

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

**Lecture - 02**  
**Introduction to DBMS/1**

(Refer Slide Time: 00:42)

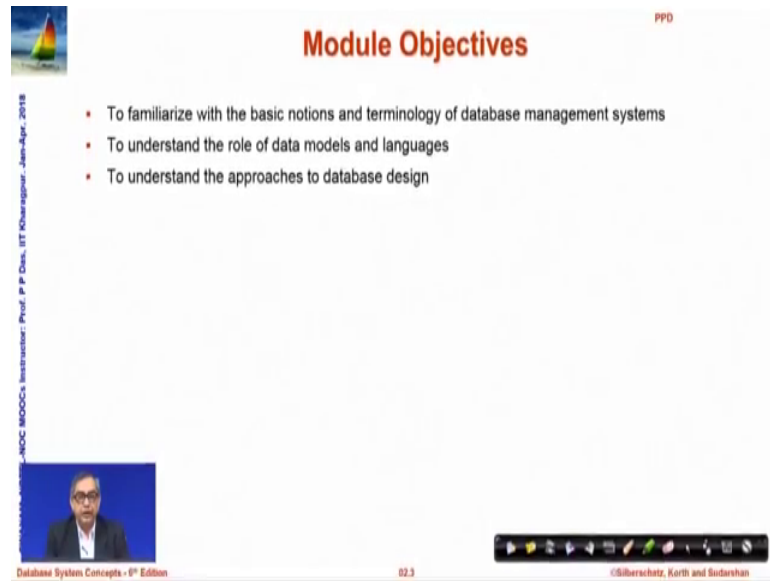
**Module Recap**

- Why Databases?
- KYC: Know Your Course
  - Course Prerequisite
  - Course Outline
  - Course Text Book
  - Course TAs

Database System Concepts - 9th Edition 02.2 ©Silberschatz, Korth and Sudarshan

Welcome to module two of database management systems. In this module, we will talk primarily about the introduction to the DBMS. This discussion will span two modules that is the current one module 2 and the next one that is module 3. Just to quickly recap in the last module we have discussed about why we need databases, and we have introduced you to different aspects of the course.

(Refer Slide Time: 00:54)



PPD

## Module Objectives

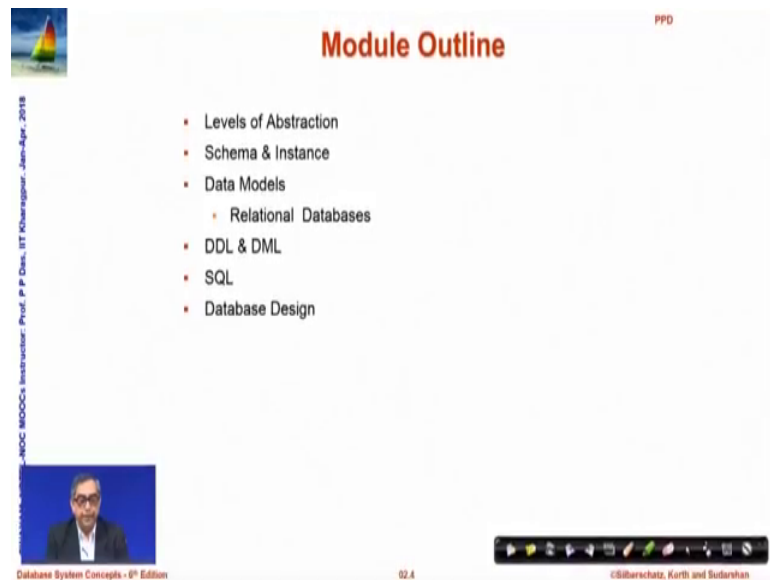
- To familiarize with the basic notions and terminology of database management systems
- To understand the role of data models and languages
- To understand the approaches to database design

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

Database System Concepts - 9<sup>th</sup> Edition 02.3 ©Silberschatz, Korth and Sudarshan

In view of this, in this module, we would familiarize you with basic notions and terminology of a database management system just at an introductory level. We will try to understand the role of data models and specific languages for database systems. And we would also outline the approach to database design.

(Refer Slide Time: 01:21)



PPD

## Module Outline

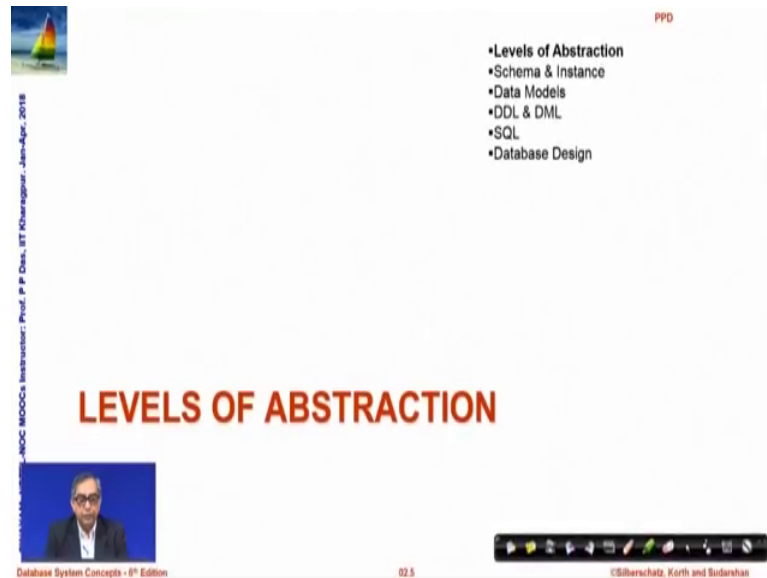
- Levels of Abstraction
- Schema & Instance
- Data Models
  - Relational Databases
- DDL & DML
- SQL
- Database Design

