


Database Systems
Prof. Sreenivasa Kumar
Computer Science and Engineering Department
Indian Institute of Technology-Madras

Lecture-02
Database Architecture

Okay, so let us begin in the last lecture I was talking to you about this idea of data models right. A data model, let us briefly recall that and then proceed with the rest of the material.

(Refer Slide Time: 00:35)




Data Model

Collection of conceptual tools to describe the database at a certain level of abstraction.

- *Conceptual Data Model*
 - a high level description
 - useful for requirements understanding.
- *Representational Data Model*
 - describing the logical representation of data without giving details of physical representation.
- *Physical Data Model*
 - description giving details about record formats, file structures etc.

7

Prof P Sreenivasa Kumar
Department of CS&E, IITM



Basically a data model will allow us to describe the information system or the database system at a certain level of abstraction, I want you to understand clearly is that why do you need several levels of abstraction for the information system. We need that mainly because information system is a very complex entity, it is a sometimes a large pieces of information and very detailed information is you know descriptions possible, etc.

So it is a complex entity and so, we need to have a description of this thing this entire information system at a level where you know, it can be easily understood by the stakeholders of this information system. There are several stakeholders for this information system. First thing is the people who want it to be built and then use okay. And then we have what are called I will tell you a little later what are called main users of the information system.

Who just want to interact with the information system, just feed information and then you know like that. So, there are various groups of people who are interacting with this information system. And when we are collecting the requirements for building an information system, there is really no need for us to kind of expose all the internal details of how this information system is going to be organized to the stakeholders.

First thing is, they would not like to know, they do not want to know those details. So we should have the ability to hide those hide certain unnecessary details from these users. And then describe the database at a level of abstraction which is comfortable for them. So that is why we need various again, of course when you have to build the system itself. Then we need all the details of that.


So, we as computer science people would like to look at the information system in a much more detailed level right. And so we need a different abstraction level for us and ultimately the information system, you know is going to be stored on a secondary storage, which is discs and tapes if necessary, if the amount of information is too huge, then we may even want to use tapes, the usage of tapes is very limited these days.

But then the disk the actual database is going to lie there in the storage medium on the disks. So, we need some to understand how exactly the database is going to be organized in the form of always as files on disk systems, so that is a different level of detail for me. So, because there are these various levels of details at which we can describe a database system, we need different toolkits for doing that.

So basically a data model provides that toolkit, the conceptual toolkit to kind of describe the database as at a level of abstraction. So, for a high level description, will use this conceptual data models and this is basically useful for understanding the requirements and also collecting them. We use representational data models to describe the database at a logical level.

This is what actually we computer science people would like to use, okay. So, this will give the full logical details about the database. And then we have also a data model that can be used to describe the full details of the record formats, file structures everything. And that is what we normally call as a physical level data model physical data model okay. So, let me spend a little bit time with each of at least the first 2 of these things.

(Refer Slide Time: 05:36)



E/R (Entity/Relationship) Model

- A conceptual level data model.
- Provides the concepts of *entities*, *relationships* and *attributes*.

The University Database Context

Entities: *student*, *faculty member*, *course*, *departments* etc.


Relationships: *enrollment* relationship between student & course,
employment relationship between faculty member, department etc.

Attributes: *name*, *rollNumber*, *address* etc., of *student* entity,
name, *empNumber*, *phoneNumber* etc., of *faculty* entity etc.

More details will be given in the E/R Model Module.

Prof P Sreenivasa Kumar
Department of CS&E, IITM

8




So, a popular conceptual level data model is what is called the entity relationship model. E/R model. So what this provides is basically, concepts like entities, relationships and attributes, basically entities are things that we would like to keep track of the important things in the information system that we would like to keep track off. And relationships basically, or associations between these important entities in the religion in the information system that we would like to keep track of.

