example,
  - $\alpha = \text{dept\_name}$
  - $\beta = \text{building, budget}$
  - $\text{dept\_name} \rightarrow \text{building, budget}$

*inst\_dept is replaced by*

- $(\alpha \cup \beta) = (\text{dept\_name, building, budget})$  ✓  $R_1$
- $\text{dept\_name} \rightarrow \text{building, budget}$  ✓
- $(R - (\beta - \alpha)) = (\text{id, name, dept\_name})$  ✓  $R_2$
- $\text{id} \rightarrow \text{name, salary, dept\_name}$  ✓

So, if a relational scheme is not in BCNF, then the question naturally is can I make it into BCNF so then that process is the process of decomposition. So, what you do? You divide the set of attributes into two or more sets of attributes. So, here let us say that we have a relational schema which has a non-trivial dependency,  $\alpha \rightarrow \beta$ , where  $\alpha$  is not a super key. So, with respect to this functional dependency, the relational schema is not in BCNF, then we can decompose  $R$  by two sets. One is  $\alpha \cup \beta$  take the  $\cup$  of these two attribute sets and remove  $\beta - \alpha$  from  $R$ , take the difference of  $\beta - \alpha$  and remove that from  $R$ . The resulting pair of relations, relational schemas will be in Boyce-Codd Normal Form with respect to this particular functional dependency.

So, let us see an example. So, if  $\alpha$  is department name  $\beta$  is (**building, budget**), we have department name  $\rightarrow$  building budget. So,  $\alpha \rightarrow \beta$  and we have already seen that it does not hold. It is not satisfied by the **inst\_dept**. So, you replace it by taking  $\alpha \cup \beta$ . So,  $\alpha \cup \beta$  is this set of relational, this relational schema and you do  $R - (\beta - \alpha)$ . Naturally if this is  $\beta$  and  $\alpha$ , then  $\beta - \alpha$  is necessary building budget because if the department does not occur in  $\beta$ .

So, this set is building budget and if I remove it from  $R$  which means that id, name, salary and department name are retained, but building and budget gets removed. So, I get another relational schema which has these four names and it holds the functional dependency id determinant. So, even now if I look into this schema  $R_1$  and this schema

R2, there are different dependencies that hold on R1 and with respect to that dependency R1 is in BCNF because department name is the super key, is the primary key and with respect to this dependency, R2 is in BCNF because id is a key.

So, I can see that the original combined relational schema was not in BCNF with respect to this functional dependency, but when I do this decomposition, I get two schemas which are each in BCNF normal form. So, this is the basic process and we will see depending on the normal form and different notions of functional dependencies, we will see how these conversions can be done, but this is a basic approach of converting a schema into a normal form.

(Refer Slide Time: 06:24)

**BCNF and Dependency Preservation**

- Constraints, including functional dependencies, are costly to check in practice unless they pertain to only one relation
- If it is sufficient to test only those dependencies on each individual relation of a decomposition in order to ensure that *all* functional dependencies hold, then that decomposition is *dependency preserving*.
- Because it is not always possible to achieve both BCNF and dependency preservation, we consider a weaker normal form, known as *third normal form*.

SWAYAM-NETTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr. 2018  
Database System Concepts - 8<sup>th</sup> Edition  
17.8 ©Silberschatz, Korth and Sudarshan

Now, the question is if the constraints including the functional dependencies if we look at, then functional dependencies will have to be checked on different instance. Now, in general it is difficult to check a functional dependency  $\alpha$  determining  $\beta$  if the attributes  $\alpha$  and the attributes of  $\beta$  or the attributes of  $\beta$  are distributed between multiple relations because naturally how do I check if they are true, how do I check that two tuples which match on  $\alpha$  is indeed matching on  $\beta$  unless I perform a costly join operation. So, the objective is to be able to come to designs where it is sufficient to test only those dependencies on individual relations of the decomposition and with that I must be able to ensure that all functional dependencies hold. So, it is a very interesting situation.

So, we are saying that we will decompose, get into a number of relational schema. Every schema will have a number of dependencies, functional dependencies and those functional dependencies if they involve only the attributes of that relational schema, they can be tested very easily and if these functional dependencies together mean ensure that all functional dependencies hold that is if the closure of this set of functional dependencies is same as the closure of the earlier set, the original set, then we say that the decomposition that we have achieved is dependency preserving because I can actually effectively compute.

This is dependency preserving because I can effectively compute whether every dependency is satisfied by checking on every individual relation, but the unfortunate part of the reality is that it is not always possible to achieve a Boyce-Codd Normal Form Decomposition which also preserves the dependencies. See if there are in some cases will be able to do like the example we saw just now the instructor and department, but it is not always possible. So, we usually need another weaker form, normal form which is known as a third normal form and we will subsequently look into those.

(Refer Slide Time: 09:09)

The slide has a decorative header featuring a sailboat on water. The title 'Third Normal Form' is centered in red. The content is organized into two columns. The left column contains a vertical footer with text: 'SWAYAM-NPTEL-NOCS Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018'. The right column contains a bulleted list of conditions for a relation to be in 3NF:

- A relation schema  $R$  is in **third normal form (3NF)** if for all:  
 $\alpha \rightarrow \beta$  in  $F^+$   
at least one of the following holds:
  - $\alpha \rightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$ )
  - $\alpha$  is a superkey for  $R$
  - Each attribute  $A$  in  $\beta - \alpha$  is contained in a candidate key for  $R$   
(NOTE: each attribute may be in a different candidate key)
- If a relation is in BCNF it is in 3NF (since in BCNF one of the first two conditions above must hold)
- Third condition is a minimal relaxation of BCNF to ensure dependency preservation (will see why later)

At the bottom, there is a video frame showing a man in a suit, a navigation bar with icons, and footer text: 'Database System Concepts - 6<sup>th</sup> Edition', '17.9', and '©Silberschatz, Korth and Sudarshan'.

A third normal form is again a relational schema is there and for all attribute, for all dependencies that belong to the closure of the functional dependencies, this following conditions must hold either  $\alpha \rightarrow \beta$  is trivial which is a condition which BCNF or  $\alpha$  is a super key of R which is also a condition that we say in BCNF or each attribute in  $\beta - \alpha$

that is right hand side difference the left hand side is contained in a candidate key for R. It is not very obvious as to why we need that. That will unfold slowly. This is the condition we did not have in BCNF. So, naturally you can see that based on the first two conditions, you can always say that if a relational schema is in BCNF, it necessarily is in 3NF, but not the reverse.

There could be some schema which is in 3 NF because of the third condition where there exists a functional dependency. So, that  $\beta - \alpha$  is contained in a candidate key for R, but it is not in the BCNF form and also you can note that the attributes of that are contained in  $\beta - \alpha$  must be in some candidate key, not necessarily in the same candidate key. If they exist in some candidate key, then itself 3NF condition will get satisfied. So, if a relation is in BCNF, it is in 3NF. We have already seen that. So, third condition minimally relaxes BCNF to ensure that we have a dependency preservation. We will see this more later. So, I am just introducing the concept of a relaxed normal form here.

(Refer Slide Time: 11:11)

The slide has a header 'Goals of Normalization' in red. On the left is a small sailboat icon. The main content is a bulleted list:

- Let  $R$  be a relation scheme with a set  $F$  of functional dependencies
- Decide whether a relation scheme  $R$  is in "good" form
- In the case that a relation scheme  $R$  is not in "good" form, decompose it into a set of relation schemes  $\{R_1, R_2, \dots, R_n\}$  such that
  - each relation scheme is in good form
  - the decomposition is a lossless-join decomposition
  - Preferably, the decomposition should be dependency preserving

At the bottom, there is a video frame showing a man speaking, the text 'Database System Concepts - 8<sup>th</sup> Edition', the page number '17.10', and the copyright notice '©Silberschatz, Korth and Sudarshan'.

So, what is a goal of this normalization is if to summarize let  $R$  be a relational scheme,  $F$  is a set of functional dependencies, we need to decide whether the relational scheme  $R$  is in a good form which means that it should not have unnecessary redundancy. It should be impossible to acquire information by doing lossless join. So, in case it is not in good form. We can convert it by decomposition into  $N$  relational schema, such that each schema is in good form. The decomposition has a lossless join, so that I can get back the

original relation from this and preferably the decomposition should preserve the dependencies. So, that is what we will target henceforth.

(Refer Slide Time: 12:08)

The slide has a title 'How good is BCNF?' in red at the top right. To the left is a small sailboat icon. On the left margin, there is vertical text: 'SWAYAM-NETELNOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr., 2018'. At the bottom left is a video player showing a man in a suit, with the text 'Database System Concepts - 6th Edition' below it. The bottom right shows a navigation bar with icons for back, forward, search, etc., and the text '17.11' and '©Silberschatz, Korth and Sudarshan'.

**How good is BCNF?**

- There are database schemas in BCNF that do not seem to be sufficiently normalized
- Consider a relation
  - inst\_info (ID, child\_name, phone)
    - where an instructor may have more than one phone and can have multiple children

ID	child_name	phone
99999	David	512-555-1234 ✓
99999	David	512-555-4321 ✗
99999	William	512-555-1234 ✓
99999	Willian	512-555-4321 ✗

So, when we do that let us quickly evaluate as to we have seen BCNF. So, how good really BCNF is. So, if I have something in BCNF should I really be very happy always. So, let us look at a relational schema. This is an information relating the idea of a person, the name of the child and the phone number and naturally the person, the instructor may have more than one phone and may have multiple children. So, this is a possible instance that you can see though all of these belong to the same instructor. He has naturally you can see that two children and there are this is here, this is here.

So, this is here and this is here. So, there are two different phone numbers. So, naturally you have four possible combinations that you need to look at.

(Refer Slide Time: 13:08)

The slide has a header 'How good is BCNF? (Cont.)' in red. On the left is a small sailboat icon. The right side contains a video player window showing a man in a suit speaking. Below the video player are some navigation icons. The footer includes text: 'SYAYAM-NETEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '17.12', and '©Silberschatz, Korth and Sudarshan'.

- There are no non-trivial functional dependencies and therefore the relation is in BCNF
- Insertion anomalies – i.e., if we add a phone 981-992-3443 to 99999, we need to add two tuples
  - (99999, David, 981-992-3443)
  - (99999, William, 981-992-3443)

So, now there is no non-trivial functional dependency in this relation. So, since there is no non-trivial functional dependency, this relation naturally is in BCNF form because that is the existence of non trivial dependency is what makes a schema not conform to the BCNF form. So, there is no such. So, this is in BCNF form and now, if you look at, but what did we see the key thing that we saw if we just go back, the key thing that we saw that there is ample redundancy of data, the same data is entered multiple times.

So, the consequence of that could be insertion anomaly. If we want to add a phone number to the same instructor, then we need to add two tuple because the instructor also has two children. If the instructor and three children will need to add three and unless this is maintained always, then we will have difficulty.

(Refer Slide Time: 14:15)

The slide features a small sailboat icon in the top-left corner. The title 'How good is BCNF? (Cont.)' is centered at the top in red. On the left, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018'. Below the title, a bulleted list says: 'Therefore, it is better to decompose *inst\_info* into:'. Two tables are shown side-by-side. The first table, 'inst\_child', has columns 'ID' and 'child\_name', with data: ID 99999, child\_name David; ID 99999, child\_name David; ID 99999, child\_name William; ID 99999, child\_name Willian. The second table, 'inst\_phone', has columns 'ID' and 'phone', with data: ID 99999, phone 512-555-1234; ID 99999, phone 512-555-4321; ID 99999, phone 512-555-1234; ID 99999, phone 512-555-4321. A note below the tables states: 'This suggests the need for higher normal forms, such as Fourth Normal Form (4NF)'. At the bottom, there is a navigation bar with icons for back, forward, search, etc., and the text 'Database System Concepts - 8<sup>th</sup> Edition', '17.13', and '©Silberschatz, Korth and Sudarshan'.

So, the redundancy consequences anomaly that we are getting into, so it could be better to decompose this to say that I make this orthogonal; I keep the child information with id and I keep the phone number information in the id separately. So, if I do that, then I can decompose it in this manner and if I decompose that, this have just shown that if you are dividing that table in two parts. So, naturally these are not required, neither are these required. So, these are the entries that I get and you can convince yourself that you can actually do a lossless join to get back the information.

So, BCNF not necessarily give you good designs and we will see later on that there are other normal forms which can be used to improve on BCNF.

(Refer Slide Time: 15:05)

The slide has a header 'PPD' at the top right. On the left, there is a small sailboat icon and vertical text: 'SWAYAM-NPTEL-MOOC Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018'. The main title 'FUNCTIONAL DEPENDENCY THEORY' is in large red capital letters. To the right of the title is a bulleted list: '• Decomposition Using Functional Dependencies', '• Functional Dependency Theory'. At the bottom, there is footer text: 'Database System Concepts - 8<sup>th</sup> Edition', '17.14', and '©Silberschatz, Korth and Sudarshan'.

Now, let us formally get into how do we convert decomposed relation into a third normal form and how we assess that we need to understand more of the functional dependencies.

(Refer Slide Time: 15:20)

The slide has a header 'Functional-Dependency Theory' at the top right. On the left, there is a small sailboat icon and vertical text: 'SWAYAM-NPTEL-MOOC Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018'. The main title is in large red capital letters. Below the title is a bulleted list: '• We now consider the formal theory that tells us which functional dependencies are implied logically by a given set of functional dependencies', '• We then develop algorithms to generate lossless decompositions into BCNF and 3NF', '• We then develop algorithms to test if a decomposition is dependency-preserving'. At the bottom, there is a video frame showing a man in a suit, footer text: 'Database System Concepts - 8<sup>th</sup> Edition', '17.15', and '©Silberschatz, Korth and Sudarshan'.

So, we will consider now a little bit of formal theory on them and then, develop algorithms that can generate lossless join decomposition into BCNF and 3 NF and we will also create algorithm to test if decomposition preserves the dependency.

(Refer Slide Time: 15:39)

Closure of a Set of Functional Dependencies

- Given a set  $F$  of functional dependencies, there are certain other functional dependencies that are logically implied by  $F$ 
  - For e.g.: If  $A \rightarrow B$  and  $B \rightarrow C$ , then we can infer that  $A \rightarrow C$
- The set of all functional dependencies logically implied by  $F$  is the closure of  $F$
- We denote the closure of  $F$  by  $F^*$

SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr., 2018  
Database System Concepts - 8<sup>th</sup> Edition  
17.16 ©Silberschatz, Korth and Sudarshan

So, just quickly to recap we have already introduced the closure set of a functional dependencies. It is all dependencies that are logically implied by it. Now, the question certainly is given a set how do I compute this closure of a set?

(Refer Slide Time: 15:57)

Closure of a Set of Functional Dependencies

- We can find  $F^*$ , the closure of  $F$ , by repeatedly applying **Armstrong's Axioms**:
  - if  $\beta \sqsubseteq \alpha$ , then  $\alpha \rightarrow \beta$  (reflexivity)
  - if  $\alpha \rightarrow \beta$ , then  $\gamma \alpha \rightarrow \gamma \beta$  (augmentation)
  - if  $\alpha \rightarrow \beta$ , and  $\beta \rightarrow \gamma$ , then  $\alpha \rightarrow \gamma$  (transitivity)
- These rules are
  - sound (generate only functional dependencies that actually hold), and
  - complete (generate all functional dependencies that hold)

SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr., 2018  
Database System Concepts - 8<sup>th</sup> Edition  
17.17 ©Silberschatz, Korth and Sudarshan

So, to do this we make use of three rules known by Armstrong's Axiom named after the person who first observed them. So, the first rule is reflexivity which says that if  $\beta \sqsubseteq \alpha$ , then  $\alpha \rightarrow \beta$ . Always  $\alpha \rightarrow \beta$ . So, this is basically reflexivity you can see is a different

way of saying specifying about trivial dependencies. Next comes important thing augmentation which says that if  $\alpha \rightarrow \beta$ , then  $\gamma \alpha$  where  $\gamma$  is some set of attributes in R.

Then,  $\gamma \alpha \rightarrow \gamma \beta$  which is very easy to see because  $\alpha \rightarrow \beta$  means two tuples who match on  $\alpha$  will necessarily match on  $\beta$ . Now, if that happens and whatever is  $\gamma$  if two tuples match on  $\gamma$  and  $\alpha$ , then certainly they will match and  $\gamma$  and  $\beta$  because  $\alpha \rightarrow \beta$  tells me that they will match on  $\beta$  and  $\gamma$  is the same set of attributes. So, augmentation also is easy. Then, we have transitivity which we earlier saw also if  $\alpha \rightarrow \beta$  and  $\beta \rightarrow \gamma$ , then obviously  $\alpha \rightarrow \gamma$ .

So, these are the foundational rules observed which can be made used to compute the closure of the set of functional dependencies. Now, these rules as they say this is more for you know understanding the theory better. These rules are sound as well as complete. Soundness mean that if I use these rules repeatedly in a set of dependencies, then it generates functional dependencies all of which actually hold. So, it will never generate a functional dependency which is not correct, which will not hold and the second it is complete which means that if I keep on using these rules, then all functional dependencies that can at all hold will eventually get generated.

(Refer Slide Time: 18:06)


**Example**

- $R = (A, B, C, G, H, I)$
- $F = \{ A \rightarrow B$   
 $A \rightarrow C \checkmark$   
 $CG \rightarrow H$   
 $CG \rightarrow I \checkmark$   
 $B \rightarrow H \}$
- Some members of  $F^*$ 
  - $A \rightarrow H$ 
    - by transitivity from  $A \rightarrow B$  and  $B \rightarrow H$
  - $AG \rightarrow I \checkmark$ 
    - by augmenting  $A \rightarrow C$  with G, to get  $AG \rightarrow CG$  and then transitivity with  $CG \rightarrow I$
  - $CG \rightarrow HI$ 
    - by augmenting  $CG \rightarrow I$  to infer  $CG \rightarrow CGI$ , and augmenting of  $CG \rightarrow H$  to infer  $CGI \rightarrow HI$ , and then transitivity

SWAYAM: NPTEL-NOCO MOOCs Instructor: Prof. P. P. Desai, Jai Narayanpur - Jan-Apr. 2018
Database System Concepts - 6<sup>th</sup> Edition
17.18
©Silberschatz, Korth and Sudarshan

So, that is a very strong result and that is what leads to say the following example. So, when we are trying to compute the functional, the closure of the function set of functional dependencies here. So, there are six attributes in the set. There are six

different functional dependencies and we identify some members of the closure. For example, we can see that  $A \rightarrow B$  and  $B \rightarrow H$ . So, transitivity clearly shows that  $A \rightarrow H$ , very clear. So, in the closure that must be there.

Similarly, we can see that  $A \rightarrow C$ . Now, if we augment it with G, that is put G on both sides, then  $AG \rightarrow CG$  and we know that  $CG \rightarrow I$ . So, if we combine these two by transitivity, then we can get a new functional dependency which has  $AG \rightarrow I$ . So, in this manner you can do the next one also and you can try to infer several other functional dependencies that can be inferred by different applications of the Armstrong's axiom, the three rules in any multiple different ways.

(Refer Slide Time: 19:36)

**Procedure for Computing  $F^+$**

- To compute the closure of a set of functional dependencies  $F$ :

```

 $F^+ = F$ 
repeat
    for each functional dependency  $f$  in  $F^+$ 
        apply reflexivity and augmentation rules on  $f$ 
        add the resulting functional dependencies to  $F^+$ 
    for each pair of functional dependencies  $f_1$  and  $f_2$  in  $F^+$ 
        if  $f_1$  and  $f_2$  can be combined using transitivity
            then add the resulting functional dependency to  $F^+$ 
    until  $F^+$  does not change any further

```

**NOTE:** We shall see an alternative procedure for this task later

SWAYAM-NPTEL-NOOC Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018

Database System Concepts - 8<sup>th</sup> Edition      17.19      ©Silberschatz, Korth and Sudarshan

So, to get the closure what we need to do is, now very simple is certainly we will have a repetitive algorithm to get the closure. The first algorithm will start with the set of functional dependencies that we have. So, the closure must include the given set of functional dependencies. So,  $F^+$  must have  $F$ . So, let us start with initial value of  $F^+$  as  $F$ , then for every functional dependency is  $F^+$ . This is what we keep on repeating. Look at the outer loop, every functional dependency that we have  $F^+$  now will apply reflexivity and augmentation and add the resulting functional dependency in  $F^+$ . It is possible that the same functional dependency gets generated and added multiple times does not matter.  $F^+$  is a set. It will naturally eliminate duplicates.

Then, for each pair of functional dependencies because reflexivity and augmentation applies to one functional dependency only, but transitivity applies to two functional dependencies. So, for every pair of functional dependencies, we check whether they can be combined by transitivity. If they do, then the transitive closure of the transitive functional dependency that arise out of that is also added to  $F^+$  and mind you more and more functional dependencies you add, there are more and more opportunities to apply the Armstrong's Axiom rules and newer functional dependencies will continue to get added, but eventually you reach a point where  $F^+$  does not change any further and when that is achieved, we know that the functional, the closure of the functional dependencies have been obtained and that is our final set.

(Refer Slide Time: 21:32)

**Closure of Functional Dependencies (Cont.)**

- Additional rules:
  - If  $\alpha \rightarrow \beta$  holds and  $\alpha \rightarrow \gamma$  holds, then  $\alpha \rightarrow \beta\gamma$  holds (**union**)
  - If  $\alpha \rightarrow \beta\gamma$  holds, then  $\alpha \rightarrow \beta$  holds and  $\alpha \rightarrow \gamma$  holds (**decomposition**)
  - If  $\alpha \rightarrow \beta$  holds and  $\gamma \beta \rightarrow \delta$  holds, then  $\alpha\gamma \rightarrow \delta$  holds (**pseudotransitivity**)

The above rules can be inferred from Armstrong's axioms

SWAYAM-NPTEL-NOOC Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018  
Database System Concepts - 6<sup>th</sup> Edition  
17.20 ©Silberschatz, Korth and Sudarshan

We can also observe that based on the rules of Armstrong, the Armstrong's Axioms we can also generate lot of derived rules. Some of those are shown here. For example, if  $A \rightarrow$ , if  $\alpha \rightarrow \beta$  holds and if  $\alpha \rightarrow \gamma$ , that also holds, then  $\alpha \rightarrow \beta$  and  $\gamma$  together. This is called the **union** set. So, if there are two functional dependencies which are the same left hand side set of attribute, then we can take the **union** of their right hand side attributes and that functional dependency will hold obviously, it is trivial to prove this.

If  $\alpha \rightarrow \beta\gamma$ , then  $\alpha \rightarrow \beta$  holds and  $\alpha \rightarrow \gamma$  holds. This is called decomposition. So, kind of the other side of the **U** which also is trivial because  $\alpha \rightarrow \beta\gamma$  says if two tuples match on  $\alpha$ , they match on  $\beta$  as well as  $\gamma$  attributes. So, obviously you take the first part, you get  $\alpha$

→  $\beta$ . You take the second part of the observation, you get  $\alpha \rightarrow \gamma$ . So, that is a composition rule. The third is interesting. It is called the pseudo transitivity which says that  $\alpha \rightarrow \beta$  if that holds and  $\gamma \beta \rightarrow \delta$  if that holds, then  $\alpha \gamma \rightarrow \delta$  which is not difficult to get because if this holds, then I can augment  $\gamma$  on both sides. I get  $\beta \gamma$  and then, I have given  $\beta \gamma \rightarrow \delta$ . So, if I combine these two in terms of transitivity, I get  $\alpha \gamma \rightarrow \delta$ .

So, this is called pseudo transitivity because here you are adding another attribute in the transitivity. So, often times it becomes easier to make use of these additional rules to quickly get to the closer set.

(Refer Slide Time: 23:45)

**Closure of Attribute Sets**

- Given a set of attributes  $\alpha$ , define the **closure** of  $\alpha$  under  $F$  (denoted by  $\alpha^+$ ) as the set of attributes that are functionally determined by  $\alpha$  under  $F$
- Algorithm to compute  $\alpha^+$ , the closure of  $\alpha$  under  $F$

```

result := α; ✓
while (changes to result) do
  for each β → γ in F do
    begin
      if β ⊆ result then result := result ∪ γ
    end
  
```

The hand-drawn diagram shows a Venn diagram with three overlapping circles labeled  $\alpha$ ,  $\beta$ , and  $\gamma$ . Arrows point from  $\alpha$  to  $\beta$  and from  $\beta$  to  $\gamma$ . A large arrow points from the union of  $\alpha$  and  $\beta$  to the intersection of  $\alpha$  and  $\beta$ , representing the step where the union of  $\alpha$  and  $\beta$  is updated to include  $\gamma$  if  $\beta$  is a subset of the current closure.

So, given a set of attributes we also compute the closure of a set of attributes. This is a second concept we have seen how to give the set of functional dependencies, how to compute the closure of the functional dependencies. Now, we are given a set of attributes and we want to define the closure of this set of functional, this set of attributes under the set of functional dependencies and as the closure of functional dependencies  $F$  is denoted by  $F^+$ , the closure of a set of attributes  $\alpha$  under  $F$  is denoted by  $\alpha^+$ . So, this set of closure attributes of  $\alpha$  is a set of attributes that are functionally determined by  $\alpha$  under  $F$ . So, all set of attributes that functionally determined by  $\alpha$  under the set of functional dependencies is member of  $\alpha^+$ .

So, the following simple algorithm can compute the closure naturally. Initially let us say the result is the final closure set. So, initially we can say that result can be initialized with

$\alpha$  because certainly the whole of  $\alpha$  would necessarily belong to  $\alpha^+$  by the reflexivity condition, then for each functional dependency  $\beta \rightarrow \gamma$ , we check if  $\beta$  is a subset the result. If  $\beta$  is a subset the current set of attributes that form result which mean that  $\alpha \rightarrow \beta$ , it will have to because result is the set of all attributes that  $\alpha$  functionally determines. So, if  $\beta$  is a subset the result, then necessarily  $\alpha \rightarrow \beta$  is a consequence of this and we know that this is their  $\beta \rightarrow \gamma$  combined by transitivity.

So, I know  $\alpha \rightarrow \gamma$ . If function  $\alpha \rightarrow \gamma$ , then it must get into the result and this is exactly what the statement is saying that take result and add  $\alpha$ , add  $\gamma$ , the set of attributes  $\gamma$  to the result. How long should you do that? Naturally you will do that as long as over a full iteration of functional dependencies in F, if there is no change to the result, then you know that all future iterations will have no change. So, you reach a fixed point and you declare that the closure of the set of attributes have been obtained.

(Refer Slide Time: 26:44)

**Example of Attribute Set Closure**

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$
- $(AG)^*$ 
  1.  $result = AG$
  2.  $result = ABCG$  ( $A \rightarrow C$  and  $A \rightarrow B$ )
  3.  $result = ABCGH$  ( $CG \rightarrow H$  and  $CG \subseteq AGBC$ )
  4.  $result = ABCGHI$  ( $CG \rightarrow I$  and  $CG \subseteq AGBCH$ )

$AG \rightarrow ABCGHI$

SWAYAM NPTEL MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jam-Apr-2018  
 Database System Concepts - 8<sup>th</sup> Edition

17.22 ©Silberschatz, Korth and Sudarshan

Now, this closure information is very interesting and we just show an example here based on the same set of attributes and same set of functional dependencies.

So, we are trying to find the closure of the set of attributes AG. So,  $AG^+$  initially it will be AG. Now, since  $A \rightarrow C$ , so given that I can say that C will get included in this set in the same iteration. If I look at  $A \rightarrow B$ , so B will get included in this set. So, after this first iterative loop I will have the result as ABCG. If ABCG is there and I am looking at the next iteration, then  $CG \rightarrow H$ . So, H comes into the set because CG is a subset that I

comes into the set because  $CG \rightarrow I$  and at this point, it eventually ends in this case. In this particular example, you can see that all attributes have got included. So, you can see that it immediately gives you another information as a byproduct of the closure that closure of AG is all attributes which mean that AG is a key. It has to be a key because  $AG \rightarrow ABCGHI$ .

So, what is the meaning of  $AG^+$  being this? So, if the meaning of this is  $AG \rightarrow ABCGHI$ , right, so we will see that this closure set has a lot of valuable information in this.

(Refer Slide Time: 28:31)

**Example of Attribute Set Closure**

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$
- $(AG)^*$ 
  1.  $result = AG$
  2.  $result = ABCG$  ( $A \rightarrow C$  and  $A \rightarrow B$ )
  3.  $result = ABCGH$  ( $CG \rightarrow H$  and  $CG \subseteq AGBC$ )
  4.  $result = ABCGHI$  ( $CG \rightarrow I$  and  $CG \subseteq AGBCH$ )
- Is AG a candidate key?
  1. Is AG a super key?
    1. Does  $AG \rightarrow R? == ls(AG)^* \sqsupseteq R$
    2. Is any subset of AG a superkey?
      1. Does  $A \rightarrow R? == ls(A)^* \sqsupseteq R$
      2. Does  $G \rightarrow R? == ls(G)^* \sqsupseteq R$

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

Database System Concepts - 8<sup>th</sup> Edition

17.22

©Silberschatz, Korth and Sudarshan

So, we can say that AG is a candidate key and because of this, we can also check whether AG is a super key or not. All that we need to do is drop some member from AG, we drop G and check whether  $A \rightarrow R$  which means we check whether  $A^+ == R$  or not. We check we drop A from AG and check whether  $G \rightarrow R$  which means  $G^+ == R$  and by that we can easily determine whether the set of attributes is a key or not.

(Refer Slide Time: 29:14)

The slide has a header 'Uses of Attribute Closure' with a sailboat icon. The text discusses several uses of the attribute closure algorithm, including testing for superkeys, testing functional dependencies, and computing closure of F. A sidebar on the left provides course information: SWAYAM-NPTEL-NOC MOOCs, Instructor: Prof. P.P. Deshpande, IIT Kharagpur, Date: June-Apr., 2018.

- Testing for superkey:
  - To test if  $\alpha$  is a superkey, we compute  $\alpha^+$  and check if  $\alpha^+$  contains all attributes of  $R$ .
- Testing functional dependencies
  - To check if a functional dependency  $\alpha \rightarrow \beta$  holds (or, in other words, is in  $F^*$ ), just check if  $\beta \subseteq \alpha^+$ .
  - That is, we compute  $\alpha^+$  by using attribute closure, and then check if it contains  $\beta$ .
  - Is a simple and cheap test, and very useful
- Computing closure of  $F$ 
  - For each  $\gamma \subseteq R$ , we find the closure  $\gamma^+$ , and for each  $S \subseteq \gamma^+$ , we output a functional dependency  $\gamma \rightarrow S$ .

So, there are several ways. The attribute closure can be used as we have just seen. It helps you determine whether something is a super key. We can check for testing functional dependencies because if we have to check whether a functional dependency  $\alpha \rightarrow \beta$  hold, all that we will have to do is to compute the closure of the set of attributes  $\alpha$  that is  $\alpha^+$  and check whether  $\beta$  is a subset that. If it is, then certainly holds. If it is not, then it does not hold. So, it is simple and useful test that can be made use of.

So, it can also be used in computing the closure of  $F$  that for example, for every subset  $\gamma$  of  $R$ , if we find  $\gamma^+$  that is a closure of the set of attributes of  $\gamma$  and then, for each subset  $\gamma^+$ , we know that there is a functional dependency  $\gamma \rightarrow S$  which is just is the same statement being made in you know or in different forms and the closure of attributes is a very nice concept which help you play around in this multiple ways and we will see subsequently many of the algorithms for normalization.

(Refer Slide Time: 30:35)

**Module Summary**

- Discussed issues in 'good' design in the context of functional dependencies
- Introduced the theory of functional dependencies

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition

17.24

©Silberschatz, Korth and Sudarshan

How they make effective use of this closure set, notion of both closure of functional dependencies and in very practical implementation algorithms, the closure of attributes. So, to summarize this module, we have discussed issues further issues in the good design in the context of functional dependencies and in the process, we have also extended the theory of functional dependencies and we will continue it this in the next module to get more insight into the algorithms that actually work with the functional dependencies.

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

**Lecture - 18**  
**Relational Database Design (Contd.)**

Welcome to module 18 of Database Management Systems. We have been discussing about relational database design. This is a part 3 of that.

(Refer Slide Time: 00:30)

**Module Recap**

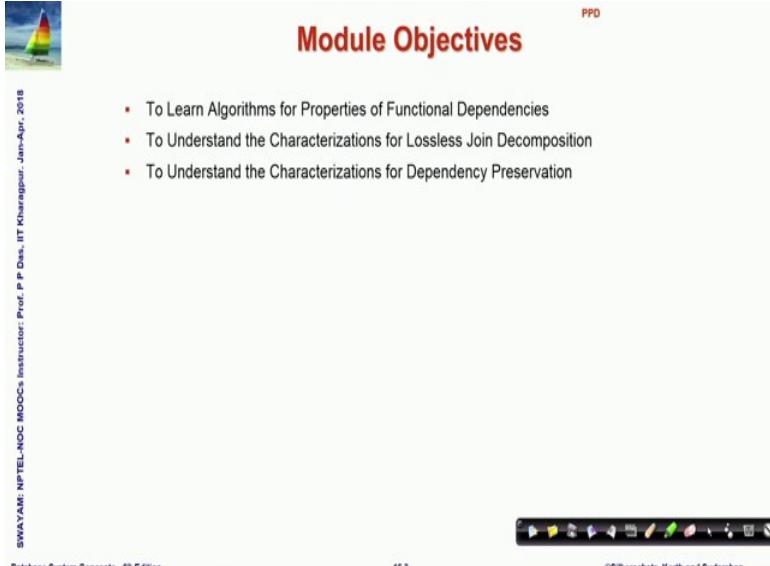
- Decomposition Using Functional Dependencies
- Functional Dependency Theory

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

Database System Concepts - 8<sup>th</sup> Edition      16.2      ©Silberschatz, Korth and Sudarshan

In the last module, we discussed about the Notion of functional dependency and decomposition based on that in an elementary level and certain bit of its theory.

(Refer Slide Time: 00:39)



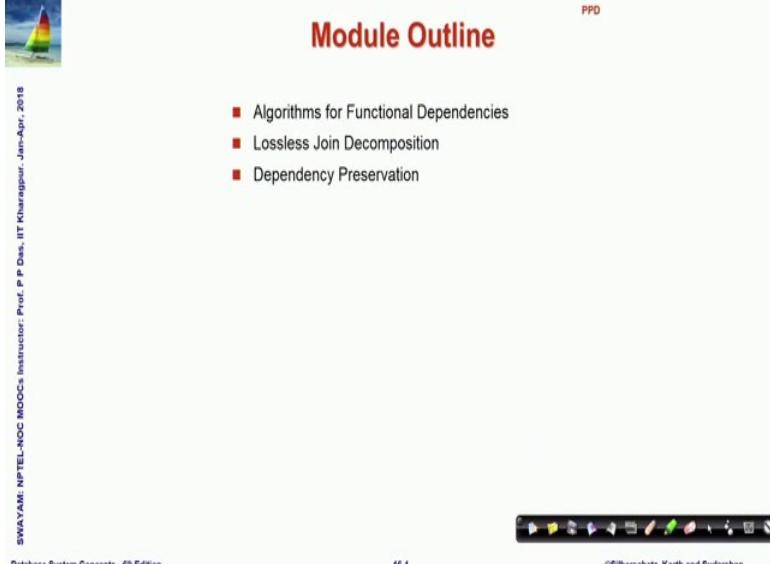
The slide is titled "Module Objectives" in red. It features a small sailboat icon in the top left corner and a "PPD" watermark in the top right. A vertical sidebar on the left contains the text "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018". The main content area lists three objectives:

- To Learn Algorithms for Properties of Functional Dependencies
- To Understand the Characterizations for Lossless Join Decomposition
- To Understand the Characterizations for Dependency Preservation

At the bottom, there is a navigation bar with icons for back, forward, search, and other presentation controls. The footer includes "Database System Concepts - 8<sup>th</sup> Edition", "16.3", and "©Silberschatz, Korth and Sudarshan".

In this current module, we learnt different algorithms that use the functional dependencies and can make conclusions about the design or make changes to the design. We will also try to understand the characterization for lossless, join decomposition and the notion of dependency preservation.

(Refer Slide Time: 01:06)



The slide is titled "Module Outline" in red. It features a small sailboat icon in the top left corner and a "PPD" watermark in the top right. A vertical sidebar on the left contains the text "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018". The main content area lists three topics:

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

At the bottom, there is a navigation bar with icons for back, forward, search, and other presentation controls. The footer includes "Database System Concepts - 8<sup>th</sup> Edition", "16.4", and "©Silberschatz, Korth and Sudarshan".

Therefore, this module will have these three topics algorithms for functional dependencies, lossless join decomposition and dependency preservation.

(Refer Slide Time: 01:15)

The slide has a header 'Example of Attribute Set Closure' with a sailboat icon. On the left, there's a vertical sidebar with text: 'SWAYAM-NPTEL-NOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr. 2018'. Below this is a video frame showing a man with glasses and a blue shirt. The main content area contains the following text:

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$
- $(AG)^*$ 
  1.  $result = AG$
  2.  $result = ABCG$  ( $A \rightarrow C$  and  $A \rightarrow B$ )
  3.  $result = ABCGH$  ( $CG \rightarrow H$  and  $CG \subseteq AGBC$ )
  4.  $result = ABCGHI$  ( $CG \rightarrow I$  and  $CG \subseteq AGBCH$ )
- Is AG a candidate key?
  1. Is AG a super key?
    1. Does  $AG \rightarrow R? == Is(AG)^* \supseteq R$
    2. Is any subset of AG a superkey?
      1. Does  $A \rightarrow R? == Is(A)^* \supseteq R$
      2. Does  $G \rightarrow R? == Is(G)^* \supseteq R$

At the bottom right are navigation icons and the text '©Silberschatz, Korth and Sudarshan'.

So, first we start with the algorithms and I quickly reproduce what we had ended in the last module in terms of computing the closure of a set of attributes. So, if we have a relation having these attributes and a set of functional dependencies, then for a given subset of attributes, in this case AG we can iteratively compute the closure set when no further changes can be done, and with using that we can make different conclusions.

For example, if our question is whether AG can be a candidate key, we would first like to check whether it is a super key that is whether its closure has all the attributes of R and we would like to check if we have taken a subset of AG. If we take just as a attribute A or attribute G whether the closure of that will actually work as a key or not.

(Refer Slide Time: 02:12)

The slide has a header 'Uses of Attribute Closure' with a sailboat icon. The text discusses several uses of the attribute closure algorithm, including testing for superkeys, functional dependencies, and computing closures. It also includes a video player showing a speaker and navigation icons.

There are several uses of the attribute closure algorithm:

- Testing for superkey:
  - To test if  $\alpha$  is a superkey, we compute  $\alpha^+$  and check if  $\alpha^+$  contains all attributes of  $R$ .
- Testing functional dependencies
  - To check if a functional dependency  $\alpha \rightarrow \beta$  holds (or, in other words, is in  $F^*$ ), just check if  $\beta \subseteq \alpha^+$ .
  - That is, we compute  $\alpha^+$  by using attribute closure, and then check if it contains  $\beta$ .
  - Is a simple and cheap test, and very useful
- Computing closure of  $F$ 
  - For each  $\gamma \subseteq R$ , we find the closure  $\gamma^+$ , and for each  $S \subseteq \gamma^+$ , we output a functional dependency  $\gamma \rightarrow S$ .

SWAYAM/NIIT-NOCS Instructor: Prof. P.P. Desai, IIT Kharagpur Date: Jan-Apr-2018

16.7 ©Silberschatz, Korth and Sudarshan

So, this algorithm of attribute closure turns out to be a very powerful one, where as we have just seen it can be used for checking super keys, the candidate keys, primary, non-primary attributes and so on. It can be used for checking functional dependencies. For example, let us suppose that if we have to check that whether if a particular functional dependency  $\alpha \rightarrow \beta$  holds, then rather in other words whether  $\alpha \rightarrow \beta$  is in the closure of the set of functional dependencies  $F$ , then all that we need to do is to compute  $\alpha^+$  that is a closure of the set of attributes on the left hand side of the dependency and check if  $\beta$  is a subset of that. If  $\beta$  is a subset of that, then I know that  $\alpha \rightarrow \beta$  actually holds.

So, in this manner it can also be used to compute the closure of the whole set of functional dependencies  $F$ . So, if I mean at least at A, rudimentary level we can think of that. If we take any subset of the set of attributes and find the closure and then, all attributes that belong to that closure set are actually functionally dependent and therefore, those functional dependencies will exist.

(Refer Slide Time: 03:42)

The slide has a title 'Canonical Cover' in red at the top right. On the left is a small logo of a sailboat on water. The main content area contains a bulleted list of points about functional dependencies:

- Sets of functional dependencies may have redundant dependencies that can be inferred from the others
  - For example:  $A \rightarrow C$  is redundant in:  $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$
  - Parts of a functional dependency may be redundant
    - E.g.: on RHS:  $\{A \rightarrow B, B \rightarrow C, A \rightarrow CD\}$  can be simplified to  $\{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$ 
      - In the forward: (1)  $A \rightarrow CD \rightarrow A \rightarrow C$  and  $A \rightarrow D$  (2)  $A \rightarrow B, B \rightarrow C \rightarrow A \rightarrow C$
      - In the reverse: (1)  $A \rightarrow B, B \rightarrow C \rightarrow A \rightarrow C$  (2)  $A \rightarrow C, A \rightarrow D \rightarrow A \rightarrow CD$
    - E.g.: on LHS:  $\{A \rightarrow B, B \rightarrow C, AC \rightarrow D\}$  can be simplified to  $\{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$ 
      - In the forward: (1)  $A \rightarrow B, B \rightarrow C \rightarrow A \rightarrow C \rightarrow A \rightarrow AC$  (2)  $A \rightarrow AC, AC \rightarrow D \rightarrow A \rightarrow D$
      - In the reverse:  $A \rightarrow D \rightarrow AC \rightarrow D$
  - Intuitively, a canonical cover of  $F$  is a "minimal" set of functional dependencies equivalent to  $F$ , having no redundant dependencies or redundant parts of dependencies

At the bottom left is the text 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kanpur Date: Jan-Apr., 2018'. At the bottom center is 'Database System Concepts - 8<sup>th</sup> Edition' and '16.8'. At the bottom right is '©Silberschatz, Korth and Sudarshan'.

Now, we move forward from there and talk about what is known as a canonical cover. A set of functional dependencies may have a number of redundant dependencies also. So, we need to understand that because there are lot of dependencies which can be inferred from a certain set of dependencies, for example if you look into this set, you will easily understand that in this whole set if I actually have just this, we will be able to by transitivity, we will be able to conclude about  $A \rightarrow C$ . So, in that way  $AC$ ,  $A \rightarrow C$  is a redundant dependency.

So, here I am just showing you some examples. For example, say I have a set of functional dependencies as this set and I want to know whether I can replace it by a simpler set here where this particular attribute on the right hand side of this dependency may be extraneous. So, if I have to do that, then what we need to perform is, we need to show that given the set of functional dependencies, the original set whether this can imply this set that is from this set of functional dependencies, whether I can logically conclude the simplified set.

So, using the rules we will need to do that I have worked that out here under the forward scheme and we would also need to establish that if I have the simplified set, then can I go to the original set that was given. So, if the simplified set also logically  $\rightarrow$  the original set, then we can say that these are in a way equivalent and therefore, I would like to use a simpler set.

So, there is another example following here where I have another set given, where if we look into this, I would like to check whether I can get rid of this C on the left hand side and as it stands, we can actually do that and here in this whole process, I have shown it in terms of using the Armstrong's Axioms how you can prove this, but what we can do to systematize this whole process, we can again make use of the notion of closure of attributes and compute whether these two sets are equivalent, whether simplification can be done. So, we will say a cover is canonical. If it is in a sense minimal and still equivalent to the original set of dependencies and we will formally introduce what is minimal.

(Refer Slide Time: 06:44)

**Canonical Cover: RHS**

- $\{A \rightarrow B, B \rightarrow C, A \rightarrow CD\} \xrightarrow{\quad} \{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$ 
  - (1)  $A \rightarrow CD \xrightarrow{\quad} A \rightarrow C$  and  $A \rightarrow D$  (2)  $A \rightarrow B, B \rightarrow C \xrightarrow{\quad} A \rightarrow C$
  - $A^+ = ABCD$
  
- $\{A \rightarrow B, B \rightarrow C, A \rightarrow D\} \xrightarrow{\quad} \{A \rightarrow B, B \rightarrow C, A \rightarrow CD\}$ 
  - $A \rightarrow B, B \rightarrow C \xrightarrow{\quad} A \rightarrow C$
  - $A \rightarrow C, A \rightarrow D \xrightarrow{\quad} A \rightarrow CD$
  - $A^+ = ABCD$

SWAYAM: NPTEL-NOCOs Instructor: Prof. P. P. Das, IIT Kharagpur. Date: Apr. 2018

Database System Concepts - 6<sup>th</sup> Edition      16.9      ©Silberschatz, Korth and Sudarshan

Before that let us just look at the same examples again. So, we are trying to show the forward direction in the first case and the reverse direction in the first case, but the only difference that I wanted to highlight is in terms of showing that you do not need to really explore on the Armstrong's Axioms, but what you can do is, you can simply take the left hand side attribute and compute its closure and see whether the right hand side is included. That is basically testing for whether the given functional dependency is actually implied.

(Refer Slide Time: 07:19)

The slide has a header 'PPD' and a title 'Canonical Cover: LHS'. It features a small sailboat icon in the top left. The main content consists of two bulleted lists under red square icons:

- $\{A \rightarrow B, B \rightarrow C, AC \rightarrow D\} \rightarrow \{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$ 
  - $A \rightarrow B, B \rightarrow C \rightarrow A \rightarrow C \rightarrow A \rightarrow AC$
  - $A \rightarrow AC, AC \rightarrow D \rightarrow A \rightarrow D$
  - $A^+ = ABCD$
- $\{A \rightarrow B, B \rightarrow C, A \rightarrow D\} \rightarrow \{A \rightarrow B, B \rightarrow C, AC \rightarrow D\}$ 
  - $A \rightarrow D \rightarrow AC \rightarrow D$
  - $AC^+ = ABCD$

On the left margin, vertical text reads 'SWAYAM/NIIT-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018'. At the bottom, there's a video frame showing a man speaking, a progress bar from 16.10 to 16.11, and a copyright notice '©Silberschatz, Korth and Sudarshan'.