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

Database System Concepts - 9<sup>th</sup> Edition 02.4 ©Silberschatz, Korth and Sudarshan

So, the module outline would be like this. And as we go along you will be able to follow which particular aspect of this outline we are discussing about.

(Refer Slide Time: 01:35)



PPD

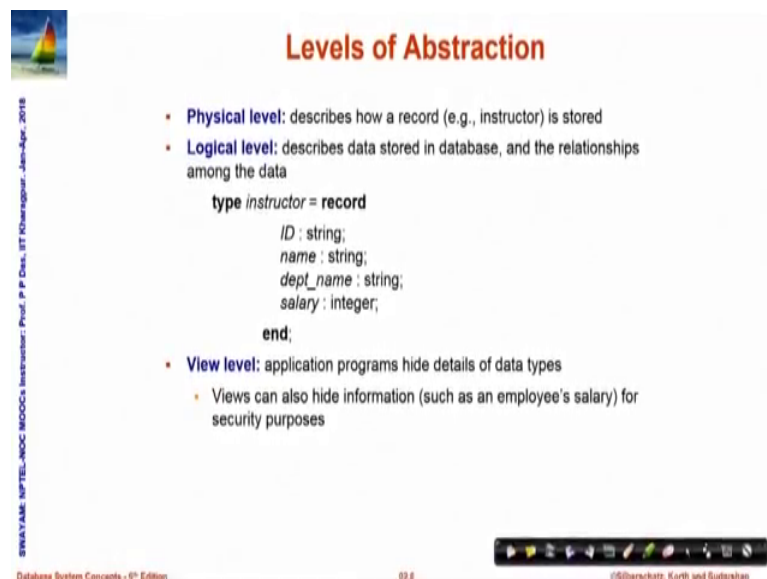
- Levels of Abstraction
  - Schema & Instance
  - Data Models
  - DDL & DML
  - SQL
  - Database Design

## LEVELS OF ABSTRACTION

Database System Concepts - 9<sup>th</sup> Edition 02.5 ©Silberschatz, Korth and Sudarshan

So, to start with we talk about levels of abstraction.

(Refer Slide Time: 01:40)



## Levels of Abstraction

- **Physical level:** describes how a record (e.g., instructor) is stored
- **Logical level:** describes data stored in database, and the relationships among the data

```
type instructor = record
    ID : string;
    name : string;
    dept_name : string;
    salary : integer;
end;
```

- **View level:** application programs hide details of data types
  - Views can also hide information (such as an employee's salary) for security purposes

Database System Concepts - 9<sup>th</sup> Edition 02.6 ©Silberschatz, Korth and Sudarshan

A database system like any other system is conceptualized in terms of three levels of abstraction. At the lower most level is the, what we say is a physical data level or the physical level which describes how a record is actually stored, so that is about the physical storage in the system. At the next level, we talk about it we say it is a logical level which describes the data stored in databases and its relationship amongst the data. So, you can any data that is stored you can think about it as a record. So, if we here we

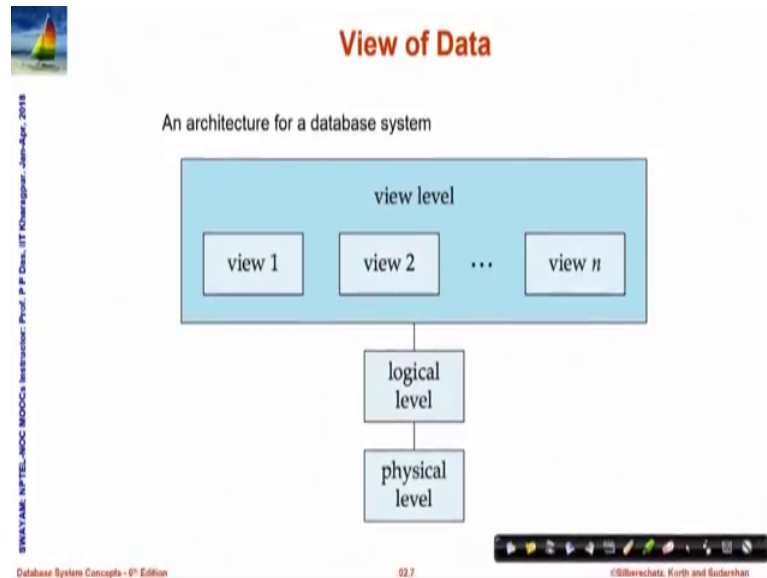
are illustrating the record of an instructor who teaches a course. So, as you know the record is a collection of multiple fields of different types. So, here we have field to describe the ID identifying number or shrink of an instructor, we have the name of the instructor, the name of the department, the salary and so on.