And attributes are like, for example, the details such as the names, roll numbers, addresses of a student entity. And name employee number, phone number of faculty entity. These are specific attributes. And here are some examples of relationships. So, for example, the enrollment relationship between student and course, students enroll for courses and we'd like to keep track of these association.

And the employment relationship between faculty and a specific department. Now, so the it is very easy for us to kind of make use of these terms like, you know, entities and relationships and attributes. These are the conceptual tools that we will use, and then sit across with the stakeholders who want this information system to be developed, and then discuss with them what are your entities, what are the relationships you bother about, etc.

And then describe that entire database in terms of these 3 concepts, there is a few more concepts, then we will see them when we actually discuss this model in a separate module in the course. Okay, I will give you more details little later, but I just want to give you some idea about what conceptual level data model looks like okay, any questions in this.

(Refer Slide Time: 08:13)



E/R (Entity/Relationship) Model

- A conceptual level data model.
- Provides the concepts of *entities*, *relationships* and *attributes*.

The University Database Context

Entities: *student*, *faculty member*, *course*, *departments* etc.


Relationships: *enrollment* relationship between student & course,
employment relationship between faculty member, department etc.

Attributes: *name*, *rollNumber*, *address* etc., of *student* entity,
name, *empNumber*, *phoneNumber* etc., of *faculty* entity etc.

More details will be given in the E/R Model Module.

Prof P Sreenivasa Kumar
Department of CS&E, IITM

8



Then we will move on to a popular representational level data model, which is actually the relational data model. Relational data model is going to be the central you know concept for this course and we will spend a lot of time understanding this relational data model and also other data is connected with the relational data model. So, to give you a brief idea about this basically it provides this concept of a relation.

And a relation is nothing but what we are familiar with in discrete mathematics course, which is a subset of cross product of sets, basically, a relationship. So, we are going to use that and that will play a central role in this data model. So, for example, in the context of a university data

model database, we can think of a relation called student. So, this is the name of the relation and then there are here as the various attributes for the student relation.


So, yes name, roll number, the joining year, birth date and the program in which the student is enrolled for and what is the department in which the student is studying. So, these are the attributes and we also call that as the schema of the relation. And here is a specific pieces of data, which we will talk about a specific student called Sriram, who has all these details.

So, we call this the entire thing as a data tuple okay, data tuple okay and a relation is basically will consist of a set of such data tuples set, okay. Please remember this definition with calling it as a set of tuples not a sequence of tuples. There is slight difference between these 2 things right. So, this is a set of tuples, a finite set of tuples will constitute our relational data instance we call it instance.

So, in general relational database is going to have several relations with these kind of attributes etc. and a bunch of these finite number of these relations will constitute our relational database and we will go into much more details about this relational model when we take it up in when we go into the relational data model okay. Now, let us move on basically the I want you to understand that there is this important notion called a data model which is a collection of conceptual tools to describe the database at a certain level of abstraction.

You can see that this level of abstraction this gives you, you know a different tool to describe the database. This is the relation to whereas in the previous slide, we have seen entities and relationships because in relationships, etc. okay, so please do not hesitate to stop me if you have any questions.

(Refer Slide Time: 11:58)




Data versus Schema or Meta-Data

- DBMS is generic in nature
 - not tied to a single database
 - capable of managing several databases at a time
- Data and schema are stored separately.
- In RDBMS context:
 - Schema – table names, attribute names with their data types for each table and constraints etc.
- Database definition – setting up the skeleton structure
- Database Loading/populating – storing data

Prof P Sreenivasa Kumar
Department of CS&E, IITM

10



Now, this point we have also mentioned earlier that the distinction between data and metadata is one of the central ideas in the database systems field and the DBMS database management system basically becomes generic in nature, because it is able to separate these data and metadata and then store them separately. So, when a DBMS has to deal with a particular database, okay.

What it will do is to console the metadata corresponding to that particular database. Open that up in some is as you know, and then plan to do certain modifications or whatever is required for that particular database. And that is why it is not tied to a single database. It is capable of managing several databases at a time. I will give you some more details about the architecture of a RDBMS as we go along.

