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

Lecture - 03
Introduction to DBMS/2

Welcome to module 3 of Database Management Systems course. We started discussions introducing the database management systems in module 2. This is the second and concluding part of that discussion.
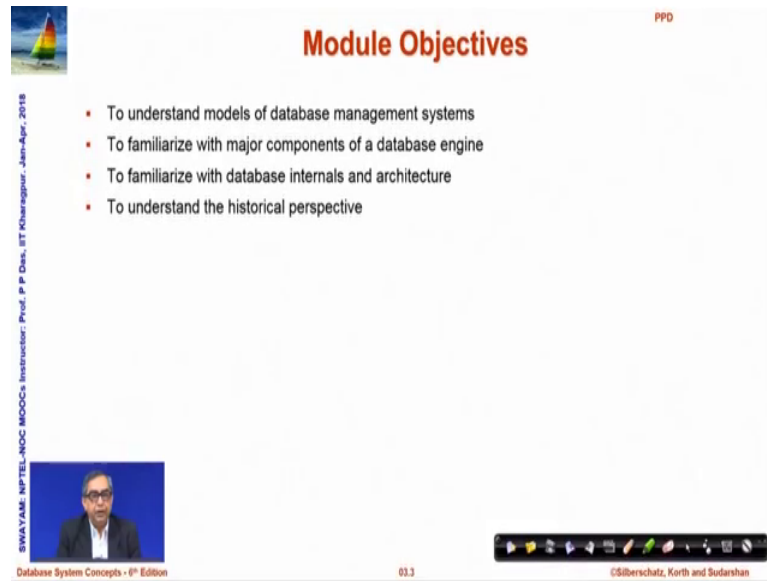
(Refer Slide Time: 00:37)

So, this is what, these are the aspects that we are discussed earlier starting from level of abstraction to the outline of database design.

(Refer Slide Time: 00:47)

In the current module, we would like to understand the modules of database management systems little bit more and we will try to familiarize with the concept of major components of database engine, will elaborate on those and will familiarize. So, with the basic architecture of a database management system, some of the internal components and we will present a brief historical perspective of the DBMS.

(Refer Slide Time: 01:21)

So, this is the outline that we will follow.

(Refer Slide Time: 01:25)

(Refer Slide Time: 01:34)

(Refer Slide Time: 01:46)

So, we have already discussed about the database design, I would like to raise a few further issues about that. So, we have seen that, there is a logical design which is driven by the business decisions and refined by the computer science decisions, there is a physical design as well; and based on that we had at presented this particular table asking, whether, this database is a good design or not.
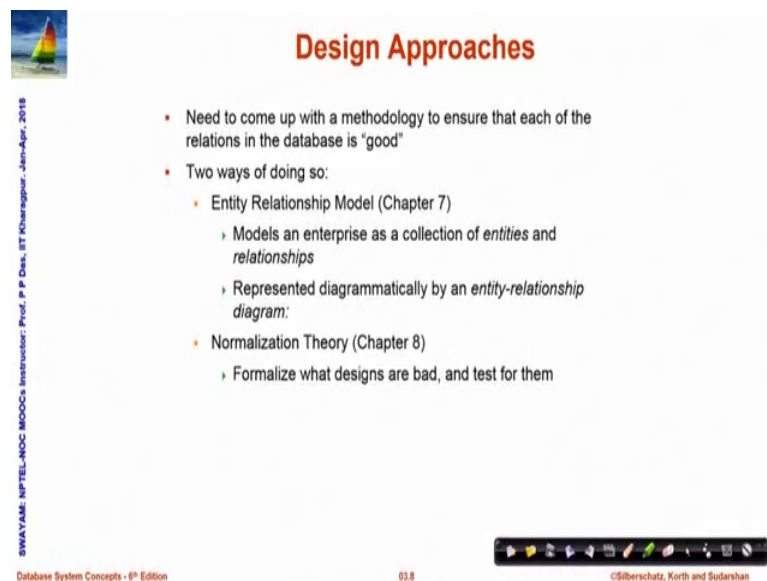
So, let us have a little look into this for example, we have introduced the department name and the building in which the department is housed. So, if we look at there are multiple instructors say, let us say Professor Einstein, who teaches in the Physics department, that is housed in the Watson building and if we look through there is a Professor Gold, who also teaches in the Physics department and naturally that is housed in the Watson building.

Now, the question is; so, Physics department, if it is housed in the Watson building then, all the instructors in this table, all the instructors who are part of the Physics department, would have their department housed in the Watson building. So, there is a certain issue of redundancy in these two, that is, the same information is given more than once, which is not a very desirable thing.