Similar things can be done to simplify the left hand side also. So, this is the other example I showed and I am just showing you that how you conclude this based on the closure of attributes algorithm.

(Refer Slide Time: 07:32)

The slide has a header 'PPD' and a title 'Extraneous Attributes'. It features a small sailboat icon in the top left. The main content is a bulleted list:

- Consider a set  $F$  of functional dependencies and the functional dependency  $\alpha \rightarrow \beta$  in  $F$ .
  - Attribute  $A$  is **extraneous** in  $\alpha$  if  $A \in \alpha$  and  $F$  logically implies  $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$ .
  - Attribute  $A$  is **extraneous** in  $\beta$  if  $A \in \beta$  and the set of functional dependencies  $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$  logically implies  $F$ .

On the right side of the slide, there are handwritten notes in red ink:  
 $\alpha \rightarrow \beta$   
 $A \in \beta$

On the left margin, vertical text reads 'SWAYAM/NIIT-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018'. At the bottom, there's a video frame showing a man speaking, a progress bar from 16.11 to 16.12, and a copyright notice '©Silberschatz, Korth and Sudarshan'.

So, now I can formally define these possible removals. So, if I can remove an attribute as I have shown I can remove it from the right hand side or I can remove it from the left hand side. So, if an attribute can be removed, then it is called extraneous. So, if I have a functional dependency, let us say  $\alpha \rightarrow \beta$  and I have an attribute  $A \in \alpha$ , then we can check

whether it is possible to remove A from  $\alpha$ . So, to test that what we do is, we form a new set by removing the original functional dependency and adding the new functional dependency where the left hand side does not have that A and if F logically  $\rightarrow$  this, then certainly we can conclude that A on the left hand side of the functional dependency was extraneous.

Similar thing can be done for checking if there is an extraneous attribute on the right hand side of a dependency and in this case, naturally what we will need to do is, we will need to work out the simpler set and then check whether F is implied by that because as you can understand that if you are making the left hand, if you are removing an attribute from the left hand side, then you are making your precondition softer.

So, you need to see whether that is implied by the original set and on the other hand, if you are removing something on the right hand side, then you are making your consequence simpler. So, you need to understand whether that set  $\rightarrow$  the original set.

(Refer Slide Time: 09:27)

The slide has a header 'Extraneous Attributes' with a small sailboat icon. The content is organized into sections:

- Consider a set  $F$  of functional dependencies and the functional dependency  $\alpha \rightarrow \beta$  in  $F$ .
  - Attribute A is **extraneous** in  $\alpha$  if  $A \in \alpha$  and  $F$  logically implies  $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$ .
  - Attribute A is **extraneous** in  $\beta$  if  $A \in \beta$  and the set of functional dependencies  $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$  logically implies  $F$ .
- Note:* Implication in the opposite direction is trivial in each of the cases above, since a "stronger" functional dependency always implies a weaker one
- Example: Given  $F = \{A \rightarrow C, AB \rightarrow C\}$ 
  - B is extraneous in  $AB \rightarrow C$  because  $(A \rightarrow C, AB \rightarrow C)$  logically implies  $A \rightarrow C$  (i.e. the result of dropping B from  $AB \rightarrow C$ ).
  - $A^+ = AC$  in  $\{A \rightarrow C, AB \rightarrow C\}$
- Example: Given  $F = \{A \rightarrow C, AB \rightarrow CD\}$ 
  - C is extraneous in  $AB \rightarrow CD$  since  $AB \rightarrow C$  can be inferred even after deleting C
  - $AB^+ = ABCD$  in  $\{A \rightarrow C, AB \rightarrow D\}$

SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P P Date, IIT Kanpur - Jan-Apr., 2018

Database System Concepts - 8<sup>th</sup> Edition      16.11      ©Silberschatz, Korth and Sudarshan

So, if you look into that and obviously, the other directions of this implication is not necessary to be proven because that will automatically follow because in the first case when I am removing an attribute, extraneous attribute from the left hand side of a functional dependency, naturally the set that I get that will always imply the original set because it is always possible to add additional attributes on the left hand side and so on. So, here are some examples worked out. So, here where I show that given a set AC and

$\mathbf{AB} \rightarrow \mathbf{C}$ , B is actually extraneous because as you can see if I remove B, then I get  $\mathbf{A} \rightarrow \mathbf{C}$  which is originally already there in the set. You can establish that by computing the closure of the attribute set.

Another example where you are trying to see an extraneous attribute on the right hand side; so in this example, C on the right hand side of the set  $\mathbf{AB} \rightarrow \mathbf{CD}$  is extraneous because it can be inferred even after because  $\mathbf{AB} \rightarrow \mathbf{C}$  can be inferred even after deleting this C from the right hand side.

(Refer Slide Time: 10:42)

The slide has a title 'Testing if an Attribute is Extraneous' at the top right. On the left, there is a small logo of a sailboat on water. The main content is a bulleted list of steps:

- Consider a set  $F$  of functional dependencies and the functional dependency  $\alpha \rightarrow \beta$  in  $F$ .
- To test if attribute  $A \in \alpha$  is extraneous in  $\alpha$ 
  1. Compute  $((\alpha - A)^*)^*$  using the dependencies in  $F$
  2. Check that  $((\alpha - A)^*)^*$  contains  $\beta$ ; if it does,  $A$  is extraneous in  $\alpha$
- To test if attribute  $A \in \beta$  is extraneous in  $\beta$ 
  1. Compute  $\alpha^*$  using only the dependencies in  $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ ,
  2. Check that  $\alpha^*$  contains  $A$ ; if it does,  $A$  is extraneous in  $\beta$

At the bottom left, it says 'SWAYAM: NPTEL-NOCs; Instructor: Prof. P. P. Deshpande, IIT Kanpur - Jan-Apr., 2018'. At the bottom center, it says 'Database System Concepts - 8<sup>th</sup> Edition' and '16.12'. At the bottom right, it says '©Silberschatz, Korth and Sudarshan'.

So, these are using this notion. We can formalize a test for whether an attribute is extraneous. So, this is the formal steps of the step are given here, but I am sure you have already understood through the example.

(Refer Slide Time: 10:58)

The slide has a header 'Canonical Cover' in red. In the top right corner, there is a small logo with the letters 'PPD'. On the left side, there is a small image of a sailboat on water. The main content area contains the following text:

- A **canonical cover** for  $F$  is a set of dependencies  $F_c$  such that
  - $F$  logically implies all dependencies in  $F_c$ , and
  - $F_c$  logically implies all dependencies in  $F$ , and
  - No functional dependency in  $F_c$  contains an extraneous attribute, and
  - Each left side of functional dependency in  $F_c$  is unique
- To compute a canonical cover for  $F$ :  
repeat
  - Use the union rule to replace any dependencies in  $F$   
 $\alpha_1 \rightarrow \beta_1$  and  $\alpha_1 \rightarrow \beta_2$  with  $\alpha_1 \rightarrow \beta_1 \beta_2$
  - Find a functional dependency  $\alpha \rightarrow \beta$  with an extraneous attribute either in  $\alpha$  or in  $\beta$   
/\* Note: test for extraneous attributes done using  $F_c$ , not  $F^*$ /
  - If an extraneous attribute is found, delete it from  $\alpha \rightarrow \beta$until  $F$  does not change
- Note: Union rule may become applicable after some extraneous attributes have been deleted, so it has to be re-applied

At the bottom of the slide, there is a navigation bar with icons for back, forward, search, and other presentation controls. The footer contains the text 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr- 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '16.13', and '©Silberschatz, Korth and Sudarshan'.

So, given this a canonical cover of a set of functional dependencies  $F$ , it is denoted by  $F_c$  will mean that it is a set which is equivalent to  $F$  which means  $F$  will logically imply all dependencies in  $F_c$  and  $F_c$  will logically imply all dependencies in  $F$ .

No functional dependency in  $F_c$  will contain any extraneous attribute. So, all of them will be required attributes and each left hand side of the functional dependency in  $F_c$  must be unique. So, it is a minimal set of functional dependencies. Please note on these two core points. A cover is canonical if it is a minimal set and it is an irreducible set.

So, neither you can remove any dependency nor you can remove any extraneous attribute from this dependency set. So, here is the algorithm. So, I am not going through the steps of the algorithm. You can go through that and convince yourself that it indeed computes the canonical cover and practice more on that.

(Refer Slide Time: 12:07)

The slide has a header 'Computing a Canonical Cover' with a sailboat icon. On the left, there's a vertical sidebar with text: 'SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kanpur Date: Jan-Apr., 2018'. The main content area contains a bulleted list of steps for computing a canonical cover:

- $R = (A, B, C)$
- $F = \{A \rightarrow BC$ 
  - $B \rightarrow C$
  - $A \rightarrow B$
  - $AB \rightarrow C\}$
- Combine  $A \rightarrow BC$  and  $A \rightarrow B$  into  $A \rightarrow BC$ 
  - Set is now  $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$
- $A$  is extraneous in  $AB \rightarrow C$ 
  - Check if the result of deleting  $A$  from  $AB \rightarrow C$  is implied by the other dependencies
    - Yes: in fact,  $B \rightarrow C$  is already present!
  - Set is now  $\{A \rightarrow BC, B \rightarrow C\}$
- $C$  is extraneous in  $A \rightarrow BC$ 
  - Check if  $A \rightarrow C$  is logically implied by  $A \rightarrow B$  and the other dependencies
    - Yes: using transitivity on  $A \rightarrow B$  and  $B \rightarrow C$ .
    - Can use attribute closure of  $A$  in more complex cases

The canonical cover is: 
$$\begin{array}{l} A \rightarrow B \\ B \rightarrow C \end{array}$$

Navigation icons and copyright information at the bottom right: ©Silberschatz, Korth and Sudarshan

So, here I have shown an example where we want to compute the canonical cover here. So, first since all left hand sides have to be unique, so first we combine two, these two into in terms of  $A \rightarrow BC$ . So, it becomes a simpler set. So,  $A \rightarrow B$  is removed, then I would check for  $A$  being extraneous in  $AB \rightarrow C$  and we find that it indeed is extraneous.

So, because  $B \rightarrow C$  is already there, you can do the formal test in terms of the closure. So, the set gets even simpler. I will check if  $C$  is extraneous in  $A \rightarrow BC$ . I find that it indeed is and again you can use transitivity to get here or can use attribute closure and finally, I get that the set of the original set  $F$  is covered by a canonical set where just you have  $A \rightarrow B$  and  $B \rightarrow C$ .

So, this set is logically implied by the original set and this set can logically imply the original set and we will often use the canonical cover for simplicity and for ease of application.

(Refer Slide Time: 13:32)

The slide has a header 'Equivalence of Sets of Functional Dependencies' with a sailboat icon. It includes a sidebar with course information: SWAYAM: NPTEL-NOC MOOCs, Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr. 2018, and a footer 'PPD'. The main content discusses the equivalence of sets F & G of functional dependencies, defining it as  $F^+ = G^+$ . It lists four cases based on whether F covers G or G covers F:

Condition	CASES			
	F Covers G	True	True	False
G Covers F	True	False	True	False
Result	$F=G$	$F \supset G$	$G \supset F$	No Comparison

A photograph of the professor is on the left, and navigation icons are at the bottom right.

Naturally this is strongly using the underlying concept of equivalence of two sets of functional dependencies F and G. They are equivalent if their closures are equal or in other words, if F covers G and G covers F, that is F logically  $\rightarrow$  G and G logically  $\rightarrow$  F. So, this table shows you at different conditions where you can conclude whether F and G are equivalent sets of functional dependencies. So, they will have to, both covers have to be true for the sets to be equivalent.

(Refer Slide Time: 14:09)

The slide has a header 'Practice Problems on Functional Dependencies' with a sailboat icon. It includes a sidebar with course information: SWAYAM: NPTEL-NOC MOOCs, Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr. 2018, and a footer 'PPD'. The main content asks to find implied functional dependencies from a set of given ones:

- Find if a given functional dependency is implied from a set of Functional Dependencies:
  - For:  $A \rightarrow BC$ ,  $CD \rightarrow E$ ,  $E \rightarrow C$ ,  $D \rightarrow AEH$ ,  $ABH \rightarrow BD$ ,  $DH \rightarrow BC$ 
    - Check:  $BCD \rightarrow H$
    - Check:  $AED \rightarrow C$
  - For:  $AB \rightarrow CD$ ,  $AF \rightarrow D$ ,  $DE \rightarrow F$ ,  $C \rightarrow G$ ,  $F \rightarrow E$ ,  $G \rightarrow A$ 
    - Check:  $CF \rightarrow DF$
    - Check:  $BG \rightarrow E$
    - Check:  $AF \rightarrow G$
    - Check:  $AB \rightarrow EF$
  - For:  $A \rightarrow BC$ ,  $B \rightarrow E$ ,  $CD \rightarrow EF$ 
    - Check:  $AD \rightarrow F$

A photograph of the professor is on the left, and navigation icons are at the bottom right.

Next what I have done is, we have put a number of practice problems for various kind of things that you can do with functional dependencies. The first set of problems. Find in first set of problems you have to find if a given functional dependency is implied from a set of functional dependencies. So, there are three problems where three sets of functional dependencies are given and you are given to check one or more functional dependencies if it is implied from that set. So, use the attribute closure and the algorithm that we have discussed to practice these problems and become master of that.

(Refer Slide Time: 14:54)

The slide has a header 'Practice Problems on Functional Dependencies' with a sailboat icon. On the right, it says 'PPD'. The left margin contains vertical text: 'SWAYAM NPTEL-NOCO MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jain-Apr-2018'. The main content area has a bullet point '■ Find Super Key using Functional Dependencies:' followed by two numbered examples. Below this is a video player showing a man speaking, with the source 'Source: http://www.edugrabs.com/how-to-find-super-key-from-functional-dependencies' and a timestamp '16.18'. The bottom right corner says '©Silberschatz, Korth and Sudarshan'.

You can also check if you can find candidate key using the functional dependencies. The sets are given. Your task would be to find the candidate keys.

You can also use the algorithms to find super keys for a given set of functional dependencies. So, do practice these problems.

 Practice Problems on Functional Dependencies PPD

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

■ Find Prime and Non Prime Attributes using Functional Dependencies:

1. R(ABCDEF) having FDs {AB→C, C→D, D→E, F→B, E→F}
2. R(ABCDEF) having FDs {AB → C, C → DE, E → F, C → B}
3. R(ABCDEFGHIJ) having FDs {AB → C, A → DE, B → F, F → GH, D → IJ}
4. R(ABDLPT) having FDs {B → PT, A → D, T → L}
5. R(ABCDEFGH) having FDs {E → G, AB → C, AC → B, AD → E, B → D, BC → A}
6. R(ABCDE) having FDs {A → BC, CD → E, B → D, E → A}
7. R(ABCDEH) having FDs {A → B, BC → D, E → C, D → A}

- Prime Attributes – Attribute set that belongs to any candidate key are called Prime Attributes
  - It is union of all the candidate key attribute: {CK1 ∪ CK2 ∪ CK3 ∪ .....}
  - If Prime attribute determined by other attribute set, then more than one candidate key is possible.
  - For example, If A is Candidate Key, and X→A, then, X is also Candidate Key .
- Non Prime Attribute – Attribute set does not belongs to any candidate key are called Non Prime Attributes

Source: <http://www.edugrabs.com/prime-and-non-prime-attributes/>

Database System Concepts - 8<sup>th</sup> Edition 16.19 ©Silberschatz, Korth and Sudarshan

You can find prime and non-prime attributes using functional dependencies. Prime attributes are attributes that belong to any candidate key, not necessarily the same candidate key. All attributes that belong to some candidate key, you take a set together and you call them as a prime attribute and non prime attributes are those that do not belong to any candidate key at all. So, here your task is to find the prime and non-prime attributes using the sets of functional dependencies given.

(Refer Slide Time: 15:50)

 Practice Problems on Functional Dependencies PPD

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

■ Check the Equivalence of a Pair of Sets of Functional Dependencies:

1. Consider the two sets F and G with their FDs as below :
  1. F : A → C, AC → D, E → AD, E → H
  2. G: A → CD, E → AH
2. Consider the two sets P and Q with their FDs as below :
  1. P : A → B, AB → C, D → ACE
  2. Q : A → BC, D → AE

Source: <http://www.edugrabs.com/equivalence-of-sets-of-functional-dependencies/>

Database System Concepts - 8<sup>th</sup> Edition 16.20 ©Silberschatz, Korth and Sudarshan

You can check for equivalents for a pair of sets of functional dependencies. There are couple of problems given on that. So, please try them out.

(Refer Slide Time: 16:01)

**Practice Problems on Functional Dependencies**

PPD

■ Find the Minimal Cover or Irreducible Sets or Canonical Cover of a Set of Functional Dependencies:

1.  $AB \rightarrow CD, BC \rightarrow D$
2.  $ABCD \rightarrow E, E \rightarrow D, AC \rightarrow D, A \rightarrow B$

Source: <http://www.edugrabs.com/questions-on-minimal-cover/>

SWAYAM-NIETEL-NOC-MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur Date: 2018

16.21 ©Silberschatz, Korth and Sudarshan

For here for the different sets, you have to compute the minimal cover or the irreducible set or canonical cover of the set of functional dependencies.

So, please practice on this problem, so that you become comfortable with using this algorithms for dealing easily with the functional dependency sets of functional dependencies, individual functional dependencies and so on. So, after this week is closed and your assignments are also done, then we will publish the solutions for these practice problems as well next let me take up a little characterization of the concept that we had introduced earlier in terms of the lossless join decomposition.

(Refer Slide Time: 16:48)

The slide has a title 'Lossless-join Decomposition' at the top right. On the left, there is a small logo of a sailboat on water. The main content area contains a bulleted list of points about lossless join decomposition. A callout box provides conditions for identifying lossless or lossy decomposition.

**Lossless-join Decomposition**

- For the case of  $R = (R_1, R_2)$ , we require that for all possible relations  $r$  on schema  $R$   
 $r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$
- A decomposition of  $R$  into  $R_1$  and  $R_2$  is lossless join if at least one of the following dependencies is in  $F^+$ :
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$
- The above functional dependencies are a sufficient condition for lossless join decomposition; the dependencies are a necessary condition only if all constraints are functional dependencies

*To Identify whether a decomposition is lossy or lossless, it must satisfy the following conditions :*

- $R_1 \cup R_2 = R$
- $R_1 \cap R_2 \neq \emptyset$  and
- $R_1 \cap R_2 \rightarrow R_1$  or  $R_1 \cap R_2 \rightarrow R_2$

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

Database System Concepts - 8<sup>th</sup> Edition 16.23 ©Silberschatz, Korth and Sudarshan

So, in the lossless join decomposition, the problem is say that you have a relational scheme  $R$  and you are trying to divide that into two relational schemes  $R_1$  and  $R_2$ . So, both  $R_1$  and  $R_2$  are having a set of attributes and  $R$  naturally has a set of attributes which is a union of the attributes of  $R_1$  and  $R_2$ , then is it possible that if I take a relation, project it on the attributes of  $R_1$  and on the attributes of  $R_2$ , the two relations that we get. If I take a natural join of that, do I get back  $R$ ?

If I do, then I say that I have a lossless join. If I do not, then I have lost some information due to this projection and re-computation of the original relation based using the natural join. This requirement of lossless join decomposition is determined if at least one of the following dependencies exist in the closure set of  $F$  which this is saying that if I do  $\mathbf{R1} \cap \mathbf{R2}$ , that is A attributes which are common. You will recall that when we do natural join, it is this set of attributes which take part because these set of attributes will help you  $\Pi_{R1}(r) \ \Pi_{R2}(r)$ .

So, if  $\mathbf{R1} \cap \mathbf{R2} \rightarrow \mathbf{R1}$  or  $\mathbf{R1} \cap \mathbf{R2} \rightarrow \mathbf{R2}$ , that is if the intersection set of attributes is a super key either in  $R_1$  or in  $R_2$  or both then, we say that the join will be a lossless join. Note that this is a sufficient condition which means that there could be some instances where this property is not satisfied yet the join is lossless, but we need guarantees for our design. So, we make use of the fact that if one of these conditions are satisfied, then it is a sufficient condition to say that the join will must, join will necessarily be lossless.

(Refer Slide Time: 19:10)

**Example**

- Consider Supplier\_Parts schema: Supplier\_Parts(S#, Sname, City, P#, Qty) ✓
- Having dependencies: S# → Sname, S# → City, (S#, P#) → Qty ✓
- Decompose as: Supplier(S#, Sname, City, Qty) × Parts(P#, Qty)
- Take Natural Join to reconstruct: Supplier ⋈ Parts

S#	Sname	City	P#	Qty	S#	Sname	City	Qty	P#	Qty	S#	Sname	City	P#	Qty
3	Smith	London	301	20	3	Smith	London	20	301	20	3	Smith	London	301	20
5	Nick	NY	500	50	5	Nick	NY	50	500	50	5	Nick	NY	500	50
2	Steve	Boston	20	10	2	Steve	Boston	10	20	10	5	Nick	NY	20	10
5	Nick	NY	400	40	5	Nick	NY	40	400	40	2	Steve	Boston	20	10
5	Nick	NY	301	10	5	Nick	NY	10	301	10	5	Nick	NY	400	40

We get extra tuples! Join is Lossy!

Common attribute Qty is not a superkey in Supplier or in Parts

Does not preserve (S#, P#) → Qty

Source: <http://www.edugrabs.com/lossy-join-decomposition/>

So, here I give you a quick example to show the idea. So, we have a supplier relationship here which has five attributes. Here is an instance of that and we know that these are the dependencies that hold the **supplier number → the supplier name** and the **(supplier city, supplier number, product number) → quantity** and we decompose them in this manner, we put a supplier relationship where we have the number, name, city and quantity of supplier and then, we have parts relation where we just have a product name and the quantity.

So, this is the projected supplier relation instance. This is a projected parts relation instance and then, we take a natural join to reconstruct. So, we are taking a natural join to reconstruct and we get this relationship. Now, our desire was that we must get back the original relation, but if you compare, you will find that this is not the case here. We have one tuple here and we have another tuple here. I have specifically highlighted them in red which were not there in the original relation.

They have come in because when I did the join naturally, the join had to be performed on this common attribute quantity and based on that value. So, Nick, 5 Nick NY, then we have 10, 5 Nick NY 10, this entry and we have two entries of 10 and 10 here. So, the combination of this with this where the product number is 20 is actually not present in the original instance of the relation and that is what shows up here a similar one exists here.

So, we get extra tuples and mind you though we are actually getting extra tuple, we will say that this join is lossy because if you get extra tuple, then you are losing information, you are losing correctness. So, being lossy is actually losing correctness. So, even though we have more tuples, we say that this is a lossy join and you can now go back and analyze it. The common attribute QTY is not a super key either in this or in this. So,  $R1 \cap R2 \rightarrow R1$  or  $R1 \cap R2 \rightarrow R2$  does not hold. So, it does not and in addition it also does not preserve this functional dependency because these are not, no more determined.

Now, let us see it. So, we saw a case where the join decomposition that we did and then, the subsequent join that we performed did not prove to be a lossless join. We lost information. So, let us take a look as to can we actually do a decomposition which will be lossless where we will not lose information.

(Refer Slide Time: 22:29)

**Example**

- Consider Supplier\_Parts schema: Supplier\_Parts(S#, Sname, City, P#, Qty)
- Having dependencies:  $S\# \rightarrow Sname$ ,  $S\# \rightarrow City$ ,  $(S\#, P\#) \rightarrow Qty$
- Decompose as: Supplier(S#, Sname, City) Parts(S#, P#, Qty)
- Take Natural Join to reconstruct: Supplier  $\bowtie$  Parts

S#	Sname	City	P#	Qty	S#	Sname	City	S#	Sname	City	P#	Qty
3	Smith	London	301	20	3	Smith	London	3	Smith	London	301	20
5	Nick	NY	500	50	5	Nick	NY	5	Nick	NY	500	50
2	Steve	Boston	20	10	2	Steve	Boston	2	Steve	Boston	20	10
5	Nick	NY	400	40	5	Nick	NY	5	Nick	NY	400	40
5	Nick	NY	301	10	5	Nick	NY	5	Nick	NY	301	10

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

S#	Sname	City	P#	Qty	S#	Sname	City	S#	Sname	City	P#	Qty
3	Smith	London	301	20	3	Smith	London	3	Smith	London	301	20
5	Nick	NY	500	50	5	Nick	NY	5	Nick	NY	500	50
2	Steve	Boston	20	10	2	Steve	Boston	2	Steve	Boston	20	10
5	Nick	NY	400	40	5	Nick	NY	5	Nick	NY	400	40
5	Nick	NY	301	10	5	Nick	NY	5	Nick	NY	301	10

We get back the original relation. Join is Lossless.

Common attribute S# is a superkey in Supplier

Preserves all dependencies

Source: <http://www.edugrabs.com/desirable-properties-of-decomposition/lossless>

So, I take the same example the supplier, but the decomposition, the same set of dependencies also, but the decomposition is different. Now, we have name number, name and city in one supplier relation and supplier name, number, product number and quantity in the other parts relation and then, we again go back and perform the join.

Now, we find that from the original relation, this was the original relation, this is the projected supplier relation, these are projected parts relation and this is the natural join of these two relations. So, this is the natural join of these two relations and we find that they exactly match with the original relation.

So, we have not lost any information we get it back. So, we say that the join is lossless and the reason we could guarantee that is because if you look into the set of functional dependencies, you will find that S number, the supplier number is a key in the supplier relationship because  $S\# \rightarrow S \text{ name}$ ,  $S \text{ city}$ . So,  $R_1 \cap R_2 \rightarrow R_1$  is true here and therefore, it actually gives you a lossless join.

It also preserves all the dependencies because if you look into these dependencies, you can check for this dependency. In this relation, you can check for this dependency also in this relation and you can check for this dependency in also in the parts relation which is something which we were not able to do in the last decomposition that we have.

(Refer Slide Time: 24:33)

**Example**

- $R = (A, B, C)$   
 $F = \{A \rightarrow B, B \rightarrow C\}$ 
  - Can be decomposed in two different ways
- $R_1 = (A, B), R_2 = (B, C)$ 
  - Lossless-join decomposition:  
 $R_1 \cap R_2 = \{B\}$  and  $B \rightarrow BC$
  - Dependency preserving
- $R_1 = (A, B), R_2 = (A, C)$ 
  - Lossless-join decomposition:  
 $R_1 \cap R_2 = \{A\}$  and  $A \rightarrow AB$
  - Not dependency preserving  
(cannot check  $B \rightarrow C$  without computing  $R_1 \bowtie R_2$ )

SWAYAM/NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr. 2018

Dat... 16.26 ©Silberschatz, Korth and Sudarshan

So, naturally this is a type of decomposition that we will prefer. So, here we have I have given some more examples which you can practice and I show that given a very simple schema having three attributes and two dependencies, one decomposition into AB and BC is lossless join decomposition whereas, the other one AB and AC is a lossy decomposition.

(Refer Slide Time: 24:57)

The slide features a sailboat icon in the top left corner and the text 'PPD' in the top right corner. The title 'Practice Problems on Lossless Join' is centered at the top. Below the title, there is a list of numbered problems under the heading 'Check if the decomposition of R into D is lossless:'. The problems involve decomposing various sets of functional dependencies (F) into sets of relations (D) and then checking if the decomposition is lossless. The source of the problems is cited as <http://www.edugrabs.com/questions-on-lossless-join/>. A small video thumbnail of a man speaking is on the left, and a navigation bar with icons is at the bottom.

I have given a number of practice problems on lossless join, so that you can practice and become master of these kind of algorithm. Finally, let me quickly go over the dependency preservation concept.

(Refer Slide Time: 25:17)

The slide features a sailboat icon in the top left corner and the text 'SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P. P. Des., IIT Kharagpur - Jan-Apr- 2018' on the left side. The title 'Dependency Preservation' is centered at the top. Below the title, there is a list of points under the heading 'Let  $F_i$  be the set of dependencies  $F^+$  that include only attributes in  $R_i$ '. The points explain what dependency preserving means and note that it may require computing joins. A box contains a detailed explanation of dependency preservation: 'Let R be the original relational schema having FD set F. Let  $R_1$  and  $R_2$  having FD set  $F_1$  and  $F_2$  respectively, are the decomposed sub-relations of R. The decomposition of R is said to be preserving if  $F_1 \cup F_2 \equiv F$  (Decomposition Preserving Dependency). If  $F_1 \cup F_2 \subset F$  (Decomposition NOT Preserving Dependency) and  $F_1 \cup F_2 \supset F$  (this is not possible)'.

Dependency preservation is if you have a relation which you have decomposed into n different relations, so if you decompose a relation into a number of relations, then naturally all functional dependencies you cannot check on all the relations because a

dependency may involve attributes all of which may not be present in a particular decomposed relation that you have. It may be distributed amongst different.

So, when you do this decomposition, for every relation you get a new set of subset of functional dependencies. So, the decomposed relation  $R_i$  the  $i^{\text{th}}$  relation will have a set of dependencies  $F_i$  which is a subset of the original set  $F$  and involves only the attributes which exist in  $R_i$ . So, the decomposition will be said to be dependency preserving if I can take the union of all these functional dependencies, what is projected on  $R_1$ , on  $R_2$  and  $R_n$ ,  $F_1, F_2, F_n$ . If we can take union of and if we take  $F$ , they must be equivalent sets which we know the requirement.

So, equivalence mean that their covers will have to be equal. If it is not, then some there will be at least one dependency which you will not be able to check in any one of the projected relations and to be able to check that, you will have to compute the natural join and that is as we know is a very expensive process and we would not be able to do that on a regular basis.

(Refer Slide Time: 27:12)

The slide has a title 'Testing for Dependency Preservation' in red. To the left is a small logo of a sailboat on water. On the right is a video frame showing a man speaking. The slide contains the following text:

To check if a dependency  $\alpha \rightarrow \beta$  is preserved in a decomposition of  $R$  into  $R_1, R_2, \dots, R_n$  we apply the following test (with attribute closure done with respect to  $F$ )

- $\text{result} = \alpha$
- **while** (changes to  $\text{result}$ ) **do**
- for each**  $R_i$  **in the decomposition**
- $t = (\text{result} \cap R_i)^* \cap R_i$
- $\text{result} = \text{result} \cup t$
- If  $\text{result}$  contains all attributes in  $\beta$ , then the functional dependency  $\alpha \rightarrow \beta$  is preserved.
- We apply the test on all dependencies in  $F$  to check if a decomposition is dependency preserving
- This procedure takes polynomial time, instead of the exponential time required to compute  $F^*$  and  $(F_1 \cup F_2 \cup \dots \cup F_n)^*$

At the bottom, it says 'SWAYAM NPTEL-NOC's Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr- 2018'. There is also a footer with 'Database System Concepts - 8<sup>th</sup> Edition', '16.30', and '©Silberschatz, Korth and Sudarshan'.

So, here I have written down the algorithm to test if a decomposition actually preserves the dependency or not. So, I will not go through the steps. I will leave that for you to understand, but what I will do, I will just show you a simple set of worked out example and reason on that.

(Refer Slide Time: 27:34)

The slide has a title 'Example' in red at the top right. On the left, there is a small logo of a sailboat. The main content area contains the following text and tables:

- R(ABCDEF):
- F = {A→BCD, A→EF, BC→AD, BC→E, BC→F, B→F, D→E}
- D = {ABCD, BF, DE}
- On projections:

ABCD (R1)	BF (R2)	DE (R3)
A → BCD BC → AD	B → F	D → E

Below the table, there is a list of dependencies and their preservation status:

- Need to check for: A→BCD, A→EF, BC→AD, BC→E, BC→F, B→F, D→E
- (BC)+/F1 = ABCD, (ABCD)+/F2 = ABCDF, (ABCD)+/F3 = ABCDEF. Preserves BC→E, BC→F
- (A)+/F1 = ABCD, (ABCD)+/F2 = ABCDF, (ABCD)+/F3 = ABCDEF. Preserves A→EF

At the bottom left, there is a video feed of a professor. The bottom right shows a toolbar with various icons.

So, I show you two different methods of doing this. So, here we have a set of attributes given the dependencies that work in that and a particular decomposition. So, given the set of attributes and the decomposition if we project, now if we project the set of functional dependencies and these are the sets that we get. So, on R1, we have two dependencies on R2, we have three dependence, one dependency and R3 we have one dependency again.

So, if we now think about the U of these and the closure for that, then we can see that these four dependencies which occur here and therefore, I have struck them off in this set. These four dependencies can be checked directly on the projected relations. So, that leaves us with three dependencies in the original set which cannot be checked on any one of R1, R2 or R3. For example, if you consider **BC** → **E**, then B exist on R1 and C also exist on R1, but E is not there. So, you cannot check that dependency on R1, you cannot check that on R2 because C and E do not exist and you cannot check them on R3, check it on R3 because none of them actually exist.

So, what we will need for the dependency preservation to hold is the dependencies which are already existing four dependencies that are struck off if they collectively can logically imply these dependencies, so that they can be checked. Then, we will be able to say that this is dependency preserving. So, what you do is something very simple. You want to say, you want to check whether this is preserved. So, we start with the left hand

side and compute the closure. The only difference you compute the closure first with the set of functional dependencies projected on R1, that is F1, the set closure set that you get, you take that and compute its closure with respect to the second set of functional dependencies F2.

The closure that you get, you take that and you compute the closure with respect to the third set of functional dependencies which is on R3 and that is your final closure set. So, this closure set includes the right hand side attribute E. So, we can conclude that  $\text{BC} \rightarrow \text{E}$  and that relationship will be preserved because we have starting from BC. We have seen that in every projected relation what all implied functional dependencies that can be checked which is what the meaning of the closure set of attributes R and since that set eventually has E, we will know that this can be, this will be preserved.

This set also has F. So, the other one will also be preserved. So, this is preserved, this is preserved to check whether this dependency is preserved. We need to again repeat the process and find whether EF belongs to the final closure set which it does and therefore, we conclude that this decomposition is dependency preserving.

(Refer Slide Time: 31:04)

**Example**

PPD

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Doshi, IIT Kanpur - Jan-Apr - 2018

- R(ABCDEF): F = {A→BCD, A→EF, BC→AD, BC→E, BC→F, B→F, D→E}. D = {ABCD, BF, DE}
- On projections:
 

ABCD (R1)	BF (R2)	DE (R3)
-----------	---------	---------

 A → B, A → C, A → D, BC → A, BC → D, B → F, D → E
- Infer reverse FD's:
  - B+F = BF: B → A cannot be inferred
  - C+F = C: C → A cannot be inferred
  - D+F = DE: D → A and D → BC cannot be inferred
  - A+F = ABCDEF: A → BC can be inferred, but it is equal to A → B and A → C
  - F+F = F: F → B cannot be inferred
  - E+F = E: E → D cannot be inferred
- Need to check for: A→BCD, A→EF, BC→AD, BC→E, BC→F, B→F, D→E
  - (BC)+E = ABCDEF. Preserves BC→E, BC→F ✓ ✓
  - (A)+F = ABCDEF. Preserves A→EF

With the same example I will just show you a little different way of ah doing the same exercise. I have not written down the algorithm for this in longhand, but the example should be quite illustrative. So, we are what you do when you project, you check if some dependency has multiple attributes on the left hand, on the right hand side, then you

write them in a separately decomposed manner. So,  $A \rightarrow BCD$  is written in terms of three dependencies.  $A \rightarrow B$ ,  $B \rightarrow C$  and  $C \rightarrow D$ . So, you make sure that all dependencies are written in a form where the right hand side has a single attribute, then you compute what is known as the reverse functional dependencies that is you take the right hand side and compute whether the right hand side can imply the left hand side.

So, I will just show you one. So, in case the right hand side here is B, you have AB on the right hand side. So, you compute the closure with respect to F. The original set, not the projected set of D and you get BF. So, you know that this inverse, this reverse functional dependency which is  $AB \rightarrow A$  which is the reverse dependency cannot be inferred and you do this for each of the right hand side single attribute and check if some, if the reverse dependencies can be inferred or not.

The interesting case occurs here where if you try to do the closure of A, you actually find that  $A \rightarrow BC$  which is a reverse of this functional dependency can be inferred, but you do not consider that as a violation because it is you already have  $A \rightarrow B$  and  $A \rightarrow C$ . So, that logically implying that  $A \rightarrow BC$ . So, it is not a new violation that is getting imposed.

So, with this your test for reverse functional dependencies is passed and then, you finally check for whether the three dependencies which are not part of the projected set of dependencies, you take the closure of the left hand side with respect to in this case. Again, the original set of functional dependencies, not the projected one and check if the right hand side belongs there. If they do, then combined with these two strategies you say that the set of functional dependencies are preserved under this decomposition.

So, this is the process to follow. You can follow any one of the two approaches to solve.

(Refer Slide Time: 34:01)

The slide features a small sailboat icon in the top left corner. The title 'Practice Problems on Dependency Preservation' is centered at the top in a red header. Below the title, a bullet point says 'Check whether the decomposition of R into D is preserving dependency:' followed by five numbered questions. The footer contains the text 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018', 'Source: http://www.edugrabs.com/question-on-dependency-preserving-deco...', 'Database System Concepts - 8<sup>th</sup> Edition', '16.33', and '©Silberschatz, Korth and Sudarshan'.

So, I have given some practice problems on dependency preservation which you should practice on to.

(Refer Slide Time: 34:06)

The slide features a small sailboat icon in the top left corner. The title 'Module Summary' is centered at the top in a red header. Below the title, there is a bulleted list of three items. The footer contains the text 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '16.34', and '©Silberschatz, Korth and Sudarshan'.

Summarize we have studied the algorithms for properties of functional dependencies and we have understood the characterization and determination algorithm for lossless join decomposition and for dependency preservation in a decomposition. In the coming module, we will make use of these and discuss about how to improve these designs of relational schemas through the use of different normal forms.

**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.

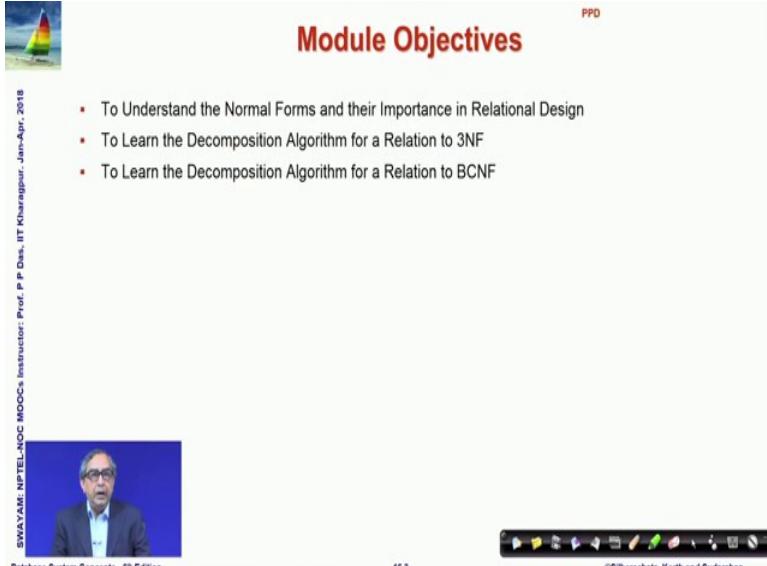
(Refer Slide Time: 00:38)

The slide is titled "Module Recap" in red at the top right. It features a small sailboat icon in the top left corner. On the far left, there is vertical text: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur, Jan-Apr., 2018". In the bottom left, it says "Database System Concepts - 8<sup>th</sup> Edition". The bottom right contains copyright information: "©Silberschatz, Korth and Sudarshan". The main content area lists three bullet points under the heading "PPD":

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

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)

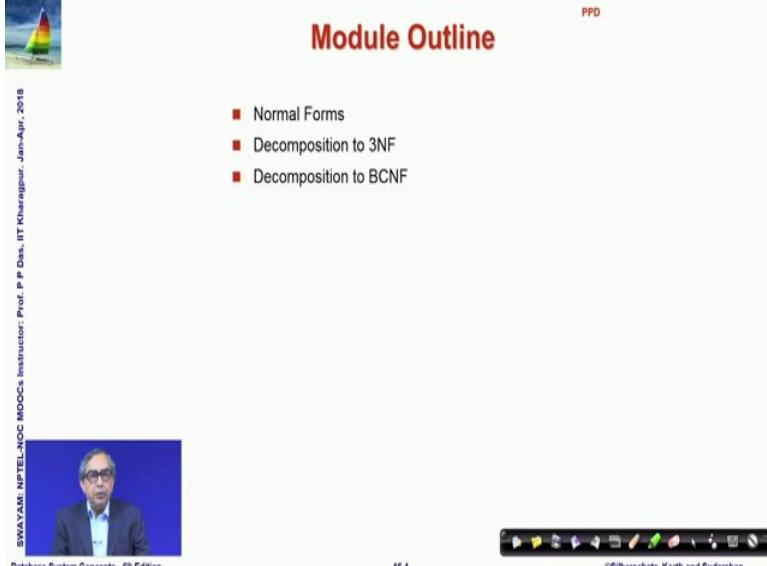


The slide is titled "Module Objectives" in red. It features a small sailboat icon in the top left corner and a video player window showing a man speaking in the bottom left. The footer includes the text "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr., 2018", "Database System Concepts - 8<sup>th</sup> Edition", "16.3", "PPD", and "©Silberschatz, Korth and Sudarshan".

- 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

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 is titled "Module Outline" in red. It features a small sailboat icon in the top left corner and a video player window showing a man speaking in the bottom left. The footer includes the text "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr., 2018", "Database System Concepts - 8<sup>th</sup> Edition", "16.4", "PPD", and "©Silberschatz, Korth and Sudarshan".

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

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

(Refer Slide Time: 01:27)

The slide features a small sailboat icon in the top left corner. In the top right, the text "PPD" is written vertically. Below it, a bulleted list includes "Normal Forms", "Decomposition to 3NF", and "Decomposition to BCNF". The main title "NORMAL FORMS" is centered in large red capital letters. Below the title is a video frame showing a man speaking. The bottom of the slide contains the text "Database System Concepts - 8<sup>th</sup> Edition", the page number "16.5", and the copyright notice "©Silberschatz, Korth and Sudarshan". On the far left, vertical text reads "SWAYAM: NPTEL NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr. 2018".

So, starting with the normal forms.

(Refer Slide Time: 01:29)

The slide has a small sailboat icon in the top left. The title "Normalization or Schema Refinement" is centered in large red capital letters. To the right, the text "PPD" is written vertically. Below the title is a bulleted list detailing the purpose and techniques of normalization, mentioning insertion, update, and deletion anomalies, and the goal of eliminating redundancy. The bottom of the slide includes the text "Database System Concepts - 8<sup>th</sup> Edition", the page number "16.6", and the copyright notice "©Silberschatz, Korth and Sudarshan". On the far left, vertical text reads "SWAYAM: NPTEL NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr. 2018".

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 anomalies 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**

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur Date: Jan-Apr., 2018

**1. Update Anomaly:** Employee 519 is shown as having different addresses on different records

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

**2. Insertion Anomaly:** Until the new faculty member, Dr. Newsome, is assigned to teach at least one course, his details cannot be recorded

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	?

**3. Deletion Anomaly:** All information about Dr. Giddens is lost if he temporarily ceases to be assigned to any 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

**Resolution: Decompose the Schema**

1. Update: (ID, Address), (ID, Skill)
2. Insert: (ID, Name, Hire Date), (ID, Code)
3. Delete: (ID, Name, Hire Date), (ID, Code)

Database System Concepts - 8<sup>th</sup> 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 → employee address** 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 an insertion anomaly. Similarly I could have a deletion anomaly 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.

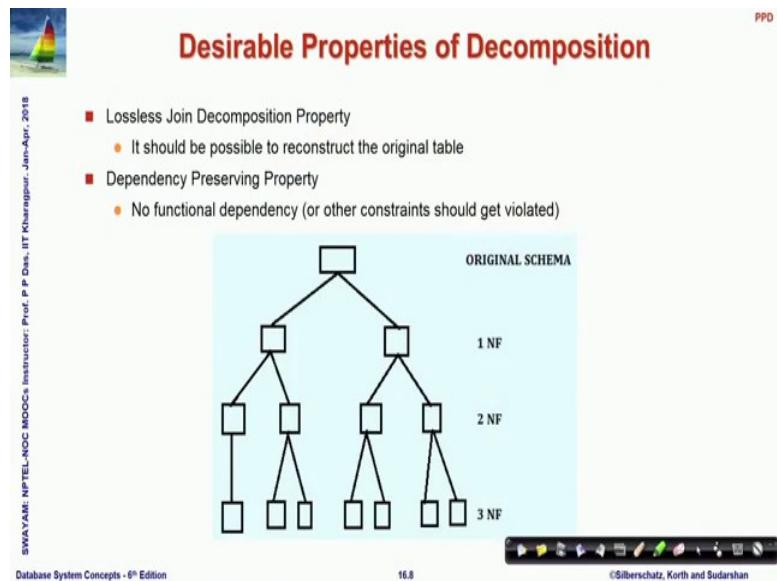
So, the association between 389 and the corresponding course code will be removed and once you remove that you remove this whole record in the process you actually lose the whole of the faculty information the ID, name and hire date. So, these are difficulties in these relational schemas and that lead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



Now, when we decompose then we would desire certain properties to be 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 MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018

Database System Concepts - 8<sup>th</sup> 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 MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018



Database System Concepts - 8<sup>th</sup> 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			Students		
SID	Sname	Cname	SID	Sname	Cname
S1	A	C,C++	S1	A	C
S2	B	C++,DB	S1	A	C++
S3	A	DB	S2	B	C++
<b>SID : Primary Key</b>			S2	B	DB
<b>MVA exists → Not in 1NF</b>			S3	A	DB
			<b>SID : Primary Key</b>		

**No MVA → In 1NF**

Source: <http://www.edugrabs.com/normal-forms/#fn>

Database System Concepts - 6<sup>th</sup> 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 think about a relationship where you have a student relationship between student 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)

**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.

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

Database System Concepts - 8<sup>th</sup> 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)

**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

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

Database System Concepts - 8<sup>th</sup> Edition

16.13

©Silberschatz, Korth and Sudarshan

So, it is a possible that if I have a functional dependency  $X \rightarrow Y$  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 we have shown an

instance of this relationship is 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 1 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)

The slide has a header 'Second Normal Form (2 NF)' in red. On the left is a small sailboat icon. On the right is a small 'PPD' logo. The main content area contains the following text:

- 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 2 NF.