So in our DBMS, relational database management system context, the schema, basically, or the metadata basically consists of all these table names and we call them table also, it is actually in the last slide I call it as relation right. Relations are informally called also as tables. So the names of the relations, names of the attributes, along with their data types for each of these relations.

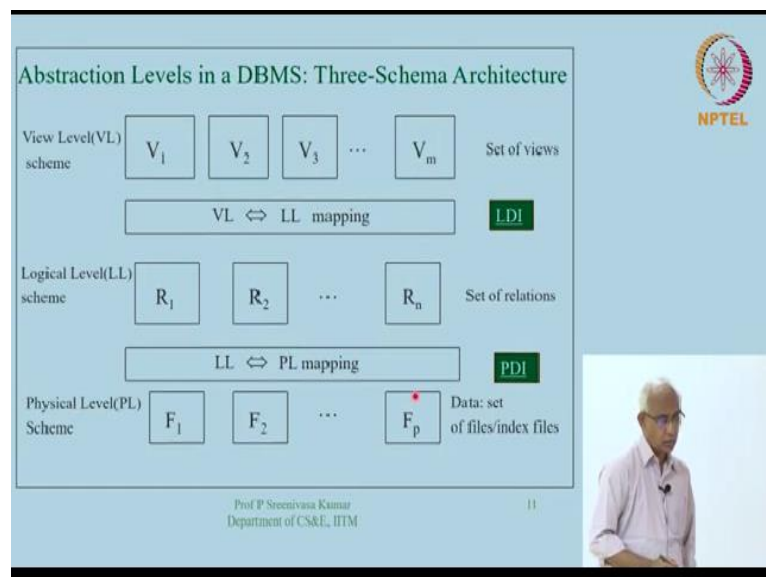
And we will also see later on that there are certain kinds of constraints. For example, if you put an age attribute there, you cannot put a negative number right. So, there are certain kind of constraints about these attributes and also about table relations in general. So, all this information

together is what is called the metadata or the schema information and this will be stored separately from the data.

And then whenever we have to deal with the database, we will conserve this schema and then operate on the on the database. So, the database definition basically talks about you know when you say that we are defining a database, what we mean is first designing this is that schema for the database and then we will when you say database loading data is getting updated what we mean is to actually store data into this the skeletal structure which we have set up.

The skeletal structure is basically the table of tables with table names, attribute teams, etc. at blanks in it. So that is what the distinction between the schema and data versus schema. This is a very central idea in database systems. Okay, so moving on.

(Refer Slide Time: 15:50)



Let me take you to this interesting architecture diagram which describes the internal abstraction levels that are available within a database management system. Now, we are talking about a database management system, specifically relational database management system and what are the different abstractions it provides in as part of its architecture. So this is popularly called the three schema architecture.

And we will see how exactly what are the various things that are there. Now, basically, the middle layer is the actually the important layer, which is this logical level representation of the database. It consists of a set of relations. Guys I was previously telling you, the relational database basically consists of a bunch of relational relations as set of relations. So, this is what is the representational data model, which is the relational data model and the description of the database is given as a bunch of relations.


Now at a low level of detail, we have the actual data is sitting in files, there are a set of files, okay, which are actually organized on the disk. And there are also what are called index files. Basically, index files help us to retrieve record efficiently from a data file. A data file is the one that actually contains a lot of records in it. Each record basically is somewhat similar to our structure that you have defined as part of C programs etc. right.

So, these are records. And then we also have index files, index files are again basically, they implement some kind of a data structure, and then they will help us retrieve a data record from a data file efficiently. So, we will spend some a lot more time trying to understand what are the various kind of indexing techniques later on in the course. But right now, we can just say that index files help us locate data records efficiently.