The consequence of this could be suppose, tomorrow the university decides to move this Physics department from Watson to the Taylor building. This will mean that once this is moved, then this Watson will have to be changed to Taylor, also this Watson will also have to be changed to Taylor. All instances of Watson that corresponded to the Physics

department in this table, will have to be changed to Taylor, and that is not a good scenario. So, it is not only that we have redundancy, we have potential for anomaly; that is program the application programmer might forget to update the building say, for this entry then, we will be in an inconsistent database. So, to put it in simple terms that this is not a good design and there are several issues to consider, in terms of whether some design is good or some design needs refinement.

(Refer Slide Time: 04:22)

So, we need to come up with a methodology to ensure that, each of the relations in the database is good. So, we primarily follow two approaches in doing this. One is, using the entity relationship model, which models the enterprise as a collection of entities or concepts or if you are familiar with the object orientation classes and the relationships that hold between these entities.
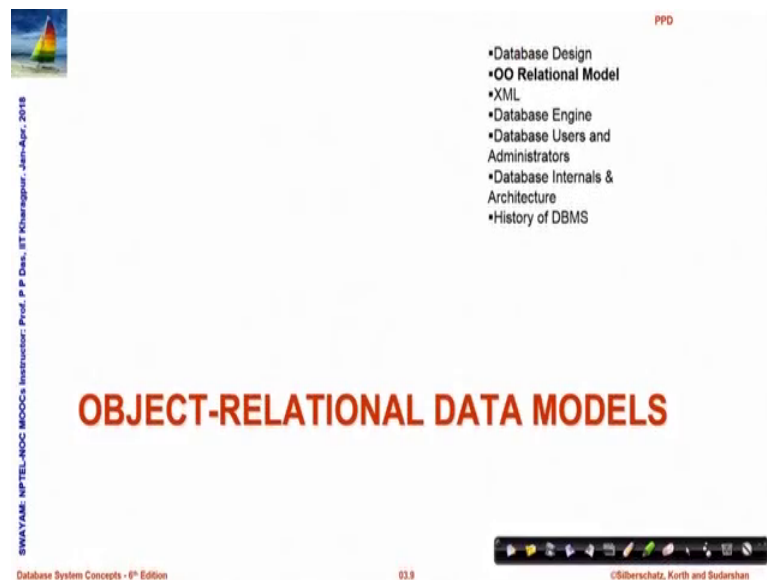
So, in a university database the entities are students, courses, teachers and the relationships are a teacher teaches a set of courses, a student attends a set of courses and so on, the teachers supervise a set of students for projects and so on and then represent them diagrammatically in terms of a ER diagram entity relationship diagram and once that has been done then, we try to follow a certain normalization theory.

This normalization theory tries to capture that what are the properties that must hold in this database design, that must be satisfied on this database design, in terms of what is known as database dependencies, there are, varied forms of dependencies functional

dependencies, multi value dependencies, joint dependencies and so on and try to formalize and evaluate whether a design is good or it is bad, test them for quality and then normalize to make them better; make them the best possible that can happen.

So, this is something that is starting from the entity relationship model, which captures the real world to the actual database schema, there is a process of representation and then capturing of ground truths, that should hold in the database system and then normalize the database is a basic requirement of the design approach.

(Refer Slide Time: 06:34)
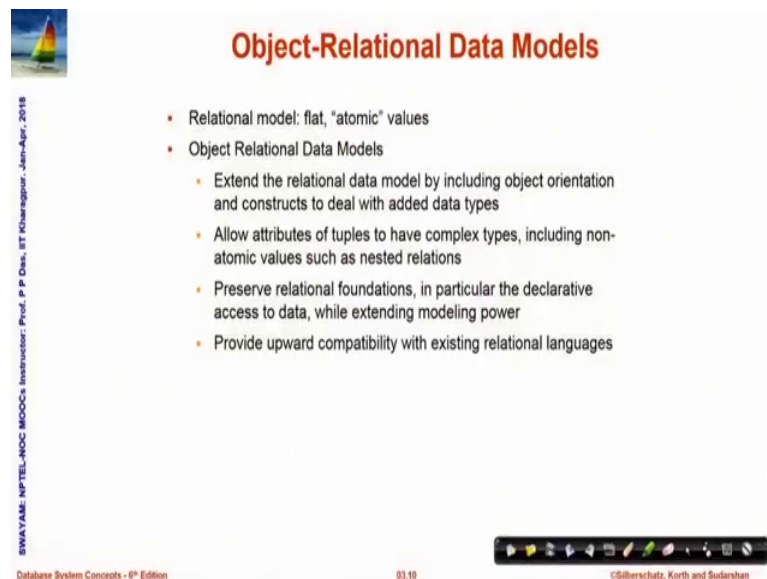
(Refer Slide Time: 06:39)

We have talked about object relational data models for a few more points about that, that in a relational model everything is flat, every value is atomic, in the sense that everything if you, look back and think in terms of C then, every field is a value which can be a simple, you know, built in type like integer, like fixed length string, variable length string, a floating point number like that, but I cannot have a composite you know, object kind of fields.

