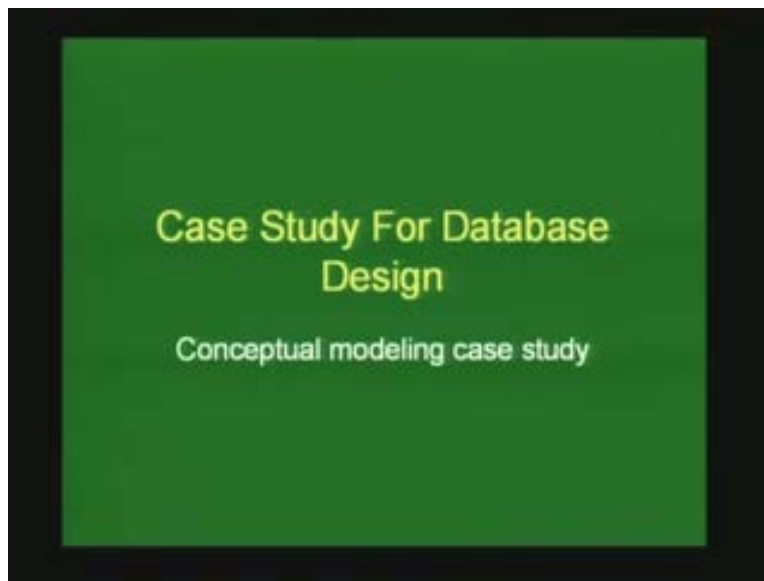


Database Management System
Dr. S. Srinath
Department of Computer Science & Engineering
Indian Institute of Technology, Madras
Lecture No. # 41
Case Study - Part One Database Design

Hello every one. Welcome to another session in database management systems. Until now we have seen different aspects of DBMS design. We have seen what a typical life cycle of a database management system looks like. Essentially we saw that a DBMS is something like or could be treated analogous to the engine of an information system. And what is an information system? Anything, a part of a larger system that deals with information flow, management, storage, retrieval, handling and so on.

So everything to do with information is usually driven by a database management system at the core. So what I assume that you should know by today's class is that you should be familiar with what are the roles of different, what are the roles of a typical DBMS system, what are the different kinds of actors that exist in a typical DBMS and conceptual modeling using the ER model, we saw little bit about ER or entity relationship based design for conceptual modeling. And also the relational model which is the physical model or rather it's not exactly the physical model as in terms of the disk storage that's used but it's still called the physical schema because that is the way in which the database schema appears to all the programs that utilize this DBMS system. So the relational model and different terms as to what is meant by a table relation or normalization, functional dependencies and so on.

(Refer Slide Time: 02:15)



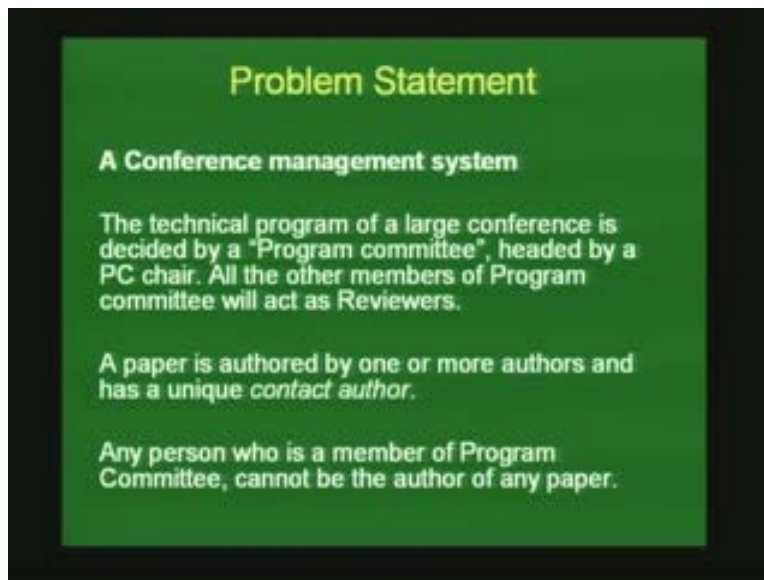
And also a little bit of set of rules as to how to convert a given conceptual model in ER diagram to a given to its correspondent relational schema. So today what will do is let us

look at a typical case study database design case study. So how do we go about designing an application around a DBMS system. Note that we are not here talking about the design of a DBMS itself but we are talking about design of an application on top of a DBMS. So when I was talking about UODs in one of my earlier class, this is what we are going to look at that is we are going to consider particular universe of discourse and then take it up. So rather than taking toy examples and rather than taking an example comprising of just a little bit of database or data management requirements.