So, the data is actually lying in the physical level on the disk in the form of these files index files and a few other things, okay. So that is the physical level scheme. So to say the metadata at that level. So the metadata here basically consists of what is a file, what is the record structure etc. is that record a fixed length record or is that record having variable number of fields in it, various other details like that.

So, that is the physical level scheme. So, we will come back to this diagram. Let me first take you, okay, so at the top level is this view level.

(Refer Slide Time: 19:47)



Three-schema Architecture(1/2)

View Level Schema

Each view describes an aspect of the database relevant to a particular group of users.

For instance, in the context of a library database:


- Books Purchase Section
- Issue/Returns Management Section
- Users Management Section

Each section views/uses a portion of the entire data.

Views can be set up for each section of users.

Prof P. Sreenivasa Kumar
Department of CS&E, IITM

12



So, I have a slide for that. So, let us look at what is view level. So, view level logical level physical level. These are the 3 levels we are talking about. The view level basically consists of a finite number of views, what is the view, a view describes certain aspect of the database which is relevant to a particular group of users okay. A aspect of the database which is relevant to a particular group of users, let me give you an example.

For instance, let us look at the context of a library database system. Okay if you have gone to the university library or the institute library, you will see that there is this place where issues of books, return of books and all of them are managed, and most of them you will interact with that particular section because that is where you borrow items, return items, probably pay fines and things like that.

If you are delayed in returning an item, they will collect, you collect fines from you and all that right. So, that is the issue return management section. And then when new users, new students come into the campus, they need to be enrolled into the library. So they have to be given smart cards or they have to be given their details need to be taken down, etc. So, that is the users management section you will see that it is there in a separate place okay.

And then of course, the library has a huge collection of books and so faculty keep requesting for new books to be added to the library, etc. And so the books have to be purchased. So there is a

procurement section. A complete section dedicated for interacting with the suppliers of books, placing orders for books, receiving books, etc. that you do not see, because you do not need that, right you do not need to know it is it is kind of hidden from you.

But to manage a library of our kind, you can imagine that you need a separate section, which will keep taking the requests from the faculty taking appropriate approvals, you know, looking at the budget and then placing orders receiving books and managing books. Once the books are purchased, then they will be stopped and so that issues and returns can happen. And so, **so** you can see that a large even a enterprise like a library has different groups of people who are concerned with different aspects of the same enterprise.

It is the same library enterprise, but you can see that there are a group of people who are bothered about books, purchase of books, dealing with suppliers and things like that, there is a group of people who are actually are issuing and written the books returning and managing all that. So, you can see that these are best model as some kind of a view of the entire database specifically meant for a particular section like a book purchase section okay.


So you can see that each of these sections views a portion of the entire data. For example, the books purchase section does not really bother about what is happening in the issue return section. There is not concerned with that. In a similar way, this issue returns section, you know, simply does not bother about what the book purchase for where people are doing, they have their own task cut out for them.

So, they keep doing. So this is little interaction between these 2 people, but all of them are interacting with the same enterprise database system. Now, how nice it would be to kind of you know, give you a view of the entire enterprise to this book section people separately, so that they can always deal with that part of the database only okay. And this is possible by what is called creating of these views.

So, we can create views which are nothing but actually virtual relations. Okay, they can answer they may not be actually be physical you know relations as part of the entire relational schema,

but they will be some kind of virtual relation. So, it is as far as that group is concerned that relation exists and they can interact with that schema. So, a part of the, so that is what we actually mean by a view. A View describes an aspect of the database relevant to a particular group of users.

(Refer Slide Time: 25:37)



Three-schema Architecture(2/2)

Logical Level Schema

- Describes the logical structure of the entire database.
- No physical level details are given.

Physical Level Schema


- Describes the physical structure of data in terms of record formats, file structures, indexes etc.