So, this logically says this is the entity this is a record or this is the structure of a record that I want at a logical way. So, this in contrast to the physical level, logical level is not concerned particularly with how that these data, how this string the number and all that will be actually stored, and how these multiples of hundreds and thousands of records will actually be stored so that they can be efficiently used. But we are just concerned with the logical view that I should be able to deal with records as they are.

At the third level which is it can say the topmost level is called the view level where the application program tries to view the data. And when the application program tries to view the data, it deals with the details that it needs to; and rest of the details are usually hidden from this view. For example, if here we talked about the university database in the last module, so if you are talking about the university database, and then you are a student. And you are when you access the database you should be able to see what all courses you are enrolled in or where is that course being held who is the instructor of that course and so on. But you should not be able to access or see the view of what is the instructors salary or for that matter, what are the grades that are obtained to by different students in different courses and so on.

Whereas, an instructor may be able to view the performance of the students in multiple different courses particularly the ones that he or she is involved in evaluation, but she again may not be allowed to check the salary of other instructors. So, view level is a high level where of abstraction where you try to provide the information about the data in terms of what the application needs, what the users of that application need, but you do not actually deal with the details of all the records that the logical level has.

(Refer Slide Time: 05:29)



So, these are three levels form the basic structure of a database system architecture of a database system. As you can see the physical level using that a logical level of records are formed. Physical level typically is in terms of database files is binary in nature, the organization of those files. The logical level deals with the records and the different fields of the records the schema of the database and so on.

And the view level is something which is constructed from the logical level in terms of different views that the different applications need. I am sure at this stage you may not understand the whole of these levels and their implications, but this is just to give you an idea of the existence of three levels, and the need to deal with the three levels. And as we go along, we will see that we will refer to these levels more and more and discuss about the specific aspects of those.

(Refer Slide Time: 06:27)

PPD

- Levels of Abstraction
- Schema & Instance
- Data Models
- DDL & DML
- SQL
- Database Design

## SCHEMA AND INSTANCE

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

Database System Concepts - 9<sup>th</sup> Edition 02.9 ©Silberschatz, Korth and Sudarshan

Next, let us talk about schema and instance.

(Refer Slide Time: 06:34)

PPD

## Schemas and Instances

- Similar to types and variables in programming languages
- **Schema**
  - **Logical Schema** – the overall logical structure of the database
    - Analogous to type information of a variable in a program
    - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
      - Customer Schema
  - Account Schema
- **Physical Schema** – the overall physical structure of the database

Name	Customer ID	Account #	Aadhaar ID	Mobile #
------	-------------	-----------	------------	----------

Account #	Account Type	Interest Rate	Min. Bal.	Balance
-----------	--------------	---------------	-----------	---------

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

Database System Concepts - 9<sup>th</sup> Edition 02.9 ©Silberschatz, Korth and Sudarshan

We will very regularly keep on referring about schemas and instance. The schema is in a very simple terms say if we talk about first a logical schema, it is a way a certain data needs to be organized, it is a plan for organizing data. So, if you can draw a parallel then say when a building is constructed, a plan is prepared. And according to that plan several buildings a in a say residential complex may be constructed. So, there is a difference between the plan which gives you the layout of where different rooms should be where

there is a staircase where is the courtier and so on and the actual building when or the group of buildings which are constructed. So, the schema primarily talks about what is the plan to organize the data.

So, if we talk about a customer schema, it has multiple different fields, it should talk about the name of the customer, ID of the customer, it is account possibly the other ID the mobile number and so on. So, the fact that these the fields need to be present for describing a customer, forms a customer schema. Whereas, when we talk about a specific schema of account that the customer holds with a bank, then we need the account number, account type, interest rate, minimum balance, the current balance and so on. These are the fields of information that we need; and we need to know what is the type of every such field, and all those and those kind of information from the schema information.

And again in line with the abstractions of physical logical and view as we did, schema also can be at a logical schema which is corresponding to the logical level of abstraction. And we may also have a physical schema which tells actually how the data is physically organized in the database, what are the different database files, how they are indexed and so on.

So, all these information which we can say is kind of a metadata information. This is not actually that it is not the customer schema is not saying who the customers are, the account schema is not saying, what are the accounts, what is their balance. But it is saying that if a customer needs to be defined; then what is the information that you need to know, what is the information that you need to manage. If an account need to be described then what is the different fields that are important. So, this schematic or this metadata is called the schema of a database.