I have taken up fairly comprehensive example. At the same time one should be aware of the fact that real life databases for example the moment when we talk about databases, we are first reminded about banks and railway reservations and so on. I have not taken either of them because they are massive database systems, Indian railways for example huge in terms of the amount of transactions that happen and amount of data that is generated and stored every day. So it would be a disservice, in fact it would be plain wrong to take up such a massive database as a case study and in fact we would be simplifying it so much that you will not appreciate the actual complexity that lies in managing such a huge system.

So what I have done here is to take up an actual system that you might actually want to implement as part of a class project or something which and several of such database management systems exists in practice.

(Refer Slide Time: 5:04)



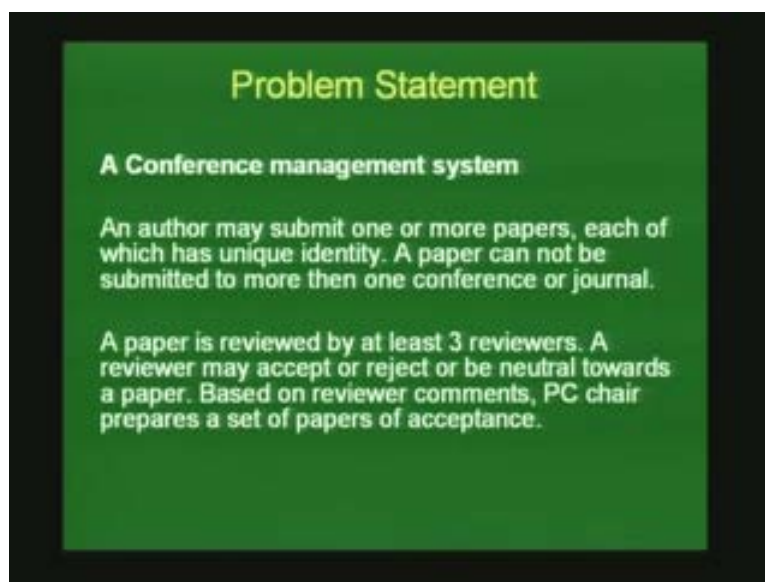
So the case study that we are taking is shown here it is a conference management system. As you know several conferences today are managed by a web based interface where you can manage all the activities and data that are related to the conference. So what is the conference management system contain? So, here is a brief description of the UOD that the universe of discourse and the different kinds of requirements that make up this UOD.

So let me read it out from the requirements itself. Of course this is a simplified conference management system, it does not make sense to take up a real life database in its complete gory details but at least what I hope is that the gist of a particular requirements of a given UOD should be captured by these requirements. So let us look at the requirements once again. The technical program of a large conference is decided by a program committee. So there is a committee of people who decide which paper should be published or which paper should be presented and which paper should not be presented in a given conference. And the program committee is headed by a PC chair or a program committee chair. All other members of program committee will act as reviewers. So people who would submit papers to the conference and they would be reviewed by different members of the program committee.

Now that's about the program committee. So let us look at the next set of requirements. What about a paper? A paper is authored by one or more authors of course and it should have a unique contact author. So there should be one author in the paper who should take responsibility of the paper. So it is to this author, its to him or her that all further correspondence will be addressed to. So correspondence regarding whether the paper is accepted finally or is it rejected or it should be accepted after another process of review and what kinds of changes to be made in the paper and so on and so forth.

So look at the other set of requirements (Refer Slide Time: 07:10). So any person who is a member of the program committee cannot be an author of any paper that is published in the conference. Of course in real life conferences, it's a bit more relaxed than this that is you can actually submit papers to a conference even though you are a program committee member. But for our purposes let us keep it kind of stringent, stringent meaning it's just going to make things simpler. So as long as you are on the program committee, you cannot publish any papers in this conference.

(Refer Slide Time: 08:27)



Problem Statement

A Conference management system

An author may submit one or more papers, each of which has unique identity. A paper can not be submitted to more then one conference or journal.