$(Y \rightarrow A)$  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>

16.14

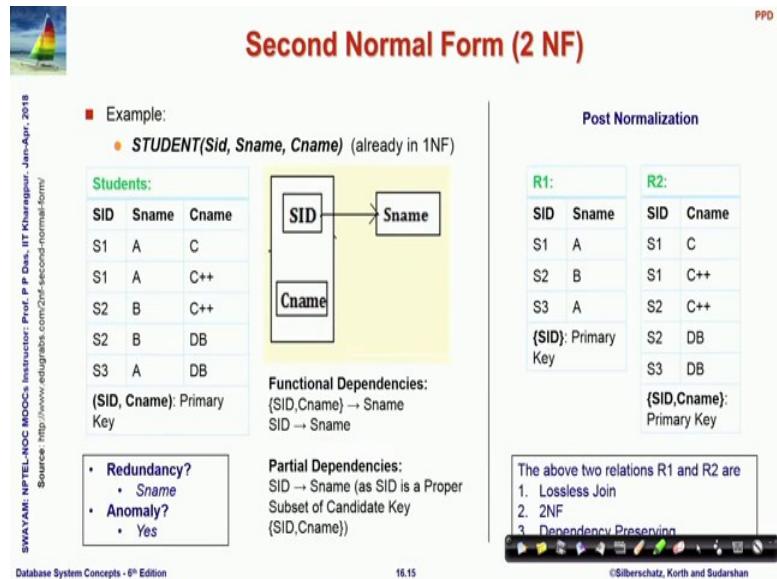
©Silberschatz, Korth and Sudarshan

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 functionally determines A if 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)



So, here I were showing an example where on the left you can see that SID and Cname together forms a key and  $\text{SID} \rightarrow \text{Sname}$ . So,  $(\text{SID}, \text{Cname}) \rightarrow \text{Sname}$  naturally  $\text{SID} \rightarrow \text{Sname}$  is a partial dependency because the left hand side  $\text{SID} \subseteq$  candidate key {SID, Cname}. And Sname is not featuring in any candidate key. So, Sname 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 Sname 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 Sname in one table and SID and Cname in other table. Naturally then the dependency that the partial dependency that you had disappears because  $\text{SID} \rightarrow \text{Sname}$  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 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)

**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	Kamal	P1	115
S2	10	Kamal	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 :	Sup_Qty :
SID   Status   City	SID   PID   Qty

**FDD of Sup\_City :**

```

graph TD
    SID --> Status
    SID --> City
    Status --> City

```

**FDD of Sup.Qty :**

```

graph TD
    Qty --> SID
    Qty --> PID

```

**Drawbacks:**

- **Deletion Anomaly** – If we delete a tuple in Sup\_City, then we not only lose the information about a supplier, but also lose 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.
- **Updation 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 - 8<sup>th</sup> 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 → 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 :

- $\text{City} \rightarrow \text{Status}$
- $\text{Non Key Attribute} \rightarrow \text{Non Key Attribute}$

In the **STUDENT** relation:

- $\text{SID} \rightarrow \text{Cname}$
- $\text{Proper Subset of one CK} \rightarrow \text{Proper Subset of other CK}$

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

Database System Concepts - 8<sup>th</sup> 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!)

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

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 I should not if I have  $X \rightarrow Y$  and  $Y \rightarrow Z$ ; then I should not have  $X \rightarrow Z$  which can be inferred transitively as you know through the angstrom axiom.

Alternately, there was an alternate definition given later on by Zanilo 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  $X \rightarrow A$  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)

The slide features a small sailboat icon in the top-left corner. The title 'Third Normal Form (3 NF)' is centered at the top in a red box. The main content consists of several bullet points explaining transitive dependency:

- 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

At the bottom left, vertical text reads: SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018. At the bottom right, it says: Database System Concepts - 8<sup>th</sup> 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)

The slide features a small sailboat icon in the top-left corner. The title 'Third Normal Form (3 NF)' is centered at the top in a red box. The main content consists of several bullet points explaining transitive dependency, followed by a table of book data:

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

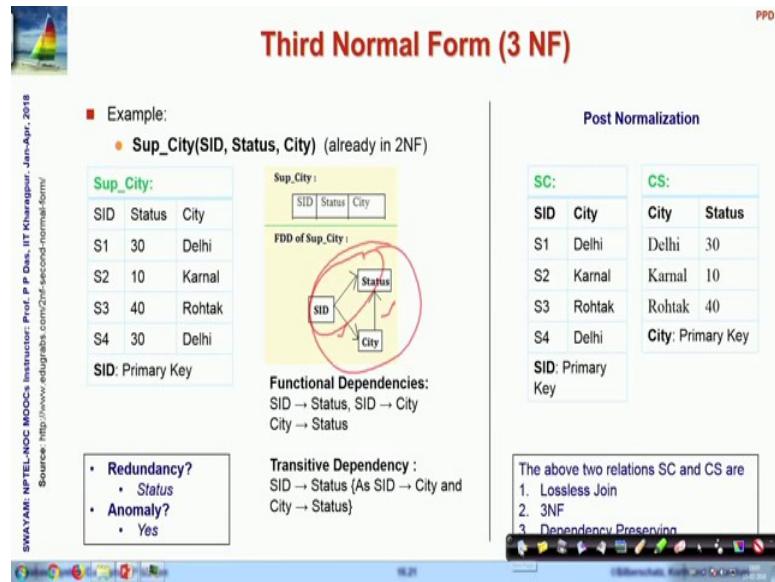
Book	Genre	Author	Author Nationality
Twenty Thousand Leagues Under the Sea	Science Fiction	Jules Verne	French
Journey to the Center of the Earth	Science Fiction	Jules Verne	French
Leaves of Grass	Poetry	Walt Whitman	American
Anna Karenina	Literary Fiction	Leo Tolstoy	Russian
A Confession	Religious Autobiography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

At the bottom left, vertical text reads: SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018. At the bottom right, it says: Database System Concepts - 8<sup>th</sup> 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 does not actually determine the book because the author may have written multiple books. But given that **book → author** and **author → author**

**nationality** we have that **book → 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)



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 → city** and **city → status**. So, this is it this is the transitive dependency that **SID → 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 → the city** which is a primary key in SC and **city → status** which is the primary key in the CS.

(Refer Slide Time: 19:50)

The slide has a header 'Third Normal Form (3 NF)' with a sailboat icon. On the left, vertical text reads 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018'. The main content includes a bulleted list of points about 3NF, a box defining 3NF, and footer information.

**Third Normal Form (3 NF)**

- Example
  - Relation `dept_advisor`:
- `dept_advisor(s_ID, i_ID, dept_name)`
- $F = \{s\_ID, \text{dept\_name} \rightarrow i\_ID, i\_ID \rightarrow \text{dept\_name}\}$
- Two candidate keys: `s_ID, dept_name`, and `i_ID, s_ID`
- $R$  is in 3NF
  - $s\_ID, \text{dept\_name} \rightarrow i\_ID$ 
    - `s_ID, dept_name` is a superkey
  - $i\_ID \rightarrow \text{dept\_name}$ 
    - `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!)

Database System Concepts - 8<sup>th</sup> Edition      16.22      ©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 s\_ID department name is a super key and this relationship is in the third normal form. Because **i\_ID → 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)

The slide has a header 'Redundancy in 3NF' with a sailboat icon. On the left, vertical text reads 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018'. The main content includes a bulleted list of redundancy issues, a table showing data, and a list of problems. The footer includes a toolbar and copyright information.

**Redundancy in 3NF**

- There is some redundancy in this schema
- Example of problems due to redundancy in 3NF ( $J: s\_ID, L: i\_ID, K: \text{dept\_name}$ )
  - $R = (J, L, K)$   
 $F = \{JK \rightarrow L, L \rightarrow K\}$

	$J$	$L$	$K$
$j_1$	$l_1$	$k_1$	
$j_2$	$l_1$	$k_1$	
$j_3$	$l_1$	$k_1$	
null	$l_2$	$k_2$	

- repetition of information (e.g., the relationship  $l_1, k_1$ )
  - $(i\_ID, \text{dept\_name})$
- need to use null values (e.g., to represent the relationship  $l_2, k_2$  where there is no corresponding value for  $J$ ).
  - $(i\_ID, \text{dept\_name})$  if there is no separate relation mapping instructors to departments

Database System Concepts - 8<sup>th</sup> Edition      16.23      ©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

PPD

■ 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)

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018  
 Source: <http://www.edugrabs.com/2nf-second-normal-form/>  
 Database System Concepts - 6<sup>th</sup> 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)



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

## DECOMPOSITION TO 3NF

PPD

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018  
 Database System Concepts - 6<sup>th</sup> 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 no 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 has a title 'Third Normal Form: Motivation' in red. To the left of the title is a small image of a sailboat on water. On the right side of the slide, there is a vertical sidebar with text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr. 2018'. Below the title is a bulleted list of points:

- 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 of the slide, there is a video player interface showing a thumbnail of a person speaking, the text 'Database System Concepts - 8<sup>th</sup> Edition', the time '16.26', and the copyright notice '©Silberschatz, Korth and Sudarshan'.

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 where third normal form does that.

(Refer Slide Time: 22:49)

The slide features a small sailboat icon in the top left corner. The title 'Testing for 3NF' is centered at the top in a red font. Below the title is a bulleted list of optimization points:

- 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  $\alpha$  is a superkey.
- If  $\alpha$  is not a superkey, we have to verify if each attribute in  $\beta$  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

On the left side of the slide, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018'. At the bottom left is the text 'Database System Concepts - 8<sup>th</sup> Edition'. The bottom right shows a navigation bar with icons and the text '©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)

The slide features a small sailboat icon in the top left corner. The title '3NF Decomposition Algorithm' is centered at the top in a red font. Below the title is a bulleted list of steps:

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

On the left side of the slide, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018'. At the bottom left is the text 'Database System Concepts - 8<sup>th</sup> Edition'. The bottom right shows a navigation bar with icons and the text '©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 cover you eliminate extraneous attributes eliminate redundant FDs and you have the canonical cover  $F_c$  from  $F$  then you create for every functional dependency  $X \rightarrow Y$  that exists in the canonical cover.

You compute you make a relation say the  $i^{\text{th}}$  relation taking  $X \cup Y$ . So, you call it the relation  $XY$  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  $R_j$  is contained in another schema  $R_k$ 
        then /* delete  $R_j$  */
             $R_j = R_k$ ;
             $i := i - 1$ ;
return  $(R_1, R_2, \dots, R_i)$ 

```

SVAYAM: NPTEL NOC Instructor: Prof. P. P. Deshpande, IIT Kanpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition

16.29

©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 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)

The slide has a title '3NF Decomposition Algorithm' at the top right. On the left, there is a small sailboat icon. Below the title, there is a bulleted list of points:

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

On the left side of the slide, there is vertical text: 'SWAYAM: NPTEL NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr- 2018'. At the bottom left, it says 'Database System Concepts - 8<sup>th</sup> Edition'. In the bottom center, it shows '16.30'. At the bottom right, it says '©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)

The slide has a title 'Example of 3NF Decomposition' at the top right. On the left, there is a small sailboat icon. Below the title, there is a bulleted list of points:

- Relation schema:  
 $cust\_banker\_branch = (customer\_id, employee\_id, branch\_name, type)$

Below the list, there is a large screenshot of a computer screen showing a database diagram or code editor. The code appears to be a SQL query or a series of statements related to the 'cust\_banker\_branch' schema. The text is mostly illegible due to the high contrast between the black background and the white text.

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 - 8<sup>th</sup> 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) → type**. So, for that you generate a schema customer ID, employee ID and type again you take the second functional dependency **employee ID → 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 MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018

Database System Concepts - 8<sup>th</sup> 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 MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018

Database System Concepts - 8<sup>th</sup> 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 MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> 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  $\alpha^+$  (the attribute closure of  $\alpha$ ), 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 MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> 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  $\alpha^+$  (the attribute closure of  $\alpha$ ) 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 MOOCs Instructor: Prof. P P Desai, IIT Kharagpur - Jan-Apr- 2018  
Database System Concepts - 8<sup>th</sup> 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

PPD

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  $F_{2+}$

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Desai, IIT Kharagpur - Jan-Apr- 2018  
Database System Concepts - 8<sup>th</sup> 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  $A \rightarrow B$  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  $A \rightarrow B$  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 U 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  $A \rightarrow B$ .

So,  $A \rightarrow A B$  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 Rj in result that is not in BCNF)
        then begin
            let α → β be a nontrivial functional dependency that
            holds on Rj such that α → Rj is not in F+,
            and α ∩ β = ∅;
            result := (result - Rj) ∪ (Rj - β) ∪ (α, β);
            end
        else done := true;

```

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

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Desai, IIT Kanpur - Jan-Apr-2018  
Database System Concepts - 8<sup>th</sup> 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)

The slide features a sailboat icon in the top left corner. The title 'Example of BCNF Decomposition' is centered at the top in red. To the right of the title is a list of bullet points. On the left side, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr- 2018'. In the bottom left corner, there is a small video thumbnail showing a man speaking. The bottom right corner contains the copyright notice '©Silberschatz, Korth and Sudarshan'.

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

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  $A \rightarrow B$ ;  $B \rightarrow C$  naturally A is the key; R is not in BCNF because  $B \rightarrow C$  is a functional dependency where B is not a super key.

(Refer Slide Time: 30:15)

The slide features a sailboat icon in the top left corner. The title 'Example of BCNF Decomposition' is centered at the top in red. To the right of the title is a list of bullet points. On the left side, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr- 2018'. In the bottom left corner, there is a small video thumbnail showing a man speaking. The bottom right corner contains the copyright notice '©Silberschatz, Korth and Sudarshan'.

- 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)

So, you can decompose them in terms of. So, you can decompose in terms of BC as one relation and AB 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 Instructor: Prof. P P Desai, IIT Kharagpur - Jan-Apr. 2018  
Database System Concepts - 8<sup>th</sup> 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,room number)→ capacity** where building room number together is not a super key. So, you split it again and you replace class 1 in terms of two new relations class room and section and both of them are in BCNF and you are done with this.

(Refer Slide Time: 31:31)

The slide has a title 'BCNF and Dependency Preservation' at the top right. On the left, there is a small logo of a sailboat on water. The background is white with a light blue gradient at the bottom. The text area contains the following points:

- 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

At the bottom left, it says 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018'. At the bottom center, it says 'Database System Concepts - 6<sup>th</sup> Edition' and '16.45'. At the bottom right, it says '©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  $JK \rightarrow L$  and that will require a join. So, this will not preserve the dependencies in terms of the decomposition.

(Refer Slide Time: 32:06)

The slide has a title 'Practice Problem for BCNF Decomposition' at the top right. On the left, there is a small logo of a sailboat on water. The background is white with a light blue gradient at the bottom. The text area contains the following list of schema examples:

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

At the bottom left, it says 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018'. At the bottom center, it says 'Database System Concepts - 6<sup>th</sup> Edition' and '16.46'. At the bottom right, it says '©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 MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018  
 Database System Concepts - 8th 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)

**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 MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: 2018

Database System Concepts - 8<sup>th</sup> 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.

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

**Lecture – 20**  
**Relational Database Design (Contd.)**

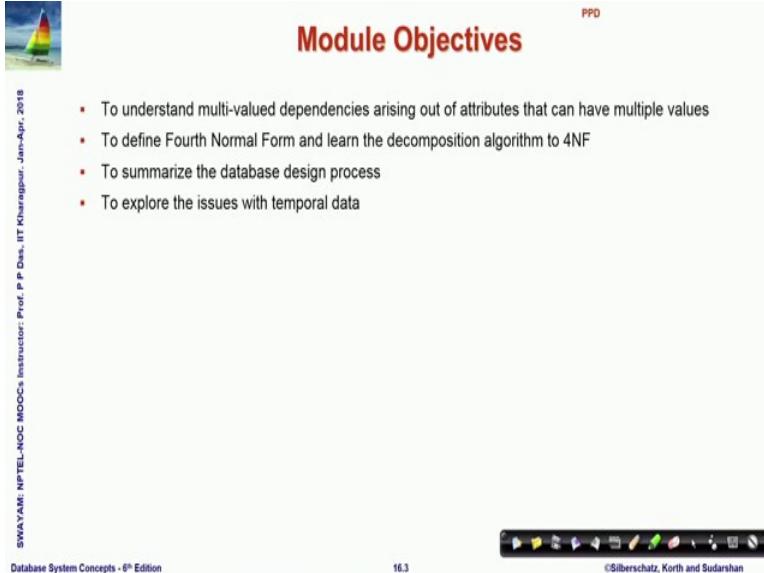
Welcome to module 20 of Database Management Systems. We have been discussing about relational database design, since the last 4 modules and this will be the concluding part of relational database design.

(Refer Slide Time: 00:33)

The slide is titled "Module Recap" in red at the top right. It features a small sailboat icon in the top left corner. On the far left, there is vertical text: "SWAYAM: NPTEL-NOCO's MOOCs", "Instructor: Prof. P. P. Das", "Module: Database System Concepts - 8<sup>th</sup> Edition", and "Date: Jan-Apr., 2018". In the top right corner, it says "PPD". The main content area contains a bulleted list: "■ Normal Forms", "■ Decomposition to 3NF", and "■ Decomposition to BCNF". At the bottom, there is a navigation bar with icons for back, forward, search, and other presentation controls. The page number "16.2" is at the bottom center, and the copyright notice "©Silberschatz, Korth and Sudarshan" is 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)



This slide is titled "Module Objectives" in red at the top right. It features a small sailboat icon in the top left corner. The background is white with a light gray sidebar on the left containing text. A decorative footer bar with various icons is at the bottom.

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 MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018

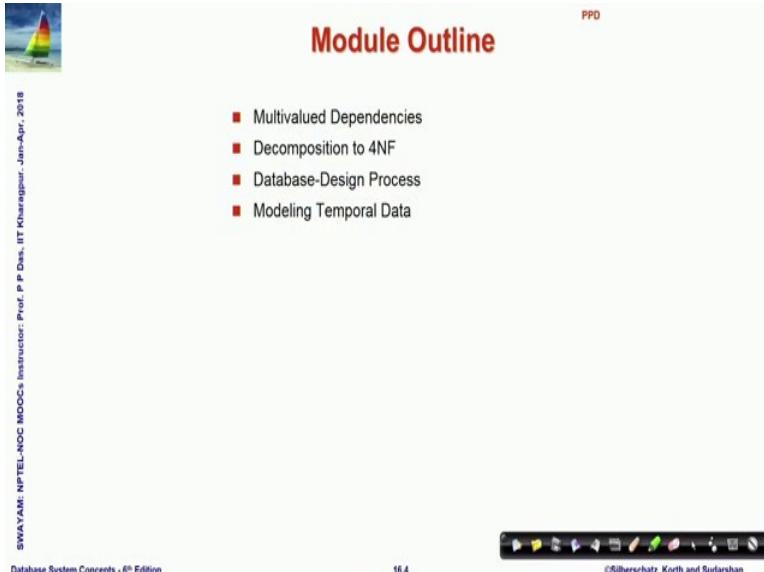
Database System Concepts - 8<sup>th</sup> Edition

16.3

©Silberschatz, Korth and Sudarshan

In view of that in the background of that, in this module we would 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 can take 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 4 NF and then we will summarize this whole set of discussions of relational database design, and talk little bit about what happens, when you have temporal data in your system.

(Refer Slide Time: 01:39)



This slide is titled "Module Outline" in red at the top right. It features a small sailboat icon in the top left corner. The background is white with a light gray sidebar on the left containing text. A decorative footer bar with various icons is at the bottom.

PPD

## Module Outline

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

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

Database System Concepts - 8<sup>th</sup> Edition

16.4

©Silberschatz, Korth and Sudarshan

(Refer Slide Time: 01:47)

**Multivalued Dependency**

PPD

■ 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  $\rightarrow\!\!\!\rightarrow$  Phones
- Man  $\rightarrow\!\!\!\rightarrow$  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 - 8<sup>th</sup> 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 three 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 one, two, three 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** →→ **phones**, **man** →→ **dog likes**. So, there are two 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	S1	A	C2	B
S2	B	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 - 8<sup>th</sup> Edition

16.7

©Silberschatz, Korth and Sudarshan

So, let us move on. So, this is just another example, we have two 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 two 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** →→ **CID** and **SID** →→ **Cname**. So, these are the two different multiple values that you can, find against the SID and this is ah. So, if two 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 MOOCs Instructor: Prof. P. P. Deshpande, IIT Kanpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition

16.8

©Silberschatz, Korth and Sudarshan

This is another example of two 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 three 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)**

PPD

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

$$\begin{aligned} t_1[\alpha] &= t_2[\alpha] = t_3[\alpha] = t_4[\alpha] \\ t_3[\beta] &= t_1[\beta] \\ t_3[R - \beta] &= t_2[R - \beta] \\ t_4[\beta] &= t_2[\beta] \\ t_4[R - \beta] &= t_1[R - \beta] \end{aligned}$$

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

- course  $\rightarrow\rightarrow$  book
- course  $\rightarrow\rightarrow$  lecturer

Test: course  $\rightarrow\rightarrow$  book

	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  $\alpha \rightarrow\rightarrow \beta$ , 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 two more tuples  $t_3$  and  $t_4$  such that this condition sold, 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  $\rightarrow\rightarrow$  book**. So, these are the two, four 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  $- (\beta - \alpha)$  ok**.

So, the first condition that all these tuples will have to match on  $\alpha$  yes, they do, all four 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  $t_3$  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  $\rightarrow\!\!\!\rightarrow$  book**, then it is trivial that course will also multi determine lecturer.

(Refer Slide Time: 10:36)


**Example**

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

- Let  $R$  be a relation schema with a set of attributes that are partitioned into 3 nonempty subsets.  $Y, Z, W$
- We say that  $Y \rightarrow\!\!\!\rightarrow Z$  ( $Y$  **multidetermines**  $Z$ ) if and only if for all possible relations  $r(R)$ 

$$\langle y_1, z_1, w_1 \rangle \in r \text{ and } \langle y_1, z_2, w_1 \rangle \in r$$

then

$$\langle y_1, z_1, w_2 \rangle \in r \text{ and } \langle y_1, z_2, w_2 \rangle \in r$$
- Note that since the behavior of  $Z$  and  $W$  are identical it follows that  
 $Y \rightarrow\!\!\!\rightarrow Z$  if  $Y \rightarrow\!\!\!\rightarrow W$





Database System Concepts - 6<sup>th</sup> 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,  $Y \rightarrow\!\!\!\rightarrow Z$ , 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 with 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  $Y \rightarrow\!\!\! \rightarrow Z$ , then you can you have ZY 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:  
 $ID \rightarrow\!\!\! \rightarrow child\_name$   
 $ID \rightarrow\!\!\! \rightarrow phone\_number$
- 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  $Y \rightarrow\!\!\! \rightarrow Z$
  - Indeed we have (in above notation)  $Z_1 = Z_2$   
The claim follows.

SWATAM: NPTEL/NOC INDOCS Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018  
Database System Concepts - 8<sup>th</sup> Edition  
16.11 ©Silberschatz, Korth and Sudarshan

So, this is in terms of our example, you can now clearly understand that  $ID \rightarrow\!\!\! \rightarrow child\_name$ ,  $ID \rightarrow\!\!\! \rightarrow phone\_number$ , in the earlier example that we took and ah. So, we can also note that if there is a functional dependency,  $Y \rightarrow Z$  then; obviously,  $Y \rightarrow\!\!\! \rightarrow Z$ , that is that is just quite obvious.

(Refer Slide Time: 12:06)

The slide has a header 'Use of Multivalued Dependencies' with a sailboat icon. On the left, there is a vertical sidebar with the text 'SWAYAM: NPTEL-NCOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018'. The main content area contains a bulleted list:

- We use multivalued dependencies in two ways:
  1. To test relations to **determine** whether they are legal under a given set of functional and multivalued dependencies
  2. 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  $r$ .

At the bottom, there is a navigation bar with icons and the text 'Database System Concepts - 8<sup>th</sup> Edition', '16.12', and '©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)

The slide has a header 'Theory of MVDs' with a sailboat icon. On the left, there is a vertical sidebar with the text 'SWAYAM: NPTEL-NCOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018'. The main content area contains a table of MVD rules:

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

Below the table, there is a bulleted list:

- A MVD  $X \rightarrow\rightarrow Y$  in  $R$  is called a trivial MVD if
  - $Y$  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.

At the bottom, there is a navigation bar with icons and the text 'Source: http://www.edugrabs.com/multivalued-dependency-mvd/', 'Database System Concepts - 8<sup>th</sup> Edition', '16.13', and '©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 ah 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 three 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  $X \rightarrow\!\!\!\rightarrow Y$ , then  $X \rightarrow\!\!\!\rightarrow R - (X \cup Y) \rightarrow\!\!\!\rightarrow$  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  $X \rightarrow\!\!\!\rightarrow Y$  and  $Y \rightarrow\!\!\!\rightarrow Z$  then,  $X \rightarrow\!\!\!\rightarrow Z - Y$ .

So, these are the these are the 3 rules which are basically these three are rules that, involve only multi valued dependencies and the other two 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,  $X \rightarrow\!\!\!\rightarrow Y$  we call that MVD to be trivial. If either  $Y \subseteq X$  which is the notion we used for functional dependencies or there is a second condition here, that the  $X \cup Y$  that left hand right hand side gives you the whole set of attributes, otherwise it is a non-trivial multivalued dependency and we have to repeat the values. So, these are the two 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)

The slide has a title 'Theory of MVDs' in red at the top right. On the left is a small sailboat icon. The main content is a bulleted list:

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

At the bottom left is vertical text: SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018. At the bottom center: Database System Concepts - 8<sup>th</sup> Edition, 16.14. At the bottom right: ©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 say dependencies, which means that there could be functional dependencies, as well as multivalued dependencies. Given 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 an 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)

The slide has a header 'Fourth Normal Form' in red. On the left is a small sailboat icon. The main content is a bulleted list:

- 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  $\alpha \rightarrow\!\!\rightarrow \beta$ , where  $\alpha \subseteq R$  and  $\beta \subseteq R$ , at least one of the following hold:
  - $\alpha \rightarrow\!\!\rightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$  or  $\alpha \cup \beta = R$ )
  - $\alpha$  is a superkey for schema  $R$
- If a relation is in 4NF it is in BCNF

At the bottom left is a video thumbnail showing a man speaking, with the text 'SWAYAM-NPTEL NOC Instructor: Prof. P.P. Desai, IIT Kanpur- Jan-Apr- 2018'. At the bottom right are navigation icons and the text 'Database System Concepts - 8<sup>th</sup> 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 NF if, every multi valued dependency  $\alpha \rightarrow\!\!\rightarrow \beta$ , 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 a 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)

The slide features a sailboat icon in the top left corner. The title 'Restriction of Multivalued Dependencies' is centered at the top in red. On the left, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs', 'Instructor: Prof. P. P. Deshpande', 'IIT Kharagpur - Jan-Apr - 2018'. The main content is a bulleted list:

- 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  $D^+$

At the bottom left is a video thumbnail showing a man speaking. The bottom right shows a navigation bar with icons and the text 'Database System Concepts - 8<sup>th</sup> Edition', '16.17', and '©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)

The slide features a sailboat icon in the top left corner. The title '4NF Decomposition Algorithm' is centered at the top in red. On the left, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs', 'Instructor: Prof. P. P. Deshpande', 'IIT Kharagpur - Jan-Apr - 2018'. The main content is a numbered list:

- For all dependencies  $A \twoheadrightarrow B$  in  $D^+$ , check if  $A$  is a superkey
  - By using attribute closure
- If not, then
  - Choose a dependency in  $F^+$  that breaks the 4NF rules, say  $A \twoheadrightarrow B$
  - Create  $R_1 = A B$
  - 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
- 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  $D_{2+}$

At the bottom left is a video thumbnail showing a man speaking. The bottom right shows a navigation bar with icons and the text 'Database System Concepts - 8<sup>th</sup> Edition', '16.18', and '©Silberschatz, Korth and Sudarshan'.

So, the decomposition algorithm into 4 NF is exactly like the decomposition algorithm of the Boyce Codd Normal form BCNF. Only difference being that, now you may be doing this crucial step of 2A 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 4 NF 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)

The slide features a small sailboat icon in the top left corner. The title '4NF Decomposition Algorithm' is centered at the top in red. On the left edge, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr- 2018'. The main content is a pseudocode algorithm:

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

At the bottom, there is footer text: 'Database System Concepts - 8<sup>th</sup> Edition', '16.19', and '©Silberschatz, Korth and Sudarshan'. A standard presentation toolbar is visible at the very bottom.

(Refer Slide Time: 20:13)

**Example of 4NF Decomposition**

**Example:**

- Person\_Modify(Man(M), Phones(P), Dog\_Likes(D), Address(A))
- FDs:**
  - FD1 : Man  $\rightarrow\!\!\rightarrow$  Phones
  - FD2 : Man  $\rightarrow\!\!\rightarrow$  Dogs\_Likes
  - FD3 : Man  $\rightarrow$  Address
- Key = MPD
- All dependencies violate 4NF

**Post Normalization**

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.)

Source: Prof. P. P. Desai, IIT Kanpur - Jan-Apr-2018  
http://www.edugrabs.com/4nf-normal-form/

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.

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 two 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 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 FD 1; **man U phones** that gives you the first relation and then the rest of it. Then again you split based on FD 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 three 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, FD1 in this has become a relation where, the multivalued dependency **man  $\rightarrow\!\!\rightarrow$  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**

- $R = (A, B, C, G, H, I)$
- $F = \{ A \rightarrow\!\!> B$
- $B \rightarrow\!\!> HI$
- $CG \rightarrow\!\!> H \}$
- $R$  is not in 4NF since  $A \rightarrow\!\!> B$  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  $R_6$ )
    - $A \rightarrow\!\!> B$  and  $B \rightarrow\!\!> HI \rightarrow A \rightarrow\!\!> HI$  (MVD transitivity), and
    - and hence  $A \rightarrow\!\!> I$  (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-NOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018  
Database System Concepts - 8<sup>th</sup> 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 three multivalued dependencies and you can see that,  $A \rightarrow\!\!> B$  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 decompose  $R_4$  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)

The slide has a title 'Design Goals' in red at the top right. On the left is a small image of a sailboat on water. The main content is a bulleted list of goals for relational database design:

- 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

At the bottom left is vertical text: 'SWAYAM: NPTEL-NOCOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr. 2018'. At the bottom center: 'Database System Concepts - 8<sup>th</sup> Edition' and '16.23'. At the bottom right: '©Silberschatz, Korth and Sudarshan' and a set of icons.

So, again to remind you the goal for our design is to have a relational database which is in BCNF or 3 NF has a lossless join due to the decomposition, and dependency preservation. If we cannot achieve that, I am sorry earlier what I meant is BCNF or 4 NF not BCNF and 3 NF. So, the idea would be I have a decomposition in BCNF 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 join 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 can be specified or checked in SQL. You can do that using assertions, 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 BCNF or 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 has a header 'Further Normal Forms' in red. On the left is a small logo of a sailboat on water. The main content is a bulleted list:

- 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

Small text at the bottom left: SWAYAM: NPTEL-NOC MOOCs; Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018  
Database System Concepts - 8<sup>th</sup> Edition

Small text at the bottom right: 16.24 ©Silberschatz, Korth and Sudarshan

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)

The slide has a header 'Overall Database Design Process' with a sailboat icon. On the left, there's a vertical sidebar with text: 'SWAYAM-NPTEL NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018'. The main content area contains a bulleted list:

- 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

At the bottom left is a video thumbnail of a professor, and at the bottom right are navigation icons and the text 'Database System Concepts - 8<sup>th</sup> 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)

The slide has a header 'ER Model and Normalization' with a sailboat icon. The text discusses normalization rules and provides an example of a functional dependency between department\_name and building. It also notes that most relationships are binary. The footer includes the title 'Database System Concepts - 8<sup>th</sup> Edition', the date '2018', and the author's name '©Silberschatz, Korth and Sudarshan'.

SWAYAM-NPTEL NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018

## 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

Database System Concepts - 8<sup>th</sup> 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 to 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)

The slide has a title 'Denormalization for Performance' at the top right. On the left is a small logo of a sailboat on water. The main content is a bulleted list of points:

- 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*  $\bowtie$  *Prerequisite*
  - Benefits and drawbacks same as above, except no extra coding work for programmer and avoids possible errors

At the bottom left is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018'. At the bottom center: 'Database System Concepts - 8<sup>th</sup> Edition' and '16.27'. At the bottom right: '©Silberschatz, Korth and Sudarshan'.

It should also be kept in your view, that at the 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 denormalized 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 if 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 join in 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 denormalization 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 denormalized design in your database.

(Refer Slide Time: 31:59)

The slide has a header 'Other Design Issues' with a sailboat icon. The main content lists issues with earnings tables and company\_year tables. A footer includes course details and copyright information.

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kanpur - Jan-Apr- 2018

## 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 - 8<sup>th</sup> 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 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  $r(R)$ .

SWAYAM: NPTEL-NOC MOOCs; Instructor: Prof. P P Das, IIT Kanpur - Jan-Apr - 2018

Database System Concepts - 8<sup>th</sup> 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 a 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 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 change as well. So, these are the things that you will have to take care ah.

(Refer Slide Time: 35:30)

The slide has a title 'Modeling Temporal Data (Cont.)' in red. To the left is a small image of a sailboat on water. On the right is a video feed of a man with glasses and a blue shirt. The video feed has a vertical caption 'SVAYAM: NPTEL/NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr. 2018'. At the bottom left is the text 'Database System Concepts - 8th Edition'. At the bottom right is the text '©Silberschatz, Korth and Sudarshan' and a set of icons. The slide content is a bulleted list:

- 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

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)

The slide is titled "Module Summary" in red at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list of learning objectives:

- 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

On the left margin, vertical text reads: "SWAYAM-NPTEL MOOCs Instructor: Prof. P.P. Desai, IIT Kharagpur - Jan-Apr. 2018". At the bottom left, it says "Database System Concepts - 8<sup>th</sup> Edition". In the bottom center, it shows "16.32". At the bottom right, it says "©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 four modules, this is the fifth 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.

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

**Lecture – 21**  
**Application Design and Development**

Welcome to module 21 of Database Management Systems in this module and the next too we will discuss about application design and development.

(Refer Slide Time: 00:31)

PPD

### Week 04 Recap

<ul style="list-style-type: none"><li>▪ <b>Module 16: Relational Database Design/1</b><ul style="list-style-type: none"><li>◦ Features of Good Relational Design</li><li>◦ Atomic Domains and First Normal Form</li><li>◦ Functional Dependencies</li></ul></li><li>▪ <b>Module 17: Relational Database Design/2</b><ul style="list-style-type: none"><li>◦ Decomposition Using Functional Dependencies</li><li>◦ Functional Dependency Theory</li></ul></li><li>▪ <b>Module 18: Relational Database Design/3</b><ul style="list-style-type: none"><li>◦ Algorithms for Functional Dependencies</li><li>◦ Lossless Join Decomposition</li><li>◦ Dependency Preservation</li></ul></li></ul>	<ul style="list-style-type: none"><li>▪ <b>Module 19: Relational Database Design/4</b><ul style="list-style-type: none"><li>◦ Normal Forms</li><li>◦ Decomposition to 3NF</li><li>◦ Decomposition to BCNF</li></ul></li><li>▪ <b>Module 20: Relational Database Design/5</b><ul style="list-style-type: none"><li>◦ Multivalued Dependencies</li><li>◦ Decomposition to 4NF</li><li>◦ Database-Design Process</li><li>◦ Modeling Temporal Data</li></ul></li></ul>
---	--

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

Database System Concepts - 8<sup>th</sup> Edition      21.2      ©Silberschatz, Korth and Sudarshan

In the last week we have for the whole week in the five modules we have discussed about relational database design; in depth we have looked into what are the different aspects of that.

(Refer Slide Time: 00:43)

The slide is titled "Module Objectives" in red bold text at the top center. In the top right corner, there is a small "PPD" logo. On the left side, there is a small image of a sailboat on water. At the bottom left, it says "SWAYAM: NPTEL-MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur, Date: Apr. 2018". At the bottom right, it says "©Silberschatz, Korth and Sudarshan". The main content is a bulleted list of objectives:

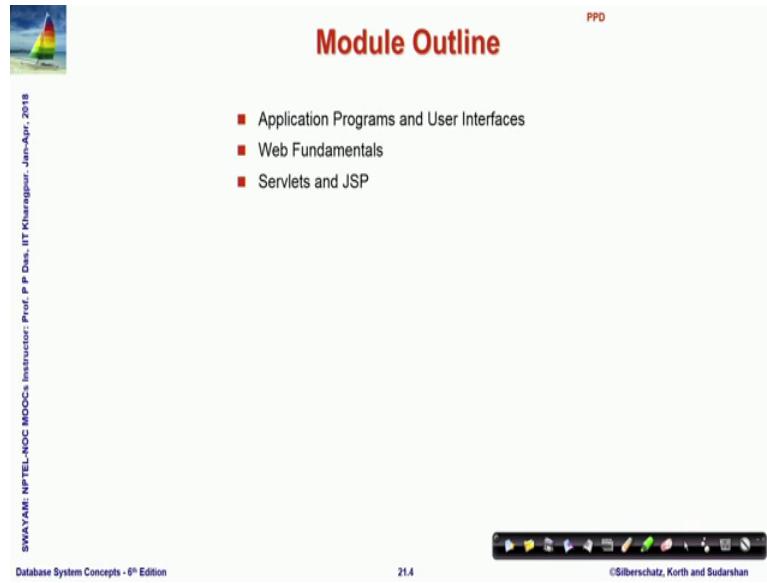
- To understand the requirements of Database Application Programs and User Interfaces
- To familiarize with the fundamental notions and technologies of Web
- To learn about Servlets and Java Server Pages

At the very bottom, there is a decorative footer bar with various icons.

And now we get into the core issue of if I have a relational database design existing and that is populated with the data then based on that how do we develop, how do we create an application where the user can interact and actually get answers to the questions that the user has or the user can actually update the data create new data, remove old data and so, on.

So, we would in that process like to familiarize with the fundamental notions of notions and technologies of web applications and specifically we would learned about servlets and Java server pages.

(Refer Slide Time: 01:30)



The slide is titled "Module Outline" in red at the top right. In the top left corner, there is a small image of a sailboat on water. The footer contains the text "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Desai, IIT Kharagpur - Jan-Apr. 2018", "PPD" in the top right corner, and a navigation bar with icons at the bottom right.

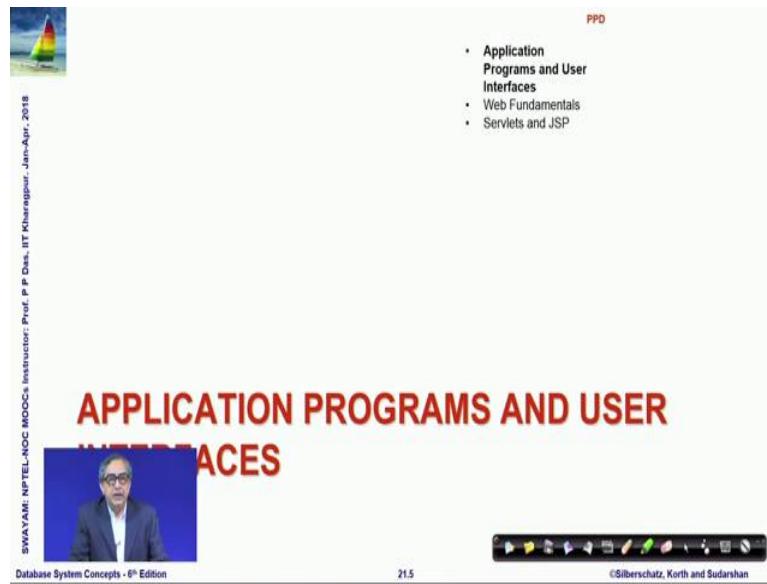
**Module Outline**

- Application Programs and User Interfaces
- Web Fundamentals
- Servlets and JSP

Database System Concepts - 8<sup>th</sup> Edition      21.4      ©Silberschatz, Korth and Sudarshan

So, these are the free topics to be covered.

(Refer Slide Time: 01:34)



The slide features a large title "APPLICATION PROGRAMS AND USER INTERFACES" in red at the top center. Below the title is a video frame showing a man speaking. The footer contains the text "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Desai, IIT Kharagpur - Jan-Apr. 2018", "PPD" in the top right corner, and a navigation bar with icons at the bottom right.

**APPLICATION PROGRAMS AND USER INTERFACES**

- Application Programs and User Interfaces
- Web Fundamentals
- Servlets and JSP

Database System Concepts - 8<sup>th</sup> Edition      21.5      ©Silberschatz, Korth and Sudarshan

So, we first start with application programs and user interfaces.

(Refer Slide Time: 01:41)

**Application Programs and User Interfaces**

■ Most database users do not use a query language like SQL  
■ An application program acts as the intermediary between users and the database  
■ Applications split into  
    ‣ frontend  
    ‣ middle layer  
    ‣ backend  
■ Frontend or Presentation Layer: user interface  
    ‣ Forms, Graphical user interfaces  
    ‣ Many interfaces are Web-based or Mobile App  
■ Middle Layer or Application / Business Logic Layer  
    ‣ Functionality of the Application – links front and backend  
■ Backend or Data Access Layer  
    ‣ Persistent data, large in volume, needs efficient access

**Overview of a 3-tier Architecture**

The diagram illustrates a 3-tier architecture with three layers: Presentation tier, Logic tier, and Data tier.

- Presentation tier:** The top-most level of the application is the user interface. The main function of this tier is to transfer data directly to something the user can understand.
- Logic tier:** This tier contains the application processing components, handle logical decisions and evaluations, and perform calculations. It receives and processes data between the two surrounding layers.
- Data tier:** Here information is stored and retrieved from a database or file system. The application in this tier passes data to the logic tier for processing and then eventually back to the user.

The diagram shows a flow from the user interface (Presentation tier) sending a "GET LIST OF ALL SALES LAST YEAR" query to the logic tier. The logic tier then sends a "SALES 1", "SALES 2", "SALES 3", and "SALES 4" response back to the user interface. The logic tier also interacts with a "Database" and "Storage" component.

So, the situation is the where we do have a relational database design it is populated with the required data, but how about the interaction with the user incidentally most of the you database users do not interact with the database or query the database using language like SQL because as you have seen it is not a very friendly language and it is not presentable in a way which I would we would always expect or we would like.

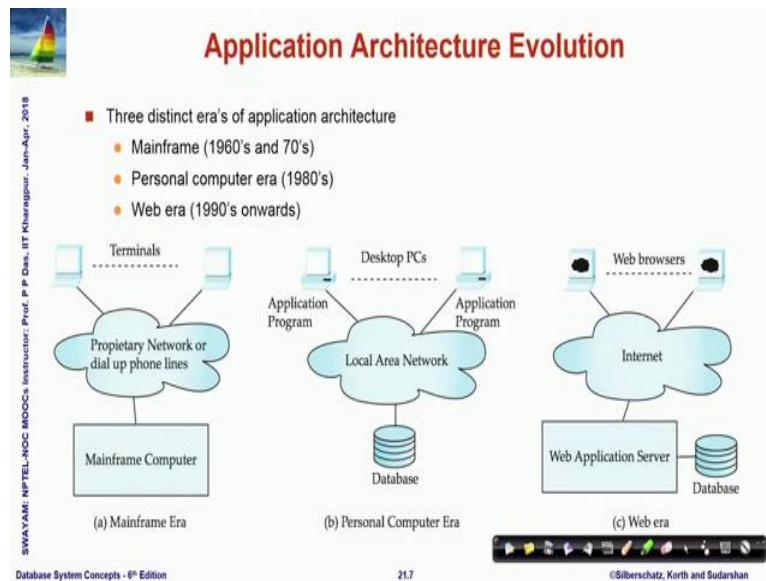
So, usually an application program acts as a as we say as an intermediately between the user and the database. And it is often split into three layers as we will say frontend middle layer and backend. So, on the right I have shown these three layers we will talk more about the this is just a representative diagram. So, the frontend is the user interface it is also called the presentation layer. So, it is the layer where it is the part of the application where there are forms GUIs and different ways to input as well as get output of the data.

Which is directly interacting with the user then we have a middle layer or its also called the business logic layer or the application layer where the functionality of the application the required operations of the or desired behavior of the application are coded and it is kind of acts as a link between the frontend and the backend and what is the backend?.

Backend is an actual data access layer or it is where the persistent data the database that we have created exist it is typically large in volume need sufficient access and so, on. So, in this module we will try to understand as to how such what are the different

requirements and what are the different technologies involved in creating such a layered application.

(Refer Slide Time: 03:55)



Now, if we historically look at; so, here we are just showing three phases initially 60s and 70s where the first database applications started then the interaction used to be takes based from the terminal.