(Refer Slide Time: 09:25)

**Schemas and Instances**

- Instance
  - The actual content of the database at a particular point in time
  - Analogous to the value of a variable
  - Customer Instance

Name	Customer ID	Account #	Aadhaar ID	Mobile #
Pavan Laha	6728	917322	182719289372	9830100291
Lata Kala	8912	827183	918291204829	7189203928
Nand Prabhu	6617	372912	127837291021	8892021892

- Account Instance

Account #	Account Type	Interest Rate	Min. Bal.	Balance
917322	Savings	4.0%	5000	7812
372912	Current	0.0%	0	291820
827183	Term Deposit	6.75%	10000	100000

Database System Concepts - 6th Edition 02.10 ©Silberschatz, Korth and Sudarshan

Now, based on this schema specific instances of databases happen, instances when you actually have populated different records according to the schema. Now, naturally once the schema is fixed, your records will need to have values in all of those fields that the schema has; and every value must be of the type that the particular field is specified with. So, I have just shown here certain sample instance of customer schema, where you can see three customers with their name, customer ID, account number, other ID, and mobile number, these are all fictitious data of course, but this is just to give you an idea of how this customer instance would look like.

Similarly, we have shown what is a accounts instance, so you can see that there is a some kind of a relationship that you can see between these instances. For example, the first customer ID on the customer instance can be seen as a first I am sorry the first account ID in the customer instance can be seen as a first entry first record in the account instance and so on. So, when we actually populate the schema with different records and this is what keeps on changing.

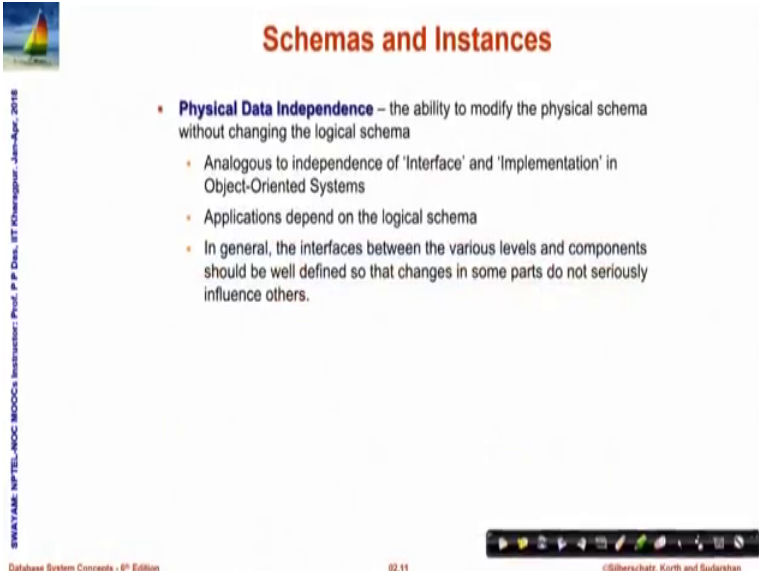
So, certainly when we do operations on the database, then certainly very regularly new records will get added, some records might get deleted from this instance, and fields of certain records may keep on changing. For example, in an accounts instance very regularly whenever a transaction is done, the account balance will get changed; maybe at a certain time the bank might decide to change the interest rate for a certain type of



account then the interest rate field will get changed, new customers may come into the customer instance and so on. So, instances keep on regularly being updated manipulated added deleted updated, but the schema remains unchanged.

So, change of the schema is very rare in a database and needs to be done only when the database is designed or when it is being upgraded. Because once you change the schema, it changes the way you look at the whole world, you look at the whole database scenario. So, if you are changing the schema at a logical level, then naturally the your view will also get affected, because you are using these schemas to decide how you would like to present a transaction application to the user or for a balance check application to the user and so on.

(Refer Slide Time: 12:22)



The slide features a small image of a sailboat on the left. The main content is a list of bullet points under the heading 'Schemas and Instances'. The text is as follows:

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Analogous to independence of 'Interface' and 'Implementation' in Object-Oriented Systems
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

At the bottom of the slide, there is a navigation bar with icons and the text '©Silberschatz, Korth and Sudarshan'.

But, of course, what do we would want is between the physical schema and the logical schema we normally would want certain independence. What it means is the logical schema is what you need to deal with, because it is linked with the view that you have at a higher level of abstraction. On the other end, the logical schema is based on the physical schema; physical schema is how you are actually organizing the information in terms of the binary files the database files.

Now, certainly you want that logical schema not to change because if it changes then at a view level all your applications will have to change. But it is quite possible that your physical schema the way you have organizing files and so on might need a change,

because maybe it is just that you had designed the database for 10,000 records and you already have 9000 records and you would like to expand it to maybe 1,00,000 records.

And the physical this system needs to be different you may need to reorganize the files and so on, you may need to index it in a different way and all this, but you would like to do that change in the physical level without requiring any change at a logical schema. So, this property of a database schema is very required which we say is a physical data independence or the ability to change the physical schema without actually affecting the logical schema or the view level. So, that will be a critical factor that will have to keep in mind.