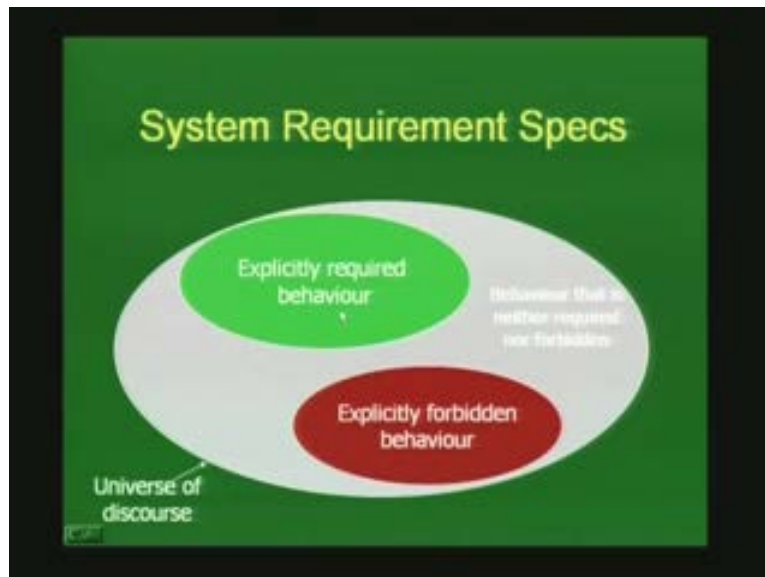
A paper is reviewed by at least 3 reviewers. A reviewer may accept or reject or be neutral towards a paper. Based on reviewer comments, PC chair prepares a set of papers of acceptance.

What is the reason for that? Because a program committee member should not push his or her own papers into the conference, so they should act only as reviewers. What about authors? An author may submit one or more paper, there is no limitation on that. But each paper has a unique identity. So we are selecting papers and not authors, so that's an important thing here. And a paper cannot be submitted to more than one conference or a journey. So if I submit a paper somewhere, I cannot submit the same paper to somewhere else and I cannot obviously also submit a published paper somewhere else.

Now the last set of requirements is, the last block of requirements is that a paper is reviewed by at least 3 reviewers. So when I send a paper to a conference, it goes to at least 3 other reviewers and a reviewer will give suggestion as to whether to accept or reject the paper. So that is shown here (Refer Slide Time: 08:44), a reviewer may either accept or reject a paper or be neutral towards a paper, if the reviewer cannot take decision the reviewer just says that I am neutral to this paper. So the actual decision should be taken by the other two reviewers and in very rare cases all three reviewers would be neutral and well the program committee chair or the PC chair should take a call on such papers. So based on reviewer comments PC chair prepares a set of paper for acceptance and then those set of papers are accepted into the conference.

Now let me pause for a little while here and go through the requirements once again. So carefully look at the requirements of your end user, there is a program committee, there is a PC chair. Ultimately what is that we have to do? We have to take care of the conference activities.

(Refer Slide Time: 09:47)



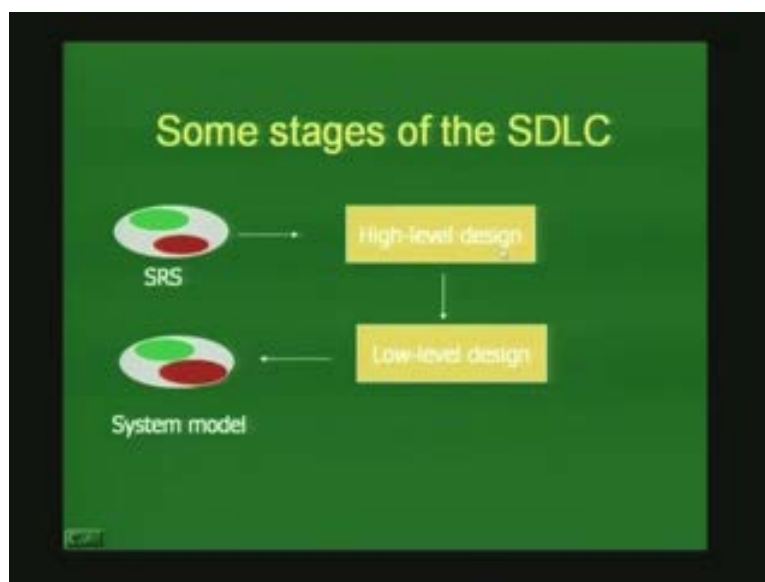
Now if you look at any set of requirements carefully, you'll see that there are two things that a requirement says. A set of requirements will indicate an explicit set of required behavior. That is these things have to be there, the paper says that every paper has to be reviewed by 3 reviewers and every paper has to have a contact author and so on.