But in a relational data model we extend in the object relational data model, we extend the relational model by including the object orientation and the constituent constructs to deal with added data types, higher data types where attributes are allowed to have complex types, non atomic values that may allow things like nested relation; that is a value could itself be a relation, could itself be a table but, we try to preserve the relational foundation and we will see what those foundations mean and provide upward compatibility to existing relational databases. So, this is what the basic concept of object relational data models are and as I said that, we will just glimpse through it, but this is not the primary objective that we will try to cover.

(Refer Slide Time: 08:04)

 (Refer Slide Time: 08:07)

In contrast, the XML extensible markup language was defined by W3C, and it was originally intended for marking up document languages. It was not designed as a database language, it was designed for marking up. So, it is kind of saying that this particular element should be put in capital, this should be in blue colour, this means a verb, this means a paragraph, there should be a page break here, those kind of markups, But subsequently, it turned out that the way XML deals with different components in terms of tags, and the ability to create nested tags, makes a great language for exchange of data, As I explained in the last module also.
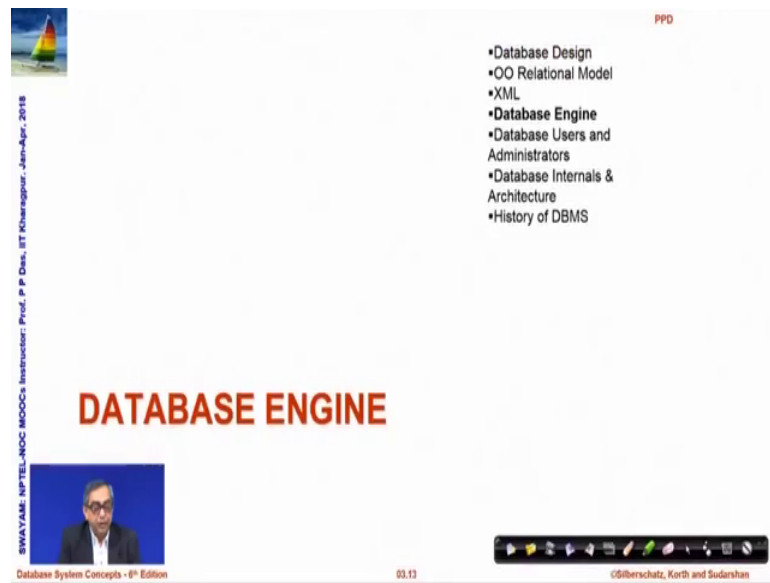
So, has become the basis for all kinds of new generation data interchange format. So, as I explained, that any database should be able to convert the data instances of the tables in terms of corresponding XML format and then you take it to some other database, in which you are intending to interchange the data and that target database should be able to import from that XML structure, and it has become widely available that you have different tools for parsing, browsing and querying XML content document data and so on. So, if you are familiar with C programming, I hope so you are! You can look up certain XML parsing and try out, there are great tools to learn.

(Refer Slide Time: 09:53)
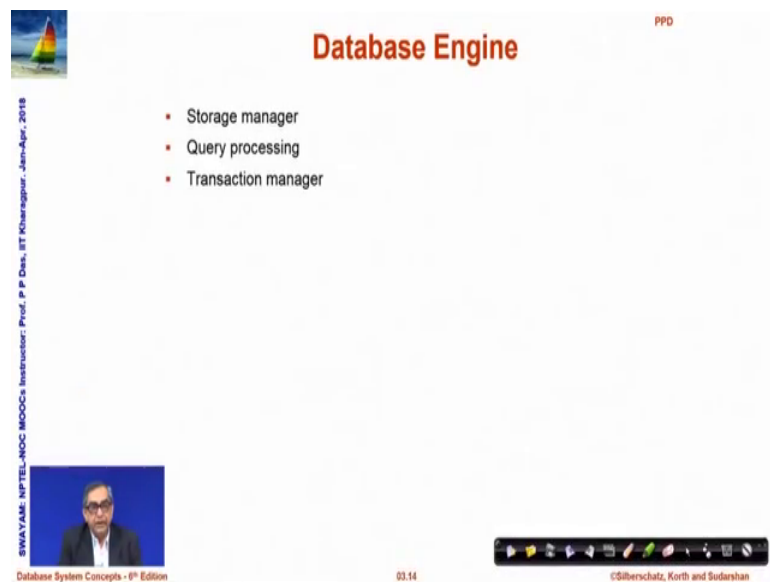
Moving on, let us briefly look at what is the core of a database management system, the database engine.