And those to directly connect to the main frame computer where the data existed through either a direct connection dial up phone or proprietary network. And as we move to the 80s; then we saw the advent of local area networks to application programs or desktop would interact to with the database through these local area networks. And beyond that we have the what we call the web era which is 1990 onwards we in the that is that is about roughly the last 30 years where typically the applications are now based on web browsers.

So, the frontend where we actually interact are web browsers and that connect to the web application server the database everything through an internet. And I must tell you at this point that when you say this is the architecture it does not necessarily mean that the cloud shown as internet will have to be the web, it could be an internet which is created with a set of systems within your organization which we typically call as an intranet or couple of organizations across.

Which we say are extranet or it could even be a set of systems which are connected through the internet protocol within your lap or it could even be a single computer in which all these layers are integrated together, but by internet we mean it is a internet protocol and technologies will be used for doing this interaction.

(Refer Slide Time: 05:52)

The slide has a title 'Web Interface' at the top right. On the left, there is a small image of a sailboat on water. The main content area contains two bulleted lists under red square bullet points. The first list discusses web browsers as the de-facto standard user interface to databases, mentioning their ability to enable large numbers of users to access databases from anywhere, avoid the need for specialized code, and run scripts like Javascript and Flash transparently. It also lists examples such as bank, airline, and rental car reservations, university course registration, and grading. The second list discusses mobile interfaces in mobile apps, noting they are getting popular, similar to web architecture but with significant differences, and will be discussed later. At the bottom left is a video thumbnail showing a person speaking. The footer includes the text 'SWAYAM: NPTEL-NOC MOOCs Initiator: Prof. P P Desai, IIT Kharagpur - Jan-Apr - 2018', 'Database System Concepts - 8th Edition', '21.8', and '©Silberschatz, Korth and Sudarshan'.

So, web interface has become the de-facto standard which gives a very distributed access to the database enables large number of users to access together. And it avoids the requirement of downloading or installing specialized code into that all that we need is just a web browser. And we have seen we are living through a variety of applications which are of this kind the banking application, the airline and railway reservations car rental hotel booking or web mail systems we will check mail in Gmail or Yahoo those are all different web interfaces through which we actually access a the required set of databases.

And every even every enterprise operations the ERP are now web based and that is become a de facto standard. So, in the web interface along with a web interface what has been imagined of let of the last about 10 years are mobile interfaces that we are getting use to using such applications from our mobile phone or tablet. And these are similar in architecture and workflow as of the web application, but there are significant differences to and at a later point in the next module, we will discuss about the specific requirements of mobile apps in this context as well.

(Refer Slide Time: 07:27)



PPD

- Application Programs and User Interfaces
- **Web Fundamentals**
- Servlets and JSP

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr., 2018

## WEB FUNDAMENTALS



Database System Concepts - 8<sup>th</sup> Edition

21.9

©Silberschatz, Korth and Sudarshan

So, before we move forward in terms of the details of how to build these applications; we need to familiarize ourselves with the basic notions of the web as such. So, we call them as web fundamentals.

(Refer Slide Time: 07:43)



- The Web is a distributed information system based on hypertext
- Most Web documents are hypertext documents formatted via the HyperText Markup Language (HTML)
- HTML documents contain
  - text along with font specifications, and other formatting instructions
  - hypertext links to other documents, which can be associated with regions of the text
  - forms, enabling users to enter data which can then be sent back to the Web server

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr., 2018



Database System Concepts - 8<sup>th</sup> Edition

21.10

©Silberschatz, Korth and Sudarshan

So, web is a distributed information system which is based on hyper text a hyper text is one where you have a part of the text available to you and then you have links we say our hyper links which can connect to the different documents contents.

Which are located at other places at other servers and typically most web documents are hyper text documents which are formatted in terms of what we know as HTML Hyper Text Markup Language; you will be familiar with that I am sure. So, they contain text and along with that they can have other components like images, video, the text as specifications for font colors style all that. And in addition there are forms which can be used to enter data and send them back to the web server.

(Refer Slide Time: 08:35)

**Uniform Resources Locators**

- In the Web, functionality of pointers is provided by Uniform Resource Locators (URLs).
- URL example:  
<http://www.acm.org/sigmod>
  - The first part indicates how the document is to be accessed (protocol)
    - "http" indicates that the document is to be accessed using the Hyper Text Transfer Protocol.
  - The second part gives the unique name of a machine on the Internet
  - The rest of the URL identifies the document within the machine
- The local identification can be:
  - The path name of a file on the machine: A file at C:/WINDOWS/media/Alarm01.wav of local machine can be accessed as:
    - <file:///C:/WINDOWS/media/Alarm01.wav>
    - <file:///localhost/c:/WINDOWS/media/Alarm01.wav>
- An identifier (path name) of a program, plus arguments to be passed to the program: Searching google.com with 'silberschatz' has the url:
  - <http://www.google.com/search?q=silberschatz>

Now, naturally when we operate on the web we need to have the functionality of pointing to different resources and this is done by what is known as URL or uniform resource locator. So, that is a URL is a procedure to which you can identify and point to a certain specific location of content. So, here I am showing an example of such a URL all of you be familiar with URLs.

But just to look into the different components the first component http : this http is actually a tells us the way the content would be accessed and this is typically called a protocol http its stands for hyper text transfer protocol which allows you different text to be accessed.

The second component in this URL which is between the two forward slash and the next slash www [dot] acm [dot] org identifies uniquely identifies a machine on the internet you will understand this is the symbolic name and the actual machine has what is known as an IP address.

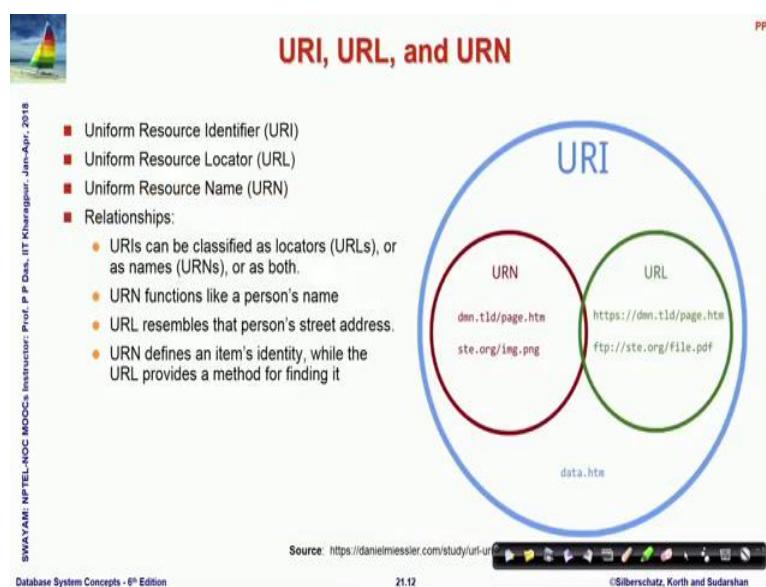
And we will not go into those details, but for us it is enough to understand that www [dot] acm [dot] org here is uniquely relatable to a particular machine on the internet. And the last part which is remaining is unique and if there are more and more parts, then it identifies the document inside within that machine. So, URL can be used in a multiple other ways also.

For example I can use URL locator to specify a file in my machine for example, I have shown an example of an AVI file in my C drive in windows and that is done through a similar URL where the protocol is not http the protocol is file telling me that it is actually residing in my local machines.

So, I have shown two ways to look at that and you will see that between the two forms; if you look at the second form you can easily understand that the machine to be identified is called the local host. And in the first form that part is missing because it is by default the machine where I am running this code and rest of it is same which is basically identifying the document to be located in that machine.

Similarly, this such URL can also in the last example you can see that www [dot] Google [dot] com is the basic machine where I am putting the URL and then the rest of it is search.

(Refer Slide Time: 11:56)



So, which actually takes it to a document where I tell the search to be performed and then there are parameters to this form the parameter is q is equal to silberschatz which is equivalent to same that I am asking Google [dot] com to search for contents which have silberschatz in it. So, this is the basic purpose of the uniform resource locator or URL incidentally you may have heard the names like URI and URN in addition to URL.

So, they are related, but they mean little bit different things as this Venn diagram shows. So, a URI can be either a URL or a URN or it could be both. So, URN functions like a person's name; so, you can conceive it that way universal resource name and URL resembles that of a person's street address. So, URN says what is the name of the content and URL says where that can be found. And URI in general could be either the name or the address or both of them.

In this context of our discussion we will continue to use the term URL only, but I just wanted you to be aware of the other two terms in case you come across them in the text.

(Refer Slide Time: 12:53)

**HTML and HTTP**

- HTML provides formatting, hypertext link, and image display features
  - including tables, stylesheets (to alter default formatting), etc.
- HTML also provides input features
  - Select from a set of options
    - ▶ Pop-up menus, radio buttons, check lists
  - Enter values
    - ▶ Text boxes
  - Filled in input sent back to the server, to be acted upon by an executable at the server
- HyperText Transfer Protocol (HTTP) used for communication with the Web server

SWAYAM-NIITEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018

Database System Concepts - 8<sup>th</sup> Edition

21.13

©Silberschatz, Korth and Sudarshan

So, we know by now that http HTML provides the formatting the hyper text links images and so, on and http provides the protocol through which the contents are exchanged between different machines in the internet. So, you can select from a set of options in terms of a HTML pop-up menus, radio buttons, check boxes and so, on; you can enter values to text box and once a form has been filled up that form will be sent back to the

server from where it came and would be acted upon by the server http helps in that transfer mechanism.

(Refer Slide Time: 13:39)

The slide title is "Sample HTML Source Text". It contains the following HTML code:

```
<html>
<body>
<table border>
<tr> <th>ID</th> <th>Name</th> <th>Department</th> </tr>
<tr> <td>00128</td> <td>Zhang</td> <td>Comp. Sci.</td> </tr>
...
</table>
<form action="PersonQuery" method="get">
Search for:
<select name="persontype">
<option value="student" selected>Student </option>
<option value="instructor"> Instructor </option>
</select> <br>
Name: <input type="text" size=20 name="name">
<input type="submit" value="submit">
</form>
</body> </html>
```

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

Database System Concepts - 8<sup>th</sup> Edition 21.14 ©Silberschatz, Korth and Sudarshan

So, the here is a sample HTML code let me show you the effect of this in the next slide.

(Refer Slide Time: 13:46)

The slide title is "Display of Sample HTML Source". It shows a rendered table and the corresponding HTML source code. The table data is:

ID	Name	Department
00128	Zhang	Comp. Sci.
12345	Shankar	Comp. Sci.
19991	Brandt	History

The rendered HTML source code is identical to the one shown in the previous slide. A video player window at the bottom left shows a person speaking.

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

Database System Concepts - 8<sup>th</sup> Edition 21.15 ©Silberschatz, Korth and Sudarshan

So, if you look in here then you can see that these are this is what is known as a tag `<>` and `</>` it kind of the tags are kind of given in the form of `( )` notation.

So, it has a opening and a closing and these have to have to actually match and you can see that the opening is written within corner brackets and the closing is write the same tag name, but you put a forward slash before the name. And then between the closing opening and the closing you can write more tabs in a nested manner.

So, here it says that I have HTML which has a body and the body expanse this much and then we are saying that there is a table. So, this is the table that you can get to see and then it also says that there is a form and this is the form that you get to see within the table you can see. So, it is saying that this is an ID there is a name; so, it is describing the first row the department. So, you can see each one of them the ID is here the name is here the department is here similarly this is a next row where it is saying it is 0 1; 00128 Zhang Computer Science.

So, this is how you can you actually in an HTML in a text form all these details will be given and when it is rendered by the web browser then you will see a table like this. Similarly here we are I am showing a an instance of a form which is use to input data. So, we are saying that here is a drop down and it is written out here in terms of options.

So, the first options student is visible here if you drop down you will actually see another option instructor here you will not see that because it is a frozen image. So, and then I have a qualifier name and there is a input text box where you can input any strain up to size 20. And once this has been done then you have an input which is submit, which is the submit button here which shows that you can now submit and then this form filled up form will be sent back to the web browser from where it originally came.

(Refer Slide Time: 16:23)

The slide has a header 'Web Servers' in red. On the left is a small sailboat icon. The main content is a bulleted list:

- A Web server can easily serve as a front end to a variety of information services
- The document name in a URL may identify an executable program, that, when run, generates a HTML document
  - When an HTTP server receives a request for such a document, it executes the program, and sends back the HTML document that is generated
  - The Web client can pass extra arguments with the name of the document
- To install a new service on the Web, one simply needs to create and install an executable that provides that service
  - The Web browser provides a graphical user interface to the information service
- Common Gateway Interface (CGI): a standard interface between web and application server

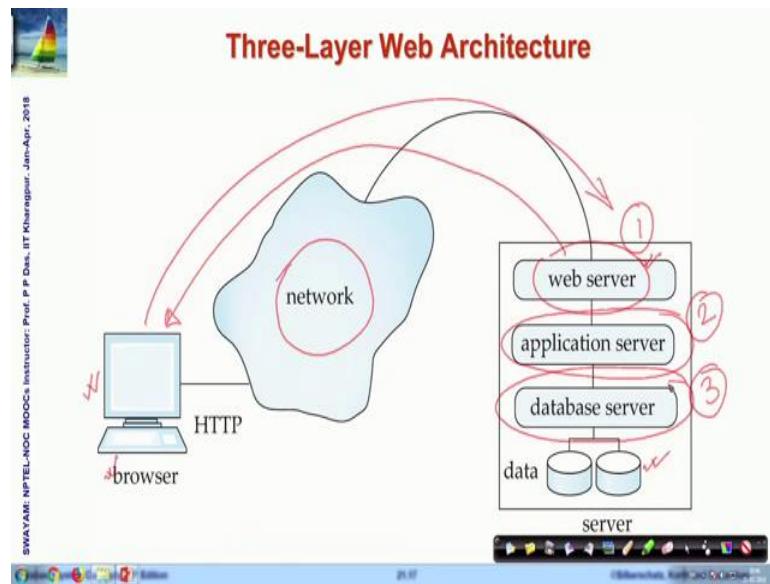
At the bottom left is a video thumbnail of a professor. The footer includes 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kanpur', 'Database System Concepts - 8<sup>th</sup> Edition', '21.16', and '©Silberschatz, Korth and Sudarshan'.

So, this is the basic mechanism of HTML you can learn little bit more about that and get familiar with it. So, a document name in a URL may identify a program that is written that generate. So, HTML could be either at the URL in the web server you can either have a HTML which we say is a static HTML or you could actually have a program which when you send the request it actually.

For example, when you are doing Google you said [www.google.com/search?q=silberschatz](http://www.google.com/search?q=silberschatz). Then actually at that location there is no HTML currently existing which contains the search result, but instead there is a program which will be executed based on the submission of this form and when run that will generate a HTML document. And once that is generated then this will be passed back to the to your web browser. So, that is a basic mechanism.

So, if you want a new service on the web then you all that you simply need to do is to create and install a new program that will provide that service and through this process we will see how easily this can be done and how web browser provides a graphical interface to this information service. There is there has been another other mechanisms of doing similar things also which was particularly popularly are called the common gateway interface or CGI, but now we have various other ways of doing the same thing.

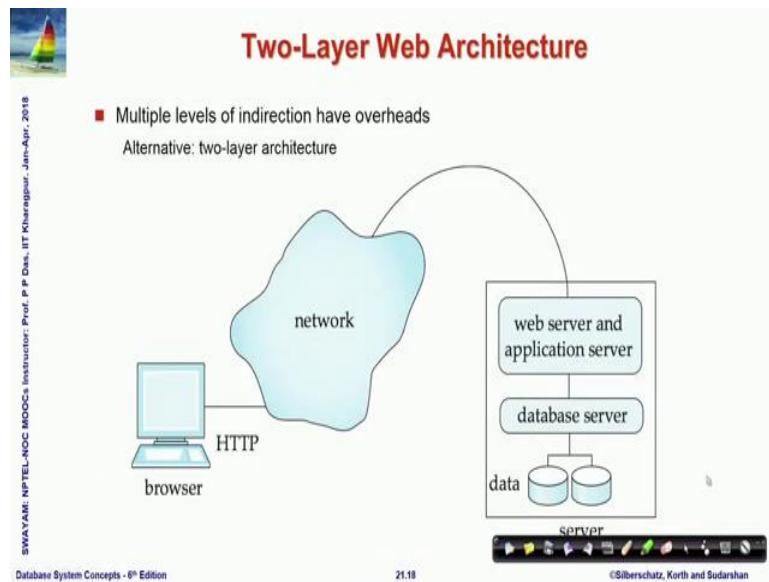
(Refer Slide Time: 18:03)



So, in this context then the basic three layers that we started discussing with are here. So, this is the most common way an application is. So, this is where your frontend is where you have the browser where you see that this is the network; we are just in general writing network it could be internet it could be intranet and a web server. So, when you send a request this is received by the server; web server somehow computes a result HTML and that send back to the browser and that is how the interaction keeps on happening.

So, the browser and the web server together often would be referred to as a frontend because that gives a presentation that presents the results the interaction to you. The next is the application layer which is the business logic where you write and then you have the data access layer or the database server and these are the actual disk where the data exist. So, this is a tier 1 this is tier 2 and this is tier 3 which is a very typical way a web application will be architected and these are the three layers or three tiers that we will usually find.

(Refer Slide Time: 19:22)



So, often actually three is not a very magical number in terms of tiers; it is possible that you could have more some applications have more tiers. And some applications may choose to have multiple functionality in the same layer for example, web server and application server functionality could be clubbed together and when this is done we say that we will then we have only two layers. So, the frontend and the middle layer are merge together and the backend or the database server becomes a second layer.

So, we would often might want to do that the reason we do that I will just take to back to the three layer view. So, if the question is naturally if we have the web server and we have these connections this is clear; now the question is what is this connection and what is this connection? Is it necessary that they will have to be on the same server physically or will they be on can be on different server and servers could be connected through a LAN or they themselves could be on different servers over the internet and may they may be connected through internet.

So, all of these are possibilities and the way we connect is the way we will write the application will be will not depend on the way these servers are connected between each other we will often assume that as if they are connected over a net and write it in a way so, that even when they may be connected over a LAN or even when they may actually be on the same machine things will work in the same way.

(Refer Slide Time: 20:57)

The slide has a title 'HTTP and Sessions' at the top right. On the left, there is a small logo of a sailboat on water. The main content area contains a bulleted list under several sections:

- The HTTP protocol is **connectionless**
  - That is, once the server replies to a request, the server closes the connection with the client, and forgets all about the request
  - In contrast, Unix logins, and JDBC/ODBC connections stay connected until the client disconnects
    - retaining user authentication and other information
  - Motivation: reduces load on server
    - operating systems have tight limits on number of open connections on a machine
- Information services need session information
  - E.g., user authentication should be done only once per session
- Solution: use a **cookie**

At the bottom left, it says 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr- 2018'. At the bottom center, it says 'Database System Concepts - 8<sup>th</sup> Edition' and '21.19'. At the bottom right, it says '©Silberschatz, Korth and Sudarshan'.

Now, one point that should be born in mind in terms of the http protocol is it is connectionless. So, this is a very very critical concept and it means that once I start the process I have an URL; I submit that and that goes to the server the server runs an application or it is a static page server picks up and returns me that HTML; the http loop is closed.

So, that is the all that happens and when I submit again something based on that. So, I have gone to Gmail **mail.google.com** I submit that and I get back a form which tells me to put my user ID and login I do that I submit and that when that goes then there is no memory about the earlier interaction that has happen.

So, when my login submission goes it is authenticated in the backend I am able to login and I am given back the first screen of my mail box which is the inbox screen. And as soon as I get that inbox the HTML containing the inbox on my browser that transaction has also been over. So, if I now want to look at a specific mail it has to be a new query and it is not remember anything from the previous query.

So, this connectionless property naturally makes it makes certain things more difficult; you will you will realize that many of the other connections that we do for example, if we login to UNIX system or to a window system if we use some database connections they are connected till the we disconnect them, but in http it is not. So, it is connectionless every time you do you have a separate session. So, naturally the question

this was I mean the there are there are reasons of why this is done this way this is to reduce load from the server and so, on.

But naturally the consequences therefore, we cannot remember information from one request response loop to the next. So, if I have logged in to my mail Gmail account and I have seen the I have got the inbox then when I want to check my first mail the system does not know any more that I am logged in because that session request response has is over and now I am making a new request that show me the first mail and I expect to see the whole body.

So, there is no information that is carried from one request to the other which makes http difficult to work with. So, the solution for that is something which is known as a cookie.

(Refer Slide Time: 23:46)

The slide has a title 'Sessions and Cookies' in red at the top right. On the left is a small image of a sailboat on water. The main content is a bulleted list under a blue header:

- A **cookie** is a small piece of text containing identifying information
  - Sent by server to browser
    - ↳ Sent on first interaction, to identify session
    - Sent by browser to the server that created the cookie on further interactions
      - ↳ part of the HTTP protocol
    - Server saves information about cookies it issued, and can use it when serving a request
      - ↳ E.g., authentication information, and user preferences
  - Cookies can be stored permanently or for a limited time

At the bottom left is a video thumbnail of a man speaking. The footer includes text: 'SWAYAM: NPTEL-NOC/NOC/OOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '21.20', and '©Silberschatz, Korth and Sudarshan'.

So, a cookie is a small piece of text which contain information which is identifying and which can go back and forth between the browser and the server. So, first it is sent by the sever to the browser and the what the browser does.

So, this happens the first time. So, when we logged in my browser has got a got some cookie from the mail Gmail server. Then the browser can send it back to the server when it is doing the next request so, that I can be identified as a logged in person and. So, the browser can keep it as a I mean locally in its memory or locally here and that is a part of the http protocol.

So, this keeps on this cookie keeps on going back and forth back and forth. So, every time I send a request the cookie actually has to go to tell the server that yes this is the Partha Pratim Das who is already logged in an authenticated himself for checking his mails. So, cookies are a big convenience and they are very important factor of the web applications. So, they can be stored permanently or for a limited period of time.

(Refer Slide Time: 25:04)

The slide has a title 'Servlets' in red at the top right. On the left is a small image of a sailboat. The main content area contains a bulleted list:

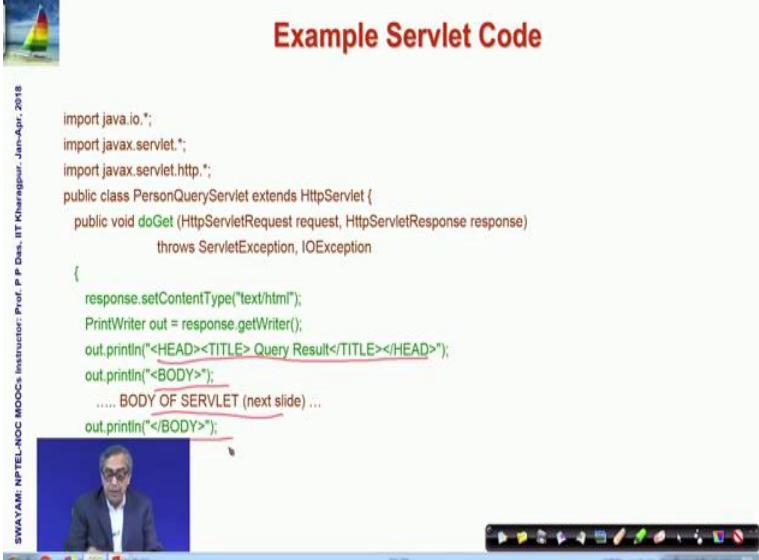
- Java Servlet specification defines an API for communication between the Web/application server and application program running in the server
  - E.g., methods to get parameter values from Web forms, and to send HTML text back to client
- Application program (also called a servlet) is loaded into the server
  - Each request spawns a new thread in the server
    - thread is closed once the request is serviced

At the bottom left is a video frame showing a man speaking, with the text 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr- 2018'. At the bottom center is the text 'Database System Concepts - 8<sup>th</sup> Edition' and '21.22'. At the bottom right is a navigation bar with icons for back, forward, search, etc., and the text '©Silberschatz, Korth and Sudarshan'.

Next let us look into some of the core technologies that are involved the first technology is called a servlet. Servlet is nothing but as you can understand from the name itself is as you have book booklet booklet is a small very small book servlet is a very small server.

So, it is a Java application which can do certain tasks; so, it is an kind of an application program and every time we request then the server actually spawns a new thread and in that thread this servlet would be running and once the request is serviced the thread will be closed.

(Refer Slide Time: 25:42)

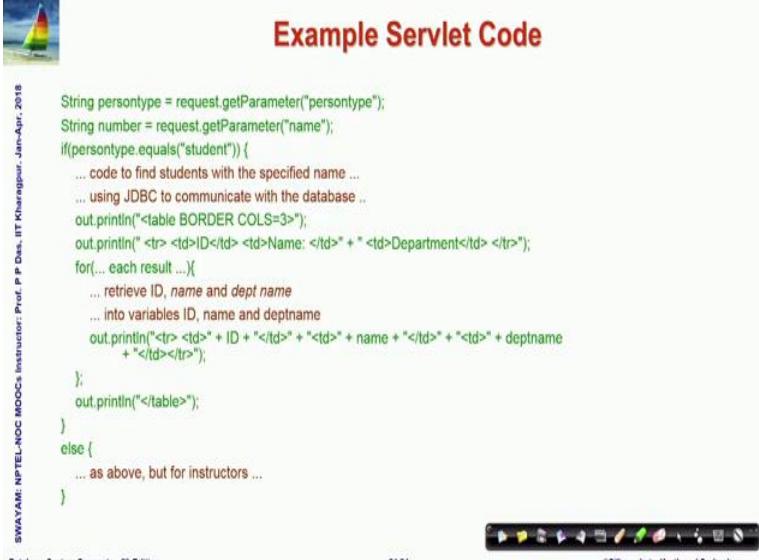


The screenshot shows a Java code editor with the title "Example Servlet Code". The code is a Java class named PersonQueryServlet that extends HttpServlet. It overrides the doGet method to handle HTTP requests. The code prints an HTML response with a title and a body containing placeholder text "..... BODY OF SERVLET (next slide) ...". A watermark on the left side of the code area reads "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018". Below the code editor is a video player window showing a man in a suit speaking. The video player has a progress bar at 21.23 and a copyright notice "©Siberschatz, Korth und Sudarshan".

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PersonQueryServlet extends HttpServlet {
    public void doGet (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HEAD><TITLE>Query Result</TITLE></HEAD>");
        out.println("<BODY>");  
..... BODY OF SERVLET (next slide) ...
        out.println("</BODY>");
```

So, this is the typical server servlet view. So, which shows that in the servlet you are creating actually creating the requested I mean possible HTML response that you would like to have.

(Refer Slide Time: 25:58)



The screenshot shows another Java code editor with the title "Example Servlet Code". This code is similar to the previous one but includes more logic for handling student or instructor queries. It uses JDBC to interact with a database and prints an HTML table with student or instructor details. A watermark on the left side of the code area reads "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018". Below the code editor is a video player window showing a man in a suit speaking. The video player has a progress bar at 21.24 and a copyright notice "©Siberschatz, Korth und Sudarshan".

```
String persontype = request.getParameter("persontype");
String number = request.getParameter("name");
if(persontype.equals("student")){
    ... code to find students with the specified name ...
    ... using JDBC to communicate with the database ...
    out.println("<table BORDER COLS=3>");
    out.println(" <tr> <td>ID</td> <td>Name: </td> <td>Department</td> </tr>");
    for(... each result ...){
        ... retrieve ID, name and dept name
        ... into variables ID, name and deptname
        out.println("<tr> <td>" + ID + "</td>" + <td>" + name + "</td>" + <td>" + deptname
        + "</td></tr>");
    }
    out.println("</table>");
}
else {
    ... as above, but for instructors ...
}
```

So, there are; so, there are different details you can read through that. So, this is the typical servlet code. So, it actually is a Java code which through print line will generate different lines of the HTML. Now naturally servlets maintain session the way we talked about.

(Refer Slide Time: 26:11)



## Servlet Sessions

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018

- Servlet API supports handling of sessions
  - Sets a cookie on first interaction with browser, and uses it to identify session on further interactions
- To check if session is already active:
  - if (`request.getSession(false) == true`)
    - .. then existing session
    - else .. redirect to authentication page
  - authentication page
    - check login/password
    - `request.getSession(true)`: creates new session
- Store/retrieve attribute value pairs for a particular session
  - `session.setAttribute("userid", userid)`
  - `session.getAttribute("userid")`

Database System Concepts - 8<sup>th</sup> Edition 21.25 ©Silberschatz, Korth and Sudarshan



So, that through an interaction I can continued to be identified and the servlet can check whether the session is on or the session is already over. So, these are these are the different ways of doing that in terms of shaking the user ID and several web servers application servers have support for servlet apache tomcat is one of the very popular one.

(Refer Slide Time: 26:33)



## Servlet Support

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018

- Servlets run inside application servers such as
  - Apache Tomcat, Glassfish, JBoss
  - BEA Weblogic, IBM WebSphere and Oracle Application Servers
- Application servers support
  - deployment and monitoring of servlets
  - Java 2 Enterprise Edition (J2EE) platform supporting objects, parallel processing across multiple application servers, etc



Database System Concepts - 8<sup>th</sup> Edition 21.26 ©Silberschatz, Korth and Sudarshan



So, which you must have heard the name of and there are, but there are several other servers as well.

(Refer Slide Time: 26:48)

The slide has a header 'Server-Side Scripting' in red. On the left is a small sailboat icon. The main content lists two bullet points:

- Server-side scripting simplifies the task of connecting a database to the Web
  - Define an HTML document with embedded executable code/SQL queries.
  - Input values from HTML forms can be used directly in the embedded code/SQL queries.
  - When the document is requested, the Web server executes the embedded code/SQL queries to generate the actual HTML document.
- Numerous server-side scripting languages
  - JSP, PHP
  - General purpose scripting languages: VBScript, Perl, Python

At the bottom, there's vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P.P. Desai, IIT Kanpur', the page number '21.27', and the copyright '©Silberschatz, Korth and Sudarshan'.

Now, along with the servlet there is another concept which is called server side scripting. server side scripting is a mechanism where you define an HTML document and to within that HTML document can be used. So, you may have some inputs to that and they can be used to directly fire embedded SQL queries.

So, we talked about a madding of SQL query in while we discuss about the basic mechanism of host language and query language. So, here the HTML kind of a language is a host and you can embed the query right in as a part of that. And so, that query goes to the database query server and you get the answer and that answer is placed where your original query was there. So, that you continue to get very easily a my complete HTML as a response.

So, this kind of a mechanism is makes it very easy because a it is quite easy to conceive of the HTML and fill in. So, if I have asked for say logged in to the my mail Gmail service then I have given the input as my user name password and when that got gets authenticated. Then I get a response which is select mail from different respective tables where my authentication is there the user name is PPD and so, on. So, it becomes quite easy to actually create the HTML and there several such scripting language is JSP and PHP are the most popular ones.

(Refer Slide Time: 28:35)

The slide features a small sailboat icon in the top left corner. The title 'Java Server Pages (JSP)' is centered at the top in a red font. Below the title, there is a list of bullet points and some code snippets.

- A JSP page with embedded Java code

```
<html>
<head> <title> Hello </title> </head>
<body>
<% if (request.getParameter("name") == null)
{ out.println("Hello World"); }
else { out.println("Hello, " + request.getParameter("name")); }
%>
</body>
</html>
```

- JSP is compiled into Java + Servlets

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur Date: Jan-Apr- 2018

Database System Concepts - 8<sup>th</sup> Edition 21.28 ©Silberschatz, Korth and Sudarshan

So, this is how a typical JSP will look like. So, you can see that this actually looks like HTML, but inside that you have a, you have some part of a Java code. So, what will happen the body will get replaced when the when the response has come for example, here the response is doing hello world.

So, when this is executed then whatever is a result will replace the body in the HTML here and the result in the HTML will get generated. There is another mechanism of scripting which is also very popular called PHP. So, this is how it is done.

(Refer Slide Time: 29:19)

The slide features a small sailboat icon in the top left corner. The title 'Client Side Scripting' is centered at the top in a red font. Below the title, there is a list of bullet points and some descriptive text.

- Browsers can fetch certain scripts (client-side scripts) or programs along with documents, and execute them in "safe mode" at the client site
  - Javascript
  - Macromedia Flash and Shockwave for animation/games
  - VRML
  - Applets
- Client-side scripts/programs allow documents to be active
  - E.g., animation by executing programs at the local site
  - E.g., ensure that values entered by users satisfy some correctness checks
  - Permit flexible interaction with the user.
    - Executing programs at the client site speeds up interaction by avoiding many round trips to server

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur Date: Jan-Apr- 2018

Database System Concepts - 8<sup>th</sup> Edition 21.30 ©Silberschatz, Korth and Sudarshan

Similarly, you could have script an client side also for example, if you are entering data for say a month and in numeric and you happened to enter 14; then in most cases the page will immediately give a error saying that 14 cannot be a valid month; so, there is a validation involved.

So, in the client side in the browser there are some small script that can run a most typically it is a Java script which can too different authentication which is possible without actually accessing the data in the database. You cannot for example, validate a mail data based on the client side scripting sitting on the browser, but you can validate small things like valid data forms, range of data and so, on.

And it is it is very important because if you could not do that then all you required is you would have send that faulty month numbered 14 to the to the backend server and got an error and you would have come back and then have to correct it, but you can do this locally at the browser itself.

(Refer Slide Time: 30:26)

The slide has a header 'Client Side Scripting and Security' with a sailboat icon. The main content lists security mechanisms for client-side scripts:

- Security mechanisms needed to ensure that malicious scripts do not cause damage to the client machine
  - Easy for limited capability scripting languages, harder for general purpose programming languages like Java
- E.g., Java's security system ensures that the Java applet code does not make any system calls directly
  - Disallows dangerous actions such as file writes
  - Notifies the user about potentially dangerous actions, and allows the option to abort the program or to continue execution.

Navigation icons and footer text are visible at the bottom.

So, client side scripting has a lot of value, but you will have to have to remember that a if you are doing client side scripting then there are security issues.

Because it is quite possible that if you are doing things on the client side that is on the browser then we might also inadvertently or by a malicious intact actually make damages to the machine on which the browser is running. So, there are different kinds of care that

is to be taken for example, Java applet which is another way of doing client side computation disallows file writes and so, on.

(Refer Slide Time: 31:03)

The slide features a small sailboat icon in the top-left corner. The title 'Javascript' is in red at the top right. The main content is a bulleted list under two main points:

- Javascript very widely used
  - forms basis of new generation of Web applications (called Web 2.0 applications) offering rich user interfaces
- Javascript functions can
  - check input for validity
  - modify the displayed Web page, by altering the underlying **document object model (DOM)** tree representation of the displayed HTML text
  - communicate with a Web server to fetch data and modify the current page using fetched data, without needing to reload/refresh the page
    - ▶ forms basis of AJAX technology used widely in Web 2.0 applications
    - ▶ E.g. on selecting a country in a drop-down menu, the list of states in that country is automatically populated in a linked drop-down menu

At the bottom left, it says 'Database System Concepts - 8<sup>th</sup> Edition'. In the bottom center, it shows '21.32'. At the bottom right, it says '©Silberschatz, Korth and Sudarshan'.

And Java script as I have said is widely used and it can function to check for input validity which I gave an example of it can modify the displayed web page, it can communicate with the web server to fetch data and so, on. And it is you should familiarize yourself with Java script more; it is a very powerful mechanism to do compute the sample things at the client side this is an example that I have given.

(Refer Slide Time: 31:20)

The slide features a small sailboat icon in the top-left corner. The title 'Javascript' is in red at the top right. The main content is a bulleted list under one point:

- Example of Javascript used to validate form input