There are some things which are explicitly required by our specifications. Similarly there are some things which are explicitly forbidden by this specification. If I have sent a paper to a conference, I cannot or I may not send the same paper to some other conference. So this is a specific forbidden condition, you shall not do this and so on. But if you see again carefully, there are number of requirements or number of things here which one might talk about which or neither required nor forbidden by the requirements.

Can you think of some requirements for the conference management system itself that is neither required nor forbidden? Let us take something like how many paper should a reviewer review? Can I say that a reviewer can review 5 papers, 10 papers or exactly 1 paper and so on. There is nothing that is said in the requirements here. If you look at this carefully the requirements says neither yes or no, so there is nothing said about this requirement itself. So that is an important thing to note in most application design. When we capture requirements, the requirements tells us something that needs to be there and tells us something that should not be there but is silent on a large number of things as well.

So that greatly affects how we design our application and whether our application, suppose you design a DBMS system and you say that because of some constraint, somewhere you say that a reviewer cannot review more than two papers. Is it correct or is it wrong? So there is no specific answer to this because the requirements neither requires this nor forbids such a thing. So usually this is how a systems development life cycle, some of the top or the early stages of a system development life cycle would look like. So if you look at the slide here, you have the systems requirements specifications were there are set of explicitly required conditions and there are set of explicitly forbidden conditions and this is the entire UOD where the number of things which are not addressed by the requirements.

(Refer Slide Time: 12:35)



Now based on these you get a high level design of your system, usually this in the form of a ER model or whatever when it comes to DBMS design. So you end up with a ER diagram here, in turn you reduce the ER diagram to a relational schema or a low level design and then you get a system model, relational schema plus transactions and so on small set of application logic and some set of constraints, triggers, stored procedures or so on and then you get a system model.

Usually what happens is this process, how do you get design from requirements or how do you move from high level design to low level design. These sets of processes involve human activity or human creativity to be more specific. And as is so common with human actions, the system model may not exactly reflect the systems requirements specs. Ideally what should the system model do? The system model should exactly reflect the requirement specification here.

So as shown here, the red spot is slightly bigger here. What is that mean? That the system model has more forbidden conditions than what is explicitly forbidden by the requirements specification itself. So it brings us to some two important concepts, when we are designing real life system, the concepts are what are called as liveness and safety.

(Refer Slide Time: 14:14)



So look at the English definitions of liveness and safety. Liveness means what, that something is alive or something is existent and so on and safety of course is obvious. Now if you look at the systems requirements specifications, why would a set of requirements. So let us go back here (Refer Slide Time: 14:31). Why would a set of requirements say that this is forbidden, why would a set of requirement say that a member of the PC committee shall not be an author of a paper.

Why would a set of requirements say? Because it would compromise the integrity of the system if that were to be alone because if I allowed a PC committee member to be an

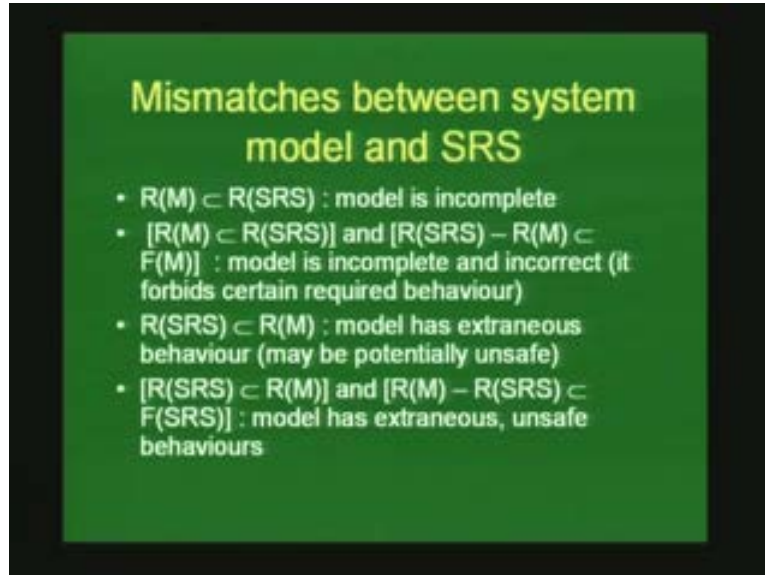
author of a paper, there is quite a likelihood or there is quite a chance that the PC committee member may push his or her own paper and have an unfair advantage over others. So it is the safety of the system is getting compromised, so that is why I forbid the behavior, forbid this activities. So essentially whatever is forbidden usually constitutes of safety requirement, in order to safe guard the system against integrity violations I say that this shall not be there.