Remarks

- Views are optional
 - Can be set up if the DB system is very large and if easily identifiable user-groups exist
- The logical scheme is essential
- Modern RDBMS' s hide details of the physical layer

Prof P Sreenivasa Kumar
Department of CS&E, IITM

13



Whereas the middle layer, which is the logical level schema, this describes the entire database. But of course, no physical level details are given at the logical level schema. Now, this is of course, useful because this is as I was telling you that this is the most important schema because it gives the entire logical structure of the whole database. Now, the physical structure of data in terms of as I was telling you record formats, file structures, indexes is given by the physical level schema.

Now, views are actually optional, you know if your enterprises not to date and you know, if there are no easily identifiable user groups who want a particular you know, aspect or a view of the database then we can probably not create these views at all okay. So, it is up to the modern all the modern relational database management systems, the software do provide this option of creating views.

So that when a group of people want to take a restricted view of the entire database and then work with that, it is possible for us to enable that in the RDBMS okay. So, the views are actually

optional. And the whereas the logical scheme of course is essential because that is the one that describes the entire database in its full form and the physical level most of the modern RDBMS software is actually completely hiding this physical level.


And giving the even the database administrators giving a limited access to this physical layer, we will see why it is it is happening. You may want to actually have access to this physical layer as to how the data is actually stored on files, formats and things like that, if you want to tune the performance of the database, how is the database performing for certain kinds of queries, if the database is taking unusually long time, you would like to go investigate as to why it is happening.

And then you may want to kind of tinker the data at the physical level. But the trend is to hide the details of the physical layer, but then of course, when there will be ways in which you can actually access the physical level details okay. So I hope this three scheme architecture is kind of clear. This is not to be confused with the data models that I was mentioning just a while ago, okay.

The 3 kinds of data models that I was mentioning a while ago. So, a while ago I mentioned conceptual level data models, those conceptual level data models are different in the within relational data model, which is a representational level data model. We are now looking at the implementation of the relational data model. The RDBMS is the one that implements the relational data model.

And while implementing the relational data model, it provides these kind of other, you know, facilities. One of them is the facility of creating views okay, and sometimes it is very useful to have that facility. Now, within this context we traditionally describe 2 things called the physical so let me take you to that slide.

(Refer Slide Time: 30:20)



Physical Data Independence

The ability to modify physical level schema without affecting the logical or view level schema.

Performance tuning – modification at physical level creating a new index etc.


Physical Data Independence – modification is localized

- achieved by suitably modifying PL-LL mapping.
- a very important feature of modern DBMS.

Three Schema Arch

Prof P. Sreenivasan Kumar
Department of CS&E, IITM

14



We call it physical data independence, okay. The notion of physical data independence is important. What it basically says is that I should have the ability to modify my physical level schema without affecting the logical or the view level schemas okay. So, the database management system should be so architected such that, I should have the ability to kind of modify the physical level details like you know, creating a new file, inserting it into a system or taking an existing file and splitting it into 2 different files etc.

Such kind of modifications I should be able to do without affecting the logical and the view level descriptions of the levels of the schemas. Why would you require that, first thing is that once you describe this database there are these what are called applications okay, that will be built based on the assumption that the database has all these details okay. So, what exactly is an application.

An application is a program written in a typical high level language like C, C++, Java which uses SQL and interacts with the database and modifies the database, retrieves results from the database okay. So, these application programs are very important and they are the ones that actually update the database okay and they in fact implement the day to day requirements of the enterprise for modification of the database okay.

Now, supposing I have already set up a few application programs like this, I do not want to redevelop all those application programs okay, because a new file has been added into the

database, you know, for example, okay. So is it really possible. That is the kind of question that we are facing. So, if it is really possible, then we will call that as a physical data independence. And actually, the three schema architecture enables this.

How does it actually do that and first of all, why do we have to do any modifications at the physical level that I supposed to be clear, because we may want to do performance tuning, a certain application is running slow, because it is trying to access the database in a certain way. And we now find that after looking through the application, we now find that if we have an additional file, which stores a slightly different kind of records.