(Refer Slide Time: 10:03)
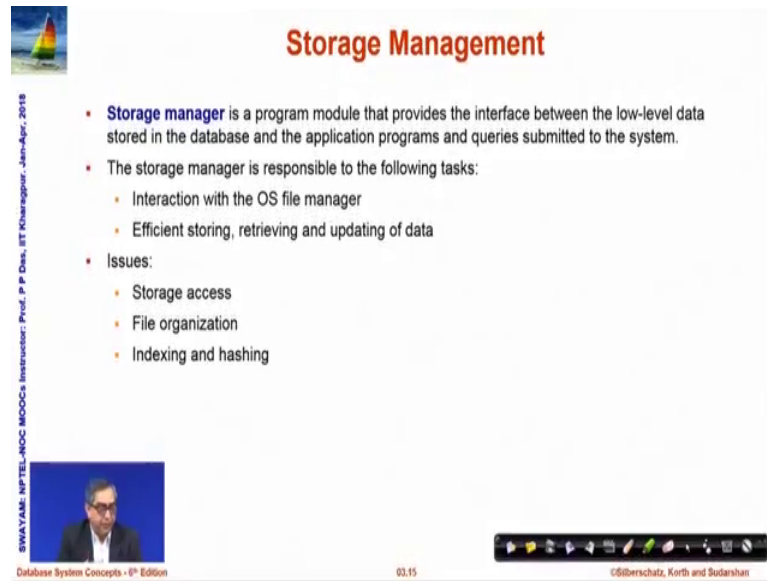
The database engine, primarily contains 3 major components; the storage manager, the query processing engine, sub engine and the transaction manager.

(Refer Slide Time: 10:14)

The storage manager, is a module or a collection of modules in a database management system, that provide the interface between the low level data and the application program. So, we have looked at the storage manager is the one, which is a bridge between the physical level of abstraction and the logical level of abstraction, then finally, to the view level of abstraction. So, the storage manager has to deal with interactions with the operating system on which the DBMS is kept, the file manager of the operating system, it is responsible for efficient storage, retrieval update of the data, it is responsible to make sure that if there are certain problems in the file system, then the data is not corrupted and so on.

So, the issues certainly that involve are :- the access to the storage, the organization of the files and very importantly indexing and hashing and we will talk about the concept of indexing later in the course. It primarily says that, if I want to, for example, you can simply understand that if you have a large chunk of data that you want to organize for efficient search then, you can use the binary search tree in simple algorithm terms. The binary search tree needs to be organized in terms of one data component.

We say that, well there is one value based on which you can say that, comparison is done. So, that at every node if that value is smaller, you go to the left sub tree, if that value is larger, you go to the right sub tree and so on. So, if we want to organize the

records of a database system in terms of such a binary search tree, then the question certainly is, which field do I use for the search tree comparison.

Now, whatever field I used for search tree comparison, on that field the searching would be very efficient, but if I want to search on a value for a different field, the searching would not remain that efficient. So, indexing is a mechanism by which, you can actually create auxiliary search trees on multiple fields. So that, the search on multiple fields can be made efficient and we will talk about this later when, that particular module comes up, but the storage manage has to deal with such issues.

(Refer Slide Time: 12:43)
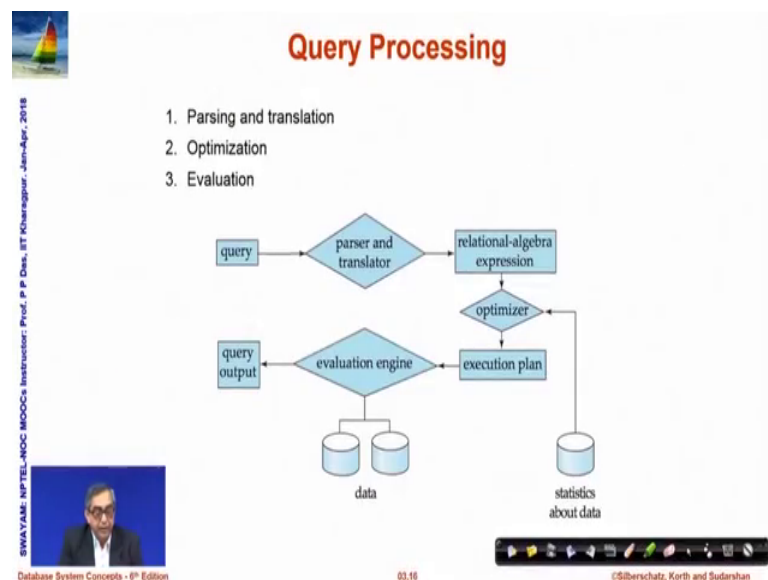
Moving on, the query processing is, if we have already talked about the language; the DDL, the DML, the query language. So, it is some kind of, like the C program, it is some kind of a text based programming code. So, naturally that code needs to be parsed and translated, as we typically do in a C compiler. So, there needs to be a query compiler. So, it parses and analyses the code, but translated unlike the C program, which translates the C program into an intermediate code and then into the binary instructions of the machine, the assembly binary instructions of the machine.