However what is the simplest form to build a safe system? How do we build a system that is absolutely safe and from any kind of integrity violations? Simple, don't start the system at all. If a system that doesn't work, it is absolutely safe. If your database system doesn't work at all, it is absolutely safe because it does not violate any integrity constraints at all. So that is why a trivial way of ensuring safety is to make a system that doesn't work. But that is not what we want. In addition to safety we need, we require certain behavior to happen. So those are what are called as liveness requirement that is the system should perform certain activities and should not perform certain activities.

So let us use some notations when just to talk about mismatches. Now suppose I say that R of SRS here (Refer Slide Time: 16:17) is the set of required behavior by the SRS or the systems requirements spec. Similarly F of SRS is the set of forbidden behavior or safety constraints specified by the SRS. Similarly let us say R of M or where M is the model that we build, the final system model that we build. So let R of M denote the set of all liveness criteria in the system model that is the system model will do this. And F of M denotes the set of safety criteria in the system model that is the system model will not do this and so on.

Now when we talk about a system model that is when we talk about building a system model from a set of requirement spec, we can think of various kinds of mismatches that can occur. So, various things can go wrong when we are talking about capturing user requirements into a system model. What are the things that can go wrong? A tentative list of things I mean these are not the only thing that can go wrong, in fact in addition to this a huge number of things can go wrong but anyway.

(Refer Slide Time: 17:20)



Now let us say what if R of M, remember what is R of M. R of M is the set of required behavior of the model. What if R of M is a proper subset of R of SRS. What is this mean? The set of required behavior by the model is a proper set of the required behavior by the SRS that means that the model is incomplete. The SRS require certain behavior to be done but you don't implement all the behavior, you don't factor all those behaviors, you factor a subset of those behaviors.

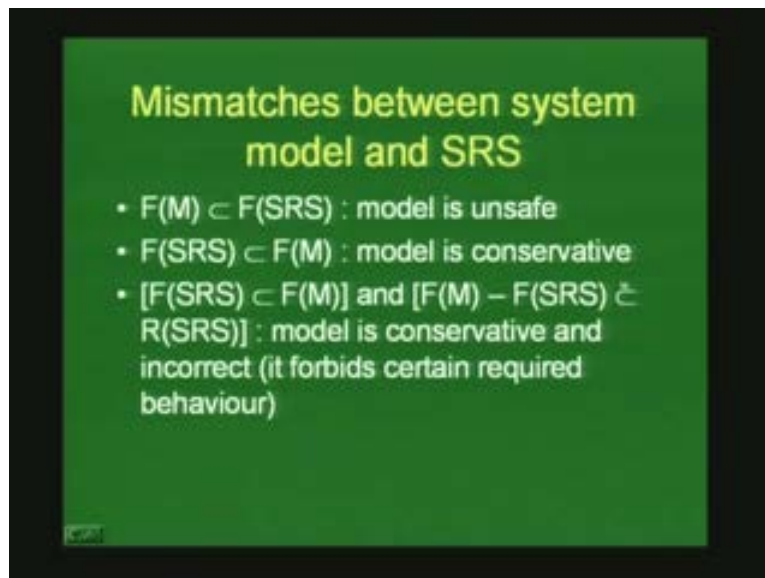
Now what if in addition to this R of M being a proper subset of R of SRS, in addition to this let us say the R of SRS minus R of M that is the set of requirements specified by the systems requirements spec which are not factored into the model is actually a part of F of M, is actually a part of the set of forbidden behaviors by the model. What is that mean? It means that not only does the model address all the requirements in the requirement spec, in fact there are certain requirements of the requirement spec that the model actually forbids that is that the model will not do. So it means that the model is not only incomplete, it is incorrect it forbids certain required behavior.