(Refer Slide Time: 14:16)

PPD

- Levels of Abstraction
- Schema & instance
- Data Models
- DDL & DML
- SQL
- Database Design

**DATA MODELS**

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

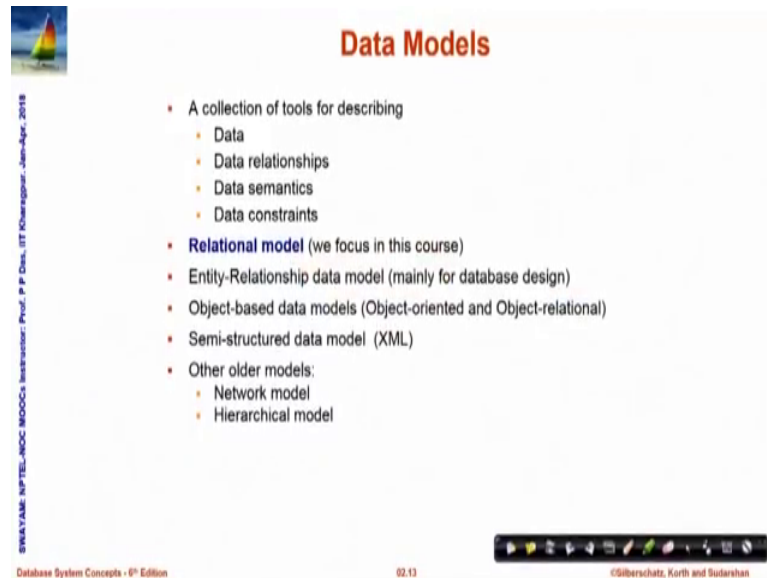
Database System Concepts - 9th Edition

02.12

©Silberschatz, Korth and Sultrehan

So, next is data models.

(Refer Slide Time: 14:20)



The slide, titled "Data Models", features a small image of a sailboat in the top left corner. The main content is a bulleted list of data models. The first bullet point is "A collection of tools for describing", which includes sub-bullets for "Data", "Data relationships", "Data semantics", and "Data constraints". The second bullet point is "Relational model (we focus in this course)", which includes sub-bullets for "Entity-Relationship data model (mainly for database design)", "Object-based data models (Object-oriented and Object-relational)", and "Semi-structured data model (XML)". The third bullet point is "Other older models:", which includes sub-bullets for "Network model" and "Hierarchical model". The slide footer contains the text "Database System Concepts - 9th Edition", "02.13", and "©Silberschatz, Korth and Sudarshan".

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- **Relational model** (we focus in this course)
  - Entity-Relationship data model (mainly for database design)
  - Object-based data models (Object-oriented and Object-relational)
  - Semi-structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Data models are a collection of tools that describe the data because we are talking about a database system. So, certainly the main focus here is to be able to model that it to be able to represent the data, so that talks about relationships between data, it talks about the meaning of the data the data semantics, it talks about data constraints. For example, it is an account balance, we just refer to account balance in the account schema in the instance, now, the question is will can the account balance be negative, the answer is yes or no.

If the bank mandates that I can only withdraw as much money up to which I have deposited, then account balance cannot be negative, but if the bank is giving me the facility to overdraft then I may be able to draw more money than I have actually deposited. So, my account balance might note negative. In some banks it could be that the bank says that well there is a minimum balance. So, minimum balance is 5,000. So, which means that my account balance cannot go below five thousand rupees the bank will not allow those transactions. So, these are examples of different constraints that might exist in the real world, and therefore, will be required in terms of the data model representation.

So, there are several data models that exist today. The most widely used the most popular and most powerful in terms of a certain section of database applications which we are commonly interested in is a relational model and that is what we focus in this course. We

will have a major discourse in terms of the relational model and lot of things will be developed based on that. But it is not easy to directly design a database in terms of the relational model, because you first need to understand the real world in which the design is happening.

So, we normally start with a less formal model known as the entity-relationship data model or an ER model, ER diagram, these are commonly called. So, if you recall your knowledge about object oriented systems, and if you happen to know uml, you already know about ER models and corresponding class diagrams that that result. So, we will talk briefly about your model and show you how to do modelling on the real world in terms of the ER diagrams. But, then they are not actually models which the database systems directly used they are subsequently converted to some relational or other model and which the database systems will use.

Next is a object-based data models. You all would be knowing familiar with the fact that objects give a better power to represent the system which objects are not like simple strings or numbers or characters like that, they are encapsulated concept of an entity which can be manipulated in a certain way. So, like in the real world, you have several objects, it would have been nice to have similar objects in the databases. So, quite a few database system have been designed used which are object-based database systems. So, there are models for those. However, we will do little of that in this course, because it is little bit advanced in notion.

The other model, which has become very popular is called the semi-structured data model. It is primarily in terms of XML. I am sure all of you would be familiar with the basics of XML, which is extensible mark up language in which you can create use tags and use different mark ups to describe the data. You can say this is the field and this is the value kind of. And this is become a very nice way to represent the data. And XML or the semi-structured models are particularly useful today to exchange data between different systems.

I may be using a my SQL kind of database system, my friend may be using an oracle system, and we need to exchange data tables between these two, these two systems will represent the data in the in physical schema which are not may not be compatible. So, we can represent both of them in terms of XML models convert the data. So, I convert the

data into XML, give it to my friend and my friend can import from that XML into the database. So, it is a XML is a data model which is frequently used in terms of data exchanges.

Then there are several other models like the network model, hierarchical model which used to be very popular in the early days of database systems before relational model came into force. They still exist in terms of the some of the databases. And some of the newer emerging big data databases actually we have started using this old concept in a new way again. So, this is a overall set of data model.