```
<html> <head>
<script type="text/javascript">
function validate() {
    var credits=document.getElementById("credits").value;
    if (isNaN(credits)|| credits<=0 || credits>=16) {
        alert("Credits must be a number greater than 0 and less than 16");
        return false
    }
}
</script>
</head> <body>
<form action="createCourse" onsubmit="return validate()">
    Title: <input type="text" id="title" size="20"><br />
    Credits: <input type="text" id="credits" size="2"><br />
    <input type="submit" value="Submit">
</form>
</body> </html>
```

At the bottom left, it says 'Database System Concepts - 8<sup>th</sup> Edition'. In the bottom center, it shows '21.33'. At the bottom right, it says '©Silberschatz, Korth and Sudarshan'.

So, you can read through and you will be able to if you know Java you will be able to understand Java script very easily, you could get through that.

(Refer Slide Time: 31:33)

The slide has a header 'USP of JSP' with a small sailboat icon to its left. The content is organized into four main sections:

- JSP vs. Active Server Pages (ASP)**
  - ASP is a similar technology from Microsoft and is proprietary (uses VB).
  - JSP is platform independent and portable.
- JSP vs. Pure Servlets**
  - JSP is a servlet but it is more convenient to write and to modify regular HTML than to have a million println statements that generate the HTML.
  - The Web page design experts can build the HTML, leaving places for the servlet programmers to insert the dynamic content.
- JSP vs. JavaScript** JavaScript can generate HTML dynamically on the client.
  - "Client Side": Java Script code is executed by the browser after the web server sends the HTTP response. With the exception of cookies, HTTP and form submission data is not available to JavaScript.
  - "Server Side": Java Server Pages are executed by the web server before the web server sends the HTTP response. It can access server-side resources like databases, catalogs.
- JSP vs. Static HTML** Regular HTML, of course, cannot contain dynamic information.

At the bottom, there is footer text: 'SWAYAM: NPTEL-NOCO Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '21.34', and '©Silberschatz, Korth and Sudarshan'.

And so, there are multiple options of doing such things, but JSP has a unique position because it is very useful in many contexts. A JSP has certain USP like active server pages which are used in terms of the Microsoft platform is also another mechanism of doing server side scripting, but JSP is better because it is portable in comparison to pure servlets which we showed you in the beginning. JSP performs easier to use.

Because JSP has the structure of the HTML page whereas, in a pure servlet we will have to use print line to print every tag of the HTML which is cumbersome. JSP in contrast with Java script is certainly a different thing because Java script runs on the browser on the client side, JSP runs on the server side and certainly JSP is compared to static HTML is more powerful because it can handle dynamic information.

(Refer Slide Time: 32:39)

The slide is titled "Module Summary" in red font at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list of learning objectives:

- Understood the requirements of Database Application Programs and User Interfaces
- Familiarized with the Fundamentals notions and technologies of Web
- Learnt the notions of Servlets and Java Server Pages

At the bottom left, it says "Database System Concepts - 8<sup>th</sup> Edition". In the center bottom, it shows "21.35". At the bottom right, it says "©Silberschatz, Korth and Sudarshan". A decorative footer bar with various icons is also present.

So, in this that brings us to the end of this current modules; so, what we have done we have understood the basic requirements of database application programs and user interfaces understood the basic terminology of the web and took a look into the core notion, core technologies of application development which is in terms of the servlets and Java server pages.

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

**Lecture – 22**  
**Application Design and Development (Contd.)**

Welcome to module 22 of database management systems, we have been discussing application, design and development this is the second part of that discussion.

(Refer Slide Time: 00:27)

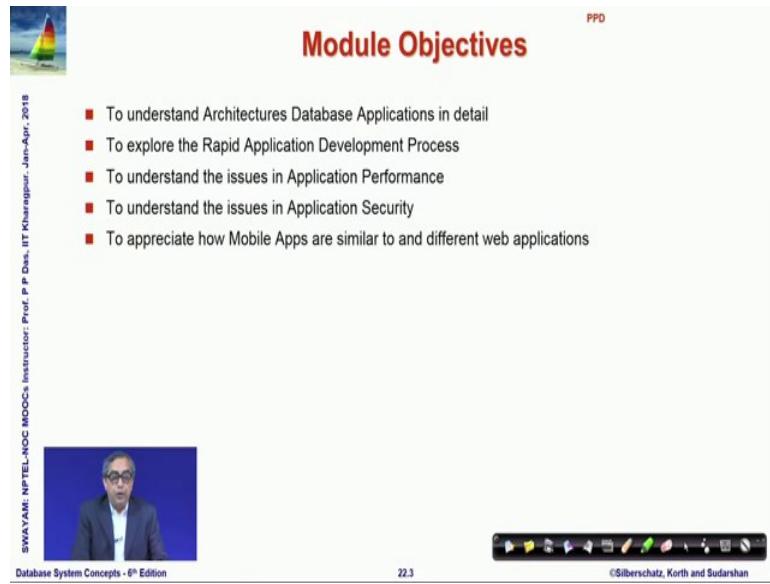
**Module Recap**

- Application Programs and User Interfaces
- Web Fundamentals
- Servlets and JSP

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018  
Database System Concepts - 8<sup>th</sup> Edition      22.2      ©Silberschatz, Korth and Sudarshan

In the last module, we have taken a quick look at the application programs and the user interfaces, looked at the fundamental notions of web and specifically the servlets and JSP.

(Refer Slide Time: 00:38)



**Module Objectives**

PPD

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

- To understand Architectures Database Applications in detail
- To explore the Rapid Application Development Process
- To understand the issues in Application Performance
- To understand the issues in Application Security
- To appreciate how Mobile Apps are similar to and different web applications

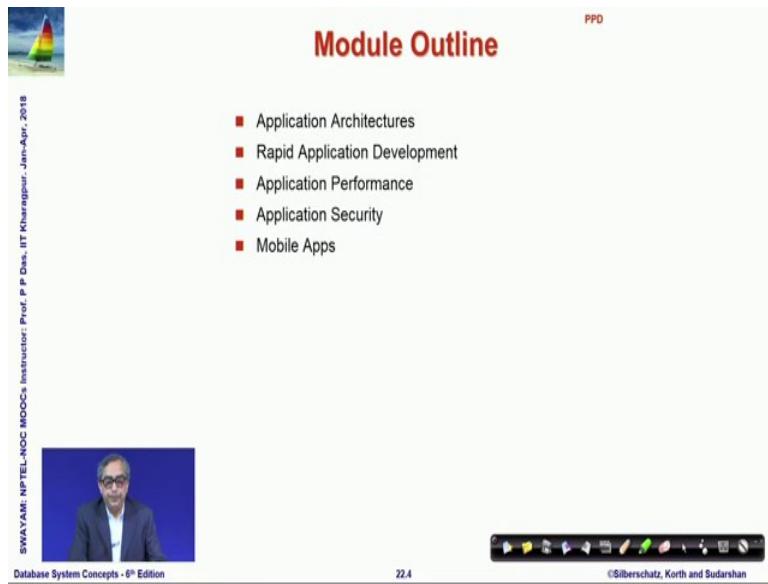
Database System Concepts - 8<sup>th</sup> Edition

22.3

©Silberschatz, Korth and Sudarshan

In the current module, we would like to understand the 3 tier architecture in little bit more detail, and explore quickly take a look into the rapid application development processes what kind of help is available, for quickly develop applications and then, we briefly we will look into the issues in terms of an applications performance and it is required security, and at the end we will discuss how a mobile app is similar to such web based database applications? And how they are different?

(Refer Slide Time: 01:14)



**Module Outline**

PPD

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

- Application Architectures
- Rapid Application Development
- Application Performance
- Application Security
- Mobile Apps

Database System Concepts - 8<sup>th</sup> Edition

22.4

©Silberschatz, Korth and Sudarshan

So, this is the outline the 5 parts.

(Refer Slide Time: 01:19)

**Application Architectures**

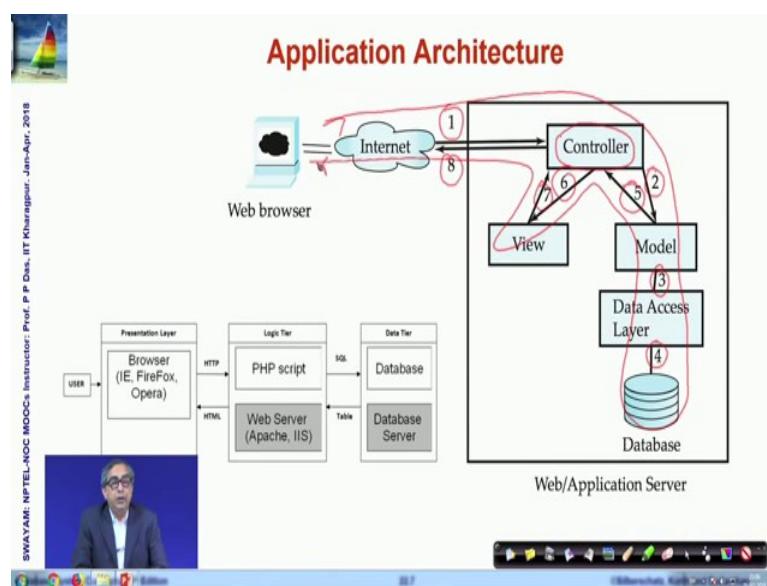
- Application layers
  - Presentation or user interface
    - **model-view-controller (MVC) architecture**
      - **model**: business logic
      - **view**: presentation of data, depends on display device
      - **controller**: receives events, executes actions, and returns a view to the user
    - **business-logic layer**
      - provides high level view of data and actions on data
      - often using an object data model
      - hides details of data storage schema
    - **data access layer**
      - interfaces between business logic layer and the underlying database
      - provides mapping from object model of business layer to relational model of database

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

Database System Concepts - 8<sup>th</sup> Edition 22.6 ©Silberschatz, Korth and Sudarshan

So, in terms of the application architecture again the presentation layer, or the user interface, business logic layer, and the data access layer, the frontend, the middle layer and the backend. Now, in the presentation layer or the user interface it is typical that applications follow, what is now known as MVC architecture, model view control architecture where, the model is the kind of the business logic that, that is implemented in terms of the frontend information. View is the actual presentation of the data, that HTML and controller is one who, receives different events execute actions and so on and then, we will go into the other layers.

(Refer Slide Time: 02:09)



So, here let us try to understand this flow. So, there is a request in the web browser say, to the service. So, this is the sequential now, let us say we are trying to log in to Gmail. So, in one we send a form HTML form which, has the username and the password and possibly encrypted, that comes to the controller ah. So, which basically controls the different events.

So, the controller knows that, it has to now decide whether, this what actions are required in terms of this input data. So, it is sends it to the model which, is the business logic here. So, the business logic model knows now well. So, at this there is an application which, deciphers the business logic which says that, business logic required here is we have a password and we have a user names now, we have to decide whether this user is a valid user and whether, he or she can be allowed to login. So, the model has to check on the user data, the user id and password data and therefore, it has to come from a database. So, it passes on this request to the data access layer, the data access layer in turn access the database.

So, you can think of data access layer is something like a SQL query layer where, you have formed a query select etc., from etc., where, user id is equal to PPD, password is equal to XXX, the database depending on what is found in the database is back to the model, and the model then sends it to the control it is says ok, this is what I have found. So, this is the result of the data that result of the request that has been prepared. So, it is says that well this is have been found and therefore, we have extracted the mails in the inbox that existed, or it is says that the authentication is not possible. So, it plugs in a error message and sends it to the controller, controller now knows that a response has to be framed. So, the controller sends it to the view of the MVC, the view we prepare the HTML that needs to go back. So, view prepares the HTML and sends it back to the controller. So, controller now has the response which it is sends back to the web browser through the internet, and we get to see that well now, my inbox and mails are all here.

So, this is a complete flow of starting from here, going through this, coming back, going here, going back here, is the is the whole route of the request, response that goes over the HTTP in a typical web or application scenario, that is the there is a way this application architecture is expected to work.

(Refer Slide Time: 05:13)

**Sample Applications in Multiple Tiers**

Application	Presentation	Logic	Data	Functionality
Web Mail	<ul style="list-style-type: none"> <li>• Login</li> <li>• Mail List View <ul style="list-style-type: none"> <li>• Inbox</li> <li>• Sent Items</li> <li>• Outbox</li> <li>• Trash</li> </ul> </li> <li>• Mail Composer</li> <li>• Filters</li> </ul>	<ul style="list-style-type: none"> <li>• User Authentication</li> <li>• Connection to Mail Server (SMTP, POP, IMAP)</li> <li>• Encryption / Decryption</li> </ul>	<ul style="list-style-type: none"> <li>• Mail Users</li> <li>• Address Book</li> <li>• Mail Items</li> </ul>	<ul style="list-style-type: none"> <li>• Send / Receive Mails</li> <li>• Manage Address Book</li> </ul>
Net Banking	<ul style="list-style-type: none"> <li>• Login</li> <li>• Account View</li> <li>• Add / Delete Account</li> <li>• Add / Delete Beneficiary</li> <li>• Fund Transfer</li> </ul>	<ul style="list-style-type: none"> <li>• User Authentication</li> <li>• Beneficiary Authentication</li> <li>• Transaction Validation</li> <li>• Connection to Banks / Gateways</li> <li>• Encryption / Decryption</li> </ul>	<ul style="list-style-type: none"> <li>• Account Holders</li> <li>• Beneficiaries</li> <li>• Accounts</li> <li>• Debit / Credit Transactions</li> </ul>	<ul style="list-style-type: none"> <li>• Check Balance and Transactions</li> <li>• Transfer Funds</li> </ul>
Timetable	<ul style="list-style-type: none"> <li>• Login</li> <li>• Add / Delete Courses, Teachers, Rooms, Slots</li> <li>• Assignments: <ul style="list-style-type: none"> <li>• Teachers → Course</li> </ul> </li> <li>• Allocations <ul style="list-style-type: none"> <li>• Course → Room, Slots</li> </ul> </li> <li>• Views</li> </ul>	<ul style="list-style-type: none"> <li>• User Authentication</li> <li>• Timetable Assignment Logic</li> <li>• Encryption / Decryption</li> </ul>	<ul style="list-style-type: none"> <li>• Courses</li> <li>• Teachers</li> <li>• Rooms</li> <li>• Slots</li> <li>• Assignments</li> <li>• Allocations</li> </ul>	<ul style="list-style-type: none"> <li>• Manage timetable for multiple courses taken by multiple teachers</li> </ul>

So, here I have created a small table, showing you the different activity is that happens at the presentation logic and data layer of different common applications like, web mail like, Google. So, at the presentation layer you will do things like, log in, mail list, view inbox, sent item, outbox so on, mail composer. So, we can write mails filters of checking at different mails. So, all the all these happens.

So, for example, if you talk about filters they might often happen in the java script itself, that trans within the browser, the logic the business logic will do user authentication, connection to mail server because, a mails have to come from a different server, they are not they may not be setting typed in terms of the database itself.

And then user encryption, decryption and the data side you will have different tables to represent the mail users, the users like you and me all who are users of the Gmail, the address book of each for each one of us the mail items and so on, and that will give us the functionality of send, receive mails, managing address book and so on. So, similarly I have listed an application for net banking which can where, we can check balance and do transactions transfer funds, or a timetable where you can manage time table for multiple courses taken by multiple teachers and so on.

So, you can if you think about an application then, you should be able to moderately map it is a functionality across the presentation logic and data layer. So, that you know what you want to do at the clients, and which is which is at the presentation layer, or what you

want at the absolute packet which is on the data, what tables and all the databases that you want to maintain, and the business logic of how actually this application will give you the result, how actually it will?

So, that is something where, you will have a whole lot of complex logic that might come in.

(Refer Slide Time: 07:18)

The slide has a header 'Business Logic Layer' with a sailboat icon. The content lists four main points about the Business Logic Layer:

- Provides abstractions of entities
  - e.g. students, instructors, courses, etc
- Enforces **business rules** for carrying out actions
  - E.g. student can enroll in a class only if she has completed prerequisites, and has paid her tuition fees
- Supports **workflows** which define how a task involving multiple participants is to be carried out
  - E.g. how to process application by a student applying to a university
  - Sequence of steps to carry out task
  - Error handling
    - e.g. what to do if recommendation letters not received on time

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jun-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition      22.9      ©Silberschatz, Korth and Sudarshan

So, coming to the business logic layer specifically, that provides abstraction of that will provide abstraction of various entities, students, instructors, courses, mails, your accounts, balance and so on, and that will enforce business tools for carrying out this. So, a student can enroll in a class only if she has completed prerequisites, you can transfer funds from one account to the other, provided, you have the authority to transfer, provided you have enough funds in the account that is going to get debited and so on. So, those are the different business tools, which will be employed by the business logic layer, and it will support a work flow which defines how a task will be carried out in terms of the multiple participants, and remember that all participants may not actually be human beings, they could be some could be human being some could be other machines or other applications as well.

So, it gives you the work flow. So, you can any of the 3 application that I just mentioned, everywhere you can find that there is a work flow, if you are want to check a mail then, there is a steps that you need to do go to the inbox, chose the particular mail item, get the

body and then if you want to reply to that, select that, the submit that, you get a new form when you write the reply, and within that form you get the original copy of the original mail and so on. So, all these kind of work flows will be supported by the business logic layer.

(Refer Slide Time: 08:47)

The slide has a header 'Object-Relational Mapping' with a sailboat icon. The main content is a bulleted list of pros and cons of O-R mapping:

- Allows application code to be written on top of object-oriented data model, while storing data in a traditional relational database
  - alternative: implement object-oriented or object-relational database to store object model
    - ▶ has not been commercially successful
- Schema designer has to provide a mapping between object data and relational schema
  - e.g. Java class *Student* mapped to relation *student*, with corresponding mapping of attributes
    - An object can map to multiple tuples in multiple relations
- Application opens a session, which connects to the database
- Objects can be created and saved to the database using `session.save(object)`
  - mapping used to create appropriate tuples in the database
- Query can be run to retrieve objects satisfying specified predicates

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kanpur - Jan-Apr - 2018

Database System Concepts - 8<sup>th</sup> Edition      22.10      ©Silberschatz, Korth and Sudarshan

Now, certainly you can understand that at the frontend, if we talk about what are different you know languages and models, that we are working within the frontend naturally our language is HTML tries for presentation, embedded with java script, at the backend in the data it is the database and the SQL query. So, it is a relational model, but what happens in between? What happens in the business layer which connects them? Now, naturally business layer you could write complex business tools.

For example, in the time table application we will have a complex algorithm, I know to find out what allocation of class rooms and slots, are feasible for assignments of teachers to courses availability of rooms and so on. So, often the business logic layer, the this tier would be convenient to write in some typical common high-level language like, C ++ or java, and naturally you would know from your experience of software engineering that, if you have a object based language then, that will be a very convenient to do that.

So, which means that, if you have say some entity as a student in your relational database then, most likely in your business logic, which is a java code you will have a class called student. So, the relation student is in the relational model, and your class student is in the

object based model, and you will need to define these in terms of certain mapping of the attributes, which we had shown when we talked about embedding of a languages, and this is what is commonly known as a object relational mapping.

So, that objects can map to multiple tuples, in multiple relations and can be viewed in a different way. So, so this mapping itself we could create a virtual view in the business logic layer, in the business logic language that you are looking at. Of course, they have been attempts to create a models, which are relational models which are also object oriented those are called object relational databases, some of them have been successful, but not really commercially successful.

So, we continue to work with SQL, and the relational database kind of things in the database level, and some kind of a high-level language like, C++, java and the object-based model in the middle tarred in the in the business logic, and continue to do the object relational mapping for solving the problems. So, you can here, I have given the points of what happens? How the objects get created? Application opens a session, which connects to the database because, we need to get the data from the database, they can be objects that can be created safe to the database.

They can be extracted from the database; new objects can be created and mapping use to create a appropriate tuples in the database. So, it is a two-way traffic that will keep on happening where, the business logic layer will continue to see entities as objects whereas, the database layer will continue to see them, as tuple in the relational database.

(Refer Slide Time: 12:07)



## Web Services

■ Allow data on Web to be accessed using remote procedure call mechanism  
■ Two approaches are widely used

- **Representation State Transfer (REST)**: allows use of standard HTTP request to a URL to execute a request and return data
  - ↳ returned data is encoded either in XML, or in **JavaScript Object Notation (JSON)**
- **Big Web Services**:
  - ↳ uses XML representation for sending request data, as well as for returning results
  - ↳ standard protocol layer built on top of HTTP

SWAYAM: NPTEL-NOOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition 22.11 ©Silberschatz, Korth and Sudarshan



So, there are also several web services, that can be used and you may be getting familiar with that, we will not go deep into this, but I will just mention that, web services a mechanism through which, you can access a data from remote server using what is known as a remote procedure call, and today very common approach for this is called rest representation state transfer, which allow standard HTTP request to a URL to execute a request and return data, and a several of the web services are based on that, and are the are big web services which you must have heard of, but I just mentioned it at this point because it is contextual, but we will not get into those in this course really. Now, coming to how do you actually develop applications?

(Refer Slide Time: 13:05)



## Rapid Application Development

■ A lot of effort is required to develop Web application interfaces

- more so, to support rich interaction functionality associated with Web 2.0 applications

■ Several approaches to speed up application development

- Function library to generate user-interface elements
- Drag-and-drop features in an IDE to create user-interface elements
- Automatically generate code for user interface from a declarative specification

■ Above features have been in used as part of **rapid application development (RAD)** tools even before advent of Web

■ Web application development frameworks

- Java Server Faces (JSF) includes JSP tag library
- Ruby on Rails
  - ↳ Allows easy creation of simple **CRUD** (create, read, update and delete) interfaces by code generation from database schema or object model

SWAYAM: NPTEL-NOOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition 22.11 ©Silberschatz, Korth and Sudarshan



Now, it is not an easy process that is a lot of effort required to develop web applications, you need to support the functionality that of current day web. So, you need several approaches to speed up applications. So, this is in parallel to if you think of how? What has been done to speed up development processes, development applications for different say C applications, or C++, or java applications.

So, one approach naturally is to variety of function library, which can help you easily get user interface elements like buttons, checkboxes, radio drag and drop features in the IDE, IDE stands for integrated development environment like, visual studio, front page these kind of which can use a, which can create user interference elements easily you can automatically generate code for user interface, and these are all parts of rapid application development tools, and some frameworks are very popular, this is primarily the java server faces or JSF is a framework where, you can rapidly develop fill in all the requirements of the different layers, in a web based database application in other very popular is ruby on rails, which allows easy creation of simple crud create, read, update, delete.

So, if you look at database applications and common applications will all applications will at least need to do this it lead to create data, read data, update data, delete data. So, you can do that quickly with ruby on rails. So, if you are looking into really the development of database applications, get yourself familiar with this rapid application development processes.

(Refer Slide Time: 14:53)

The slide has a header 'ASP.NET and Visual Studio' with a sailboat icon. The main content lists features of ASP.NET and Visual Studio, including drag-and-drop development, DataSet object association, Validator controls, JavaScript enforcement, user actions, and DataGrid usage. The footer includes copyright information for Prof. P. Das, IIT Kharagpur, Jan-Apr. 2018, and SWAYAM: NPTEL-NOOC MOOCs.

ASP.NET and Visual Studio

- ASP.NET provides a variety of controls that are interpreted at server, and generate HTML code
- Visual Studio provides drag-and-drop development using these controls
  - E.g. menus and list boxes can be associated with DataSet object
  - Validator controls (constraints) can be added to form input fields
    - ▶ JavaScript to enforce constraints at client, and separately enforced at server
  - User actions such as selecting a value from a menu can be associated with actions at server
  - DataGrid provides convenient way of displaying SQL query results in tabular format

SWAYAM: NPTEL-NOOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition 22.14 ©Silberschatz, Korth and Sudarshan

There are different other frameworks as well, another very popular and widely used framework is Microsoft specific, this is unfortunately not portable because, it is proprietary of Microsoft where, you can use the the [dot] net framework of Microsoft, which helps you with lot of an a resources and libraries which are already provided, and like we talked about JSP, we can use ASP[ dot] net here active server page in the[ dot] net framework, which provides a whole lot of controls and you have a very nice powerful id in terms of visual studio, high were being proprietary these need licenses and you need to pay for that. So, many a times, many developers may not be able to afford it or like it for that reason. Naturally, as you designed applications and create that, you will have to be careful about it is performance because certainly we all want very fast results.

(Refer Slide Time: 15:52)

The slide has a title 'Improving Web Server Performance' at the top right. On the left is a small image of a sailboat on water. The background is light green. At the bottom left is vertical text: 'SWAYAM: NPTEL-NOCO MOOCs Instructor: Prof. P. Doshi, IIT Kanpur - Jan-Apr-2018'. At the bottom right are navigation icons and the text 'Database System Concepts - 6<sup>th</sup> Edition 22.16 ©Silberschatz, Korth and Sudarshan'.

## Improving Web Server Performance

- Performance is an issue for popular Web sites
  - May be accessed by millions of users every day, thousands of requests per second at peak time
- Caching techniques used to reduce cost of serving pages by exploiting commonalities between requests
  - At the server site:
    - ▶ Caching of JDBC connections between servlet requests
      - a.k.a. **connection pooling**
    - ▶ Caching results of database queries
      - Cached results must be updated if underlying database changes
    - ▶ Caching of generated HTML
  - At the client's network
    - ▶ Caching of pages by Web proxy

So, if I log in to my Gmail application, and I would expect that as soon as I press the submit button with a couple of seconds, my inbox will be displayed on my browser, I am not ready to wait for 2 minutes, 3 minutes, 5 minutes for doing that.

So, while developing the application you will have to make an estimate of how often it will be used? How many users will use that every day and so on? how many, what is your expected heat rate? That is, when the maximum number of users are trying to use this then, what is the you know estimated number of request per second? That will come to your server. And to improve performance their different kinds of techniques can be used, the significant of them is called caching, caching is nothing but if you expect to request to be similar ah. So, that they are results should be similar then, after the first request besides sending the response back, you actually keep a local copy of that.

So, that if a similar request come in future, you can you may not re compute it, you can just send that cache copy. So, it can be done in terms of verity of JDBC connections called connection pooling, it can be done in terms of database queries, caching of generated, HTML and so on.

It can be done at the clients network side also by caching pages by web proxy; obviously, if you are using caching to improve performance, you will have to understand lot more of the web dynamics in depth because, certainly if you cache then there is a possibility, that somebody is asking is sending a request for which, the response would not be the same as what it was, when the last time the response was computed and cached. So, if

you send the cached information back then, you may be giving a dated information. So, possibly say for example, if you are checking at the net banking transaction, you have making a fund transfer and checking your account transactions after that one transfer, you would not expect a cached page whereas, if you are looking at the website of a say IIT Kharagpur then, it will be to cache that because, it is not expected to change very frequently. So, these are different factors. So, we do not have it in the scope to going to different issues of, how to improve performance? And what are the different challenges?

But I just want that you to be sensitive about these issues. Other very deep you know concerned, deep requirement about applications are the security of applications, there are this is a very involved topic, and there are several issues that are involved.

(Refer Slide Time: 18:48)

The slide has a title 'SQL Injection' in red. The content is a bulleted list of points related to SQL injection:

- Suppose query is constructed using
  - "select \* from instructor where name = " + name + ""
- Suppose the user, instead of entering a name, enters:
  - X' or 'Y' = 'Y
- then the resulting statement becomes:
  - "select \* from instructor where name = " + "X" or 'Y' = 'Y" + ""
  - which is:
    - ✖ select \* from instructor where name = 'X' or 'Y' = 'Y'
  - User could have even used
    - ✖ 'X'; update instructor set salary = salary + 10000; --
- Prepared statement internally uses:
  - "select \* from instructor where name = 'X' or 'Y' = 'Y"
- **Always use prepared statements, with user inputs as parameters**
- Is the following prepared statement secure?
  - conn.prepareStatement("select \* from instructor where name = " + name + "")

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

Database System Concepts - 8<sup>th</sup> Edition      22.18      ©Silberschatz, Korth and Sudarshan

So, and I am talking about only a few of them because, the many of them require knowledge about several other fields particularly, in the field of security and an encryption and so on. So, one that is very common in terms of SQL query is known as a SQL injection where, based on I mean there is if you are expecting some inputs to come ah, and fill up certain parts of the SQL query then, you will have to be careful that, user should not be able to give such input say such strings.

So, that the query actually mean something different, this query may be which was just a query to read something, may update something, or give some different result. So, here I have just briefly highlighted some of those issues, there are other security issues like,

leakage of password if there are important things which are based on a single password then, use the password gets compromised because, it is shared or it is broken in a middle by some hacker and so on then, you will have a lot of risks involved.

(Refer Slide Time: 19:37)

The slide features a small sailboat icon in the top left corner. The title 'Password Leakage' is centered at the top in a red font. The main content consists of two bullet points:

- Never store passwords, such as database passwords, in clear text in scripts that may be accessible to users
  - E.g. in files in a directory accessible to a web server
    - ▶ Normally, web server will execute, but not provide source of script files such as file.jsp or file.php, but source of editor backup files such as file.jsp~, or file.jsp.swp may be served
- Restrict access to database server from IPs of machines running application servers
  - Most databases allow restriction of access by source IP address

At the bottom, there is footer text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '22.19', and '©Silberschatz, Korth and Sudarshan'.

(Refer Slide Time: 19:59)

The slide features a small sailboat icon in the top left corner. The title 'Application Authentication' is centered at the top in a red font. The main content consists of two bullet points:

- Single factor authentication such as passwords too risky for critical applications
  - guessing of passwords, sniffing of packets if passwords are not encrypted
  - passwords reused by user across sites
  - spyware which captures password
- Two-factor authentication
  - e.g. password plus one-time password sent by SMS
  - e.g. password plus one-time password devices
    - ▶ device generates a new pseudo-random number every minute, and displays to user
    - ▶ user enters the current number as password
    - ▶ application server generates same sequence of pseudo-random numbers to check that the number is correct.

At the bottom, there is footer text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '22.20', and '©Silberschatz, Korth and Sudarshan'.

So, you should be you will need to make sure that, you do not store passwords you just store encrypted forms of that in which encrypted forms, you have must have observed for the last couple of years that, in many cases earlier you could just log in. So, giving just one password was enough, but now in many transactions for example, if you are logging

into a net banking application then, often you will be asked to provide some additional key information, or you will be asked to do a authentication by OTP one-time password and so on. So, these are common because, you know it is risky to work with just a single authentication, and you will find that some net banking applications for example, if you are doing a fund transfer, then they actually require two additional password authentication, one is a special password for fund transfer, and then possibly we will be asked to go through an OTP. So, depending on the criticality of your application and the potential vulnerability of the application, your authentication mechanism will have to be appropriately designed.

(Refer Slide Time: 21:09)

The slide has a title 'Application-Level Authorization' in red at the top right. On the left is a small logo of a sailboat on water. The main content area contains a bulleted list of issues with SQL authorization:

- Current SQL standard does not allow fine-grained authorization such as "students can see their own grades, but not other's grades"
  - Problem 1: Database has no idea who are application users
  - Problem 2: SQL authorization is at the level of tables, or columns of tables, but not to specific rows of a table
- One workaround: use views such as
 

```
create view studentTakes as
  select *
  from takes
  where takes.ID = syscontext.user_id()
```

  - where syscontext.user\_id() provides end user identity
    - ↳ end user identity must be provided to the database by the application
  - Having multiple such views is cumbersome

At the bottom, there is footer text: 'SWAYAM, NPTEL-NOCO MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018', 'Database System Concepts - 6<sup>th</sup> Edition', '22.21', and '©Silberschatz, Korth and Sudarshan'.

There are issues in terms of security, in terms of the application level also, for example, if you are looking at a student application where, you want to say that the student, every student can see his or her own grade, but they should not be able to see the grade of others. Now, how do you implement this? Now, two issues are one is the database cannot implement this as a part of access control because, database has no idea about who the application users are, and the second if you recall the way, we talked about authorization in SQL, that is in terms of tables or columns of the tables, but SQL has no mechanism to create authorization for rows of the table.

So, this will have to be handled, in terms of what is known as at the application layer. So, this is where you are. So, what this is saying that you are created a view where, you are created a view student text where, to do this you will have to read you are saying where, text [dot] id is equal to this function called is sys context [dot] user id. So, sys context is an object naturally, dot user I d this function called. So, what this function gives you call to the function gives you is an information about the end user identity, and that identity has to match, the identity that exist in the text ID for the result to be computed. So, this will ensure that depending on who is actually the current user, the same view will be evaluated for different results.

So, this is no not a not a very sound this is not a very comfortable situation because, here the authorization is not being implemented in the database level, but is being implemented at the application level, but that is way things a because, most of the fine grained authorization currently, is done entirely in the application level only a extension to SQL authorization, at for similar you know fine graining and so on has been proposed, but they have not been implemented because, there are several issues of implementing where there several issues of how do you represent? How do you module? And most importantly, these have potentials of slowing down the database, query process and significantly.

(Refer Slide Time: 23:08)

The slide has a header 'Application-Level Authorization (Cont.)' with a sailboat icon. The main content lists pros and cons of application-level authorization and introduces fine-grained row-level authorization. A footer notes the instructor is Prof. P. P. Dhas, IIT Kharagpur, Jan-Apr., 2018, and mentions SWAYAM: NPTEL-NOC NOOCs Instructor: Prof. P. P. Dhas, IIT Kharagpur - Jan-Apr., 2018.

**Application-Level Authorization (Cont.)**

- Currently, authorization is done entirely in application
- Entire application code has access to entire database
  - large surface area, making protection harder
- Alternative: **fine-grained (row-level) authorization** schemes
  - extensions to SQL authorization proposed but not currently implemented
  - Oracle Virtual Private Database (VPD) allows predicates to be added transparently to all SQL queries, to enforce fine-grained authorization
    - ↳ e.g. add `ID= sys_context.user_id()` to all queries on student relation if user is a student

So, often this is not a preferred mechanism either. Another way to ensure security is to keep audit trail, trail must log actions into it.

(Refer Slide Time: 23:46)

The slide has a title 'Audit Trails' in red at the top right. To its left is a small image of a sailboat on water. On the far left edge of the slide, there is vertical text that reads 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kanpur - Jan-Apr. 2018'. The main content is a bulleted list:

- Applications must log actions to an audit trail, to detect who carried out an update, or accessed some sensitive data
- Audit trails used after-the-fact to
  - detect security breaches
  - repair damage caused by security breach
  - trace who carried out the breach
- Audit trails needed at
  - Database level, and at
  - Application level

At the bottom of the slide, there is footer text 'Database System Concepts - 8<sup>th</sup> Edition' and '22.23', followed by a navigation bar with icons and the copyright notice '©Silberschatz, Korth and Sudarshan'.

So, where you write down actions in terms of who carried out and update, or who access some sensitive data, or who did a delete and so on. So, audit trails can be used later on if the need arises to detect, if some security breach that happen, or if some damage has been caused by the security breach, that can be corrected and so on create a trace.

So, auditing is necessarily a very required, in a very required activity for any data-based application and audit trail had a good mechanism for that. So, when you develop applications you have to consider has to whether, you would like to support audit trail of course, if you support audit trail then, somewhat your application will get slowed down, it will require more disk to keep that trail because, every transaction every details you will get logged in to the audit trail, but it is very, very important for critical applications like, net banking where currently it is mandated every transaction that, you do every action that you do on your account, through the net banking is trailed in the banks end.

So, that if later there is a there is some dispute, there is some breach has found then, the same can be recovered and trace back to the actions, that you are actually taken ah. At the end of this module let me take a quick look into the mobile apps.

(Refer Slide Time: 25:32)

**What Is a Mobile App?**

- A type of application software designed to run on a mobile device, such as a smartphone or tablet computer
- Developed specifically for use on small, wireless computing devices, such as smartphones and tablets
- Designed with consideration for the demands and constraints of the devices and also to take advantage of any specialized capabilities
  - Form Factor – influences display and navigation
  - Limited Memory
  - Limited Computing Power
  - Limited Power
  - Limited Bandwidth
  - ...
  - + Availability of sensors like accelerometer
  - + Availability of touchscreen – Gesture-based Navigation
  - + ...

Source: <https://www.slideshare.net/hassandar18/architecture-of-mobile-software>

PPD

SWAYAM: NPTEL-NOOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jan-Apr - 2018

Database System Concepts - 8<sup>th</sup> Edition

22.25

©Silberschatz, Korth and Sudarshan

These are somewhat different, but it is very important to today to take a notion of the mobile app. So, it is a type of a application software, which is designed to run on a mobile devise. So, it this is not necessarily mean that, it has to be a smart phone, it could be a tablet or you know some small device, which typically is handled and carried around. So, it is typically specifically for use on small wireless computing devices, and it is designed with considerations for demands and coincident of the device. So, how are how is a different? If I want to use the application or want to have an application which does my task, but I want to do it as a mobile app, I want to do it for a small wireless handled device. Certainly, there are if we if we since the small device it has lot of constraints, there are lot of demands of the device, compare to if we were using a web browser in a desktop or a laptop where, you have lot of resources.

And also on the this is this is on the one kind of the restriction side and on the other side, these devices have certain capabilities, which your desktop or laptop may not have and therefore, it may be you may be able to do more interesting application using this mobile devices, and there could be really interesting mobile apps, and as you all must be using today use 10s of if not 100s of mobile apps for different applications, many of which actually support a database at the backend. So, these are in red are some of the negatives, which are restrictive in terms of a mobile application the first thing is form factor, which is the look the aspect ratio the size and the style of the device, which influences display and influences the way you navigate. In a desk of application you will typically use a mouse, or a roller and keyboard to navigate, but that is not possible or that is not easy in terms of a mobile device. So, you are navigation may happen in in a different way.

The presentation itself may happen in a different way, you have a very limited display area. So, you might want to stack multiple responses one after the other whereas, the in a in a web application running on a desktop, you would probably have shown them on different tabs side by side, or would have just shown them side by side on the display. Then, mobile devices typically have limited memory, they have limited computing power, very importantly most of them run on battery and therefore, they have one limited power available. So, you are applications will lead to be power optimized, which is not the case with the normal web based application, you have a wireless connection, so which may be limited in bandwidth based on your connectivity and that time, and so on. So, you while developing a mobile app corresponding to a possible web application, your considerations would be very different. So, if I say that a bank say, HDFC bank has a net banking application, which runs on the browser. And it is also having a mobile app, which runs on say the android phone then, the their requirements and their style of solutions will have to be very, very different. And at the same time if I talk about a mobile app then, the mobile devices often have features which desktop do not have for example, you have a number of senses available like, accelerometer and so on which you can make to advantage for example, we I mean in in many smart phones, if we just rotate the screen, rotate that device, then the display self-rotate automatically, which we use we use some some kind of an accelerometer inputs for that, you have a touch screen which can allow a wide range of gesture based navigation, which is typically not possible in desktop and most of the laptops.

(Refer Slide Time: 29:48)

**Mobile Website vis-à-vis Mobile App**

**■ Mobile Website**

- Similar to any other website in that it consists of browser-based HTML pages
- Can display text content, data, images and video
- Typically accessed over WiFi or 3G or 4G networks
- Designed for the smaller handheld display and touch-screen interface
- Can also access mobile-specific features such as click-to-call (to dial a phone number) or location-based mapping

**■ Mobile Apps**

- Actual applications that are downloaded and installed on the mobile device
- Users download apps from device-specific portals such as App Store, Google Play Store
- The app may
  - pull content and data from the Internet, in similar fashion to a website, or
  - download the content so that it can be accessed without an Internet connection

Source: <https://www.slideshare.net/hassandar18/architecture-of-mobile-software>

SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kanpur - Jan-Apr- 2018

PPD

Database System Concepts - 8<sup>th</sup> Edition

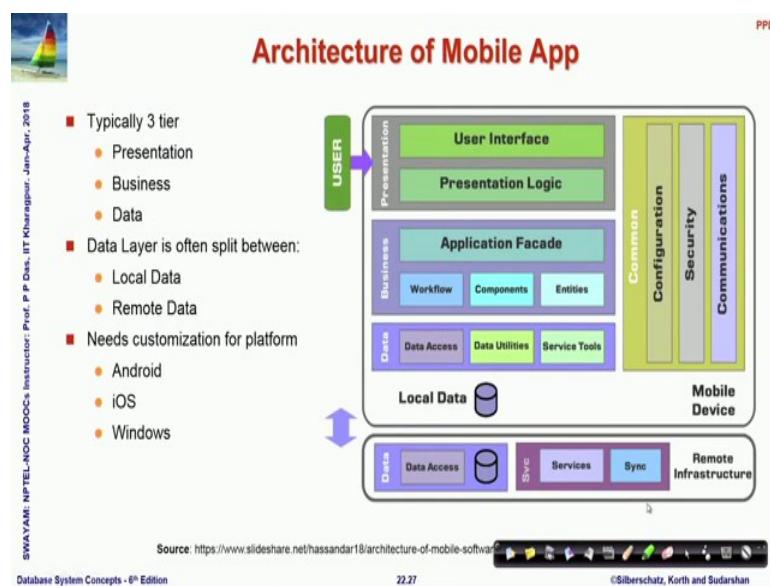
22.26

©Silberschatz, Korth and Sudarshan

So, having said that, naturally there are two aspects of mobile apps as well. So, when we talking about mobile apps, we have talking about stand alone mobile apps, there are other ways of developing applications also for example, we can do a typical web-based application, but we can use a website which is specifically designed catering to the mobile devices. So, these send back pages, which are smaller in size are stack differently and so on. So, in contrast to that a mobile app are actually, applications that are downloaded and runs on the system.