Similarly what if R of SRS is a proper subset of R of M that means that the model is performing more activities than what is required by the SRS itself. That is the model has extraneous behavior and having extraneous behavior is not that is having an added feature for example, suppose the model as for the birth date of the author when were you born and so on does not always be a desirable feature, it can actually be potentially unsafe. When is it potentially unsafe, when R of SRS is a proper subset of R of M that is what I saw here and the difference between R of M and R of SRS is actually a part of F of SRS that is the extra behavior that or the extra so called value addition that your model is doing is actually part of the set of safety conditions that is actually forbidden by your requirements. So the model has extraneous and unsafe behaviors.

So when you build a system model, you should be careful to or this is one set of guidelines by which you can measure whether your system model is good enough against the requirements. That is just try separating the requirements into set of required behavior and set of forbidden behavior and your model also into set of required behavior and a set of forbidden behavior. So let us see some more mismatches here. Now what if F of M is a proper subset of F of SRS . That is the set of all forbidden things of the model is a proper subset of the set of all forbidden things of the SRS that means that the model is unsafe. That is the requirements require you to forbid certain things which you are not forbidding.

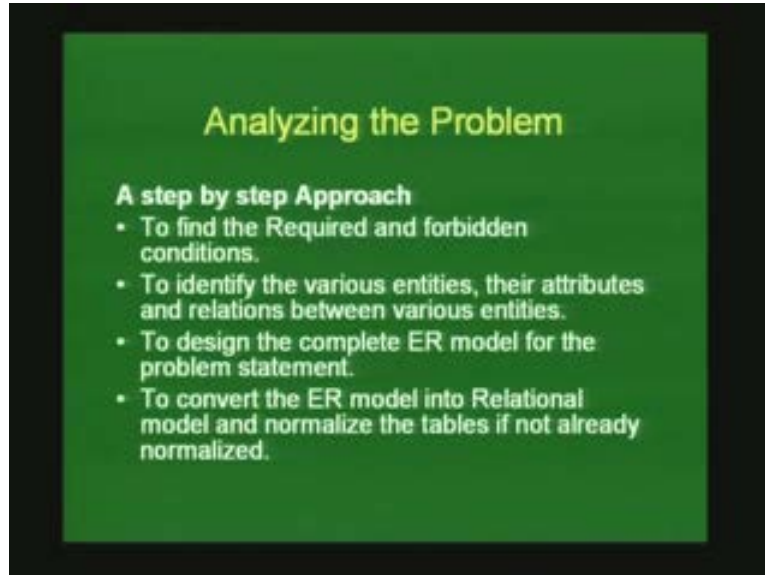
Similarly, what if it is the other way around. That is the model forbids more than what is required by the SRS then you say that the model is conservative. Now conservative again doesn't mean that you are safe. Usually in English, we say that oh let us be conservative and go about like this and take this action and so on. But just be saying let us be conservative doesn't necessarily mean that you are safe. Why? Because you could actually be violating a liveness criteria.

(Refer Slide Time: 20:58)



So this is the case here that is F of SRS is a proper subset of F of M that is the model forbids more than what is required to be forbidden by the SRS and the difference that is what the model forbids which is not forbidden by the requirements is actually part of the required behavior of the SRS. So if forbids something which actually needs to be there, so in being more conservative you are actually hampering the liveness of the system. So just being conservative doesn't always mean you are building a safe model.

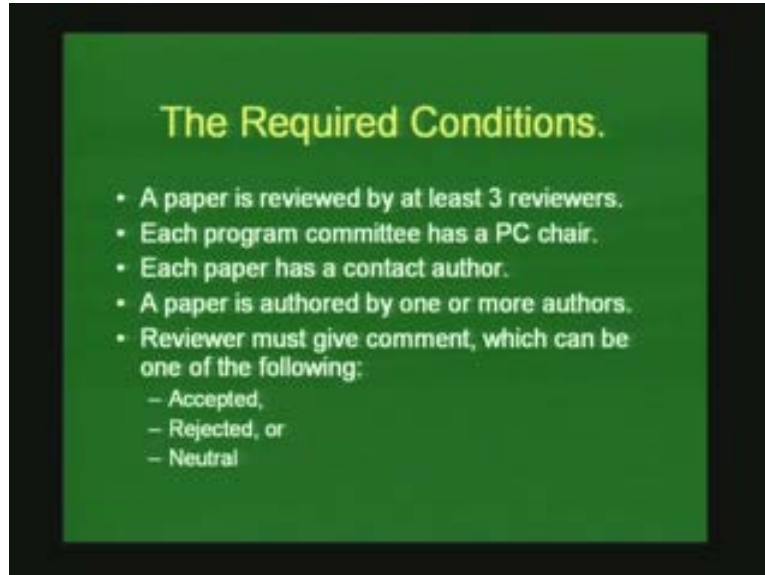
(Refer Slide Time: 22:56)