(Refer Slide Time: 19:58)

**Relational Model**

- All the data is stored in various tables
- Example of tabular data in the relational model

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

(a) The *instructor* table

The slide includes a small image of a sailboat in the top left, a vertical text string on the left side, a small video inset of a man in the bottom left, and a navigation bar at the bottom right.

So, here I am just showing an example of a relational model data which is simply looks like a table. So, you can see that on top row in the blue are the names of the different fields which describe the schema. It says that it has an ID, it has a name, it has a departments name, it has salary. They are trying to describe a particular instructor, and then a whole lot of records rows in that table, which are every row individually is a particular data entry or a record. So, columns are attributes, and rows are records that the relational model described.

(Refer Slide Time: 20:40)

**A Sample Relational Database**

ID	name	dept_name	salary
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
48565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58883	Callifert	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The instructor table

dept_name	building	budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	80000
Physics	Watson	70000

(b) The department table

Database System Concepts - 9<sup>th</sup> Edition 02.15 ©Silberschatz, Korth and Sudarshan

Some more of that the instructor table along with the department table. So, the table below describes details of a department, so that has its own schema and the individual records. We have of course seen similar instances already in terms of the customer and accounts instance that we have just discussed a couple of while ago.

(Refer Slide Time: 21:03)

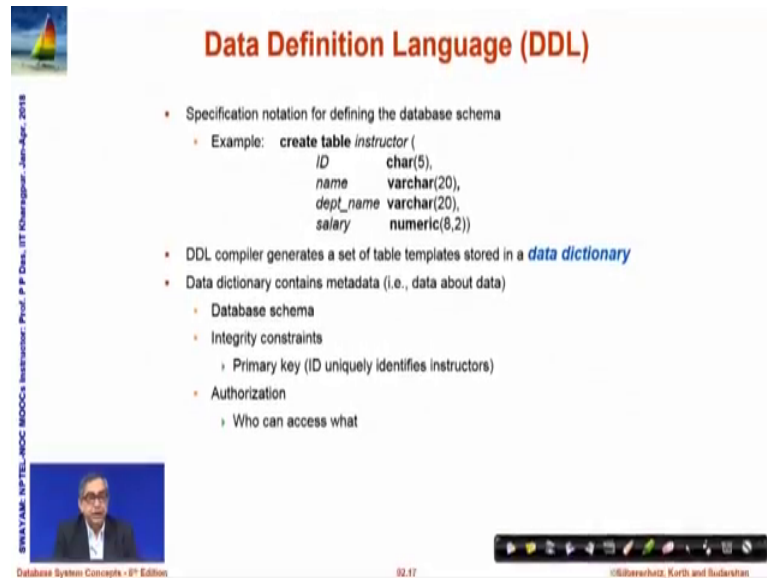
**DDL AND DML**

- Levels of Abstraction
- Schema & Instance
- Data Models
- DDL & DML
- SQL
- Database Design

Database System Concepts - 9<sup>th</sup> Edition 02.16 ©Silberschatz, Korth and Sudarshan

Let me introduce these two terms DDL and DML.

(Refer Slide Time: 21:10)



**Data Definition Language (DDL)**

- Specification notation for defining the database schema
  - Example: 

```
create table instructor (  
    ID          char(5),  
    name        varchar(20),  
    dept_name   varchar(20),  
    salary      numeric(8,2))
```
- DDL compiler generates a set of table templates stored in a **data dictionary**
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Database System Concepts - 8<sup>th</sup> Edition 92.17 ©Silberschatz, Korth and Sudarshan

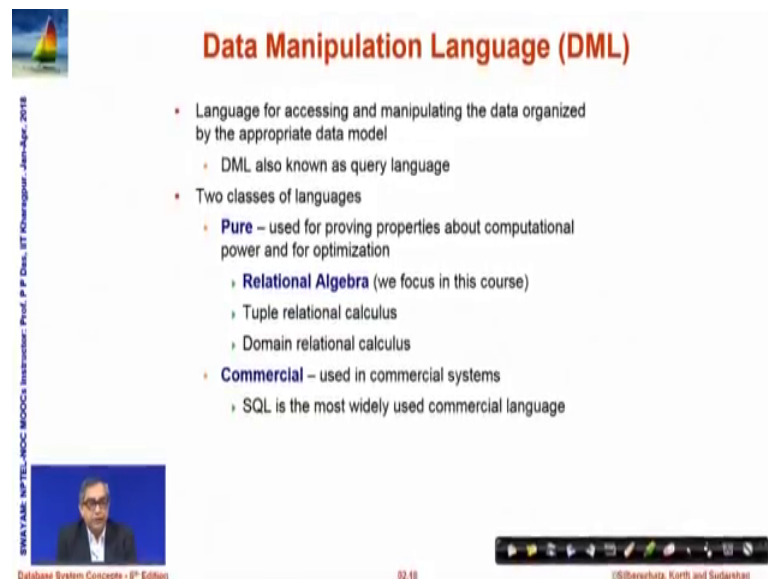
DDL talks about data definition language. So, what the concept wise what we are trying to say is certainly we have a schema and we have instance. So, we need certain language constructs to be able to define a schema and certain other language constructs to be able to manipulate the data in that schema or they are basically manipulate the instances. So, DDL is the language or part of the language which is used to define and manipulate the schema of a database that is why schema is a way to define a database. So, it is called a data definition language.