The query translator, translates the query into relational algebra expressions. I said that, there are two kinds of languages :- the commercial query language and the pure language. So, it translated in terms of a program in the pure language, which could be a

relational algebra language and then it tries to optimize. So, that is a critical term to be noted that there is an optimizer.

So, this optimizer is a critical component, which tries to make sure that the query, when it is run on your data will run with the most you know, least amount of time in an effective manner. So, then an execution plan needs to be decided, we will be able to understand this when we go to the actual relational algebra execution plan, basically says that if, there are multiple operations in that query to be performed, then how those operations, in which order they should be performed and where should temporary tables be used, where they should be skipped and so on and then, once that has been done then it passes on to an evaluation engine which actually runs that query on the data that you have, the instances of the data that you have, and that brings out the resultant query output which is another table of results that we get. So, this query processing is a core part of a database engine, which actually allows us to write text based queries and relative data efficiently, change-update data efficiently, insert data efficiently, and so on.

(Refer Slide Time: 15:12)

So, when we do this, we need to look at alternative ways of evaluating a query. There could be different ways to write the same thing, these are called equivalence expression, equivalent expressions and what are the good algorithms for doing each and every operation, there is a cost between good and bad way of evaluating. So, this has to be understood that, the same thing you can compute in a, you have seen this similar

concepts in normal programming language also, I mean we have seen for example, for sorting the several ways to sort and some are better some are not as efficient.

So, if the similar things in terms of a query need to be evaluated and the cost between good and bad ways need to be figured out. So, then we need to estimate the cost of every operation. It depends on the information of what has happened in the past, the statistical information and need to estimate those statistics for intermediate results. These are a couple of things that the query processing sub engine in a database will do.

(Refer Slide Time: 16:36)

So, beyond the storage manager and the query processor we have a transaction management system, which is very critical and core of the database system. It primarily has to deal with two fundamental issues of a database. One, what if a system fails; see, database systems are unlike the programs that you have written so far. A program starts execution ends, the program always deals with transient data, the data did not exist before your program started, it ceases to exist after your program ends. So, a program; however, complicated; however, important has a limited lifetime.

A database in contrast, has a much longer lifetime which deals with persistent data, that is very important to understand, that is the each application whether I am doing a bank fund transfer, whether I am making a credit card payment, to whether I am checking the balance or I am booking a railway ticket, whether, I am purchasing a book from Amazon, each one of the applications are like the normal program, it has a fixed lifetime.

If I start it, I do certain operations, I am done with it, but the data that is behind it the data of my accounts, my account balance, my transactions, my different bank charges, all that need to stay on and on and on and beyond every particular operation that I have done on the database. So, which means that if this database system fails, at some stage for some reason, then we have an enormous impact of that and that is not something that we can absorb that something that we can accept.

So, a database system has to come with the concept of recovery. It must be possible, if the system fails, it must be possible to recover it, to a certain earlier point where it is consistent. So, transaction management system is responsible to guarantee this kind of recover ability of databases. Then, the other question that we have discussed about earlier also, is a multiple users are you accessing the same database, the same set of data, at the same time. So, what how to make sure that more than one user can concurrently use an update without the data getting inconsistent, that is as I had mentioned, there is only one seat available, one berth available on a particular train, on a particular date and two users at the same time has initiated a booking.

It should not happen, that both of them get the booking. So, one should get, one should not get and that needs to be the complexity is high, for this kind of you know, decisions because in the databases, as applications are significantly distributed, Indian railways have no idea of who is going to do what booking, of which berth, from where at, which point of time. So, transaction management system is, as the name suggests defines something called, a transaction which always keeps the database consistent and operable. So, a transaction is a collection of operation, that performs a single logical function in a database application. This is very critical. It is a collection of operations and performs a single logical function.

So, it does not do anything and everything, it just does a single particular logical operation and that is what forms the transaction. So, a transaction management system is a component, that ensures that with all these transactions happening, hundreds and thousands of transactions happening every second in a database system, in a typical database system; the data should still remain consistent, in a consistent state, in spite of failures, in spite of concurrences, it must make sure that at no point of time it should happen that, an amount has been debited from an account and has not been credited to account or an amount has been credited to an account and has not been debited to the

corresponding account or the same seat same berth is booked by two persons at the same time and so on, so forth. So, this also includes concurrency control manager, which controls the interaction among different concurrent transactions, which ensures the consistency in the database and provides the total safety.