So let us see, let us take a step by step approach to see to let us try to divide our requirements coming back to the conference example to see what kinds of required behavior are there by the model and so, by the requirements spec and so on. So of course the kind of the example that we are seeing here is a simplified example and real life examples are far more difficult than these but anyway this gives the gist of how to factor a requirements into set of required behavior or liveness behavior or set of safety conditions and so on.

So what could be the step by step approach? Let us, the first thing is we have to find the set of required and forbidden conditions. Then once you start that then start identifying the various entities, their attributes, the relationships between entities and so on. Then build a complete ER model for the problem statement and then convert the ER model into a relational model and perform normalizations if they are not already normalized.

(Refer Slide Time: 23:46)



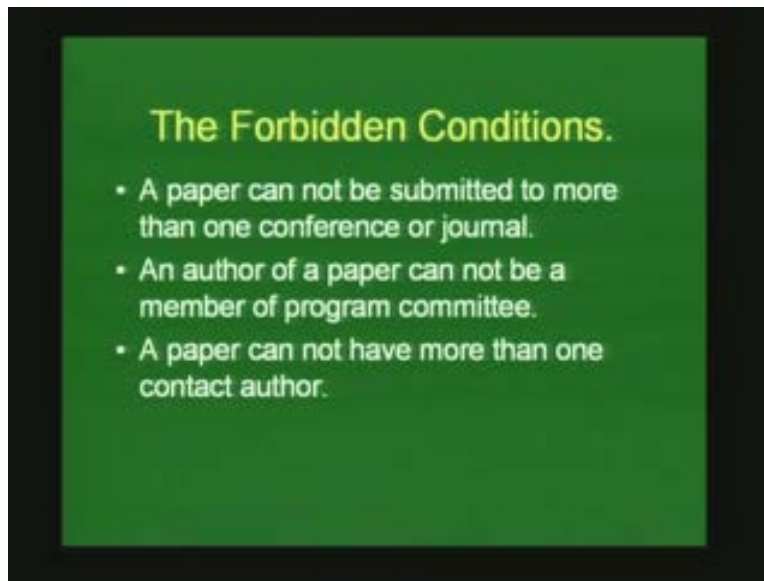
So let us look at some of the required conditions. A paper is reviewed by at least 3 reviewers that actually means that a paper should be reviewed by at least 3 reviewers. So if I try to review or if I try to accept a paper that is being reviewed by only 2 reviewers then your conference management system should flag an error, it should not allow it to do that. So this is a required condition that is a paper is reviewed by at least 3 reviewers. Each program committee has a PC chair, so this is another required condition, you cannot have a program committee without a chair.

Each paper has a contact author. If you go back through the requirements that we saw, all these have been picked from the set of requirement itself. So each paper has a contact author that means that each paper should have a contact author and so on. A paper is authored by one or more authors, so obviously this means that you should not accept a paper without any authors in it. And reviewer must comment or must give comments which can be one of the following accepted, rejected or neutral. You cannot, the reviewer cannot give any other comment other than this three and the reviewer should give one of these three.

The reviewer cannot remain silent saying that I am neither accepted nor rejected nor I am neutral about the paper and so on. And the reviewer should give only one of this. So this is a set of required condition. What are some of the forbidden conditions? We saw some examples already. A paper cannot be submitted to more than one conference or a journal. So you may not submit a paper to more than one conference and so on. An author of a paper may not be a member of a program of the program committee. So that's another forbidden, explicitly forbidden conditions so that have been explicitly stated in the requirements that these are forbidden and a paper may not have more than one contact author. So there has to be one and only one contact author, so that it may not have more than one. So these are some kinds of required conditions and forbidden conditions and so on.

So when you build a system model, what you should be able to do is take up each set of required conditions and see whether your model also has that required behavior. Take each set of forbidden conditions and see that whether your model also forbids those conditions and the other way around, take each set of required behaviors by your model and see that whether they are actually required by the set of requirements so on.