So, you can define that I will am going to have a table called instructor which will have four different fields, each having certain types of data. So, it says that the ID will be a five character data; the name would we would have a variable length, because you cannot say that the name will be of a fixed length, but it will be a variable length that is what varchar is, but the length will not exceed 20. And similarly, salary will be a numeric data with up going up to 8 figures, and having a decimal part having two parts.

So, this way of defining the schema in terms of the different attributes and their types or columns in the table or trying to define the structure of that table is a main issue of the data definition language. So, the data definition language compiler who generates a set of tables in the data dictionary, where the data dictionary basically contains metadata as I said the schema is nothing but a metadata about the database tables. So, which will have the database schema, it will have different integrity constraints, it could say that well the

account balance cannot be negative or account balance cannot be less than the minimum balance. So, these are different integrity constraints. It could say that this is the primary key, we will talk more in more depth in terms of what is key. And it could also specify the authorization as to who is allowed to access which part of the data and so on, so that is these all are part of the schema definition and forms the DDL of the language.

(Refer Slide Time: 23:36)



**Data Manipulation Language (DML)**

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - **Relational Algebra** (we focus in this course)
      - Tuple relational calculus
      - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

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

Database System Concepts - 9<sup>th</sup> Edition

02.18

©Silbarachata, Korth and Sultanshan

In contrast, the data manipulation language is a language for accessing and manipulating the data organized. So, it is for access, update, addition of new records, deletion of existing records and so on. And very commonly we will refer to the data manipulation language as a query language, because this is what you want to know what exists in the database. So, the query language will be designed, they are designed primarily in one of the two ways. One group of languages is known as a pure language, they are more mathematical in nature. They have a formal basis that can you can prove that whatever do you do in these languages are correct, and will give you the correct result.

So, they are different languages based on the relational model, they are called relational algebra, tuple relational calculus, domain relational calculus and so on. Of these three, we will in this course deal only with relational algebra. There are mathematical proof which show that whatever you can do in relational algebra you can do it in tuple relational calculus and vice versa. Similarly, whatever you can do in relational algebra, you can do in domain relational calculations and so on. In one sense that these languages



are equally powerful; the same thing can be done in any one of them, but we will just take the simplest of them and study here in terms of the relational algebra, but these are more mathematical representations are not easy to write as a program. So, normally we will use certain commercial query language which is called SQL for most of our applications and we will do the coding in that.

(Refer Slide Time: 25:19)

**SQL**

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

SRINIVAS RAO, IIT Kharagpur, Prof. P. P. Das, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 9th Edition 02.20 ©Silberschatz, Korth and Shturman

So, SQL which is a most widely used commercial language and mind you this is not a Turing equivalent language which means that everything that can be that need to be computed cannot be computed in SQL, there are certain computations which SQL cannot do. It is a limitation; it is a restricted language. So, often SQL is used in conjunction with some common high-level programming language like C or C plus plus and so on. So, whatever is there can be done in SQL in terms of data manipulation will be done in terms of the relational model, but there could be additional logic that needs to be built in, in terms of the high level language. So, application programs are typically written through them. So, we can do this through a process of embedding that is put in SQL as a part of a C program or use certain libraries which can actually take a query from C, and fire it on the SQL database.

(Refer Slide Time: 26:26)



PPD

- Levels of Abstraction
- Schema & Instance
- Data Models
- DDL & DML
- SQL
- Database Design

## DATABASE DESIGN

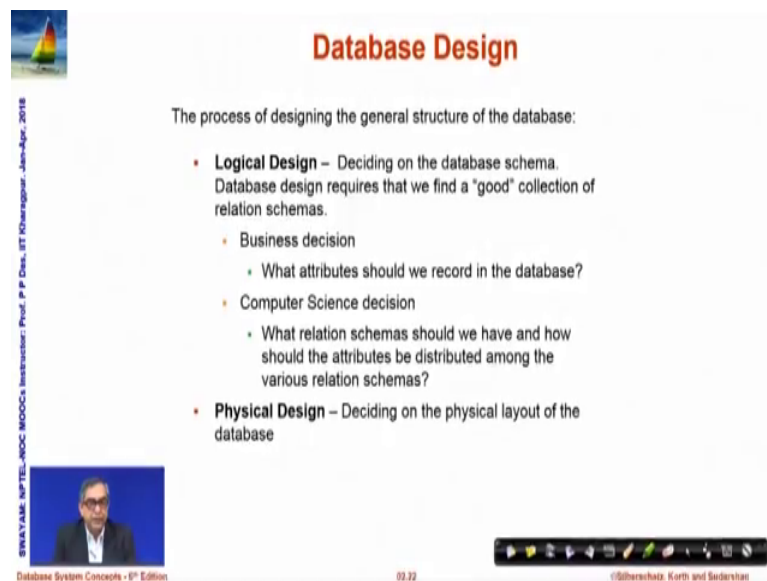
Database System Concepts - 9<sup>th</sup> Edition 02.21 ©Silberschatz, Korth and Sudarshan

SWAMYAM: NPTEL MOOC Instructor: Prof. P. P. Chiu, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 9<sup>th</sup> Edition 02.21 ©Silberschatz, Korth and Sudarshan

So, we will see how to do this in the course of time.aspect.