(Refer Slide Time: 21:32)

So, in total, we have seen the different components of the database engine comprising the storage manager, comprising the query processor, and the transaction manager. Now, we will just have a quick look in terms of who are the typical users of a database system. (Refer Slide Time: 21:57)
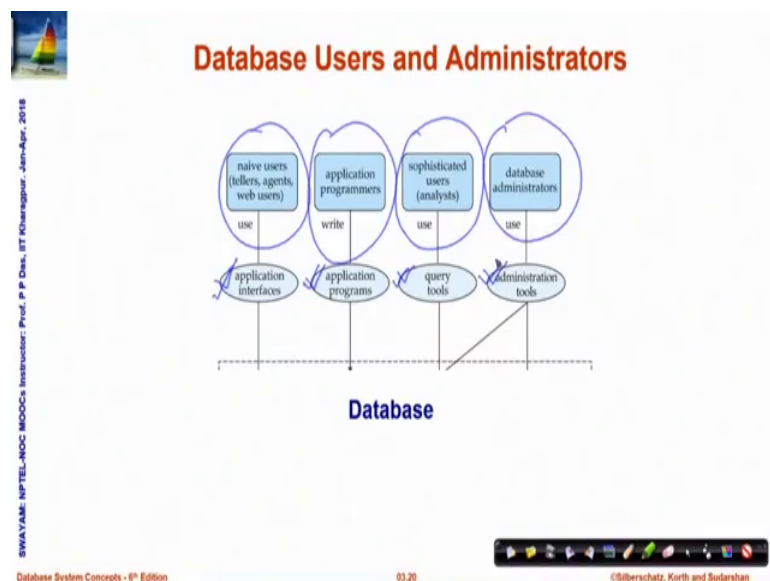
So, if we see grossly, the users of a database system can be grouped into, I mean you can group it in multiple different ways, but this is a typical way to group that, you have the naive users, those like the secretarial staff, who sits at the teller of the bank. Now, that person does not know database management system, but that person just needs to know the particular application. He knows a few set of screens; graphical screens, what needs to be filled up, where which button needs to be clicked and so on and can use this database through an application interface.

So, this is a lowest level of user. Then, you have the set of application programmers, about whom I talked about in my course overview presentation, that application programming is a big chunk of you know, IT services that databases need, who actually write the application programs, while the naive user is similarly using it application programmers are responsible for writing coding this application program. So, they need to understand the database designs they need to understand how to write the query language, how to fit with the application data, input, output all the systems.

The next levels are the analysts, who are called the sophisticated users. So, they design different kinds of query tools, they are responsible for the design of the database, that is a schema the different constraints, the authorizations and so on and manages that over a period of time, when the application requirements change they might need to redesign the schema, migrate the data from an old schema to a new schema. So, analysts are higher level of programmers, they have far more solid understanding of the database management system to be able to design different kind of query tools that, the application programmer will use and at the end there are database administrators.

So, database administrators are people with specialized rights, who can do a lot of privileged operation on the database. For example, taking backups of databases, for example, creating different users. For example, if there has been a failure then how to do the failure recovery scripts, recovering the database and so on. So, they do all kinds of administration tasks, but not the usual day to day data maintenance and you know, query processing and so on.

So, if you would like to know your positions then I would say that by this course, you are going to position yourself amongst the application programmers and the analysts and as I mentioned that the first half of the course is focused on application programming aspect

and the second half would be more focused on the analysis and some of the administrative will do little bit of administration, but not nearly serious administration tasks.
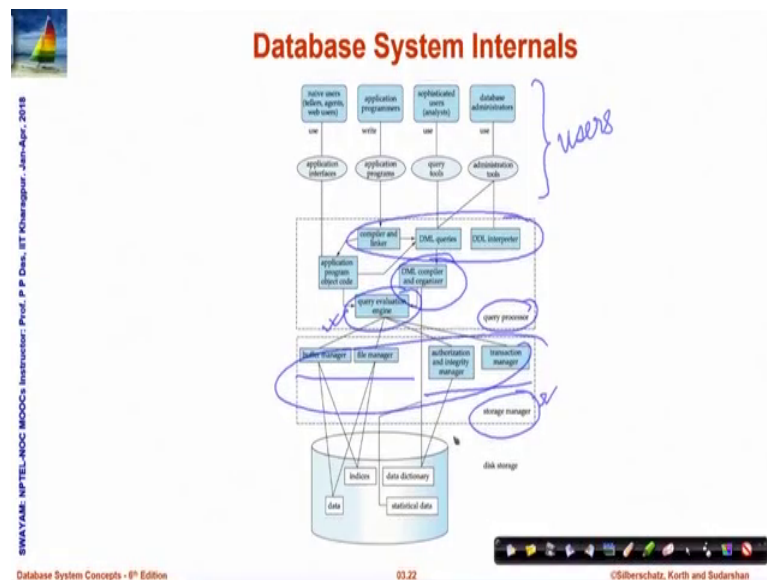
(Refer Slide Time: 25:33)


(Refer Slide Time: 25:35)

Now, we will take a quick look into the database internals and architecture. So, I will take you to this diagram. I am sorry this diagram is little small, in terms of the script size. So, please refer to the actual presentation. So, if you look at the top here is the users. So, it is trying to show what different users use. This is a query processor, that we have

known. So, the query processor gets a query and so that naturally, this query comes from the application program. So, the compiler link these are the basic processing, then the compiler organization, the evaluation engine which actually takes care of the processing of the whole query and then it goes to the storage manager which is now taking this whatever the evaluation engine needs to do, has to go through different modules in the storage manager, which we will talk about these modules, when we do have a discussion module on the storage management.