So, these are all different possibilities and even though mobile website is also becoming popular, but certainly mobile apps are very, very common in terms of a majority of critical applications of data bases that we have.

(Refer Slide Time: 30:40)



So, this is the typical architecture of a mobile app as you can see again here, that you have the three layers presentation, business and data the only difference being now, since you have a device, which can go anywhere and you have a connectivity.

So, this is what shows the connectivity, this is the device side and this is the pure backend or your service provider side. So, since your connectivity is not may not be very strong, you try to create all the layers of a presentation, business, as well as data on the mobile phone itself, but naturally all the data you will not have on your device ah, you are checking mails on the Gmail naturally, you will not have all the data on your Gmail.

So, what you do is you use the connection to connect to the remote database provided by the service, but primarily most of this layer of this presentation, business and a small part of the data layer are all realized within the phone itself, or within the mobile device itself. So, the data layer is split in this case, which is very different from how you do the web-based applications? And specifically, it might need customizations based on the kind of platform, you are using whether you have using android, iOS and windows, and these are all custom solutions for that and you could also note that, there are different types of mobile apps one large class is known as native application where, which is completely written in the native language of the platform.

So, for if you are doing an iOS mobile app then you will write it in object C, objective C if you are doing an android one you will write it in C or C++ or java is platform specific whereas, there is another class of mobile apps, which are known as web apps which run completely inside the web browser, so much like the way java scripts work.

(Refer Slide Time: 32:12)

**Types of Mobile Apps**

- **Native Apps:** Completely written in the native language of a platform
  - iOS → Objective-C, Android → Java or C/C++
  - Platform specific (heavily dependent on OS)
- **Web Apps:** Run completely inside of a Web browser.
  - Features interfaces built with HTML or CSS
  - Powered via Web programming languages →Ruby on Rails, JavaScript, PHP, or Python
  - Portable across any phone, tablet, or computer
- **Hybrid Apps:** Combines attributes of both native and Web apps.
  - Attempts to use redundant, common code that can be used across platforms, and
  - Tailors required attributes to the native system

SWAYAM: NPTEL-NOCO Instructor: Prof. P. Das, IIT Kharagpur - Jam-Apr- 2018

https://www.slideshare.net/hassandar18/architecture-of-mobile-software

Database System Concepts - 6<sup>th</sup> Edition

22.28

©Silberschatz, Korth and Sudarshan

So, they feature interfaces built with HTML and the style shades, and they have powered by different web programming languages like, ruby on rails java script to PHP and so on. And certainly there is a third kind, when you combine the attributes of both native and wave application and you try to you know. So, you can very easily understand, that if your application is a native one then it is not portable across devices, this is not an iOS application is not run on android and so on.

If it is a web app app, then it is portable because it is portable across different phone tablet or computer because, you are using generic technologies. So, you could do a hybrid app also where, you could use redundant or common code, which is usable across platform and add some tailor functionality in terms of the native system. So, these are the typical kinds of mobile apps that.

(Refer Slide Time: 33:50)

The slide has a header 'Design Issues' in red. Below it is a bulleted list of nine items:

- Determine Device
- Note Device Resources – memory, power, speed
- Consider Bandwidth
- Decide on Architecture Layers
- Select Technology
- Define User Interface
- Select Navigation
- Maintain Flow

On the left edge of the slide, there is vertical text that reads: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018'. In the top right corner, it says 'PPD'. At the bottom, it shows 'Source: <https://www.slideshare.net/hassandar18/architecture-of-mobile-software>' and a set of navigation icons. The footer of the slide includes 'Database System Concepts - 8<sup>th</sup> Edition', '22.29', and '©Silberschatz, Korth and Sudarshan'.

You might be developing and there are several factors to consider in the design issue, you have to first decide on the device, you have to take specific note of the typical resources, that your device will support memory, power, speed. You have to consider what should be the bandwidth should it be 2 G, 3 G, 4 G or wireless you know LAN, what kind of connections you would expect? Decide on the layers in the architecture, select the technology based on the device choice and the other factors define the user interface, navigation and maintain the work flow.

So, these are very variety I will not go into details of this, but just wanted to give a glimpse of the fact, that in today's time while you are talking about database applications then it is a very reality, that you will create that as a mobile app.

(Refer Slide Time: 34:32)



## Module Summary

- Studied the aspects of Database Applications Architectures
- Understood the steps in the Rapid Application Development Process
- Exposed to the issues in Application Performance
- Exposed to the issues in Application Security
- Learnt the distinctive features of Mobile Apps

SWAYAM: NPTEL-NOOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018



Database System Concepts - 6<sup>th</sup> Edition

22.30

©Silberschatz, Korth and Sudarshan

So, in this module to summarize your study aspects of database application architecture, understood the steps of rapid development, and took a quick look into the issues of application performance security, and what it takes to? What are the distinctive features of a mobile app for database applications?

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

**Lecture – 23**  
**Application Design and Development (Contd.)**

Welcome to module 23 of Database Management Systems. We have been discussing about application design and development and this is the third and concluding module in that regard.

(Refer Slide Time: 00:27)

The slide is titled "Module Recap" in red at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list of topics:

- Application Architectures
- Rapid Application Development
- Application Performance
- Application Security
- Mobile Apps

On the left side of the slide, there is vertical text that reads: "CHANDRAKANTAPURAM-NODEP-HOC", "Instructor: Prof. P. P. Das, IIT Kharagpur", "Module 23", "Date: Jan-Apr., 2018", and "Version: 1.0". At the bottom left, it says "Database System Concepts - 8<sup>th</sup> Edition". At the bottom center, it says "23.2". At the bottom right, it says "©Silberschatz, Korth and Sudarshan". The bottom of the slide has a standard Beamer navigation bar with icons for back, forward, search, and other slide controls.

In the last module we have discussed about different aspects of application architecture rapid development process issues and performance and security and took a glimpse in terms of, what is required for doing a mobile app.

(Refer Slide Time: 00:44)

The slide is titled "Module Objectives" in red at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bullet point: "■ To design the schema for a Library Information System". At the bottom left, there is a video thumbnail showing a person speaking. The bottom right corner features a navigation bar with icons for back, forward, search, and other presentation controls. The footer includes text: "Database System Concepts - 8<sup>th</sup> Edition", "23.3", and "©Silberschatz, Korth and Sudarshan". A vertical column on the far left contains the text: "CHAKRABORTY NOC MOOCs", "Instructor: Prof. P. P. Das, IIT Kharagpur", and "2018".

In this module, we will take care case study in terms of a library information system. We will try to design the schema for that as you have seen that for a database application design there are frontend designs there are middle tier business logic design and there will be issues in terms of the database design.

Since we are in the DBMS course, I will not focus on the whole aspects of application design, but we will focus specifically on the database aspect. So, we will start with a basic requirement specification for a library information system and then from the starting from that we will try to extract different entities and attributes and their relationships and we will make a relational schema and use different notions of dependency and how to write queries we will look into those aspects and refine that and finalize a schema for this problem. So, let us work through that. So, the outline the module issue is library information system.

(Refer Slide Time: 01:55)

The slide has a header 'Library Information System (LIS)' with a small sailboat icon to the left. In the top right corner, there is a red 'PPD' logo. The main content discusses an institute library with 200,000+ books and 10,000+ members. It details the need for an LIS to manage books, members, and the issue-return process. It specifies four categories of members: undergraduate students, post graduate students, research scholars, and faculty members. Each member has unique attributes like name, roll number, department, gender, mobile number, date of birth, and degree. Every member has a unique membership number. A maximum quota is set for each member based on their category. The library also issues a unique membership number to every member. The slide lists the following quotas:

- Each undergraduate student can issue up to 2 books for 1 month duration
- Each postgraduate student can issue up to 4 books for 1 month duration
- Each research scholar can issue up to 6 books for 3 months duration
- Each faculty member can issue up to 10 books for six months duration

The library has the following rules for issue:

A book may be issued to a member if it is not already issued to someone else (trivial). Also, a book may not be issued to a member if another copy of the same book is already issued to the same member. No issue will be done to a member if at the time of issue one or more of the books issued by the member has already exceeded its duration of issue. No issue will be allowed also if the quota is exceeded for the member.

Finally, it is assumed that the name of every author or member has two parts – first name and last name.

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

Database System Concepts - 8<sup>th</sup> Edition

23.5

©Silberschatz, Korth and Sudarshan

So, let us start with a basic this is a very small description. So,, but yet it will give us lot of food for thought in this work and while you actually go through the rest of the video. I would suggest that you take a print out of this page or keep this page separately because you will frequently need to refer to it.

So, let me quickly go through this we are talking about an institute library that has over 2 lakh books and over 10,000 members who can use regularly issue the books on loan and return them on the expiry of the period or before that and the library needs a library information system to manage the books the members and as well as the issue return process. So, we are just looking into these aspects not the procurement of books and organization of the books and so on.

Now what is given every book has it is title author publisher etcetera a number of different attributes ISBN number accession number in terms of a book I must explain to you at this stage that any book that you that is published has an ISBN number which is an international number given so, that you can uniquely identify that book.

So, if we are talking about the database management book we are following here then that has a ISBN number, but that does not mean that number actually is unique for the book would not for a specific copy of the book. So, if you buy three copies of the book and put it in the library, then these three copies need to be identify separately by another number which is typically called the accession number.

So, naturally the LIS need that every book has ISBN number which says which book it is and the accession number which says which specific copy it is, because they are may be multiple copies of the same book in the library on the member side. There are four categories of members broadly: students and teachers, but amongst students if they could be undergraduate postgraduate or research students and the faculty members. The student are specified by their name, roll number, department, gender, mobile number, date of birth, and the degree that they are doing and every faculty member also has a name, employee id, department, gender, mobile number.

And the date of joining the library to manage these members library also issues a unique membership number to every member and every member has a maximum quota for the number of books that she or he can issue and the maximum duration for which those books can be retain once issued. So, there are different specification for at different categories of member can have different quota and different duration the general rule as specified by this libraries the a book may be issued to a member naturally if it is not issued to someone else the book has to be available also a book may not be issued to a member if another copy of the same book is already issued to the same member.

So, you cannot issue two copies of the same book at the same time no issue will be done to a member if he is kind of a default that is the time of at the time of issue one or more of the other books already issued by the member has already exceeded the duration of the issue.

So, they are overdue for return in that case no issue will be allowed no issue will; obviously, be allowed if the quota exceeds and it is also specified that whenever we talk about name every name will have two parts the first name and the last name. So, this is the this is the given specification based on this we will need to make a good relational schema to represent the tables and manage the queries.

(Refer Slide Time: 05:45)

LIS Queries

LIS should support the following operations / query:

- Add / Remove members, categories of members, books.
- Add / Remove / Edit quota for a category of member, duration for a category of member.
- Check if the library has a book given its title (part of title should match). If yes: title, author, publisher, year and ISBN should be listed.
- Check if the library has a book given its author. If yes: title, author, publisher, year and ISBN should be listed.
- Check if a copy of a book (given its ISBN) is available with the library for issue. All accession numbers should be listed with issued or available information.
- Check the available (free) quota of a member.
- Issue a book to a member. This should check for the rules of the library.
- Return a book from a member.
- and so on

CHAKRABORTY, NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018

Database System Concepts - 8<sup>th</sup> Edition

23.6

©Silberschatz, Korth and Sudarshan

So, let us take a quick look into some of the sample queries this is just a indicative sample. Now, naturally we need a whole lot of you know insert delete update kind of queries for adding or removing members categories of members shapes the books and so, on adding or removing or changing the quota of a category of member the duration for that also we will have queries like to check.

If the library has a given book with a given title and if it is found then the details of those should be listed or we need to check if library has a book given it is author. So, if I say the author it should be possible to locate a book we should able to check, if a copy of a book if I specify the ISBN number, then whether it is available with the library for issue. So, there may be as I said multiple copies of the same book.

So, given the ISBN number it will return all the copies the accession number of all the copies and their issue status; whether they are issued or they are available it should be available it should be possible to check the quota available free quota of a member and certainly it should have features of issuing a book to a member and they should check for the rules of the library as stated it should be possible to return a book and so, on. So, these are the typical queries against which in the backdrop of which we will try to design the database system. So, what we will do?

(Refer Slide Time: 07:18)

**Entity Sets: books**

PPD

CHAKRABORTY, NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018

- Every book has title, author (in case of multiple authors, only the first author is maintained), publisher, year of publication, ISBN number (which is unique for the publication), and accession number (which is the unique number of the copy of the book in the library). There may be multiple copies of the same book in the library.
- Entity Set:
  - books
- Attributes:
  - title
  - author\_name (composite)
  - publisher
  - year
  - ISBN\_no

Database System Concepts - 8<sup>th</sup> Edition

23.7

©Silberschatz, Korth and Sudarshan

We will initially start with a specification as given. So, I hope you already have kept a copy to refer to and we will try to extract the different entity sets and the attributes. Naturally, the first entity said that we extract we let us call it books which is about books where in the beginning I have given the basic statement that is given in the so, in the specification.

So, from that we can see that books will be an entity set and it will have attributes like title author name which is a composite one, because it will have two parts into that the first name and the last name the publisher year ISBN number and accession number. So, these are the attributes available for books. So, this is my first entity set.

(Refer Slide Time: 08:11)

The slide is titled "Entity Sets: students" in red text at the top right. On the left, there is a small logo of a sailboat on water. The main content is a bulleted list:

- Every student has name, roll number, department, gender, mobile number, date of birth, and degree (undergrad, grad, doctoral).
- Entity Set:
  - **students**
- Attributes:
  - member\_no – is unique
  - name (composite)
  - roll\_no – is unique
  - department
  - gender
  - mobile\_no – may be null
  - dob
  - degree

At the bottom left, it says "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018". At the bottom center, it says "Database System Concepts - 8<sup>th</sup> Edition" and "23.8". At the bottom right, it says "©Silberschatz, Korth and Sudarshan".

The second entity set are students. So, this is the statement about the student and from that statement, we can easily extract that entity set here is student and the attributes are member number, because certainly in the context of the library system as we said the; student has to be a member. So, it should be there will be a member number which is has to be unique the composite name the student has a roll number. So, we assuming that the role number is a unique field.

So, knows to students will have the same roll number, then there is department gender mobile number which could be null the; date of birth degree that the student has done is doing. So, those are the different attributes for this entity set.

(Refer Slide Time: 09:03)

**Entity Sets: faculty**

- Every faculty has name, employee id, department, gender, mobile number, and date of joining.
- Entity Set:
  - **faculty**
- Attributes:
  - member\_no – is unique
  - name (composite)
  - id – is unique
  - department
  - gender
  - mobile\_no – may be null
  - doj

Database System Concepts - 8<sup>th</sup> Edition

23.9

©Silberschatz, Korth and Sudarshan

So, we have books we have students similarly we will have faculty again the there is this process will continue by extracting different parts from the specification. So, these are statement about the faculty and we know that there will be a faculty entity set with attributes like member number name id and so, on.

(Refer Slide Time: 09:25)

**Entity Sets: members**

- Library also issues a unique membership number to every member. There are four categories of members of the library: undergraduate students, post graduate students, research scholars, and faculty members.
- Entity Set:
  - **members**
- Attributes:
  - member\_no
  - member\_type (takes a value in ug, pg, rs or fc)

Database System Concepts - 8<sup>th</sup> Edition

23.10

©Silberschatz, Korth and Sudarshan

So, it should be quite obvious library has talked of that it can each it will issue unique membership number to every member and there are 4 categories of member. So, let us

we are just making attentive you know suggestion that there could be. So, it looks like members are in entity which the library has to interact with in terms of issue.

So, let us create an entity set members which has the member number and the member type. So, which can take different types of undergraduate post graduate these different one of these four specific values let us say.

(Refer Slide Time: 10:00)

**Entity Sets: quota**

- Every member has a maximum quota for the number of books she / he can issue for the maximum duration allowed to her / him. Currently these are set as:
  - Each undergraduate student can issue up to 2 books for 1 month duration
  - Each postgraduate student can issue up to 4 books for 1 month duration
  - Each research scholar can issue up to 6 books for 3 months duration
  - Each faculty member can issue up to 10 books for six months duration
- Entity Set:
  - quota
- Attributes:
  - member\_type
  - max\_books
  - max\_duration

Database System Concepts - 8<sup>th</sup> Edition

23.11

©Silberschatz, Korth and Sudarshan

The rules have talked about having a quota. So, every member will according to it is member category we will have different quota. So, to represent so, quota becomes an entity set. So, we would like to represent that for different member type which is a category; how many number of maximum books can be taken and what could be the maximum duration? So, let us say we will put the maximum duration say in terms of months and with these three attributes we will have an entity set which is quota.

(Refer Slide Time: 10:33)

**Entity Sets: staff**

- Though not explicitly stated, library would have staffs to manage the LIS.
- Entity Set:
  - **staff**
- Attributes: (speculated – to ratify from customer)
  - name (composite)
  - id – is unique
  - gender
  - mobile\_no
  - doj

CHIEF MINISTER NOC MOOCs Instructor: Prof. P. P. Des., IIT Kharagpur Date: Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition

23.12

©Silberschatz, Korth and Sudarshan

Here, I am talking about another entity sets staff if you again carefully read the specification you will find that there is no mention of this staff in the in the total of the specification, but if we logically think and this is what we often need to do. When we deal with a practical specification like somewhat like, the one that I have given here is that we would need to look little beyond the specification.

So, think about it if I have the library with books students faculty issue process quota and all that then certainly they will have to be some staff of the library; that will actually do all this operation. So, they will be able to log in to the database and actually issue a book return a book check for validity and so, on.

So, let us assume that we have a entity set staff which certainly a staff should have name and id which id should be unique, gender, mobile number and date of joining. So, this is kind of a speculative addition to the design of entity sets and this must be ratified from the customer; when the opportunity arise, but something like that must be there for to make the design complete.

(Refer Slide Time: 11:48)

The slide has a header 'Relationships' in red. On the left, there's a small logo of a sailboat on water. On the right, it says 'PPD'. The main content is a bulleted list:

- Books are regularly issued by members on loan and returned after a period. The library needs an LIS to manage the books, the members and the issue-return process.
- Relationship
  - book\_issue
- Involved Entity Sets
  - students / faculty
    - ▶ member\_no
  - books
    - ▶ accession\_no
- Relationship Attribute
  - doi – date of issue
- Type of relationship
  - Many-to-many

At the bottom, there's vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P.P. Desai, IIT Kanpur Date: Jan-Apr- 2018'. Below the slide are standard presentation navigation icons and the text 'Database System Concepts - 8<sup>th</sup> Edition' and '23.13 ©Silberschatz, Korth and Sudarshan'.

So, these are ah; obviously, the immediately visible entity sets that we see. So, the other part that we will now have to check on is a relationship. So, again we pick up the relevant statement in this regard books are regularly issued by members on loan and returned after a period the library needs an LIS to manage the books members and the issue return process.

So, certainly there will have to be a relationship of issue between the student or faculty in the books. So, we loosely define this relationship on one side there has to be the books and the other side would be the involved entity set would be either student or faculty so, actually though I am just noting it here as a as a single relationship, but actually there is a relationship of issue between students and books and faculty and books.

So, we will have to see how to handle these kind of situation now certainly the students or faculty are identified by the unique member number of the library that has been given books as we have said are identified by the accession number. Again, please note that we are not talking about the ISBN number, because ISBN number could be same for multiple copies of the book, but when you issue you issue a specific copy. So, that accession number which is a unique for a specific copy needs to be tracked. So, these are the two attributes, which will certainly be involved in the relationship and then you would recall that often relationships of their own attributes.

So, here we have one the date of issue needs to be recorded, because we want to check conditions like; if the borrower is has issued the book and how many for how many months he or she is keeping that book. So, if you have to check that we need to know when the book was issued. So, this is a attribute which does not exist in any of the entity sets that are involved in this relationship neither in students or faculty nor in books, but this is a attribute of the relationship which needs to be specified.

And of course, it is this relationship is a many to many relation; because every student or faculty can issue multiple books and every book may be issued by different, but we can we can in general we can be specific to say that this is many to one also if we want to maintain that at a any given instant a book can be issued only by one person. So, if you look at it from that perspective this will not be many to many this will be treated as many to one.

(Refer Slide Time: 14:24)

**Relational Schema**

- books(title, author\_fname, author\_lname, publisher, year, ISBN\_no, accession\_no)
- book\_issue(members, accession\_no, doi)
- members(member\_no, member\_type)
- quota(member\_type, max\_books, max\_duration)
- students(member\_no, student\_fname, student\_lname, roll\_no, department, gender, mobile\_no, dob, degree)
- faculty(member\_no, faculty\_fname, faculty\_lname, id, department, gender, mobile\_no, doj)
- staff(staff\_fname, staff\_lname, id, gender, mobile\_no, doj)

NPTEL MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018

Database System Concepts - 8<sup>th</sup> Edition

23.14

©Silberschatz, Korth and Sudarshan

So, having done this finding having found out this entity an attributes and the relationships let us now start with a relational schema.

So, what I have done here; in terms of the relational schema that for each of the entity set. I have created a relational schema by the same name and have put the attributes as identified as attributes of that relational schema. So, this is a very straight forward once we have identification, made this identification process from the original specification. This is a more straight forward process where you just convert every entity set into a

relational schema and also we have converted the relationship there was a there is a relationship called book issue here as you can see the book issue this also we have converted to a relational schema involving the two attributes of member number and the accession number.

So, let us see how this will span out later parts so, having done this.

(Refer Slide Time: 15:38)

The screenshot shows a presentation slide titled "Schema Refinement". The slide content is as follows:

- books(title, author\_fname, author\_lname, publisher, year, ISBN\_no, accession\_no)
  - ISBN\_no → title, author\_fname, author\_lname, publisher, year
  - accession\_no → ISBN\_no
  - Key: accession\_no
- Redundancy of book information across copies
- Good to normalize:
  - book\_catalogue(title, author\_fname, author\_lname, publisher, year, ISBN\_no)
    - ▶ ISBN\_no → title, author\_fname, author\_lname, publisher, year
    - ▶ Key: ISBN\_no
  - book\_copies(ISBN\_no, accession\_no)
    - ▶ accession\_no → ISBN\_no
    - ▶ Key: accession\_no
- Both in BCNF. Decomposition is lossless join and dependency preserving

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jun-Apr. 2018

The next task would be to work on the refinement of the schema. So, now, it is time that the relational schema is available the first schema is available. So, now, we have to apply all different notions of the relational design to refine this schema and finalize. So, for what will do; we will take every relational schema and we will try to identify the functional dependencies and use that to identify the key.

So, if I look at the books relational schema, then we easily know that **ISBN number → title, author first name; authors last name, publisher, year**. So, here also you may just note that since name is stated to be a composite attribute I have designed here to use two different attributes the first name and the last name. So, with that we have this functional dependency which tells me that given ISBN number, I know these details, but of course, that does not tell me what is the accession number because there could be multiple copies, but another functional dependency must hold because, every copy of the same book must have the same ISBN number. So, given the accession number, ISBN number should be determinable.

So, **accession number → ISBN number**. So, if you do the sample computation on this you will easily figure out that a key of this is accession number. So, this is what we currently have this this is what we have done now. So, you can see that if there are say five copies of a book having different accession numbers which are the key they are has been number would may all be same.

So, if I have the same ISBN number I can multiple accession numbers and therefore, there are different records, but if that accession number of two books are I am sorry the ISBN number of two books are same then all of that title, author and first name last name all this attributes will be repeated and if there are multiple copies of the book then each one of them will have a separate entry because they have a separate accession number.

But, their ISBN number and everything else will be redundant. So, we can easily see that the given relational schema actually has redundancy across copies of the book and if we want to look at this from formal theory we can easily check that the keys accession number. So, **ISBN number → title, author, .... etcetera** that functional dependency violates the boyce code normal form actually this is the also not in the third normal form right now.

So, we would do well to reduce this redundancy and it would be good to normalize. So, here I have not gone through the actual steps of normalization, but I am showing you more intuitively as to, how you can normalize; because it is a quite obvious that the accession number is involved only with the ISBN number and which keeps track of the copies.

So, we propose to have under normalization we propose to have one which is let me just highlight and show you. So, book copies where the ISBN number and accession number will be maintained. So, for every accession number we will be able to see the ISBN number which will tell you which book it is and rest of the book details will be kept in this say another new relational schema called book catalogue which will have all of the earlier attributes expect the accession number.

So, now, if you project the; this dependency on book catalogue the dependency will be fully projected. So, the whole dependency is preserved if you project this dependency on the book copies it will also be fully preserved. So, this decomposition is a dependency

preserving and you can easily check that these two can be joint by a natural join using ISBN number as the common attribute.

So, if we take the intersection of the two sets of attributes then ISBN number would be the attribute and **ISBN number → all attributes** in the book catalogues. So, our lossless join condition  $r_1 \cap r_2 \rightarrow r_1$  here. So, this will also give me a lossless join and if you check on each one of these relational schema then this functional dependency the left hand side is a key and therefore, book catalogue is in Boyce Codd normal form and book copies is also in Boyce Codd normal form.

So, you can you could come to the same result by doing the formal process of functional Boyce Codd normal form decomposition, but I have just shown you here as to intuitively how you get this. So, intuitively becomes very clear that you have details of the book that you keep separate in a separate table, because that is not change across the copies and we have a separate table to maintain the specific information about the copies. So, that takes care of the books relationship moving on let us look at the other.

(Refer Slide Time: 21:16)

The slide is titled "Schema Refinement". It contains the following text:

- **book\_issue(member\_no, accession\_no, doi)**
  - member\_no, accession\_no → doi
  - Key: members, accession\_no

On the left side, there is a vertical footer with the following text:  
SWAYAM: NPTEL-NOC  
MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr, 2018

At the bottom of the slide, there is a set of navigation icons and the following metadata:  
Database System Concepts - 8<sup>th</sup> Edition      23.16      ©Silberschatz, Korth and Sudarshan

So, book issue is a relational schema which comes from the relationship between the now between what now book issue is to happen between student faculty and books. So, it is I mean we cannot have two of them of these entity sets occurring in the same. So, initially let us just put that well the library has a concept of a member. So, what if the book issue is a relation simply between the members and the books of course, we

do not know the relationship between members and students or members and faculty, but we can at least refine the book issue to be between member and the book and therefore, member number and accession number together becomes the key which → the date of issue this is already in normal form as you can see.

(Refer Slide Time: 22:11)

The slide has a title 'Schema Refinement' in red at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list:

- **quota(member\_type, max\_books, max\_duration)**
  - $\text{member\_type} \rightarrow \text{max\_books, max\_duration}$
  - Key: `member_type`

At the bottom left, it says 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018'. At the bottom center, it says 'Database System Concepts - 8<sup>th</sup> Edition' and '23.17'. At the bottom right, it says '©Silberschatz, Korth and Sudarshan'.

Quota is a simple relational schema where member type → the max books and duration and that is the key in normal form

(Refer Slide Time: 22:19)

The slide has a title 'Schema Refinement' in red at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list:

- **members(member\_no, member\_type)**
  - $\text{member\_no} \rightarrow \text{member\_type}$
  - Key: `member_no`
  - Value constraint on `member_type`
    - ▶ ug, pg, or rs: if the member is a student
    - ▶ fc: if the member is a faculty
  - How to determine the `member_type`?

At the bottom left, it says 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018'. At the bottom center, it says 'Database System Concepts - 8<sup>th</sup> Edition' and '23.18'. At the bottom right, it says '©Silberschatz, Korth and Sudarshan'.

The members I mean we have identified something like a member which should list all the members of the library.

So, existence of a member number in that list will mean that someone holding that member number is actually a member and it is should also tell me what type of member or what category of member he or she is. So, member number must → the member type and the member type will be constrained in terms of possible 4, 1 of the four possible values are stated here now of course, we will still have to figure out is to where do we get this member type from and so, on and that information is not present here. So, further refinements will be required in this regard.

(Refer Slide Time: 23:02)

The slide has a header 'Schema Refinement' and a small logo of a sailboat in the top left. In the top right corner, there is a small red 'PPD' icon. On the left side, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kanpur - Jan-Apr-2018'. On the right side, there is a video frame showing a man with glasses and a blue background. Below the video frame is a toolbar with various icons. At the bottom, there is footer text: 'Database System Concepts - 8<sup>th</sup> Edition', '23.19', and '©Silberschatz, Korth and Sudarshan'.

**■ students(member\_no, student\_fname, student\_lname, roll\_no, department, gender, mobile\_no, dob, degree)**

- roll\_no → student\_fname, student\_lname, department, gender, mobile\_no, dob, degree
- member\_no → roll\_no
- roll\_no → member\_no
- 2 Keys: roll\_no | member\_no

**■ Issues:**

- member\_no is needed for issue / return queries. It is unnecessary to have student's details with that.
- member\_no may also come from **faculty** relation.
- member\_type is needed for issue / return queries. This is implicit in degree – not explicitly given.

Let us go to the students this is the whole schema that we had done. So, **roll number → all the attributes** specifically **roll number → member number**, **member number → roll number**, because every student has a unique roll number and; obviously, has a unique member number also **number → all attrib.** So, **member number → roll number** **roll number → member number** and so **roll number → rest of attributes.**

Which mean that this design this relational schema has two keys the roll number and the member number now are we happy with this design that is a question we need to ask. So, we can figure out that member number is needed for issue return or queries book issue has member number, that is; what we were thinking of; now when we want to deal with

this book issue issuing a book to a student and returning a book from the student and soon.

Is it necessary to have all the students details with that one that make every record very heavy and if we want to extra connect with that do some join or if you want to do some query unnecessarily we will have to deal with very large units of data and as we will see in the next couple of modules that if a record becomes larger dealing with it become naturally becomes more cumbersome. So, there is some kind of discomfort at this design similarly a member number is not here it is the student relation. So, it is coming from the student relation, but in general in terms of issue it may also come from faculty relations.

So, how is that handled we do not know then issue return also needs the member type which is implicit here in terms of the field in terms of the attribute degree which tells you that student is a undergraduate postgraduate or research, but it is not explicitly maintained anywhere. So, these are the issues in this form of the design that we that we came.

(Refer Slide Time: 25:06)

The slide has a header 'Schema Refinement' in red. On the left, there is a small logo of a sailboat on water. On the right, there is a 'PPD' watermark. The main content is organized into two sections:

- faculty**(member\_no, faculty\_fname, faculty\_lname, id, department, gender, mobile\_no, doj)
  - id → faculty\_fname, faculty\_lname, department, gender, mobile\_no, doj
  - id → member\_no
  - member\_no → id
  - 2 Keys: id | member\_no
- Issues:**
  - member\_no is needed for issue / return queries. It is unnecessary to have faculty details with that.
  - member\_no may also come from **students** relation.
  - member\_type is needed for issue / return queries. This is implicit by the fact that we are in faculty relation.

At the bottom, there is a footer with the text 'SWAYAM: NPTEL-NOC's Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '23.20', and '©Silberschatz, Korth and Sudarshan'. There is also a set of navigation icons at the bottom right.

Moving on look at the faculty we have a very similar situation as of the **student id → all attributes** of the faculty , **id → member\_number**, **member number → id** every faculty has two unique numbers two keys. And we have similar type of issues as we have seen in terms of student. So, we will need to do something in terms of this.

(Refer Slide Time: 25:34)

The slide has a header 'Schema Refinement' and a small logo of a sailboat in the top left corner. On the right, there is a small video window showing a person speaking. The main content is a bulleted list under the heading 'Consider a query:'.

- Consider a query:
  - Get the name of the member who has issued the book having accession number = 162715
    - \* If the member is a student,
      - SELECT student\_fname as First\_Name, student\_lname as Last\_Name
      - FROM students, book\_issue
      - WHERE accession\_no = 162715 AND book\_issue.member\_no = students.member\_no;
    - \* If the member is a faculty,
      - SELECT faculty\_fname as First\_Name, faculty\_lname as Last\_Name
      - FROM faculty, book\_issue
      - WHERE accession\_no = 162715 AND book\_issue.member\_no = faculty.member\_no;
    - Which query to fire!

At the bottom, there is a note: 'These are sample indicative code not checked for syntax.' and some navigation icons. The footer includes 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kanpur', 'Database System Concepts - 8<sup>th</sup> Edition', '23.21', and '©Silberschatz, Korth and Sudarshan'.

So, what can us; I mean what kind of issues we will get into. So, let us consider a query that the query is to get the name of a member of the member who has issued a book say having accession number some accession number 162715.

Now, if we have to write a select query for this we will need to know whether the select query should be written on the student or with the faculty. So, if this member is a student then we need the first query where we do a join between student and book issue to find out the student member number who is issued that book where the accession number gives me the particular book issue record from this result will come. Similarly, if the member is a faculty I need a different query. Now, it is a problem because if I have two possible queries and based on the member number I have to decide which query to fire we have not done any mechanism like that. So, this design has problems.

(Refer Slide Time: 26:36)

The slide has a header 'Schema Refinement' and a small logo of a sailboat in the top left. On the right, there is a small video window showing a man speaking. The main content discusses the refinement of a 'members' entity set based on member categories: undergraduate students, post graduate students, research scholars, and faculty members. It lists attributes like member\_no, member\_class (student or faculty), member\_type (ug, pg, rs, fc), roll\_no, and id. It also mentions hidden relationships between student and faculty members and the type of relationship (one-to-one).

There are four categories of members of the library: undergraduate students, post graduate students, research scholars, and faculty members. This leads to the following specialization relationships.

- Consider the entity set **members** that represent the behavior of the member of a library and refine:
  - Attributes:
    - ✖ member\_no
    - ✖ member\_class – ‘student’ or ‘faculty’, used to choose table
    - ✖ member\_type – ug, pg, rs, fc, ...
    - ✖ roll\_no (if member\_class – ‘student’. Else null)
    - ✖ id (if member\_class – ‘faculty’. Else null)
  - We can then exploit some hidden relationship:
    - students IS\_A members
    - faculty IS\_A members
  - Type of relationship
    - One-to-one

So, let us see what we can do? So, we again go back to the specification and see what the specification says; it said that there are four categories of members undergraduate, postgraduate, research scholar, and faculty members and least to the least to the specialization of this relationship and we have already done a members concept members entity we did.

Which has a member number and the member type, but now we have just seen that it actually matters as to whether the member is a student or is a faculty is a more unique deciding factor in terms of, how we design our query? So, we introduce a new attribute which was not specified explicitly say; let us call it member class which can take only two values either student or faculty and then retain the member type. So, what it will mean that the; if the student class is student then the member type could be ug, pg or rs and if the member class is faculty.

Then the person is a faculty than the member type should be only fc and then also maintain the roll number and id in this members table. So, the roll number ah; obviously, if it is if the member class is student then the person is a student and the roll number will exist, but the id which is an employee id will be null and at the same time if member class is a faculty for a record then the roll number will be null because, a faculty cannot have a roll number, but the id will be present.

So, we extend the members entity set in relational schema with these attributes and once we do that then we kind of exploit a hidden relationship which was not very explicitly stated anywhere that; now we can say that students is a members faculty is a members, that is; from the perspective of issuing books and using the library both students and faculty can be considered to be members it is kind of a you know virtual entity that we can see here and certainly there will be a one to one relationship between the students and members and faculty and members.

(Refer Slide Time: 28:39)

**Schema Refinement**

- Consider the old query again:
  - Get the name of the member who has issued the book having accession number = 162715