Then this application will now become faster. And so we would like now add that file at the physical level. So this is what is called performance tuning. And so, because of this, we should be able to, we want to have this ability of doing a modification to the physical level. And ideally, if we do this physical level modification we will not like the logical and view level schemas to change okay. So that is this ability that we are talking about. So, how is it achieved.

It is achieved by ensuring that the modification we have done is in some sense localized okay. And nothing comes free. So how does it exactly come. It is achieved this particular ability for physical data independence is achieved by suitably modifying the physical layer to logical layer mappings okay. Now, it is time to go back to this three schema architecture picture, I think I can just go back like this okay.

So, I talked about this set of relations and then I talked about this set of views, these are the ones that provide specified you know use view for a group of people etc. Now is the time to talk about these boxes that are lying between these layers. So, this is a box that will that basically captures the logical level to physical level mappings, what basically it consists of is okay, you have a relation, let us say R1.

How is it actually implemented. What are the files, where are the remembers the relation R1. R1 is a set of tuples. It is a set of tuples okay. Each tuples consists of a bunch of values. So it is a conceptual thing. Now physically, it has to be stored as a each of those tuples might be stored as

a single record, or each of those tuples might be stored as 2 records. We do not know exactly how it happens okay.

So there is a bunch of ways probably 1 file or probably 2 files or 3 files that are kind of implementing this relation okay. So that details are actually here. How is this related, mapped to actual physical right. Now, that detail is there in this particular layer box. And so, if the basically this physical data independence is basically achieved by suitably modifying this mapping, if you create a new file now at the physical layer.

Then we will record the details, will modify the details of this logical level physical level mapping suitably to reflect this new situation that we now have a new file and so and so relation that was not using this particular file will no use that etc. whatever is the appropriate modification will have to record it here. And with that we will be able to, now any application that was using relation are 1 will continue to work.

Because now the software the RDBMS package can looking while you know, servicing the request for version R1 will go through these mappings, new mappings and then look up the appropriate files okay. So that is how logical physical data independencies achieved. In a similar spirit we have okay.

(Refer Slide Time: 38:30)

Logical Data Independence

The ability to change the logical level scheme without affecting the view level schemes or application programs

Adding a new attribute to some relation

- no need to change the programs or views that don't require to use the new attribute

Deleting an attribute

- no need to change the programs or views that use the remaining data
- view definitions in VL-LL mapping only need to be changed for views that use the deleted attribute

Three-schema Architecture

Prof P Sreenivasa Kumar
Department of CS&E, IITM

NPTEL

15

So, another similar notion is this notion of logical data independence. So, the ability to change the logical level schema without affecting the view level schemas and the application programs is what is called logical data independence. Now, why do you need to do this in the first place. For example slightly you know adding a new attribute to some relation, because then the designers have now realized that they also need to keep track of this new attribute, which was not earlier thought of it is a term.

Now that has to be added. And certain attribute has to be deleted, let us say. So if these requirements come, then how do we handle them. That is the thing that we are talking about. So of course, if you add a new attribute, then to add some relation, then what we would like to do is that with this situation we like is that there should be no need to change the programs or the views that do not require to use this new attribute. Obviously, things that want to use this new attribute will have to be changed because this is what has come into picture.

So those things that are not using this new attribute can continue to be used as they are in a similar way, if we delete an attribute all those are programs that you know use the remaining data there should be no need for changing them. Whereas, the ones that are using this particular deliverable it obviously has to be changed. So, how do we achieve this logical data independence. Again, it is the views which are on the top layer, again not defined in terms of the lower level relations okay, as I was briefly mentioning earlier.


Views are basically nothing but virtual relations. So, those views would be defined in terms of the actual relation that there are the logical layer, we will see how exactly we will start to be defined when we go into the relational data model in detail. So, the logical data independence is achieved by looking at the real view level to logical level mapping and then suitably modifying that mapping we will be able to achieve this logical data independence okay.