Later in this course, and we will then talk about what is a file manager and what is authorization and so on, but these are the sub components of the storage management needs to do and then the storage manager is the only one who deals with the actual data, the actual disk storage the different files and so on.
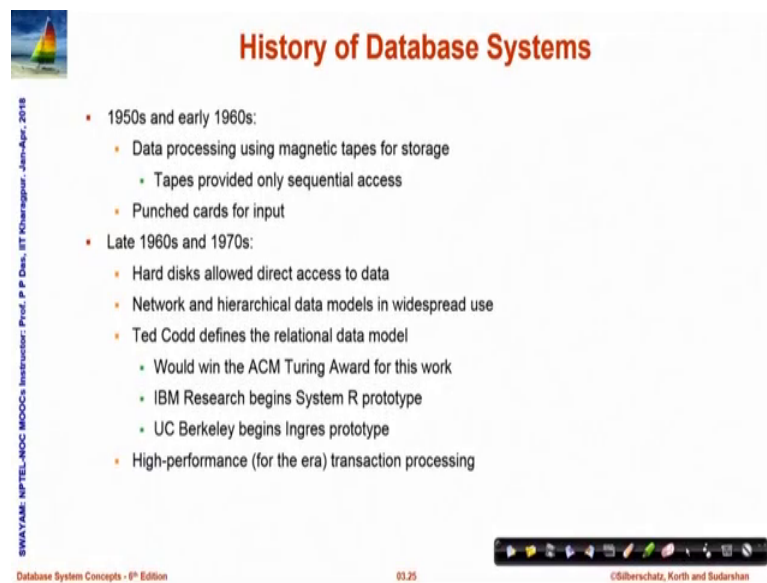
(Refer Slide Time: 27:05)
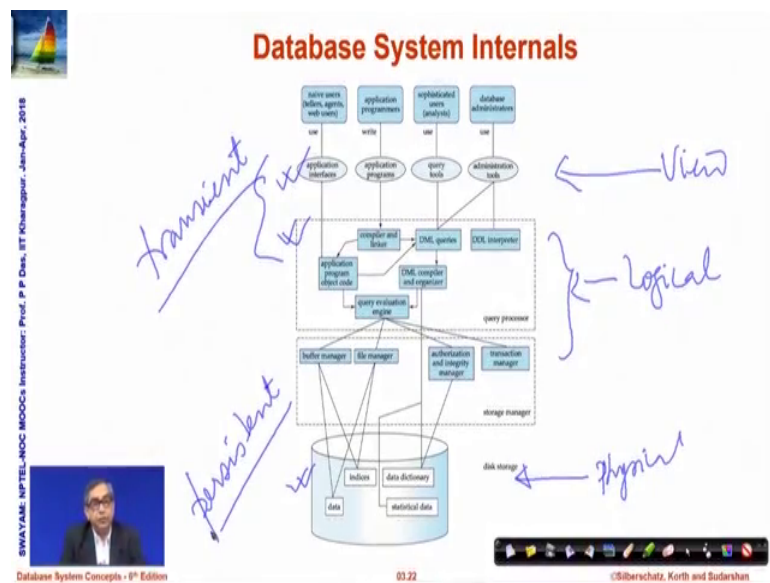
So, as you can see that, the whole system is kind of layered in terms of so, this is your basic physical layer that you have, and this is your final view layer that you have and in between this is a logical layer, that you deal with. So, you can see that the abstractions, as we had talked about are also mapped in terms of them, way the actual database system architecture is managed and finally, the data stays in the disk storage which ensures that whatever data I have is actually persistent in nature. It does not go away. Any data that stays within here or within the application interfaces transient.

So, these data are transient, they will disappear as soon as the application is over, but these data are persistent, they exist beyond this and architectures supports that whole gamut from of all applications or transiency of the applications based on the persistence of the data at the storage. So, that is a basic architectural view of a typical database systems. The actual architecture we form more complex, but we just want to take a schematic view. So, that we can understand it better.