```

SELECT
  ((SELECT faculty_fname as First_Name, faculty_lname as Last_Name
  FROM faculty
  WHERE member_class = 'faculty' AND members.id = faculty.id)
UNION
  (SELECT student_fname as First_Name, student_lname as Last_Name
  FROM students
  WHERE member_class = 'student' AND members.roll_no = students.roll_no))
FROM members, book_issue
WHERE accession_no = 162715 AND book_issue.member_no = members.me
  
```

These are sample indicative code not checked for syntax errors.

Database System Concepts - 8<sup>th</sup> Edition      23.23      ©Silberschatz, Korth and Sudarshan

So, let us see what is a consequence of that; in terms of the earlier query. So, we look at go back and look at the query again we will see that problems have got significantly solved, because we now still have to find that member name and this is the basic condition which say.

But, I am sorry this is a basic condition which says the specify the member number who has issued that book, but the actual problem of finding the name can be solved here, because I can write two queries and take a union of the first query runs on faculty the other query runs on student. Now, here since I have a member class which tell me whether the member is a faculty or the member is a student. So, one of these queries will actually return a null result will not return any record because it will not match this condition, but the other one we will match and will give me the record give me the corresponding names first name and last name of the member and certainly to ensure that

when we are looking at the faculty. We need to check the equality of member id between the members relation and the faculty relation and while dealing with the student we need to check for the equality of the roll number.

So, in this way by using intelligently using the union feature and the you know nested query feature we can easily write a query and where implicitly. Now, based on the data the switching of which table I am I am actually looking the data from will get solved.

(Refer Slide Time: 30:17)

The screenshot shows a presentation slide with the title "Schema Refinement" in red at the top center. To the left of the title is a small icon of a sailboat on water. On the right side, there is a small video window showing a man with glasses and a blue shirt, likely the lecturer. The main content area contains a bulleted list under the heading "members(member\_no, member\_class, member\_type, roll\_no, id)". The list includes:

- members(member\_no, member\_class, member\_type, roll\_no, id)
  - member\_no → member\_type, member\_class, roll\_no, id
  - member\_type → member\_class
  - Key: member\_no

At the bottom of the slide, there is some footer text: "SWAYAM: NPTEL-NOC's Instructional Platform", "Prof. P. P. Das, IIT Kharagpur - Jan-Apr, 2018", "Database System Concepts - 8th Edition", "23.24", and "©Silberschatz, Korth and Sudarshan".

So, with this refinement my members schema now turns out to be **member number → member class, member type, roll number, and id member, number** and **member type → member class**, because if I know some member type is undergraduate I know member class is student and so, on key; obviously, is one number.

(Refer Slide Time: 30:38)

The slide has a header 'Schema Refinement' and a footer with the text 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr. 2018'.

**■ students**(student\_fname, student\_lname, roll\_no, department, gender, mobile\_no, dob, degree)

- roll\_no → student\_fname, student\_lname, department, gender, mobile\_no, dob, degree
- Keys: roll\_no
- Note:
  - ▶ member\_no is no longer used
  - ▶ member\_type and member\_class are set in **members** from degree at the time of creation of a new record.

**■ faculty**(faculty\_fname, faculty\_lname, id, department, gender, mobile\_no, doj)

- id → faculty\_fname, faculty\_lname, department, gender, mobile\_no, doj
- Keys: id
- Note:
  - ▶ member\_no is no longer used
  - ▶ member\_type and member\_class are set in **members** at the time of creation of a new record

PPD

Database System Concepts - 8<sup>th</sup> Edition      23.25      ©Silberschatz, Korth and Sudarshan

So having done that, we again go back to the student and faculty, because now we do not need member number in that anymore; because these will not directly be involved in the issue return, because all that you need is just the member number which is already there in the members. So, now, it is a roll number which determines everything member number is no longer used.

And member type and member class which we have assumed is exist in the in the members will have to be derived from the degree value at the time when a new record is created. So, when a new student record is created an entry will happen in this relation as well as a corresponding entry we will need happen in the members relation where the membership number is given and the membership type will be derived from the degree similar change will happen in terms of the faculty as well.

(Refer Slide Time: 31:31)

The slide title is "Schema Refinement – Final". The content lists eight relational schemas:

- book\_catalogue(title, author\_fname, author\_lname, publisher, year, ISBN\_no)
- book\_copies(ISBN\_no, accession\_no)
- book\_issue(member\_no, accession\_no, doi)
- quota(member\_type, max\_books, max\_duration)
- members(member\_no, member\_class, member\_type, roll\_no, id)
- students(student\_fname, student\_lname, roll\_no, department, gender, mobile\_no, dob, degree)
- faculty(faculty\_fname, faculty\_lname, id, department, gender, mobile\_no, doj)
- staff(staff\_fname, staff\_lname, id, gender, mobile\_no, doj)

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

Database System Concepts - 8<sup>th</sup> Edition 23.26 ©Silberschatz, Korth and Sudarshan

So, after refinement now we have these eight relational schema from book catalogue to give the details of every book copies which keeps the information about, how many; what are the different copies book issue; is the basic issuing information. We have quota members has the virtual kind of relation that we have created to support the notion of members of the library and students and faculty are related to this members either through roll number or through id in a selective manner and staff we have not touched.

(Refer Slide Time: 32:12)

The slide title is "Module Summary". The content lists two main points:

- Using the specification for a Library Information System, we have illustrated how a schema can be designed and then refined for finalization
- Coding of various queries based on these schema are left as exercises

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

Database System Concepts - 8<sup>th</sup> Edition 23.27 ©Silberschatz, Korth and Sudarshan

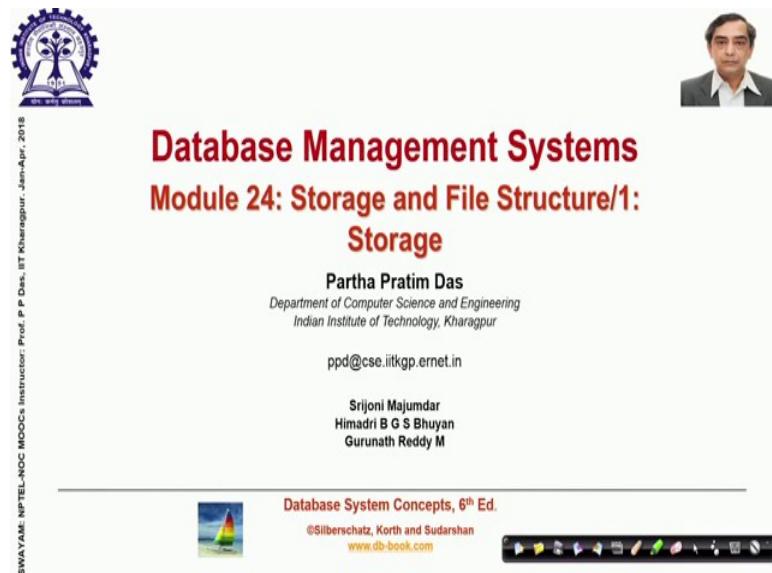
So, this is the final relational schema. In this module I have shown you illustrated you that how starting from a very simple specification you can reason and work through in terms of creating the entity sets attributes and relationships and then get into the relational schema look at the possible functionality dependency and refine that and also look into what will be the requirements of doing your query and come to a final refined schema.

So, various queries that we had talked of and others can be can now be coded on this, but I leave that as an exercise to you please try out coding those queries and you will be able to learn in terms of how to do that well and I will try to also supply some supplementary information on this offline.

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

**Lecture – 24**  
**Storage and File Structure: Storage**

(Refer Slide Time: 00:16)



The slide is titled "Database Management Systems" in red, followed by "Module 24: Storage and File Structure/1: Storage". It features a portrait of Prof. Partha Pratim Das on the right. The footer includes the book title "Database System Concepts, 6<sup>th</sup> Ed.", authors "Silberschatz, Korth and Sudarshan", and the website "www.db-book.com". A decorative footer bar with various icons is also present.

SWAYAM-NPTEL-NOOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur. Date: Apr. 2018

**Database Management Systems**  
**Module 24: Storage and File Structure/1:**  
**Storage**

**Partha Pratim Das**  
Department of Computer Science and Engineering  
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

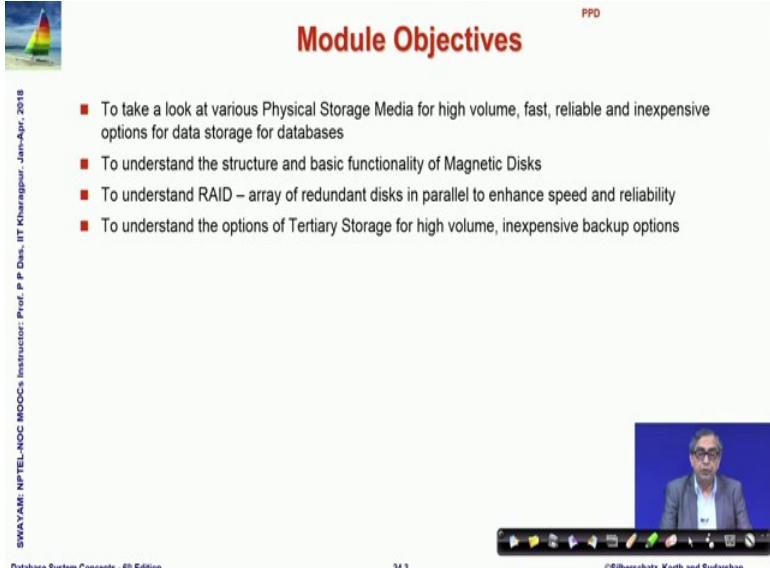
Srijoni Majumdar  
Himadri B G S Bhuyan  
Gurunath Reddy M

---

Database System Concepts, 6<sup>th</sup> Ed.  
©Silberschatz, Korth and Sudarshan  
[www.db-book.com](http://www.db-book.com)

Welcome to module 24 of database management systems in this module and the next we will take a look at the storage and file structure of database systems. So, we will start with the storage.

(Refer Slide Time: 00:30)



This slide is titled "Module Objectives" in red at the top right. It features a small sailboat icon in the top left corner and a video player window showing a man speaking on the right. The text on the slide lists four objectives:

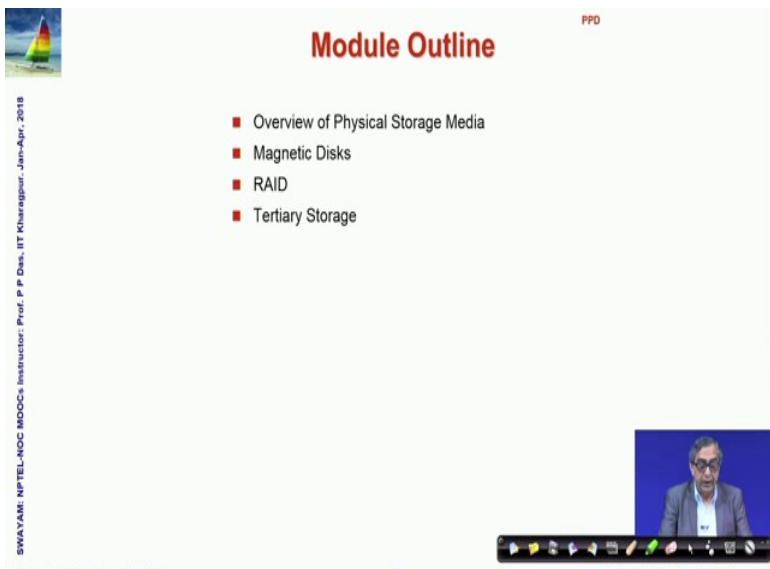
- To take a look at various Physical Storage Media for high volume, fast, reliable and inexpensive options for data storage for databases
- To understand the structure and basic functionality of Magnetic Disks
- To understand RAID – array of redundant disks in parallel to enhance speed and reliability
- To understand the options of Tertiary Storage for high volume, inexpensive backup options

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

Database System Concepts - 6<sup>th</sup> Edition 24.3 ©Silberschatz, Korth and Sudarshan

So, specifically we want to look at various physical storage medium because. So, far we have been talking only about the logical layer of the database design and now we want to actually look at the in physical terms how the data will be stored what could be the physical storage medium for high volume fast reliable inexpensive options for databases. We would like to understand the structure and basic functionality of magnetic disks, because they are they are most widely used we will try to take the glimpse about RAID which is a kind of a good option in terms of reliable databases and also look at options for the tertiary storage.

(Refer Slide Time: 01:12)



This slide is titled "Module Outline" in red at the top right. It features a small sailboat icon in the top left corner and a video player window showing a man speaking on the right. The text on the slide lists five topics:

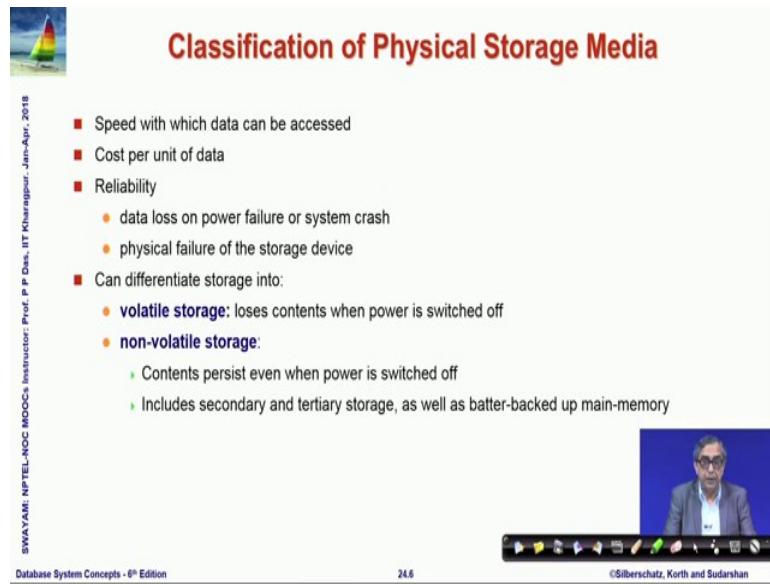
- Overview of Physical Storage Media
- Magnetic Disks
- RAID
- Tertiary Storage

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

Database System Concepts - 6<sup>th</sup> Edition 24.4 ©Silberschatz, Korth and Sudarshan

So, these are the topics that we will quickly cover in this.

(Refer Slide Time: 01:17)



The slide has a title 'Classification of Physical Storage Media' in red at the top right. On the left is a small sailboat icon. The main content is a bulleted list of factors for classification:

- Speed with which data can be accessed
- Cost per unit of data
- Reliability
  - data loss on power failure or system crash
  - physical failure of the storage device
- Can differentiate storage into:
  - **volatile storage**: loses contents when power is switched off
  - **non-volatile storage**:
    - ↳ Contents persist even when power is switched off
    - ↳ Includes secondary and tertiary storage, as well as battery-backed up main-memory

At the bottom, there is a small video frame showing a man speaking, the text 'Database System Concepts - 8<sup>th</sup> Edition', the page number '24.6', and the copyright '©Silberschatz, Korth and Sudarshan'.

So, first let us take a overview of the physical storage medium I am sure all of you have known all or parts of this. So, this is, but this is more for completeness to look at from the perspective of a database application. So, some of the the classification of storage media done on different factors the factors includes speed which is the first thing the how fast the data can be accessed the cost per unit of data you can say the rupees per bit or rupees per byte or rupees per kilobyte something like that.

So, which is a cost per unit of data the reliability that is the if we will the data get lost if power fails or if the system crashes and or if this physical you know failure of the storage device and so, on. So, what is the reliability on that; and broadly as you all know we can differentiate storage into volatile storage which loses contents. When the power is switched off and the non volatile storage which are secondary and tertiary storage where the data will continue to stay even when power is off even you have some parts of the memory which may be battery backup which also will be non volatile.

(Refer Slide Time: 02:29)

The slide has a header 'Physical Storage Media' with a sailboat icon. It contains three main sections: 'Cache', 'Main memory', and 'Volatile'. The 'Cache' section lists: fastest and most costly form of storage, volatile, managed by the computer system hardware. The 'Main memory' section lists: fast access (10s to 100s of nanoseconds; 1 nanosecond =  $10^{-9}$  seconds), generally too small (or too expensive) to store the entire database, capacities of up to a few Gigabytes widely used currently, Capacities have gone up and per-byte costs have decreased steadily and rapidly (roughly factor of 2 every 2 to 3 years). The 'Volatile' section lists: contents of main memory are usually lost if a power failure or system crash occurs. The footer includes 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '24.7', and '©Silberschatz, Korth and Sudarshan'.

So, in terms of the physical storage certainly the absolute starting point of the storage is registered in the CPU; we are not talking about that because they are primarily meant for temporary computations. So, in terms of data the first possible level is the cache which is the fastest and most costly form of storage it is volatile in nature and is managed by the computer system hardware.

So, cache typically is a fast semiconductor memory that exist between your main memory and the disk system and it is very fast to work with then comes the main memory which is which has fast access, but compare to cache it may be may be much bigger, but overall it is too small to store an entire database, but I mean every regularly the size of this main memory is increasing. So, the capacity of couple of gigabytes are common these days, but still it is small compare to the requirement of the databases and main memory typically is volatile. So, if the power goes up the system crashes all the data is lost.

(Refer Slide Time: 03:48)

The slide has a title 'Physical Storage Media (Cont.)' at the top right. On the left, there is a small logo of a sailboat on water. The main content is a bulleted list under the heading 'Flash memory'. The list includes:

- Data survives power failure
- Data can be written at a location only once, but location can be erased and written to again
  - ↳ Can support only a limited number (10K – 1M) of write/erase cycles
  - ↳ Erasing of memory has to be done to an entire bank of memory
- Reads are roughly as fast as main memory
- But writes are slow (few microseconds), erase is slower
- Widely used in embedded devices such as digital cameras, phones, and USB keys

At the bottom left, it says 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur - Jan-Apr. 2018'. At the bottom center, it says 'Database System Concepts - 8<sup>th</sup> Edition' and '24.8'. At the bottom right, it says '©Silberschatz, Korth and Sudarshan'.

We have flash memory where the data can survive across power failure; it can be they had data can be written at a location only once, but you can erase and write it again.

So, it is not like in the main memory where you can read write read write like that here you can write and then if you want to write again then you will have to erase and write it. So, the read is very fast in case of flash memory which is almost as fast as a main memory, but writes are slow particularly when you have erase and write it will be a slow process all the kinds of USB keys pen drives digital phone memory that we are often using are actually flash memory.

(Refer Slide Time: 04:35)

The slide has a header 'Physical Storage Media (Cont.)' with a sailboat icon. On the left, there is vertical text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018'. The main content is a bulleted list under 'Magnetic-disk':

- **Magnetic-disk**
  - Data is stored on spinning disk, and read/written magnetically
  - Primary medium for the long-term storage of data
    - typically stores entire database
  - Data must be moved from disk to main memory for access, and written back for storage
    - Much slower access than main memory
  - **direct-access**
    - possible to read data on disk in any order, unlike magnetic tape
  - Capacities range up to roughly 16~32 TB
    - Much larger capacity and cost/byte than main memory/flash memory
    - Growing constantly and rapidly with technology improvements (factor of 2 to 3 every 2 years)
  - Survives power failures and system crashes
    - disk failure can destroy data, but is rare

On the right, there is a video player window showing a man speaking, and at the bottom, there are navigation icons and the text 'Database System Concepts - 8<sup>th</sup> Edition' and '24.9 ©Silberschatz, Korth and Sudarshan'.

Then you have the magnetic disk where the data is stored on spinning disk and it is typically written and read magnetically. So, this is the primary medium for long term storage of large volume of data. So, data needs to be moved from disk to the main memory and written back for permanent storage. So, it is ways slower compare to the main memory and, but it is has a kind of direct access which means that it is possible to read data on this disk in any arbitrary order in compare to magnetic disk.

Which is the serial device here it is a, it is kind of a I can do things in parallel at random in any order capacity is go up to tens of terabytes easily and it can survive for failure and system crashes, because it will the magnetic recording will still be there if the disk itself fails then it will the data will get distract, but such a situation is usually rear.

(Refer Slide Time: 05:40)

The slide is titled "Physical Storage Media (Cont.)" and features a small sailboat icon in the top left corner. On the right side, there is a video player window showing a man speaking. The video player has a blue background and includes standard controls like play, pause, and volume. The main content area contains a bulleted list under the heading "Optical storage".

**Optical storage**

- non-volatile, data is read optically from a spinning disk using a laser
- CD-ROM (640 MB) and DVD (4.7 to 17 GB) most popular forms
- Blu-ray disks: 27 GB to 54 GB
- Write-one, read-many (WORM) optical disks used for archival storage (CD-R, DVD-R, DVD+R)
- Multiple write versions also available (CD-RW, DVD-RW, DVD+RW, and DVD-RAM)
- Reads and writes are slower than with magnetic disk
- Juke-box systems, with large numbers of removable disks, a few drives, and a mechanism for automatic loading/unloading of disks available for storing large volumes of data

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018

Database System Concepts - 8<sup>th</sup> Edition

24.10

©Silberschatz, Korth and Sudarshan

We have different optical storage devices CD-ROM, DVD and so, on the juke box systems and which are also non volatile and data is written optically here, that is by using a laser light on the spinning disk use a typically optical storages are removable media.

(Refer Slide Time: 06:03)

The slide is titled "Physical Storage Media (Cont.)" and features a small sailboat icon in the top left corner. On the right side, there is a video player window showing a man speaking. The video player has a blue background and includes standard controls like play, pause, and volume. The main content area contains a bulleted list under the heading "Tape storage".

**Tape storage**

- non-volatile, used primarily for backup (to recover from disk failure), and for archival data
- sequential-access**
  - much slower than disk
- very high capacity (40 to 300 TB tapes available)
- tape can be removed from drive  $\Rightarrow$  storage costs much cheaper than disk, but drives are expensive
- Tape jukeboxes available for storing massive amounts of data
  - hundreds of terabytes (1 terabyte =  $10^{12}$  bytes) to even multiple **petabytes** (1 petabyte =  $10^{15}$  bytes)

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018

Database System Concepts - 8<sup>th</sup> Edition

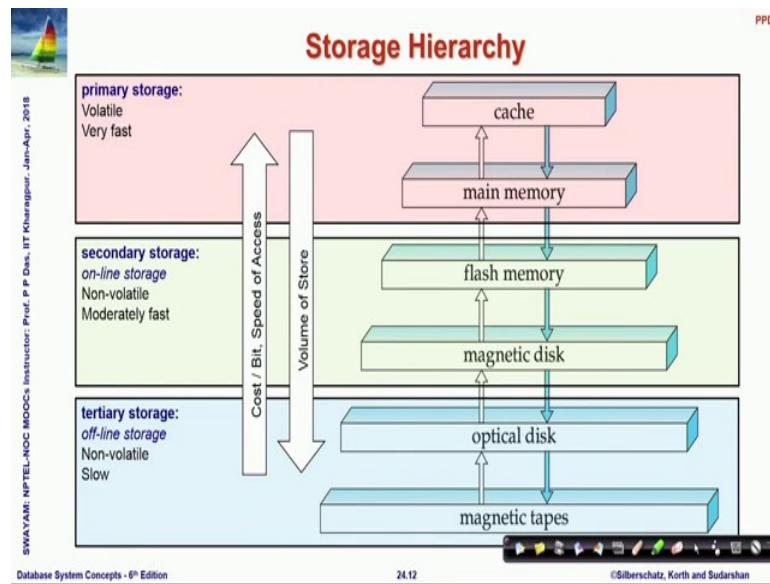
24.11

©Silberschatz, Korth and Sudarshan

Then you have the tape storage which usually is a largest volume of storage, but it is as a name suggests it is a tape it is a linear device. So, access can only be sequential. So, if you want to read the 6th record you have to skip of a record 1 to 5, but it can be a very high capacity usually it is slow.

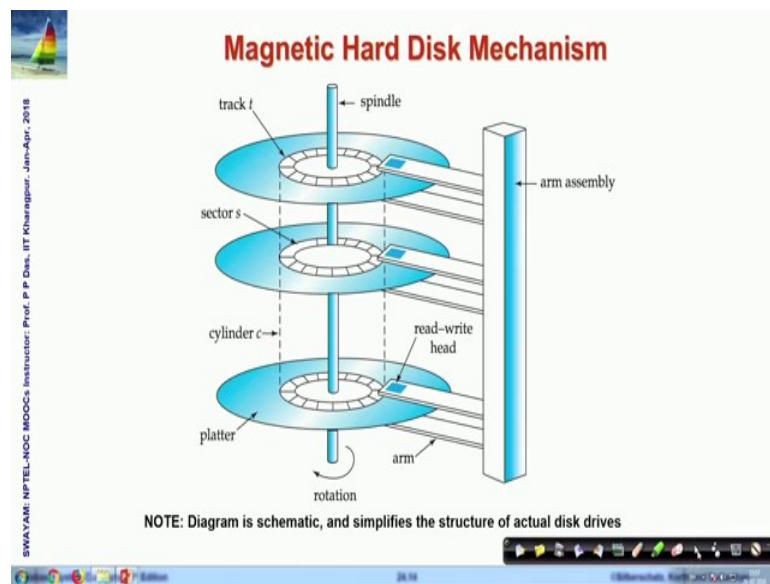
But very large in volume and tape juke boxes can support hundreds of terabytes or even multiple of petabyte. So, that is for offline storage.

(Refer Slide Time: 06:35)



This is a big medium. So, this is the basic storage hierarchy. So, we broadly classify them into three groups primary storage which is volatile and very fast cache and main memory is within that or secondary storage which is an online store which is non volatile and moderately fast and tertiary storage is called the offline store which is non volatile and slow. So, flash memory and magnetic disk are secondary storage they are non volatile and moderately fast and they are online. So, they exist with the system whereas, optical disk and magnetic disk can be removed and taken elsewhere.

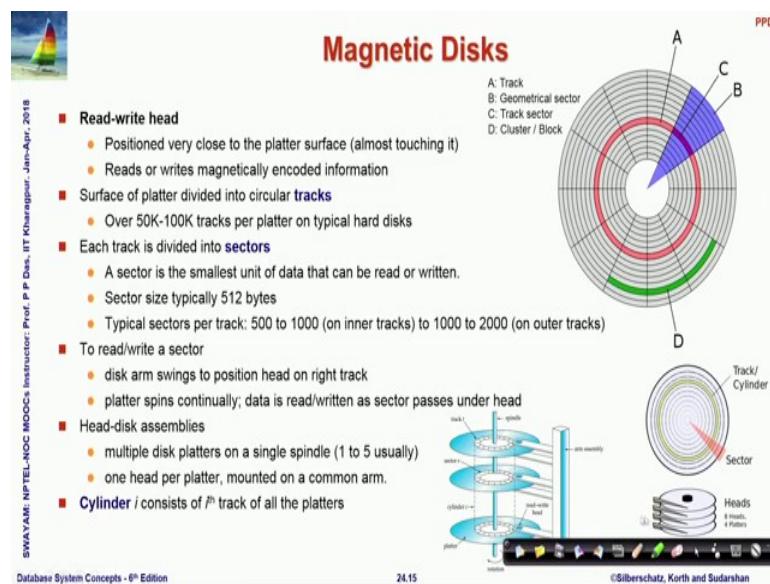
(Refer Slide Time: 07:12)



So, let us quickly take a look into the magnetic disk this is how a typical magnetic disk looks like; these are the different cylinders these are the different disks that we have and these are all different read write head.

So, as you can see all of them can work along a path backward forward like this and they can come and parallelly all of them parallelly can read from the different disks and this disks keep on spinning to help you look at the data anywhere on the disk. So, that is a typical structure of a magnetic.

(Refer Slide Time: 07:50)



Disk if you look specifically into. So, this is this is the structure we saw and if you specifically look into one particular disk then the disk is radially divided into different sectors portions. So, these are these are all separate portions and you can at a time the head can read one such sector.

So, these are called the geometric sectors and the whole of the ring that you can see the cylindrical ring that you can see is called a track. So, this is a track sector the orange one is a track sector and often we take multiple sectors from a particular track and combine them into one unit this is called the block of data and we will often talk about the block of data.

So, here at the typical numbers of how these sizes of this different units turn out to be.

(Refer Slide Time: 08:44)

The slide has a decorative header image of a sailboat on water. The title 'Magnetic Disks (Cont.)' is in red. The content is organized into two main bullet points:

- Earlier generation disks were susceptible to head-crashes
  - Surface of earlier generation disks had metal-oxide coatings which would disintegrate on head crash and damage all data on disk
  - Current generation disks are less susceptible to such disastrous failures, although individual sectors may get corrupted
- Disk controller – interfaces between the computer system and the disk drive hardware
  - accepts high-level commands to read or write a sector
  - initiates actions such as moving the disk arm to the right track and actually reading or writing the data
  - Computes and attaches **checksums** to each sector to verify that data is read back correctly
    - If data is corrupted, with very high probability stored checksum won't match recomputed checksum
  - Ensures successful writing by reading back sector after writing it
  - Performs remapping of bad sectors

Small text on the left edge: SWAYAM: NPTEL-NOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018

Small text at the bottom: Database System Concepts - 8<sup>th</sup> Edition

Small text at the bottom: 24.16

Small text at the bottom: ©Silberschatz, Korth and Sudarshan

So, the early generation where of disk were susceptible to head crashes, but now a days it is moved it quite stable there are disk controllers which regularly check and manage the; read write into in terms of the sectors naturally the. So, many heads being in parallel the data can be written on to multiple sectors at the same time and so, for doing that.

(Refer Slide Time: 09:12)

The diagram illustrates a Disk Subsystem. At the top, there is a small sailboat icon. Below it, the title "Disk Subsystem" is centered. A horizontal line labeled "system bus" runs across the middle. On the left side of the bus, there is a rectangular box labeled "disk controller". Four vertical lines descend from the controller to four circular shapes labeled "disks". A vertical line of text on the left edge reads: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018". On the right side, there is a video frame showing a man speaking, with a progress bar and other video controls below it. At the bottom, the text "Database System Concepts - 8<sup>th</sup> Edition" and "24.17" are on the left, and "©Silberschatz, Korth and Sudarshan" is on the right.

We will see what is the mechanism for that; and we also have disk subsystems where if you need a large volume of data to be managed which is larger than a single disk. Then you can connect multiple disk and through a disk controller you can actually read write data on to them and there are different such disk interface standards that you may have heard of.

(Refer Slide Time: 09:38)

The diagram illustrates a Disk Subsystem. At the top, there is a small sailboat icon. Below it, the title "Disk Subsystem" is centered. A horizontal line labeled "system bus" runs across the middle. On the left side of the bus, there is a rectangular box labeled "disk controller". Four vertical lines descend from the controller to four circular shapes labeled "disks". A vertical line of text on the left edge reads: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr- 2018". On the right side, there is a video frame showing a man speaking, with a progress bar and other video controls below it. At the bottom, the text "Database System Concepts - 8<sup>th</sup> Edition" and "24.18" are on the left, and "©Silberschatz, Korth and Sudarshan" is on the right.

We will just mention that these are the typical storages the SAN and NAN are the typical storages.

So, if you come across these terms we will not go into details of that that takes us into a different course of hardware discussion, but these are the very common storages for database disk systems.

(Refer Slide Time: 09:56)

The slide has a decorative header with a sailboat icon and the title 'Performance Measures of Disks' in red. The content is organized into sections with bullet points:

- **Access time** – the time it takes from when a read or write request is issued to when data transfer begins. It consists of:
  - **Seek time** – time it takes to reposition the arm over the correct track
    - Average seek time is 1/2 the worst case seek time
      - Would be 1/3 if all tracks had the same number of sectors, and we ignore the time to start and stop arm movement
    - 4 to 10 milliseconds on typical disks
  - **Rotational latency** – time it takes for the sector to be accessed to appear under the head
    - Average latency is 1/2 of the worst case latency.
    - 4 to 11 milliseconds on typical disks (5400 to 15000 rpm)
- **Data-transfer rate** – the rate at which data can be retrieved from or stored to the disk.
  - 25 to 100 MB per second max rate, lower for inner tracks
  - Multiple disks may share a controller, so rate that controller can handle is also important
    - E.g. SATA: 150 MB/sec, SATA-II 3Gb (300 MB/sec)
    - Ultra 320 SCSI: 320 MB/s, SAS (3 to 6 Gb/sec)
    - Fiber Channel (FC2Gb or 4Gb): 256 to 512 MB/s

Small video player window showing a person speaking is visible in the bottom right corner. The footer contains the text 'SWAYAM: NPTEL-NOCOCS Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr. 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '24.19', and '©Silberschatz, Korth and Sudarshan'.

Now basic question is for doing this read write on the disk what kind of performance measures we should look at. So, one main measure is access time if I want to access a record from the disk, then how much time shall I need. So, this is based on two major components one is a seek time now naturally as we have seen that the disk platter as a whole range of read heads which are positioned.

So, to be able to get the data the platter has to the head has to move forward or backward to the right track on which the data is there. So, this time it takes to go from current position to the correct track where, I find the data is called the seek time. Now, even when it is come to the; correct position in terms of the correct track it may not actually be on the correct sector.

Because, it is at the whole track is a is kind of a circle. So, it may be at one part of the circle and the actual data may be in a sector which is in a different part of the circle. So, the disk will have to rotate. So, that the correct sector comes under the head which is already positioned through the seek. So, the time to seek plus the rotational latency the time it takes the for the sake sector to be access is gives the total access time and then comes the other measure which is the data transfer rate as to you using this then what is

the rate how many megabytes and so, on can be transferred at from this. So, it that depends on a besides the access time you will have to see what is the rate at which the disk can be copied from the magnetic medium to the semiconductor medium and transferred.

(Refer Slide Time: 11:52)

The screenshot shows a presentation slide with a title 'Performance Measures (Cont.)'. On the right side, there is a video player window displaying a man speaking. The video player has standard controls like play, pause, and volume. Below the video player, there is some text and a small logo. The left side of the slide contains a list of bullet points under a heading 'Mean time to failure (MTTF)'. The footer of the slide includes the text 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Date, IIT Kanpur - Jan-Apr- 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '24.20', and '©Silberschatz, Korth and Sudarshan'.

- **Mean time to failure (MTTF)** – the average time the disk is expected to run continuously without any failure
  - Typically 3 to 5 years
  - Probability of failure of new disks is quite low, corresponding to a "theoretical MTTF" of 500,000 to 1,200,000 hours for a new disk
    - ▶ E.g., an MTTF of 1,200,000 hours for a new disk means that given 1000 relatively new disks, on average one will fail every 1200 hours
  - MTTF decreases as disk ages

Ah the other performance measure that we are concerned with in the database system is known as MTTF which is mean time to failure.

Because, database systems as you know are dealing with persistent data. So, data has to exist and therefore, the disk on which we keep the data must be very very reliable. So, meantime to failure is conceptually that if you consider two failures of the database of the disk to consecutive failures then what is the time the time elapsed between them the average of the time that has elapse between them now. So, if we say the, it is typically now mean time to failure for this magnetic disk that we use today at typically 3 to 5 years.

So, the probability of the failure of a new disk is very very low. So, it is kind of a theoretical MTTF we which will be said like this which; obviously, in terms of hours it it does not really make a make a physical science. So, what it in technical in in more practical terms, what it means that if you are given thousand relatively new disks then on average one of them will fail every twelve hundred hours.

So, this is what is a. So, MTTF certainly decrease it will decrease as a disk becomes old. So, it decreases with age. So, they will start failing sooner than it is to do.

(Refer Slide Time: 13:22)

**Optimization of Disk-Block Access**

- **Block** – a contiguous sequence of sectors from a single track
  - data is transferred between disk and main memory in blocks
  - sizes range from 512 bytes to several kilobytes
    - ▶ Smaller blocks: more transfers from disk
    - ▶ Larger blocks: more space wasted due to partially filled blocks
    - ▶ Typical block sizes today range from 4 to 16 kilobytes

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

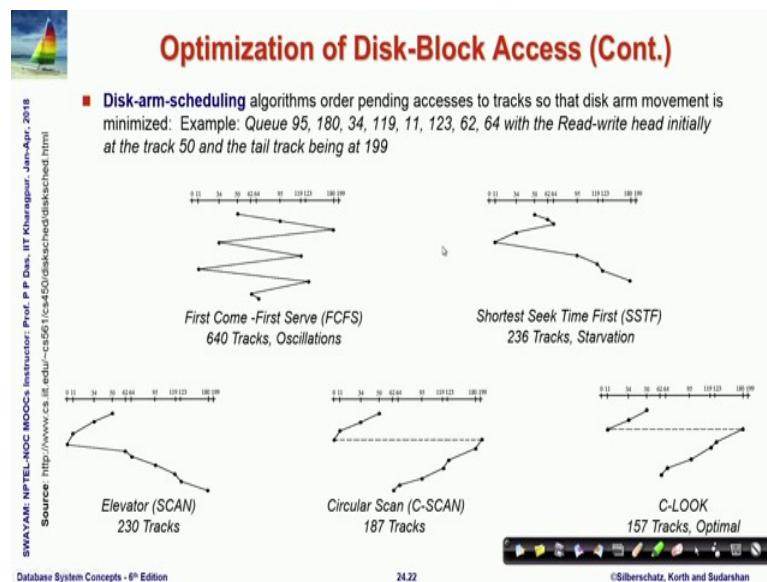
Database System Concepts - 8<sup>th</sup> Edition      24.21      ©Silberschatz, Korth and Sudarshan

Now we have to basic objective is to transfer the data at a fast rate. So, what we try to optimize is a; what is called a block the contiguous sequence of sectors from a single track which I mentioned earlier. So, this is what we will be read at one go. So, once you access the data one block will be read block size can range from 512 bytes to several kilobytes. If the blocks are smaller than naturally will need more transfers from the disk if the blocks are larger then you might waste lot of space because your part of the block will not can be used with the data.

So, with all that consideration the typical blocks size that use today is 4 to 16 kilobytes. So, you will see that in all our subsequent discussions particularly with the access and the file organization and the indexing we will consider that a block is one unit.

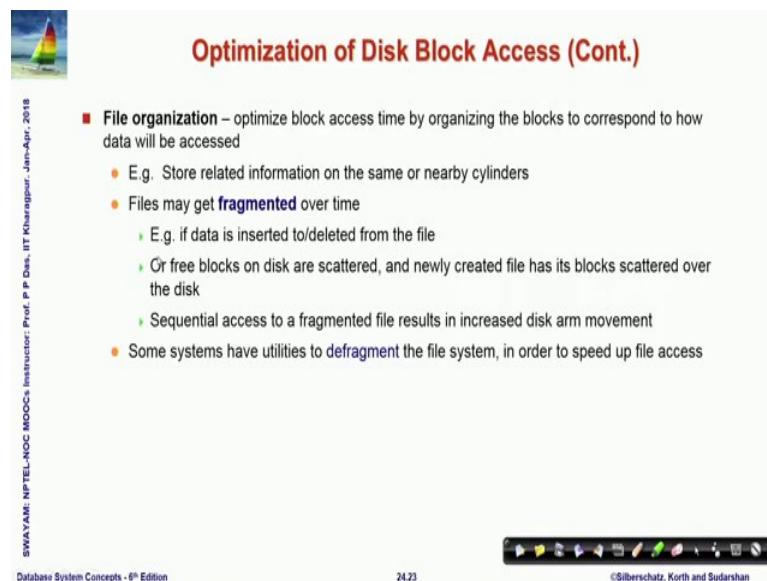
Which can be fetched in one go and that determines the size of the basic information node where the information about the records will be maintained.

(Refer Slide Time: 14:36)



This is the more details in terms of how you move your disk head. So, I think will skip this is more advance material.

(Refer Slide Time: 14:46)



So, to optimize the block access we need to organize the blocks corresponding to how the data will be accessed. So, certainly if the related data are kept in nearby blocks then naturally you are disk head and the rotation will have to be mean will get minimized. So, the basic idea is store the related information in nearby cylinders in the nearby cylindrical practice, but as you keep on using you may start with that, but as you keep on

using for various types of insert delete and overflows that keeps on happening. So, the data gets very fragmented which means that the data gets spread over a whole lot of widely separated cylinders and so, on.

So, the time to actually seek the data increases. So, we can correct that by doing what is called a defragmentation process. So, by defragmentation what you do is your data which is got distributed all over the disk you try to bring them together again to logically continuous physically contiguous blocks so, that their access time can improve. So, databases often will periodically defragment the file system.

(Refer Slide Time: 16:06)

**Optimization of Disk Block Access (Cont.)**

- Nonvolatile write buffers speed up disk writes by writing blocks to a non-volatile RAM buffer immediately
  - Non-volatile RAM: battery backed up RAM or flash memory
    - Even if power fails, the data is safe and will be written to disk when power returns
  - Controller then writes to disk whenever the disk has no other requests or request has been pending for some time
  - Database operations that require data to be safely stored before continuing can continue without waiting for data to be written to disk
  - Writes can be reordered to minimize disk arm movement
- Log disk – a disk devoted to writing a sequential log of block updates
  - Used exactly like nonvolatile RAM
    - Write to log disk is very fast since no seeks are required
    - No need for special hardware (NV-RAM)
- File systems typically reorder writes to disk to improve performance
  - Journaling file systems write data in safe order to NV-RAM or log disk
  - Reordering without journaling: risk of corruption of file system data

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

Database System Concepts - 8<sup>th</sup> Edition

24.24

©Silberschatz, Korth and Sudarshan

The other way look at the optimize block access is by using buffers. So, idea of the buffer is suppose you want to write some data to the disk. So, naturally for writing the data also you will need a seek time you will need the rotational latency then we will have to data do the data transfer.

So, if you are trying to do some write you have you can take another option that you actually write that to a buffer a buffer which is in the memory in the it is a in the a semiconductor buffer where you can very quickly write and then when you have enough data in the buffer then you can take them in a single go to the disk.

And write that or when the disk is not actually doing something some access you can use those cycles to write that now naturally if you write things onto the buffer then it is

possible that while your buffer has some data which has actually not been written to the disk if the power fails then you will lose that data. So, parts of the data which you think have been written actually have not gone to the disk. So, to take care of that often non volatile memory ram are used which are battery backed up or flash memory.

So, that even if power fails the right buffers will not be lost. So, we say that use non volatile buffers and other option that is used often is you maintain a log disk a log disk is nothing, but when you are writing to the disk you make a sequential log of the updates that you are doing. So, this kind of is a report. So, you are saying that I have written this data to this block written this data to this block.

So, if there is a failure, then if you can go through the log and you can actually retrieve the information of what was lost how far it actually worked correctly and you can use exactly like the non volatile ram and using the log you can find out. So, you are keeping a kind of a general link system you are keeping information of this is what I did this is what I did. So, all this write information are maintained in terms of the log.

(Refer Slide Time: 18:30)

The slide is titled "Flash Storage" in red at the top right. On the left, there is a small logo of a sailboat on water. The main content is a bulleted list under two main points: "NOR flash vs NAND flash" and "NAND flash".

- NOR flash vs NAND flash
- NAND flash
  - used widely for storage, since it is much cheaper than NOR flash
  - requires page-at-a-time read (page: 512 bytes to 4 KB)
  - transfer rate around 20 MB/sec
  - **solid state disks**: use multiple flash storage devices to provide higher transfer rate of 100 to 200 MB/sec
  - erase is very slow (1 to 2 millisecs)
    - erase block contains multiple pages
    - remapping of logical page addresses to physical page addresses avoids waiting for erase
      - **translation table** tracks mapping
        - also stored in a label field of flash page
      - remapping carried out by **flash translation layer**
    - after 100,000 to 1,000,000 erases, erase block becomes unreliable and cannot be used
      - **wear leveling**

You can use flash storage nowadays the NAND based flash storage is very common.

(Refer Slide Time: 18:40)

The slide has a header 'PPD' and a sidebar with the text: 'Overview of Physical Storage Media', 'Magnetic Disks', 'RAID', and 'Tertiary Storage'. It features a large red title 'RAID: REDUNDANT ARRAY OF INDEPENDENT DISKS'. The footer includes 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '24.26', '©Silberschatz, Korth and Sudarshan', and a navigation bar.

So, here are some details on that let us move on to understanding. So, we just took a look into the basic storage hierarchy and the use of magnetic disks and what are the parameters that control the performance in terms of the read write in terms of the disk RAID is a the full form is redundant array of independent disks.

(Refer Slide Time: 19:05)

The slide features a large red title 'RAID'. Below it is a bulleted list under the heading '■ RAID: Redundant Arrays of Independent Disks':

- disk organization techniques that manage a large numbers of disks, providing a view of a single disk of
  - high capacity and high speed by using multiple disks in parallel,
  - high reliability by storing data redundantly, so that data can be recovered even if a disk fails
- The chance that some disk out of a set of  $N$  disks will fail is much higher than the chance that a specific single disk will fail
  - E.g., a system with 100 disks, each with MTTF of 100,000 hours (approx. 11 years), will have a system MTTF of 1000 hours (approx. 41 days)
  - Techniques for using redundancy to avoid data loss are critical with large numbers of disks
- Originally a cost-effective alternative to large, expensive disks
  - I in RAID originally stood for "inexpensive"
  - Today RAIDs are used for their higher reliability and bandwidth
  - The "I" is interpreted as independent

The footer includes 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr., 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '24.27', '©Silberschatz, Korth and Sudarshan', and a video thumbnail of a professor.

So, the disk organization that manage a large number of disk, but the view that you get you do not get to see the multiple disk you get to see a single disk.

So, by this you can create very high capacity and very high speed and. So, because you have multiple disks so, you organize a data in such a way that when you are writing or you are reading actually you can parallelly read or write from multiple different disks. So, that not only increases capacity, but actually increases a throughput and you can get high reliability also by storing the data redundantly; that is keeping multiple copies and that is what RAID is often quite known for.

So, originally when RAID was originally designed actually the, I in red stood for inexpensive. So, it was kind of a disk array which was inexpensive to afford, but now it is not particularly the expenses is not the primary factor for which we do go for RAID, but we go for RAID for the high capacity high speed and high reliability. So, the I is now interpreted as independent array.

(Refer Slide Time: 20:21)

**Improvement of Reliability via Redundancy**

SWAYAM: NPTEL-NOCO MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018

- **Redundancy** – store extra information that can be used to rebuild information lost in a disk failure
- **Mean time to data loss** depends on mean time to failure, and **mean time to repair**
  - E.g. MTTF of 100,000 hours, mean time to repair of 10 hours gives mean time to data loss of  $500 \times 10^6$  hours (or 57,000 years) for a mirrored pair of disks (ignoring dependent failure modes)
- **Mirroring (or shadowing)**
  - Duplicate every disk. Logical disk consists of two physical disks.
  - Every write is carried out on both disks
    - » Reads can take place from either disk
  - If one disk in a pair fails, data still available in the other
    - » Data loss would occur only if a disk fails, and its mirror disk also fails before the system is repaired
      - Probability of combined event is very small
      - » Except for dependent failure modes such as fire or building collapse or electrical power surges

Database System Concepts - 6<sup>th</sup> Edition      24.28      ©Silberschatz, Korth and Sudarshan

So, we can improve. So, now, the main issue in red is to improve reliability. So, the main the key idea is have redundancy to improve the reliability that is stored the data in multiple copies and can rebuild from the copies once a disk has failed. So, we earlier looked at the MTTF mean time to failure. Now, we are look at another parameter which we say is a mean time to data loss. So, which depends on mean time to failure, because if you are if you are failed then you have lost the data, but when you have failed you have a possibility now, to recover the data to repair it; because you have redundant copies.

So, mean time to data loss it depends on the MTTF plus the mean time to repair how quickly can we use your redundant copies and get back the original data. So, mean time to data loss is the actual key factor which needs to be minimized and for this red does a mirroring or shadowing that is for every disk there is a duplicate there is a clone.

So, it logically you have one disk, but physically you have actually two disk. So, every write that you do is carried out to both the disks and when you read the read happens on either of the disk usually it switches between the disks. So, if one of the disk in the pair would fail, then the data can still be recovered from the other and the data loss would occur if a disk fail, but the mirror disk also has to fail before the system has been repair.

So, if one of the disk fail you get the data from the middle disk you continue to do that from the mirror disk you restore the other disk you may be replace and put a new one mirror it again and. So, on, but you can restore that. So, in between this time if the mirror disk also fails; then you have actually lost data, but the probability of that is very very small. So, mirroring gives a at the expense of naturally having lot more of redundant storage the mirroring can actually give you a much higher reliability.

(Refer Slide Time: 22:39)

**Improvement of Reliability via Redundancy**

- **Bit-level striping** – split the bits of each byte across multiple disks
  - In an array of eight disks, write bit  $i$  of each byte to disk  $i$
  - Each access can read data at eight times the rate of a single disk
  - But seek/access time worse than for a single disk
    - ▶ Bit level striping is not used much any more
- **Block-level striping** – with  $n$  disks, block  $i$  of a file goes to disk  $(i \bmod n) + 1$ 
  - Requests for different blocks can run in parallel if the blocks reside on different disks
  - A request for a long sequence of blocks can utilize all disks in parallel

SWAYAM: NPTEL-NOCO MOOCs. Instructor: Prof. P. Das., IIT Kharagpur. Jam-Apr. 2018

Database System Concepts - 6<sup>th</sup> Edition      24.29      ©Silberschatz, Korth and Sudarshan

That we expect today another some of the other techniques which improve reliability is bit level and byte level block level striping techniques. So, what you in the basic bit level striping what you do is a say every byte has 8 bits. So, when you are writing a byte you

do not write all the bytes to the same disk you write them to multiple disks. So, you take an array of 8 disks. So, write bit  $I$  of each byte to disk  $I$  it is. So, did interesting concept you have fragmenting it in a very peculiar way.

So, you have 8 disks and every byte first byte first bit is written to one disk second byte is second bit is written to the second disk, third bit is written to the third disk and so, on. And when you access you can access from all these 8; now naturally which means that this will decrease your throughput to some extent, because you have to collect from all of that reconstruct. So, bits levels striping is not much in use any more instead you have block level striping where with end disk a block  $I$  of a file goes to the disk  $i \bmod n + 1$ .

So, circularly so, the first goes if you have say five disks in the first block goes to disk one second to disk two-fifth to disk 5 and the 6th again back to disk 1. So, the request to different blocks can run in parallel and reside on different disk and if they can easily utilize this parallelism to improve the throughput.

(Refer Slide Time: 24:30)

**Improvement of Reliability via Redundancy**

■ **Bit-Interleaved Parity** – a single parity bit is enough for error correction, not just detection, since we know which disk has failed

- When writing data, corresponding parity bits must also be computed and written to a parity bit disk
- To recover data in a damaged disk, compute XOR of bits from other disks (including parity bit disk)

■ **Block-Interleaved Parity**: Uses block-level striping, and keeps a parity block on a separate disk for corresponding blocks from  $N$  other disks

- When writing data block, corresponding block of parity bits must also be computed and written to parity disk
- To find value of a damaged block, compute XOR of bits from corresponding blocks (including parity block) from other disks.

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

Database System Concepts - 8<sup>th</sup> Edition 24.30 ©Silberschatz, Korth and Sudarshan

At the same time improve the reliability now how do you improve the reliability. So, for improving the reliability you use the basic you know error correcting coding concept you just use a bit level that again two options one is you can do a bit inter lift parity which means a single parity bit is used which is good for error correction. Usually, we know that if we use a single parity bit we can know a single error we can detect a single error,

but here with a single bit you can correct the single error also because you know in case of a failure you know which particular disk has failed. So, then you can exalt with a data from the other disk and reconstruct the error bit.

So, the other is naturally block inter leaving of the parity which uses block level striping and keeps a parity block on a separate disk for corresponding blocks from n other disks and you can reconstruct in a very similar manner. So, by using block interleaved parity with block striping you can really have a higher throughput with a better reliability.

(Refer Slide Time: 25:44)

**Choice of RAID Level**

- Factors in choosing RAID level
  - Monetary cost
  - Performance: Number of I/O operations per second, and bandwidth during normal operation
  - Performance during failure
  - Performance during rebuild of failed disk
    - ↳ Including time taken to rebuild failed disk
- RAID 0 is used only when data safety is not important
  - E.g. data can be recovered quickly from other sources
- Level 2 and 4 never used since they are subsumed by 3 and 5
- Level 3 is not used anymore since bit-striping forces single block reads to access all disks, wasting disk arm movement, which block striping (level 5) avoids
- Level 6 is rarely used since levels 1 and 5 offer adequate safety for most applications

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

Database System Concepts - 8<sup>th</sup> Edition

24.31

©Silberschatz, Korth and Sudarshan

So, it is a win-win situation now naturally when you go for RAID you will find that there are different levels of read that are available today goes up to level 6 right. Now, and the factors that certainly has to be considered is I mean different RAID at different kind of cost the what is the performance what is the performance during failure; that is performance during failure is typically the MTTF and performance during rebuilt is a mean time to data loss so, including the rebuilding and so, on. So, based on these factors different rate levels can be looked at RAID 0 is used only when data safety is not important. So, that is not very common the RAID level 2 and 4 are never used.

Because, they are they got subsumed in level 3 and 5. So, you can ignore that level three is also not used anymore, because it used bits striping and naturally we talked of that that is not that is worse compare to the block striping which level 5 uses and level 6 is also really used. Since level 1 and 5 adequately support all applications.

(Refer Slide Time: 27:08)



## Choice of RAID Level (Cont.)

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

- Level 1 provides much better write performance than level 5
  - Level 5 requires at least 2 block reads and 2 block writes to write a single block, whereas Level 1 only requires 2 block writes
  - Level 1 preferred for high update environments such as log disks
- Level 1 had higher storage cost than level 5
  - disk drive capacities increasing rapidly (50%/year) whereas disk access times have decreased much less ( $\times 3$  in 10 years)
  - I/O requirements have increased greatly, e.g. for Web servers
  - When enough disks have been bought to satisfy required rate of I/O, they often have spare storage capacity
    - so there is often no extra monetary cost for Level 1!
- Level 5 is preferred for applications with low update rate, and large amounts of data
- Level 1 is preferred for all other applications



Database System Concepts - 8<sup>th</sup> Edition 24.32 ©Silberschatz, Korth and Sudarshan

So, the conclusions simply, is that you either use RAID level 1 or you use RAID level 5. So, RAID level 1 gives a better right performance, than RAID 5 and it is certainly level. So, therefore, level 1 is preferred for high update environments such as log disks and so, on whereas, level one has higher storage cost than 5 also and level 5 is preferred for applications that has low update rate and large volume of data. So, if you have very high update you go for level one rate. So, which will give you which will cost you more, but if you have a level 5, then you will be able to get a low update, but have large amount of data stored reliably with less amount of money invested into that.

(Refer Slide Time: 28:15)



## Optical Disks

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

- Compact disk-read only memory (CD-ROM)
  - Removable disks, 640 MB per disk
  - Seek time about 100 msec (optical read head is heavier and slower)
  - Higher latency (3000 RPM) and lower data-transfer rates (3-6 MB/s) compared to magnetic disks
- Digital Video Disk (DVD)
  - DVD-5 holds 4.7 GB, and DVD-9 holds 8.5 GB
  - DVD-10 and DVD-18 are double sided formats with capacities of 9.4 GB and 17 GB
  - Blu-ray DVD: 27 GB (54 GB for double sided disk)
  - Slow seek time, for same reasons as CD-ROM
- Record once versions (CD-R and DVD-R) are popular
  - data can only be written once, and cannot be erased.
  - high capacity and long lifetime; used for archival storage
  - Multi-write versions (CD-RW, DVD-RW, DVD+RW and DVD-RAM) also available



Database System Concepts - 8<sup>th</sup> Edition 24.34 ©Silberschatz, Korth and Sudarshan

And so, these are the RAID levels then of course, finally there are different tertiary storages compact disk CD-ROM we all are familiar that DVD the record once versions where you just record and use that particularly for different kind of distribution these.

(Refer Slide Time: 28:34)

**Magnetic Tapes**

- Hold large volumes of data and provide high transfer rates
  - Few GB for DAT (Digital Audio Tape) format, 10-40 GB with DLT (Digital Linear Tape) format, 100 GB+ with Ultrium format, and 330 GB with Ampex helical scan format
  - Transfer rates from few to 10s of MB/s
- Tapes are cheap, but cost of drives is very high
- Very slow access time in comparison to magnetic and optical disks
  - limited to sequential access.
  - Some formats (Accelis) provide faster seek (10s of seconds) at cost of lower capacity
- Used mainly for backup, for storage of infrequently used information, and as an off-line medium for transferring information from one system to another.
- Tape jukeboxes used for very large capacity storage
  - Multiple petabytes ( $10^{15}$  bytes)

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

Database System Concepts - 8<sup>th</sup> Edition

24.35

©Silberschatz, Korth and Sudarshan

storages are used there are magnetic tapes which are very large volume and provide a high transfer rate and they are go currently they go into different couple of orders of terabytes even petabytes in size, but the tapes are not really very expensive. So, we can use them to hold really really large databases they are good for Backups and so, on, but the tape drives are quite expensive. So, that will have to keep in mind.

(Refer Slide Time: 29:08)

The slide is titled "Module Summary" in red at the top right. On the left, there is a small sailboat icon. The main content area contains a bulleted list of learning objectives:

- Looked at the options of Physical Storage Media for high volume, fast, reliable and inexpensive options for data storage for databases
- Understood the structure and basic functionality of Magnetic Disks
- Understood RAID – array of redundant disks in parallel to enhance speed and reliability
- Understood the options of Tertiary Storage for high volume, inexpensive backup options

At the bottom left, it says "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P.P. Desai, IIT Kanpur Date: Jan-Apr-2018". At the bottom center, it says "Database System Concepts - 8<sup>th</sup> Edition" and "24.36". At the bottom right, it says "©Silberschatz, Korth and Sudarshan". There is also a small video player window showing a man speaking.

So, to summarize we have looked at different physical storage media. So, this was I mean besides all the details in the discussion the key take way point for us here is to understand that primarily.

We have a memory which is expensive and which is small in size very high speed. So, all operations that we need that need to be done we will finally, have to happen when the once the data is in memory and on the other side we have all the persistent data in a in some kind of a magnetic disk in certain structure and that is that can support large storage it is persistent it is reliable, but it is relatively slow to access and so, it needs to be used in a intelligent manner and one point that you have specifically noted that there is some unit for every disk system.