(Refer Slide Time: 26:32)



## Database Design

The process of designing the general structure of the database:

- **Logical Design** – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
  - Business decision
    - What attributes should we record in the database?
  - Computer Science decision
    - What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- **Physical Design** – Deciding on the physical layout of the database

Database System Concepts - 9<sup>th</sup> Edition 02.22 ©Silberschatz, Korth and Sudarshan

SWAMYAM: NPTEL MOOC Instructor: Prof. P. P. Chiu, IIT Kharagpur, Jan-Apr, 2018

Database System Concepts - 9<sup>th</sup> Edition 02.22 ©Silberschatz, Korth and Sudarshan

Coming to the database design this is a process through which the databases need to be designed. And certainly the first part of the design is the logical design where you want you need to identify what are the schemas and you know what are the constraints that apply, what is authorization required. And first set of decisions those are related to the business as we say. Business means it is basically comes from the domain. So, it is if I am doing a university database, the business decisions will come in terms of you know I

have courses, students, instructors, and the instructor teach courses, can an instructor teach multiple courses, can a course be taught by multiple instructors these kind of business decisions are critical for the database design.

And then there is a whole set of computer science decision or the data based decisions to decide as to if this is the kind of business information that you want to keep in the database, then what is the kind of relation, what is the kind of schemas that we should use what are should be the attributes, which attribute should be of what type what should be strain, what should be numbers and so on. So, these are formed the basis of the physical logical design. And of course, we then need a physical design which decides on the physical layout of the data, what are the different database files, how they should be indexed and so on.

(Refer Slide Time: 27:53)

**Database Design (Cont.)**

- Is there any problem with this relation?

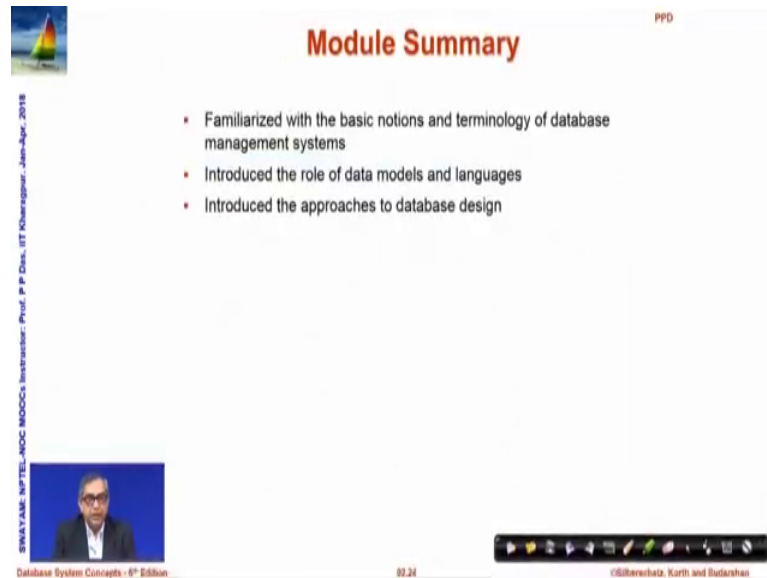
ID	name	salary	dept_name	building	budget
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	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

Database System Concepts • 9<sup>th</sup> Edition ©Silberschatz, Korth and Sudarshan

So, here we are showing an example table. So, it has multiple fields. It shows the instructors expanded form of the instructor table you saw earlier. It is expanded with the departments name and the building in which it is housed. So, if you look carefully that this certainly comes from the business decision that you need to know the department to which an instructor belongs and certainly you need to know the building in which that department exists. So, knowing the department of the instructor and the building of where that department is are critical, but the question is this a good design, is this so we

will discuss as to when why this is a good, this may not be a good design to represent the data.

(Refer Slide Time: 28:41)



The slide is titled "Module Summary" in a large, bold, orange font. It features a small sailboat icon in the top left corner and a small "PPD" logo in the top right. The main content is a bulleted list of three items. On the left side, there is a vertical text string and a small video thumbnail of a man speaking. At the bottom, there is a navigation bar with various icons and a copyright notice.

Database System Concepts - 9th Edition

02:24

©Silberschatz, Korth and Sudarshan

PPD

Module Summary

- Familiarized with the basic notions and terminology of database management systems
- Introduced the role of data models and languages
- Introduced the approaches to database design

Database System Concepts - 9th Edition

©Silberschatz, Korth and Sudarshan

So, in this module, we have taken you through the basic notions and terminology of database management systems, highlighting primarily the levels of abstraction, the schema an instance, the basic data models the languages that you need DDL, DML and the commercial SQL language. And we have also tried to give you a glimpse of the approach that is required in terms of the database design. We will elaborate on this more in the second part of our introduction to DBMS which will be taken up in module 3.