So, are there any questions on these three schema architecture, it is a kind of a abstraction, layers of abstraction that are provided within the RDBMS package or the DBMS system. Of course, if you look at some RDBMS they may not provide you know, the ability to create views. Okay so they kind of implemented the RDBMS in a partial manner. They think that okay, the users of

their RDBMS may not really require views and things like that we are not probably handling such large enterprises.

So they may, you know, optionally not provided. But the ability to create views is part of SQL standard. And so any RDBMS package has to implement the SQL standard. And, and so it should be impossible, it should be available okay.

(Refer Slide Time: 42:44)




Development Process of a Database System (1/2)

Step 1. Requirements collection

- *Data model requirements*
 - various pieces of data to be stored and the interrelationships.
 - presented using a conceptual data model such as E/R model.
- *Functional requirements*
 - various operations that need to be performed as part of running the enterprise.
 - acquiring a new book, enrolling a new user, issuing a book to the user, recording the return of a book etc.

Prof P. Sreenivasa Kumar
Department of CS&E, IITM

16



Now, let us look at the development process of a database system. How exactly we are going to do the development of a typical database okay. The first thing is we have to collect the requirements, we have to sit with the users end users of this information system and collect our requirements and for this and there are 2 kinds of things that we have to collect. First thing is what is called the data model the data model requirements.

That means, what are the various pieces of data to be stored and their interrelationships, this information and this is typically presented using a conceptual level conceptual data model like an E/R model, E/R model is entity relationship model. You also have some alternatives actually at the conceptual level model which is UML I am sure you would have heard about UML also unified modeling language.

UML is more used when you focus on you know, large scale software development. Whereas in the in the database world, the entity relationship model is more popular both of them are conceptual level data models. So, these data model requirements are presented in the form of E/R model. And then there is what is called the functional requirements from the end users that were to collect.


Functional requirements are what are the various kinds of operations that need to be performed as part of the running of the enterprise, like for example, in focusing on the library database system they routinely have to do these activities, right acquiring a new book, enrolling a new user. New students come into picture. So they have to enroll new users issuing a book to the user, recording the return of the book from a user.

All these are their functional requirements, they have to have these things. Now, each of these functional requirements will translate to an application program. The application program is written in some high level language like Java or C++. And it typically would have some graphical user interface and things like that. And it collects whatever parameters it wants to collect from the end users.

And then interacts with the database server through SQL and do the appropriate modifications in the database server and then get back to the end user. So we will see how exactly application programs have to be developed much later in the course. But these are the application program requirements, these are functional required. So, these are the 2 kinds of requirements that will need to collect before we start designing a database.

Now, once the data model requirements are collected, you would typically represented in a entity relationship model and this model we will see more details about this model in the in the next module. Basically, it also has a diagrammatic representation. So, we draw the diagram for the information system. And then the main aim here is to ensure that we have collected the correct requirements. And the end user also understands what exactly he wants okay. So, that is the requirements collection phase. This is a very important is.

(Refer Slide Time: 47:08)



Development process of a database system (2/2)

Step 2. Convert the data model into a representational level model


- typically relational data model.
- choose an RDBMS system and create the database.

Step 3. Convert the functional requirements into application programs

- programs in a high-level language that use embedded SQL to interact with the database and carry out the required tasks. *

Prof P Sreenivasan Kumar
Department of CS&E, IITM

17



Then in the next step, we will convert the data model into a representational data model which is the relational data model. So, whatever is the information that we collected as per the requirements of the database, we will internally have to will first have will have to convert that into the relational data model and choose an RDBMS system and then create the database. So, we will study as to how exactly one has to convert the information that is there in the conceptual data model into the representational net model when we study both these models in detail.

And then convert the functional requirements into application programs, programs in high level language that typically use what is called embedded SQL. So we will have to see how we can embed SQL into a high level program, they typically use embedded SQL to interact with the database and actually carry out the required tasks. So these are the various steps that are involved in the development process of a database system.