There is some unit called a block or disk block, which is a basic unit of data that will be transferred every time you access the disk. So, you are if your design is aligned with a size of the disk block which is couple of kilobytes then it will be easier to be able to design more optimal physical storage for your files and you can speed up the whole process of search and update that you want to do in the database.

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

**Lecture – 25**  
**Storage and File Structure : File Structure**

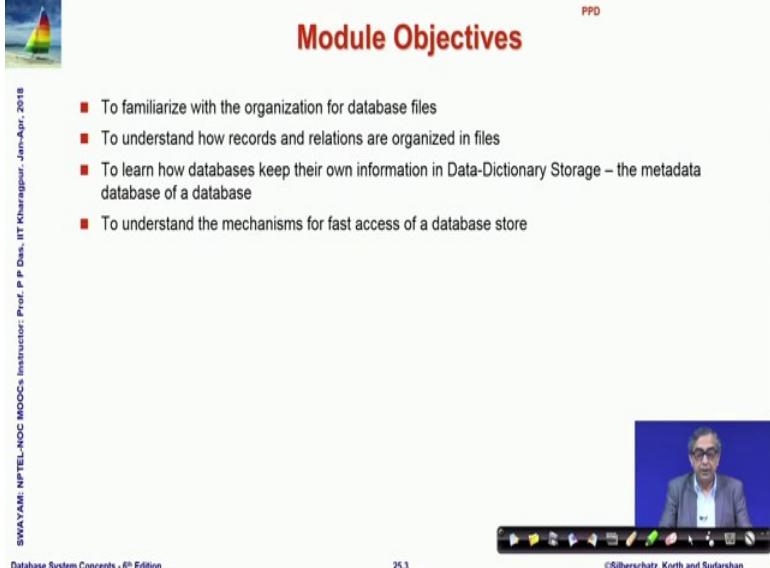
Welcome to module 25 of Database Management Systems. We have been discussing about storage and file structure.

(Refer Slide Time: 00:22)

The slide is titled "Module Recap" in red at the top right. It features a small sailboat icon in the top left corner. A vertical column of text on the left side reads: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018". At the bottom left, it says "Database System Concepts - 8<sup>th</sup> Edition". The main content area contains a bulleted list of storage topics: "■ Overview of Physical Storage Media", "■ Magnetic Disks", "■ RAID", and "■ Tertiary Storage". On the right side, there is a video player window showing a man speaking, with the caption "©Silberschatz, Korth and Sudarshan" below it. The slide has a "PPD" watermark in the top right corner.

In the last module we have talked about different storage options.

(Refer Slide Time: 00:27)



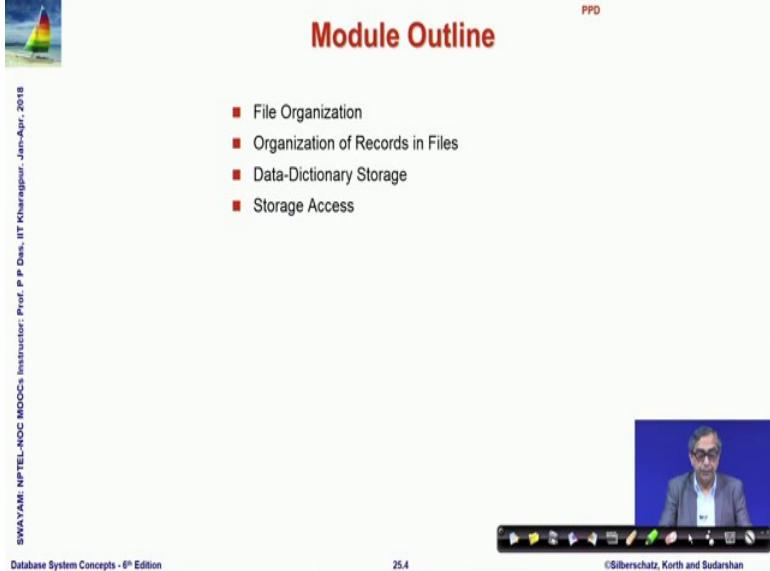
The slide is titled "Module Objectives" in red. It features a small sailboat icon in the top left corner and a video player window in the bottom right showing a man speaking. The text on the slide lists four objectives:

- To familiarize with the organization for database files
- To understand how records and relations are organized in files
- To learn how databases keep their own information in Data-Dictionary Storage – the metadata database of a database
- To understand the mechanisms for fast access of a database store

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018  
PPD  
Database System Concepts - 8<sup>th</sup> Edition  
25.3  
©Silberschatz, Korth and Sudarshan

And in this one we will talk about the; organization of database files, what should be the typical structure to store the records in the files. And how the overall database which manage itself we will talk about those issues.

(Refer Slide Time: 00:41)



The slide is titled "Module Outline" in red. It features a small sailboat icon in the top left corner and a video player window in the bottom right showing a man speaking. The text on the slide lists five topics:

- File Organization
- Organization of Records in Files
- Data-Dictionary Storage
- Storage Access

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018  
PPD  
Database System Concepts - 8<sup>th</sup> Edition  
25.4  
©Silberschatz, Korth and Sudarshan

So, the file organization; so if you look at a database; what is the database? It is a collection of relations.

(Refer Slide Time: 00:43)

The slide has a title 'File Organization' at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list under two main points:

- A database is
  - A collection of *files*. A file is
    - A sequence of *records*. A record is
      - A sequence of *fields*
- One approach:
  - assume record size is fixed
  - each file has records of one particular type only
  - different files are used for different relations

This case is easiest to implement; will consider variable length records later

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kharagpur - Jan-Apr., 2018

Database System Concepts - 8<sup>th</sup> Edition

25.6

©Silberschatz, Korth and Sudarshan

A video player interface is visible on the right side of the slide, showing a video of a man speaking. Below the video player are several small icons.

So, it is a collection of files every relation is a file. Now, what is a file? A file is a sequence of records, and what is a record? It is a sequence of fields. So, this is the hierarchy that exists and this will have to be kept in mind, when we design the organization of how we keep this data.

Now, one starting approach could be we can assume that all records are of fixed size which makes a life easier and each file has records of only one type again a simplifying assumption and different files are used for different relations. So, this is a easiest case to implement.

(Refer Slide Time: 01:30)

The slide has a header 'Fixed-Length Records' with a sailboat icon. It includes a sidebar with course information: SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018.

**Simple approach:**

- Store record  $i$  starting from byte  $n * (i - 1)$ , where  $n$  is the size of each record.
- Record access is simple but records may cross blocks
  - Modification: do not allow records to cross block boundaries

**Deletion of record  $i$ :**

alternatives:

- move records  $i + 1, \dots, n$  to  $i, \dots, n - 1$
- move record  $n$  to  $i$
- do not move records, but link all free records on a free list

record #	name	subject	score	
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Navigation icons: back, forward, search, etc.

Page footer: Database System Concepts - 8<sup>th</sup> Edition, 25.7, ©Silberschatz, Korth and Sudarshan

So, we will start with that; so this is what we have we store these are fixed size records. So, we store them one after the other and based on the fixed size, we can easily know what is the starting address of any record and we can access it accordingly. Now, if a record is deleted, then there are several things that I can do see that this is a different alternatives, that is if I record delete record I then. So, if we delete any record then we can actually move the records. So, that we consume that space or we can take the last record and move it there or we can simply do not do any move, but use an some additional pointers to denote that these records have become free rather give it to a free list.

(Refer Slide Time: 02:22)

The slide title is "Deleting record 3 and compacting". It features a small sailboat icon in the top left corner and a copyright notice for Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018 on the right. The main content is a table of student records:

	ID	Name	Major	GPA
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Below the table is a navigation bar with icons for back, forward, search, etc. At the bottom, it says "Database System Concepts - 8<sup>th</sup> Edition", "25.8", and "©Silberschatz, Korth and Sudarshan".

So, these are the three main strategies. So, here we showing the first one the record three has been removed. So, all records have moved up in this it is we have move the last record 11 in the place of record 3. So, record 3 is gone, but still the whole thing remains compact only the point that must be noted that in the earlier one, where well we moved everything then the ordering that existed here of this key of this key field is maintained, but if we move the last record the naturally that ordering has got destroyed. So, it will have implications in terms of indexed organization that will cover in the next modules.

(Refer Slide Time: 03:08)

The slide title is "Free Lists". It features a small sailboat icon in the top left corner and a copyright notice for Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018 on the right. The main content is a table of student records with arrows pointing from specific cells to a free list header:

	ID	Name	Major	GPA
header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	15151	Mozart	Music	40000
record 2	22222	Einstein	Physics	95000
record 3	33456	Gold	Physics	87000
record 4	58583	Califieri	History	62000
record 5	76543	Singh	Finance	80000
record 6	76766	Crick	Biology	72000
record 7	83821	Brandt	Comp. Sci.	92000
record 8	98345	Kim	Elec. Eng.	80000
record 11				

Arrows point from the value "22222" in record 2, the value "33456" in record 3, and the value "76766" in record 6 to a "header" row above the table. Below the table is a video frame of a professor speaking, a navigation bar with icons, and the footer information "Database System Concepts - 8<sup>th</sup> Edition", "25.10", and "©Silberschatz, Korth and Sudarshan".

The third option could be use a free list, which is a nice one because you would you do not neither here neither you destroy the order that existed and no one have to really move records which is expensive, but you just start with a pointer and keep on pointing to the empty records.

And once you delete it you use that space itself to point to the next deleted record. So, whenever you have to you know delete a record all that you need to do is adjust this pointer. So, which is pretty fast and quite efficient way of getting this linked together in terms of; so there is as such no space over it and it is a fastest possible that you can do now in contest to fix length record.

(Refer Slide Time: 03:54)

**Variable-Length Records**

- Variable-length records arise in database systems in several ways:
  - Storage of multiple record types in a file
  - Record types that allow variable lengths for one or more fields such as strings (`varchar`)
  - Record types that allow repeating fields (used in some older data models)
- Attributes are stored in order
- Variable length attributes represented by fixed size (offset, length), with actual data stored after all fixed length attributes
- Null values represented by null-value bitmap

Null bitmap (stored in 1 byte)  
0000

21, 5	26, 10	36, 10	65000	10101	Srinivasan	Comp. Sci.		
Bytes 0	4	8	12	20	21	26	36	45

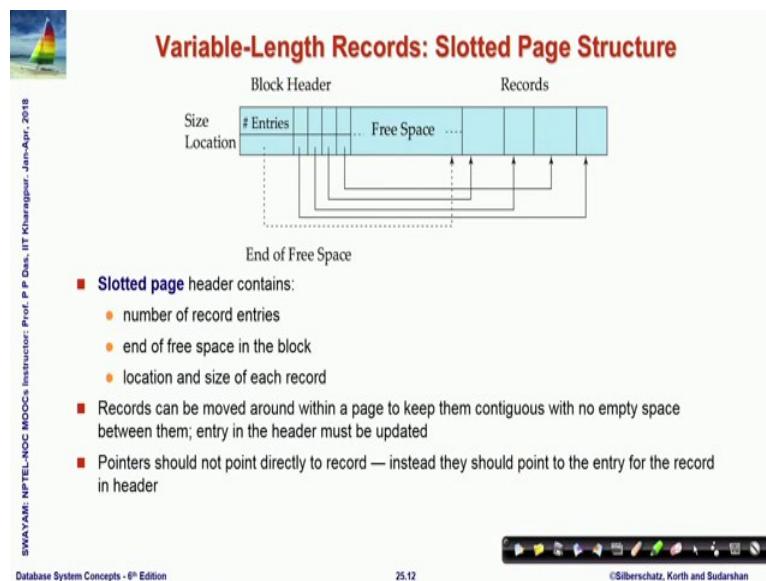
SWAYAM-NPTEL-MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018  
Database System Concepts - 8<sup>th</sup> Edition  
25.11  
©Silberschatz, Korth and Sudarshan

If the record becomes variable length, then certainly every record may of very different size and it is very common for example, we have types like varchar a lot of strings are varchar. So, we just we do not know how much it will take. So, the typical way you represent that is a you represent as to; what is a starting pointer of a particular of the actual value and the size of the value the number of bytes it will take. So, when we say **21,5** which we mean that this field will actually start from location 21 and we will have 5 locations 5 bytes, then the next one is **26, 10**.

So, this will start to 26 and go for 10 such; so what happens is; if you look into this part of the data, then that part is actually for all practical purposes the fix length 1, because here you are just keeping double x for the variable length data or you have some field

which is a fixed length data anyway or you have a null which is stored in one byte, and then you have all the variable stuff at one end. So, you can actually make part of this fixed length by using this kind of encoding. So, this is what is explained here.

(Refer Slide Time: 05:22)



So, for variable length records a one main issue is if you if you keep it like this, then since you are using actually you are using pointers here we saying that this data actually is on 21. So, what will happen is if you change the position of the record if you relocate the record, then all these references will have to be updated. So, that becomes a slotted thing. So, what the slotted page structure does is it does a [li/little] little bit of adjustment it ports a puts a records here at the at the end.

And it has a header it has a. So, it has a block header as in here and the block header has actually pointers to the records and then you have a an entry which points to the end of the free space where more records can still be stored. So, when you refer to a particular record you do not actually refer here. So, you do not refer here, but you refer here. So, what you maintain is the header is actually not changed, but if there are relocations required adjustments required, then that will be done with respect to this. And so, this value will change, but any references made to this location will remain invariant. So, that is the basic idea of the slotted page structure, which can allow you to have the variable length record with easy re locatability in the design.

Now, let us see the given this what is the organization of the records in the file.

(Refer Slide Time: 07:05)

The slide has a title 'Organization of Records in Files' at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list of four organization types: Heap, Sequential, Hashing, and Multitable clustering file organization. The 'Sequential' entry includes a note about search keys. The 'Multitable clustering file organization' entry includes a note about motivation. The footer of the slide includes the text 'SWAYAM-NPTEL-MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur- 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '25.14', and '©Silberschatz, Korth and Sudarshan'.

- **Heap** – a record can be placed anywhere in the file where there is space
- **Sequential** – store records in sequential order, based on the value of the search key of each record
- **Hashing** – a hash function computed on some attribute of each record; the result specifies in which block of the file the record should be placed
- Records of each relation may be stored in a separate file. In a **multitable clustering file organization** records of several different relations can be stored in the same file
  - Motivation: store related records on the same block to minimize I/O

So, there are different organizations that have been tried out the simplest is the heap is a record can be placed anywhere in the file where there is space. And you can link to that that is that is one way certainly there is nothing very smart in terms of doing that, but you can you would possibly like to do better than that. So, one is you can store the; records in a sequential manner let us store records in a sequential order in terms of certain search key.

So, based on the value of the search key you put them in the sequential orders. So, what it will mean that it will become easier to search the records in that way, but it has consequences or you can hash you can use a hash function on a some of the attributes of the record and the results specified on which block which disk block the record will be placed. So, these are the different option and a records of which relation may be stored in a separate file that is a basic convention, but in some cases there could be multi table clustering as well.

So, let us quickly take a look at these options.

(Refer Slide Time: 08:16)

The slide features a title 'Sequential File Organization' at the top right. On the left, there is a small logo of a sailboat on water. A vertical watermark on the left side reads 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018'. The main content consists of a table of records and a diagram showing pointer chains. The table has columns for ID, Name, Subject, and Marks. The records are:

ID	Name	Subject	Marks
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Arrows from the 'Name' column point to the next record in the sequence, illustrating the sequential organization. At the bottom, there is a navigation bar with icons and the text '25/16'.

So, these sequential file organizations. So, these things are kept sequentially here as you can see there all consequentially here and this is the link key of those.

(Refer Slide Time: 08:34)

The slide continues the topic with a title 'Sequential File Organization (Cont.)'. It includes a small sailboat logo and a watermark for 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018'. Below the title is a video frame showing a man speaking. The slide lists several points about managing sequential files:

- Deletion – use pointer chains
- Insertion – locate the position where the record is to be inserted
  - if there is free space insert there
  - if no free space, insert the record in an overflow block
  - In either case, pointer chain must be updated
- Need to reorganize the file from time to time to restore sequential order

Below the text is the same table of records as the previous slide, with arrows indicating pointer chains. At the bottom, there is a navigation bar with icons and the text '25/16'.

So, this is the issues of deletion and if you delete you use pointer chains. As you have we have discussed earlier, and if you have to insert then you look for a free space, if you find a free space you can put it there you insert it there if there is no free space then you have to use a overflow block, where you can go and place that separately as the dilemma

shows here; in a multi table clustering what you would do is more than one relation could be kept in the same file.

(Refer Slide Time: 09:00)

**Multitable Clustering File Organization**

Store several relations in one file using a **multitable clustering** file organization

<i>department</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
	Comp. Sci.	Taylor	100000
	Physics	Watson	70000

<i>instructor</i>	<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
	10101	Srinivasan	Comp. Sci.	65000
	33456	Gold	Physics	87000
	45565	Katz	Comp. Sci.	75000
	83821	Brandt	Comp. Sci.	92000

<i>multitable clustering of department and instructor</i>	<i>Comp. Sci.</i>	<i>Taylor</i>	<i>100000</i>
	45564	Katz	75000
	10101	Srinivasan	65000
	83821	Brandt	92000
	Physics	Watson	70000
	33456	Gold	87000

For example, here we are showing two relations department name building and budget these attributes doing department and instructor, id name, department name, and salary is other instructor in a way keeping them together here naturally, where we keep them together.

For example here in we have one which is here in we have one, which is and entry of record from the department relation. Similarly here is another which is from the department relation whereas, these are entries from the instructor relation; please note that since we are doing it multi table with a department we do not need to keep these information in as a part of the record, but what you mean is if there is a computer science entry here. Then all those records which follow this computer science entry are actually instructors in the computer science till I actually come across another departments entry where which will be followed by instructors for that department. So, that is a basic multi table convention that is to be followed here.

(Refer Slide Time: 10:26)

The slide title is "Multitable Clustering File Organization (cont.)". It features a small sailboat icon in the top left and a portrait of a man in the bottom left. On the right, there is a table with data and a pointer chain diagram.

Table data:

	Comp. Sci.	Taylor	100000	
45564	Katz	75000		
10101	Srinivasan	65000		
83821	Brandt	92000		
Physics	Watson	70000		
33456	Gold	87000		

Diagram: A pointer chain diagram showing two pointer chains originating from the last two columns of the table's data rows. One chain points to the first row, and the other points to the second row.

Page footer: SWAYAM-NPTEL-MOOCs Instructor: Prof. P.P. Desai, IIT Kharagpur - Jan-Apr. 2018, Database System Concepts - 8<sup>th</sup> Edition, 25.18, ©Silberschatz, Korth and Sudarshan.

Now, it is actually good for queries that involved joining department with instructor, because based on the value of the department you have the instructors club together and they could be very easily quickly taken together and it is also good for single queries with departments and it is instructors, because as you can see you can if you want to know for example, who are the faculty for at computer science department; then it be very easy to answer that, because you need to search for computer science and then you know all the list of the faculty will be in consecutive block.

So, you can easily lift that, but certainly this is not true, if you want to involve queries which have department only; because that department information are all now sparsely distributed. So, if your query has the department based information to be to be accumulated, and then this may not be a good option. So, that will result in then you can have supporting pointer chains to actually link the department information. So, this is a one kind of a design that you have ok.

Now think about; so the whole so, we have; so, far talked about the relations and relations going to either single files or multi table relations; multi table file where multiple relations are on the same file. Now, if you look at the database as a whole. So, what is a data base?

(Refer Slide Time: 12:08)

The slide has a header 'Data Dictionary Storage' with a sailboat icon. The main content discusses what the Data dictionary stores, listing various types of metadata. A footer includes course details and copyright information.

**Data Dictionary Storage**

The **Data dictionary** (also called **system catalog**) stores **metadata**; that is, data about data, such as:

- Information about relations
  - names of relations
  - names, types and lengths of attributes of each relation
  - names and definitions of views
  - integrity constraints
- User and accounting information, including passwords
- Statistical and descriptive data
  - number of tuples in each relation
- Physical file organization information
  - How relation is stored (sequential/hash/...)
  - Physical location of relation
- Information about indices

SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Deshpande, IIT Kanpur Date: Jan-Apr., 2018

Database System Concepts - 8<sup>th</sup> Edition      25.20      ©Silberschatz, Korth and Sudarshan

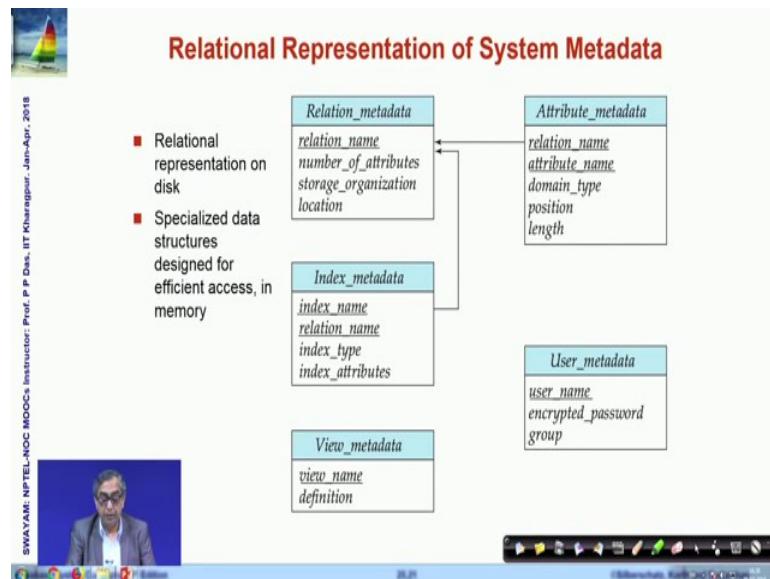
The database as a whole has a whole lot of tables; and so far we have just been focus on the fact that tables we know the tables we know their attributes and the data resides in inside, but if you think in terms of the database, then somebody; somewhere we will need to remember that what are my tables? What are my relations? For a given relation I need to know what are the attributes that the relation has, what is the; you know length type of this attributes I did remember what are the views that I have created on the database the constraints that exist.

So, all of these information which you can say is databases own metadata information needs to be also maintained; and what is done is that; also is maintained as a database within the database system. And such a metadata system is usually known as the name of data dictionary or system catalogues.

So, it has informations like this. So, you put them again, you create the schema design based on the all this metadata information that you need, also you can have you will need to maintain information for users, accounts, passwords and so, on. Then you may have statistical information, where you would like to; we will see the use of statistical information when we talk about indexing in the following week you will see that you need to know, what is the you know how frequently the different queries are fired, what are the number of peoples in each relation and so, on; you may also need to have

information in terms of what has been your choices in terms of the physical locations of file the storage options and so on the index files.

(Refer Slide Time: 14:05)



So, here is a sample one. So, what if you look into; so you can again see a number of schema. So, this is saying that the; this is the relational metadata schema which is talking about, what is the different relations? So, every record here is not keeping the data of your application, but its keeping the information that here you have these different relations. For example, couple of modules back we are talking about the library information system.

So, in the library information system we had different we designed different relations the book issue, the book catalogue, the book copies and so, on. So, those relation names and the how many attributes they have? How will you organize the storage, where is that storage will go to this particular table? Then depending on the kind of index that you are creating we have still not discussed about index we will do subsequently; but those index information can be preserved the view information we can have attribute metadata.

So, it is for the relation name what is attribute name and what is the type of that attributes. So, if the relation name is say book catalogue, then the attribute name is title, then what is the domain type. So, we will say this is a varchar; then the position of that attribute, the length if it is given the user metadata all of this are typical things that will go into this system catalogue or the data dictionary that we will require.

(Refer Slide Time: 15:47)

The slide has a header 'Storage Access' with a sailboat icon. The main content is a bulleted list:

- A database file is partitioned into fixed-length storage units called **blocks**
  - Blocks are units of both storage allocation and data transfer
- Database system seeks to minimize the number of block transfers between the disk and memory
  - We can reduce the number of disk accesses by keeping as many blocks as possible in main memory
- **Buffer** – portion of main memory available to store copies of disk blocks
- **Buffer manager** – subsystem responsible for allocating buffer space in main memory

On the left margin, vertical text reads: SWAYAM-NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr., 2018.

At the bottom left is a video thumbnail of the professor. The bottom right shows the navigation bar and copyright information: Database System Concepts - 8<sup>th</sup> Edition, 25.23, ©Silberschatz, Korth and Sudarshan.

So, finally, the access to the storage the database file as I have been repeatedly saying is partition into fix length units called blocks, because blocks are defined; so that they can be easily allocated and transfer and they are the fastest unit of data that can be transferred between the disk and the memory.

So, unlike many of our typical algorithmic considerations; so when we talk about different algorithms, what we try to minimize? We try to minimize certain expensive operations in the CPU; say the comparison operation or the assignment or may be the memory read operation. But in terms of database systems block is a basic unit of data transfer and the data transfer to and from the disk is the most time taking factor much takes much larger time compare to any in memory operation that we do. So, this kind of becomes the primary unit of cost that we want to minimize.

So, normally we will see that as we talk about index saying and other different kinds of mechanisms, our primary target is to minimize the number of block transfers. So, certainly we can do that by; can reduce the number of disk access by keeping as many blocks as possible in the main memory. So, we can how can you minimize that transfer, if we can keep more of the blocks in our main memory and naturally of course, there is a limitation because main memory is much smaller. So, often we make use of different buffers.

So, a portion of the main memory will be kept to store copies of the disk block. So, every time you need a block you may not want to need to bring it from the; disk storage. So, you keep it in the buffer in main memory, and then you have a management strategy to manage this buffer. So, whenever you want to actually access a record which should be in a particular block; you check whether that block is already available in the buffer; if it is available in the buffer it use that if it is not available in the buffer, then you take it from the disk you will need quite a bit of cycles for that and as you get that from the disk then you keep a copy in the buffer so that it can be used in future.

(Refer Slide Time: 18:33)

The slide has a title 'Buffer Manager' in red at the top right. On the left, there is a small image of a sailboat on water. The main content area contains a bulleted list:

- Programs call on the buffer manager when they need a block from disk.
  1. If the block is already in the buffer, buffer manager returns the address of the block in main memory
  2. If the block is not in the buffer, the buffer manager
    1. Allocates space in the buffer for the block
      - 1. Replacing (throwing out) some other block, if required, to make space for the new block
      - 2. Replaced block written back to disk only if it was modified since the most recent time that it was written to/fetched from the disk
    2. Reads the block from the disk to the buffer, and returns the address of the block in main memory to requester

Now as you keep on doing that, naturally soon you will run out of the buffer memory. So, you will come to a situation, where we need to bring a block from the disk to the memory, but the buffer does not have enough space. So, then we will have to create replace some of the blocks and create space for that. So, here is a basic buffer management strategy.

So, as I said if we if we start if the block is already there in the buffer, then that is given out if the block is not there in the buffer the buffer manager will need to allocate some space how do you allocate space by throwing out some other block which is not required or replace the; then replace the block return back to disk and if it was modified and make space free and then read from the disk and keep a copy in the buffer. So, that is a simple strategy as you can see.

(Refer Slide Time: 19:36)

The slide has a title 'Buffer-Replacement Policies' at the top right. On the left is a small sailboat icon. The main content is a bulleted list:

- Most operating systems replace the block **least recently used** (LRU strategy)
- Idea behind LRU – use past pattern of block references as a predictor of future references
- Queries have well-defined access patterns (such as sequential scans), and a database system can use the information in a user's query to predict future references
  - LRU can be a bad strategy for certain access patterns involving repeated scans of data
    - For example: when computing the join of 2 relations r and s by a nested loops
      - for each tuple  $tr$  of  $r$  do
      - for each tuple  $ts$  of  $s$  do
      - if the tuples  $tr$  and  $ts$  match ...
  - Mixed strategy with hints on replacement strategy provided by the query optimizer is preferable

At the bottom left, vertical text reads: SWAYAM-NPTEL-NOC-MOOCs Instructor: Prof. P.P. Desai, IIT Kanpur-2018. At the bottom right, there is a toolbar and a status bar showing '29.39'.

Now, certainly when you have to replace the block in the buffer, then the question is which block would you replace. Now if you recall from your from similar situations in the in the operating system in terms of memory management, you have read about different strategies for doing replacement and one of the very common strategy more often used is the LRU strategy of the least recently used strategy. So, the idea of behind LRU is use the past pattern of block references as to predict the future. So, least recently used is if this is not been used in the recent past. So, it has less likely hood of being used in the future.

Now, to queries; now here we are trying to do the similar thing in terms of queries. So, they have a well defined access pattern and database system can make use of that and as it turns out LRU can be a bad strategy for example, often you are doing computations in terms of such what you say such nested loops.

So, you have for each tuple you do this; so you have basically trying to do a join. So, you have two relations and you are trying to do a join. So, when you do this when you are going through the inner loop, there will be lot of transfers that will happen; and the original block where you have been holding this is here and you are not accessing that for quite some time.

So, while you are doing this if you if this block, which was which was holding  $r$  at that time if that turns out to be a LRU; then you will throw it away, but with that when you

complete this loop and come back here, you will again have to read it from the disk. So, this is not LRU for such nested computations may not be a good strategy, so maybe some kind of mixed strategy would work better.

(Refer Slide Time: 21:50)

The slide has a title 'Buffer-Replacement Policies (Cont.)' in red. On the left is a small sailboat icon. The right side contains a bulleted list of buffer replacement strategies:

- **Pinned block** – memory block that is not allowed to be written back to disk
- **Toss-immediate** strategy – frees the space occupied by a block as soon as the final tuple of that block has been processed
- **Most recently used (MRU) strategy** – system must pin the block currently being processed. After the final tuple of that block has been processed, the block is unpinned, and it becomes the most recently used block.
- Buffer manager can use statistical information regarding the probability that a request will reference a particular relation
  - E.g., the data dictionary is frequently accessed. Heuristic: keep data-dictionary blocks in main memory buffer
- Buffer managers also support **forced output** of blocks for the purpose of recovery

At the bottom, there is footer text: 'SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P P Das, IIT Kharagpur - Jan-Apr- 2018', 'Database System Concepts - 8<sup>th</sup> Edition', '25.26', and '©Silberschatz, Korth and Sudarshan'.

So, there are several that are used in terms of buffer replacement one is called pin block, where you mark a certain memory block which is not allow to be return back to the disk it has to stay in the buffer or a toss immediate strategy is quite often used. So, it frees the space occupied by a block; as soon as the final tuple has been removed.

So, it is a toss immediate. So, as soon as you are done you just throw it out you are you are done with it so you write it back. Another which is commonly uses most recently used the; if whenever the block is currently being processed, then the system will kind of keep a marker a pin. So, that it is not removed, but after the final tuple has been processed, the block will be unpinned and then it becomes a most recently used block and you can go with defining the most recently used block and having the strategy.

(Refer Slide Time: 23:04).

The slide is titled "Module Summary" in red at the top right. On the left, there is a small sailboat icon. The main content area contains a bulleted list of learning objectives:

- Familiarized with the organization for database files
- Understood how records and relations are organized in files
- Learnt how databases keep their own information in Data-Dictionary Storage – the metadata database of a database
- Understood the mechanisms for fast access of a database store

On the left side of the slide, there is vertical text that reads: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kanpur Date: Jan-Apr- 2018". Below the main content area, there is a video player showing a man in a suit, identified as Prof. P. P. Desai. At the bottom of the slide, there are several small icons for navigating through the presentation.

You can certainly use different kinds of statistical information and in summary; so on this we have talked about the basic organization of database files starting from the fixed record to variable record handling of the different file organization and try to take a look in terms of how records and relations are organized in terms of the in terms of the files and what are the options that we have and we took a look at the data dictionary storage the basic system catalogue, where database keeps its own information.

And then noted that block happens to be the major unit of data transfer between the disk and the main memory and therefore, that is the unit of defining unit of cost that we have to incur, and to minimize that we have a buffering strategy in the main memory, where the disk blocks will be kept a few disk blocks will be kept for quick use whenever required. And there needs to be various different smart replacements strategies for good management of this buffer.

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

**Lecture - 26**  
**Indexing and Hashing/1: Indexing/1**

Welcome to module 26 of Database Management Systems. In this module and the next 4; that is for the whole of this week we will talk about indexing and hashing.

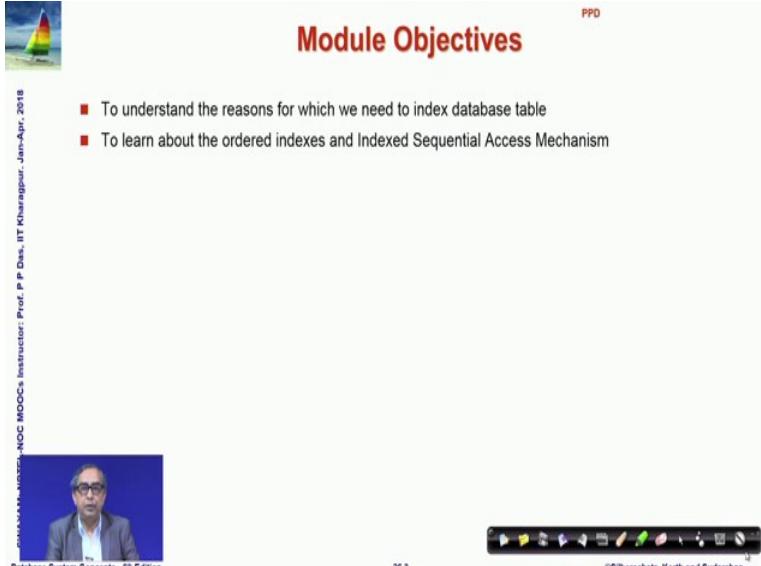
(Refer Slide Time: 00:33)

The slide is titled "Week 05 Recap" in red at the top right. It features a small sailboat icon on the left. The content is organized into four main sections, each with a bullet point and a list of topics. The sections are: "Module 21: Application Design and Development/1", "Module 22: Application Design and Development/2", "Module 23: Application Design and Development/3", and "Module 24: Storage and File Structure/1 (Storage)". A vertical sidebar on the left contains the text "SWAYAM: NPTEL MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018". At the bottom, there is a navigation bar with icons and the text "Database System Concepts - 8<sup>th</sup> Edition", "26.2", and "©Silberschatz, Korth and Sudarshan".

- **Module 21: Application Design and Development/1**
  - Application Programs and User Interfaces
  - Web Fundamentals
  - Servlets and JSP
- **Module 22: Application Design and Development/2**
  - Application Architectures
  - Rapid Application Development
  - Application Performance
  - Application Security
  - Mobile Apps
- **Module 23: Application Design and Development/3**
  - Case Studies of Database Applications
- **Module 24: Storage and File Structure/1 (Storage)**
  - Overview of Physical Storage Media
  - Magnetic Disks
  - RAID
  - Tertiary Storage
- **Module 25: Storage and File Structure/2 (File Structure)**
  - File Organization
  - Organization of Records in Files
  - Data-Dictionary Storage
  - Storage Access

So, this is a quick recap of what we did in the last week primarily talking about application development and then specifically; we focused on storage and file structure that is how a database file can be stored how it can be organized and in a manner so that, we have efficient representation for that now.

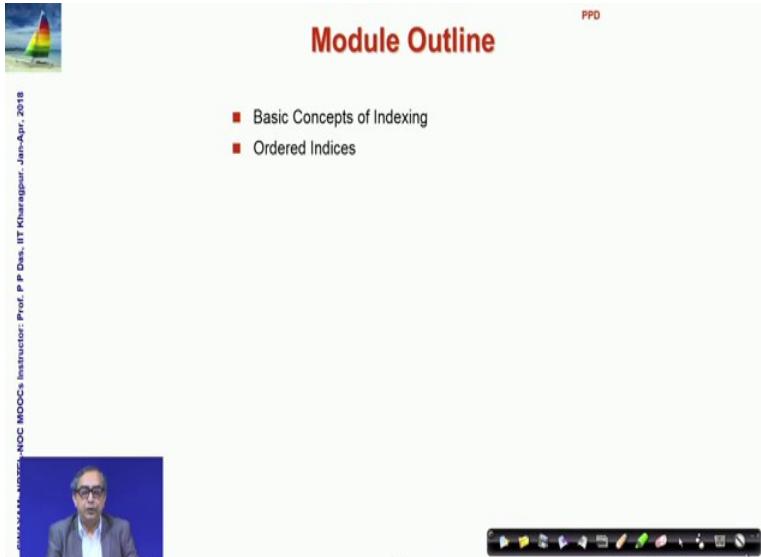
(Refer Slide Time: 01:01)



This slide is titled "Module Objectives" in red at the top right. It features a small sailboat icon in the top left corner. On the left edge, there is vertical text: "CHAKRABARTI-NODES-NOC MOOCs", "Instructor: Prof. P. P. Das, IIT Kharagpur", and "Date: Apr. 2018". The main content area contains two bullet points: "To understand the reasons for which we need to index database table" and "To learn about the ordered indexes and Indexed Sequential Access Mechanism". At the bottom left is a video thumbnail showing a man speaking. The bottom right corner includes the text "Database System Concepts - 8<sup>th</sup> Edition", "26.3", and "©Silberschatz, Korth and Sudarshan". A navigation bar with various icons is at the bottom.

We will move on from there to and focus on the basic feature of a database system, which is the ability to find information in a very fast manner the ability to update in a very fast manner. And we will see the reasons for which we do something on the database tables called indexing and we will talk about ordered index in this module and about the index sequential access mechanism.

(Refer Slide Time: 01:31)



This slide is titled "Module Outline" in red at the top right. It features a small sailboat icon in the top left corner. On the left edge, there is vertical text: "CHAKRABARTI-NODES-NOC MOOCs", "Instructor: Prof. P. P. Das, IIT Kharagpur", and "Date: Apr. 2018". The main content area contains two bullet points: "Basic Concepts of Indexing" and "Ordered Indices". At the bottom left is a video thumbnail showing a man speaking. The bottom right corner includes the text "Database System Concepts - 8<sup>th</sup> Edition", "26.4", and "©Silberschatz, Korth and Sudarshan". A navigation bar with various icons is at the bottom.

So, introduction of the basic indexing concepts and the order indices and we start with the basic concepts.

(Refer Slide Time: 01:40)

The slide is titled "Search Records" and features a table illustrating a search process. The table is divided into three main sections: "Index on 'Name'", "Table 'Faculty'", and "Index on 'Phone'".

Index on "Name"		Table "Faculty"			Index on "Phone"	
Name	Pointer	Rec #	Name	Phone	Pointer	Phone
Anupam Basu	2	1	Partha Pratim Das	81998	6	81664
Pabitra Mitra	6	2	Anupam Basu	82404	1	81998
Partha Pratim Das	1	3	Ranjan Sen	84624	2	82404
Prabir Kumar Biswas	7	4	Sudeshna Sarkar	82432	4	82432
Rajib Mall	5	5	Rajib Mall	83668	5	83668
Ranjan Sen	3	6	Pabitra Mitra	81664	3	84624
Sudeshna Sarkar	4	7	Prabir Kumar Biswas	84772	7	84772

Below the table, there are two sections of bullet points:

- Consider a table: Faculty(Name, Phone)
- How to search on Name?
  - Get the phone number for 'Pabitra Mitra'
  - Use "Name" Index – sorted on 'Name', search 'Pabitra Mitra' and navigate on pointer (rec #)
- How to search on Phone?
  - Get the name of the faculty having phone number = 84772
  - Use "Phone" Index – sorted on 'Phone', search '84772' and navigate on pointer (rec #)
- We can keep the records sorted on 'Name' or on 'Phone' (called the primary index), but not on both.

On the left side of the slide, there is vertical text: "SWAYAM: NPTEL-NOC MOOCs Instructor: Prof. P. P. Desai, IIT Kharagpur - Jan-Apr. 2018". On the right side, there is a "PPD" logo and a decorative footer bar with various icons.

So, consider a very simple example, let us consider an example where there is a relation faculty which has name and phone number and additionally. So, just focus on the middle part the pinkish part of the table which is table faculty besides the two attributes I have put a serial number for the records which kind of can be considered as internal numbers of the database to identify each record.

Now, let us say let us assume that we have to search on names suppose we have a query that get me the phone number of Pabitra Mitra. So, you can see that Pabitra Mitra the record name occurs at position 6. So, if we have to get the phone number, then naturally we have to look at all these entries from one end to the other till we come across Pabitra Mitra match the name Pabitra Mitra and we can access the phone number.

Now, similarly if we have to search for a phone number we will have similar situation. So, if we want to get the phone numbers of the faculty having phone number 84772. Again, we will have to search on the phone number from one end to the other and find the name of the faculty. Now, certainly this will mean that if there are n records in the database then we will need or the order of about  $n/2$  comparisons to be done or accesses record accesses to be done before.

We can find the desired record which is certainly not a very efficient way of doing things and you know from your knowledge of basic algorithms that; if we have a set of data and we want to find a particular one then we can make it efficient by actually keeping the

data in a sorted manner and doing some kind of a binary search that is one one possible mechanism through which you can do that.

So, we can use that same context same concept now and just look at the left side the blue part of the blue part tableware, what I have done have collected all the names of the faculty and I have sorted them in lexicographically increasing order and with that I have kept the record number as a pointer. Now, since this is a sorted array. So, here I can search for Pabitra Mitra for the first query in a binary search. So, at least in the  $\log n$  order of comparisons I should be able to find professor Pabitra Mitra and get the fact; that it is record number is 6 navigate to the record number 6 in the table in the middle and take out the phone number.

Similarly, without disturbing the actual faculty table I can build a similar kind of supportive table on the right the yellow one where I collect all the phone numbers and I have sorted on the phone numbers I can again do binary search on the phone number 84772 and find the phone number find the record number where this phone number occurs and go and find the name of the faculty.

So, you can see that naturally this gives us a alternate way of finding out and certainly you would say that we could have kept the record sorted on name or on phone number, but certainly if we keep it sorted on name the first query we will get benefited the second query will still take a linear time if we get keep it sorted by phone number the first query will take a linear time. So, if we cannot actually keep the data sorted on both and therefore, this is just an alternate mechanism called the indexing mechanism through which even though the original data is not sorted by maintaining some auxiliary data we can actually make our search mechanism more efficient and that is the core idea of indexing.

(Refer Slide Time: 05:39)

The slide has a header 'Basic Concepts' with a sailboat icon. The content includes:

- Indexing mechanisms used to speed up access to desired data.
  - For example:
    - Name in a faculty table
    - author catalog in library- Search Key** - attribute to set of attributes used to look up records in a file
- An **index file** consists of records (called **index entries**) of the form

search-key	pointer
------------	---------
- Index files are typically much smaller than the original file
- Two basic kinds of indices:
  - Ordered indices**: search keys are stored in sorted order
  - Hash indices**: search keys are distributed uniformly across "buckets" using a "hash function"

At the bottom left is a video thumbnail of a professor, at the bottom center is the text 'Database System Concepts - 8<sup>th</sup> Edition', and at the bottom right is the text '©Silberschatz, Korth and Sudarshan'.

So, indexing mechanism is used to speed up accesses to desired data. So, as you have just seen. So, what we search on is called the search key and an index file consists of the index entries as you have just seen it has a search key and the pointer, pointer is the record number or the internal address of the record where it occurs and certainly I have shown an example where there just two attributes in general there will be a large number of attributes. So, the entry of every index would be much smaller than the corresponding record.

So, the overall index file will be a much smaller one than the original and we can index it and there are two basic ways to index and; that is what we will discuss through these modules one is called ordered index where this search keys are stored in sorted order as you have just seen and the other is hash indices where the search keys are distributed randomly across different buckets. Using, concepts of hash function which you must have studied as a part of your data structure course.

(Refer Slide Time: 06:44)

**Index Evaluation Metrics**

- Access types supported efficiently. For example,
  - records with a specified value in the attribute, or
  - records with an attribute value falling in a specified range of values
- Access time
- Insertion time
- Deletion time
- Space overhead

CHAKRABORTY, NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr. 2018  
Database System Concepts - 8<sup>th</sup> Edition

26.8 ©Silberschatz, Korth and Sudarshan

Now, if we mean why should we index on the basic question is should we index on all attributes should we index on one attribute should we index on combination of attributes. So, access. So, there are certain measures to decide as to what is a good way to index. So, that will be that will be measured against the basic requirements of the database that is I should be able to index in a way.

So, that the access time the insertion time the deletion time all these should get as minimized as possible and as you have just seen indexing might mean maintaining additional structures. So, that overhead of space should also be minimal. So, with that with indexing we should be able to do at least certain kind of accesses where a particular value matches the attribute of a record or falls within a range of values in a record those kind of searches those kind of queries should become very efficient and these metrics will have to always keep in mind.

(Refer Slide Time: 07:51)

## Ordered Indices

- In an **ordered index**, index entries are stored sorted on the search key value. For example, author catalog in library
- **Primary index:** in a sequentially ordered file, the index whose search key specifies the sequential order of the file
  - Also called **clustering index**
  - The search key of a primary index is usually but not necessarily the primary key
- **Secondary index:** an index whose search key specifies an order different from the sequential order of the file
  - Also called **non-clustering index**
- **Index-sequential file:** ordered sequential file with a primary index

CHAKRABORTY, NOC MOOCs Instructor: Prof. P. P. Das, IIT Kharagpur - Jan-Apr., 2018

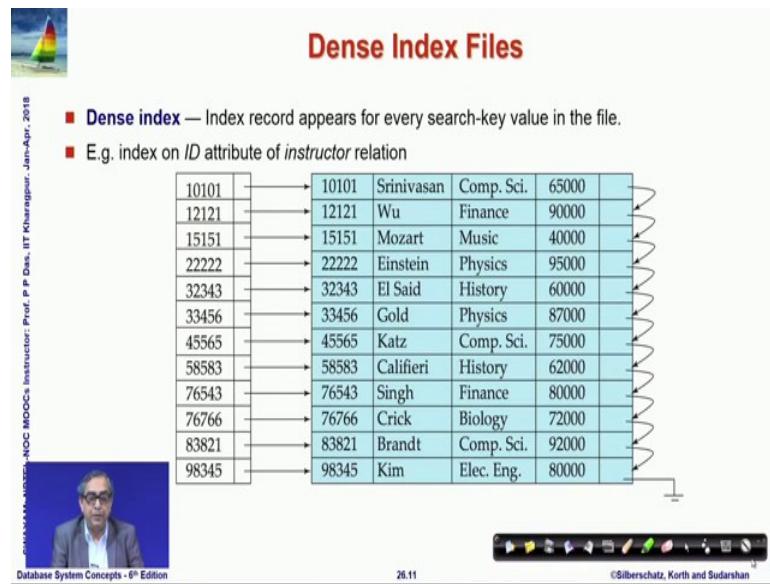
Database System Concepts - 8<sup>th</sup> Edition

26.10

©Silberschatz, Korth and Sudarshan

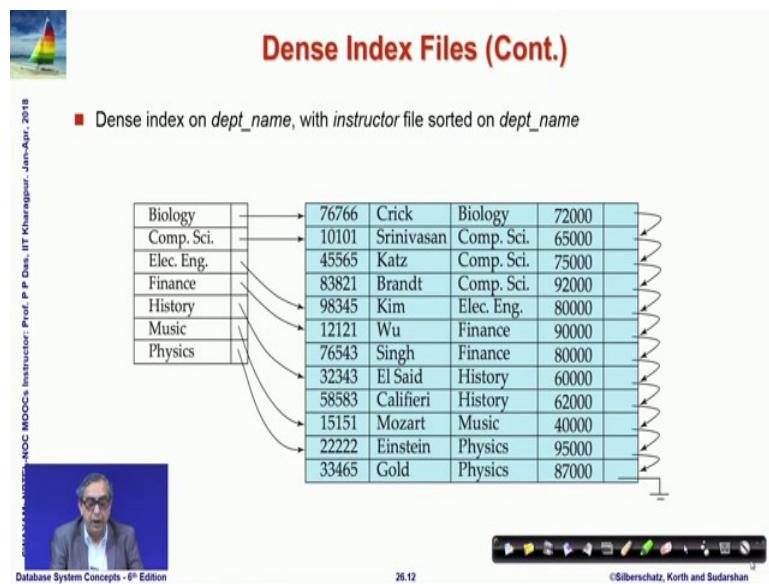
So, let us look at the first concept of ordered index which is pretty much like what we have just sent. So, in a sequentially ordered file the index whose search key specifies the sequential order is known as the primary index. So, the sequential ordering is done based on that primary index it is called also called the clustering index and please keep in mind that the search key of a primary index is usually the primary key, but not necessarily the primary key it could be different from the primary key also and if I create an index who search key is different from the sequential order of the file then we say it is a secondary index and it is also called the non clustering index. So, index sequential file is ordered sequential file which is ordered on the primary index.

(Refer Slide Time: 08:44)



So, here I show an example and this is example of dense index file. So, you can see the blue part is actual table which has different fields the id the name of the faculty, the department, and the salary and this is as you can see that it is completely sorted on the id in the increasing value of id and on the left the white is basically just the ids and with every idea we have a pointer they record the number possibly of the record that it corresponds to here all the indices that feature in the table are also put on the index file and therefore, it is called a dense index you should also note on the right that the records are interlinked through a chain I mean kind of they are being formed in terms of a linked list whose purpose would be seen later on.

(Refer Slide Time: 09:40)

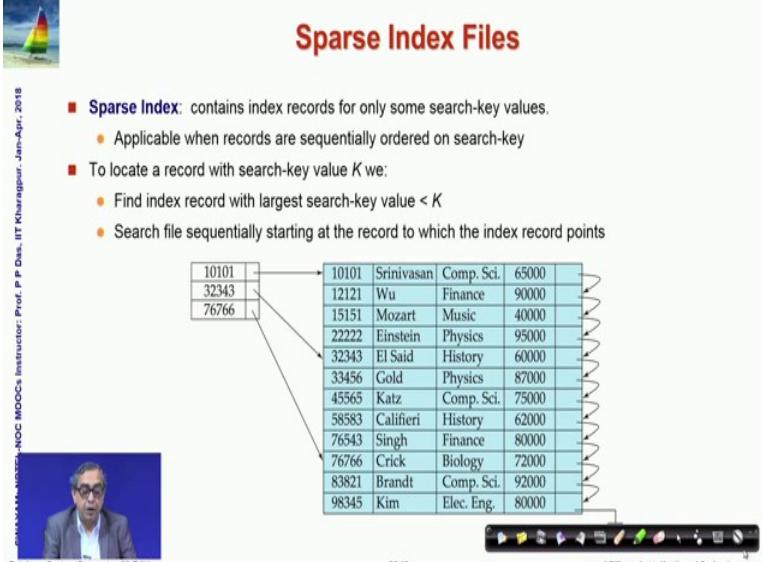


Now, instead of doing id if I want to do a dense index on department name, then naturally the sequential file has to be indexed primarily on the department name and as you can note that unlike the id which is also the primary key and therefore, every id value was unique the department name is not a primary key it is a different attribute which allows for multiple value multiple records having the same attribute value and therefore, when we sort we have one instance of biology, but three of computer science two of finance and so, on.

So, when we make the index we made the index collect all the distinct values of the department names and for every department name we put a pointer we put the index marking the first record in the sequential file with that department name. So, you can see that your computer science points to the record starting with 1 0 1 0 1 and then the; it goes on to the next two records.

So, now you can understand why we need the link list; because if we are looking for the all those teachers all those faculty who are associated with department computer science, then I can find it on the index file with the department name, computer science traveled to the record first record in that which is of Srinivasan and then keep going on this list through the linked list that we have on write and find the record for Katz and record for Brandt and till we moved to the next record for Kim which does not match the department name anymore.

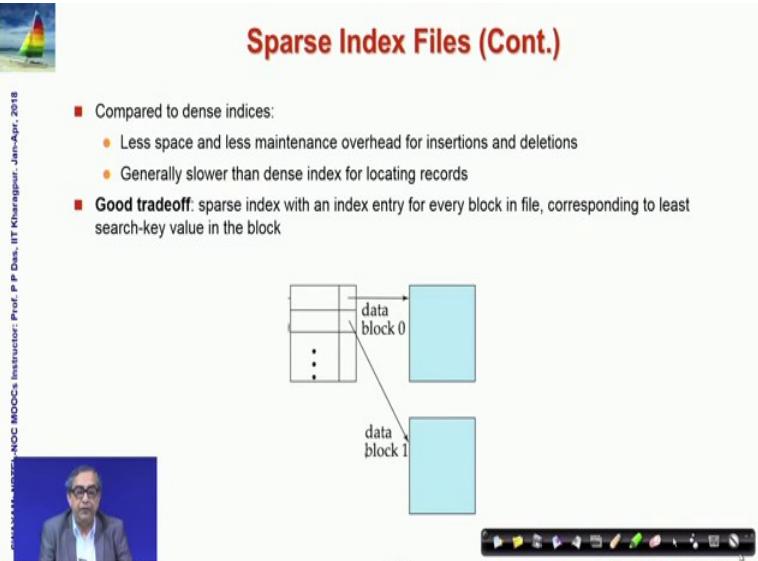
(Refer Slide Time: 11:26)



The slide is titled "Sparse Index Files". It features a small sailboat icon in the top left and a video camera icon in the bottom left showing a person speaking. The main content area contains a bulleted list and a diagram. The list describes a sparse index and how to locate a record. The diagram shows a sparse index structure with three indexed entries (10101, 32343, 76766) pointing to a sequential file of 15 records. The file records are: 10101 Srinivasan Comp. Sci. 65000, 12121 Wu Finance 90000, 15151 Mozart Music 40000, 22222 Einstein Physics 95000, 32343 El Said History 60000, 33456 Gold Physics 87000, 45565 Katz Comp. Sci. 75000, 58583 Califieri History 62000, 76543 Singh Finance 80000, 76766 Crick Biology 72000, 83821 Brandt Comp. Sci. 92000, 98345 Kim Elec. Eng. 80000.

Now instead of dense index we could also do sparse index. With parse index it means that instead of maintaining all the search key values that exist in the file we just take a subset of that and we maintain those in the index file. So, here again we are showing a index sequential structure with id being the primary index and we have chosen three of the ids and kept them in the index file. So, in the sorted order so, this if you want to say look at a record with search K value K we can find the index record with the largest key value largest search key **value id < K** and then search sequentially or onwards because these are all linked together in the sequential manner.

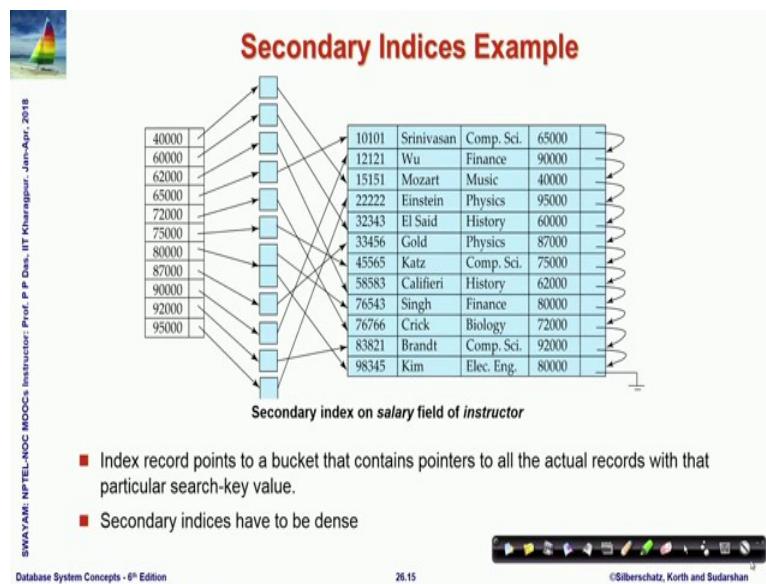
(Refer Slide Time: 12:18)



The slide is titled "Sparse Index Files (Cont.)". It features a small sailboat icon in the top left and a video camera icon in the bottom left showing a person speaking. The main content area contains a bulleted list comparing sparse indices to dense indices and discussing good tradeoffs. The diagram shows a sparse index entry with three pointers to sequential data blocks labeled "data block 0" and "data block 1".

So, naturally compared to the dense index of sparse index takes less space and therefore, less overhead for maintenance when we do insertion deletion you must have already noted that if I were dense index then every time I insert a value it will have to be the dense index file will also have to change changes will be required there for insertion as well as for deletion for sparse index it will not be required, because it will just be required when I am actually changing the record or creating a record which is existing on the sparse index.

(Refer Slide Time: 13:11).



So that way it is better than the dense index, but certainly in the dense index I can go to any record directly from indexing in the sparse index I need to first index and then go sequentially. So, it will be in general slower in terms of performance and will need to look at this tradeoff now it is also. So, these were the ways to do the primary indexing where we decide the order in which we actually keep the record sorted or the fields on which we do that and the index associated with it, but once since the data can be primarily indexed on one or couple of attributes and that gets fixed.

But we may want to search for other values also to make that efficient we create a secondary index. So, here we show an example where the primary index is id. So, the whole recall records are sorted in terms of that and we are creating an index on the salary. So, that we can quickly find the salary of and given the salary of an employee we can quickly find the employee or the employees who get that salary.

So, you can see that for secondary index now it is quite possible that there are multiple entries for the same value of salary for example, if you look at the salary eighty thousand that is received by Professor Singh as well as Professor Kim. So, if you look at the index file of the index sequence of the salary values you will find that against 80,000 we have two entries which mark to two different records corresponding to the faculty who are drawing that salary. So, secondary index naturally has to be dense and is created when we want we feel that there is a need to quickly search on that field or that set of fields and we will discuss more about those issues later on.

(Refer Slide Time: 14:46)

The slide features a sailboat icon in the top left corner. The title 'Primary and Secondary Indices' is centered in red. To the left of the title is a vertical column of text: 'MOOCs-NOC MOOCs Instructor: Prof. P. Das, IIT Kharagpur - Jan-Apr. 2018'. Below the title is a bulleted list of points:

- Indices offer substantial benefits when searching for records
- BUT: Updating indices imposes overhead on database modification --when a file is modified, every index on the file must be updated
- Sequential scan using primary index is efficient, but a sequential scan using a secondary index is expensive
  - Each record access may fetch a new block from disk
  - Block fetch requires about 5 to 10 milliseconds, versus about 100 nanoseconds for memory access

At the bottom left is a small video thumbnail showing a man speaking. The bottom right contains navigation icons and the text 'Silberschatz, Korth and Sudarshan'.

So, indices offer substantial benefits when searching for records, but updating index impose over it; because if you create an index whenever you insert a record or a delete a record or you change the value of a field which is indexed in a record then certainly all these indices will have to be also updated and therefore, while your access time significantly reduces, because of indexing your actual update insert delete update time will increase and therefore, the indexing will have to be done carefully.

So, sequential scan using primary index is efficient, but sequential scan using secondary index is expensive. So, you will have to bring them in blocks and that will require couple of millisecond versus the amount of time that you need in the memory access. So, these are the factors that we will have to take into consideration and we will talk more about those.