(Refer Slide Time: 28:24)

So, the actual architecture may again vary, based on the computer system that you are using. It could be centralized, we will talk about some of these at later modules. Centralized in the sense that there could be one database server or you know, a group of database servers at the same physical location connected together to which, all applications, all users will connect to, it could be in terms of a client server model which is a very typical client server model, that the programming systems are. So, that typically for example, any of the net based internet based database applications we are looking at are necessarily client server in nature.

What you are doing in the browser is a client and there is a server back far back there the multiple tiers in between them. Databases could be single processes or for performance they could be in terms of multiprocessor parallel databases, the data sets themselves could be so large that, it may not be possible to search on them through a single

processor, in a reasonable time, they could be distributed the data itself could be distributed, tables could get so large that I may not be able to keep them at a single point.

So, these are different aspects of you know, complex real life database system that exist and we will deal with some of those aspects over the course of time, but you have to keep in mind that a final architecture of a database and its associated application will have some of these factors that you will need to know and maybe decide on in terms of history well and I will not go through each and every point here. This is more for completeness and to give you an essence of how things have been going. So, database system started in the 50's and early 60's, then major developments of these relational models and all that started happening in the 70's.

(Refer Slide Time: 30:36)
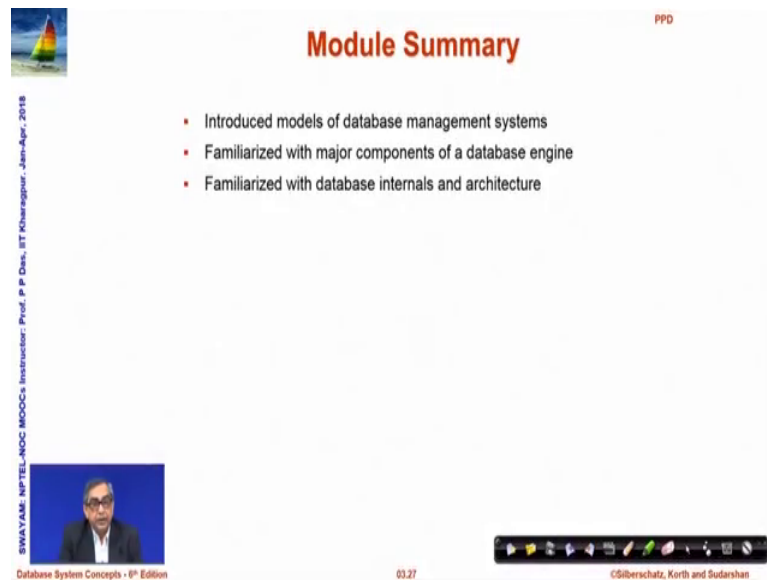
And in 80's, really it proliferated large in terms of prototypes and commercial systems, parallel distributed systems, object based systems started happening in 80's 90's. It really exploded in terms of large decision support, data mining applications, you know applications widespread use of internet based data applications and systems emergence of Google and all that. From 20's, early in early 2,000 it has been XML and automated database administration and now what we are facing in at the big data, which are certainly aspects of other courses, but at the back of back of back at the last layer then often is a strong relational system that exists.
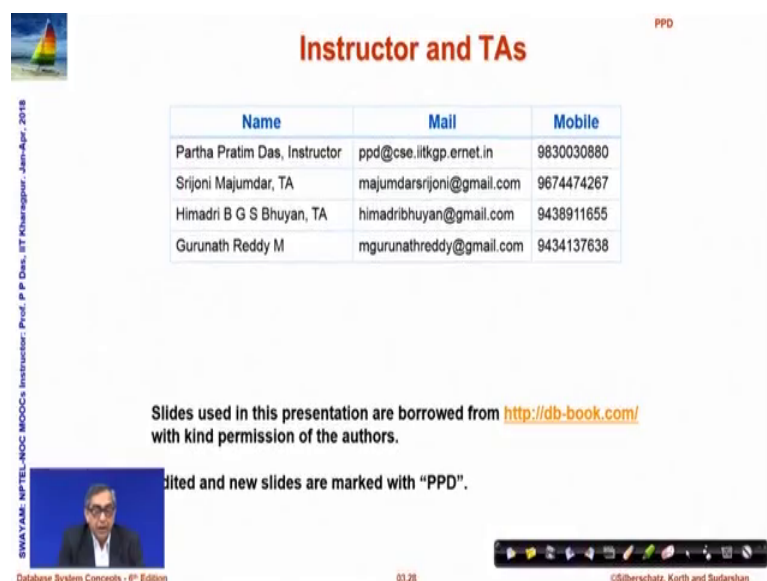
(Refer Slide Time: 31:28)

(Refer Slide Time: 31:41)

So, to summarize, in this module, we have introduced the models of database management system, the major components of a database engine and discussed about the internals and architecture, and this will conclude our discussion on the internals of database management systems, and next we will move on to exposing more on the relational model.