(Refer Slide Time: 24:53)



So let us now go to the next step and start identifying entities. So how do we identify entities and what is an entity. An entity is some logical item one could say or logical something of which has an independent existence of its own. So I was about to say logical entity which kind of becomes a circular definition in this case. So any way let us look at the problem statement once again.

(Refer Slide Time: 25:57)

Identifying Entities

Statement: "The technical program of a large conference is decided by a Program committee, headed by a PC chair."

Entities: 1) Conference
2) Program Committee
3) PC Chair

Statement: "All the other members of Program committee will act as Reviewers. ..."

Entities: 1) Reviewers

The technical program of a large conference is decided by a program committee headed by a PC chair. If you just look at that statement, you can already find several entities here. So the conference is an entity of a large conference, conference is something that has a logical existence. Program committee is an entity here essentially the nouns in this sentence and is headed by a PC chair is an another entity. And one could even say the technical program could also be an entity and so on. So just reading through each sentences, you can identify what could be potential entities in your system.

(Refer Slide Time: 27:25)

Identifying Entities

Statement: "... A paper is authorized by one or more authors. ..."

Entities: 'Paper'

Question: Is 'author' an attribute of 'paper' or an entity itself?

Author has an independent existence. He may participate in other relations, too. Author may write more than one paper. Thus, taking it as an attribute may lead to lot of redundancy.

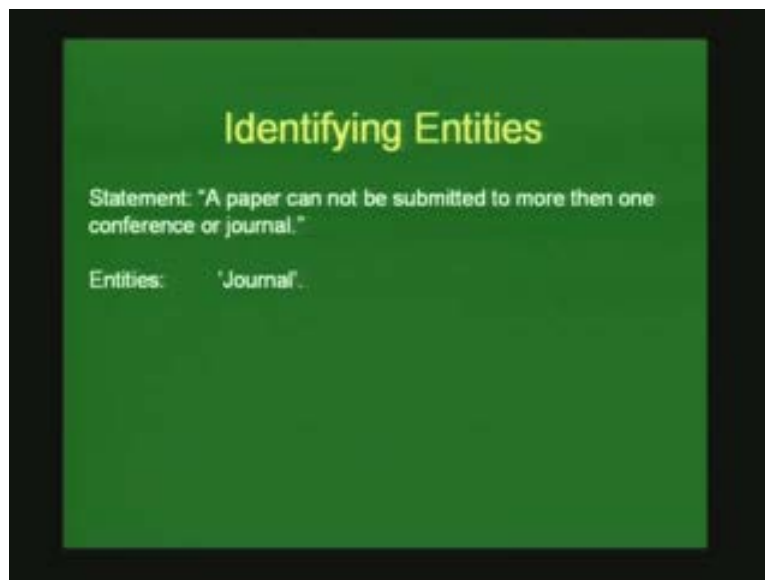
So, in our case, it is better to take 'Author' as an entity.

Similarly the next statement, all other members are some statement down here. All other members of program of the program committee will act as reviewers. So reviewers is another entity as soon as we found. So similarly a paper is authored by one or more authors, so paper is an entity. Now author is a, now here there is a question this is not as simple as that. So is author is it an entity or is it an attribute. Is an author an attribute of a paper that is a paper has an entity and this paper is authored by so and so authors and so on.

Now some cases we can make author as an attribute of a paper but here we see that author also has an independent existence. Why because we have something called a contact author, we have something called author may not be a member of the program committee and so on. So the author may actually participate in other relations as well and an author may write more than one paper. If I make author as an attribute of the paper entity, there is no way to relate or there is no relate to equate that paper one and paper two have been published by the same author and so on. So there is no way to equate those two papers. So, in our case it's better to take author or its better to design author as an entity itself.

Similarly again some more, a paper cannot be submitted to more than one conference or a journal. Again there is an entity called journal and so on. Conference we already saw is an entity. Now what about relationships? Now that we have identified entities, of course we are no way near to finishing the identification of entities but let us look at relationships. I mean you should have got the ideas as to how to go about identifying entities and its attributes and so on.

(Refer Slide Time: 28:31)



Now again take a look at the statement. Now what are the entities that you can see in a given statement? The entities would generally be the nouns of a particular statement. Now what could be the relationship here.

(Refer Slide Time: 29:05)



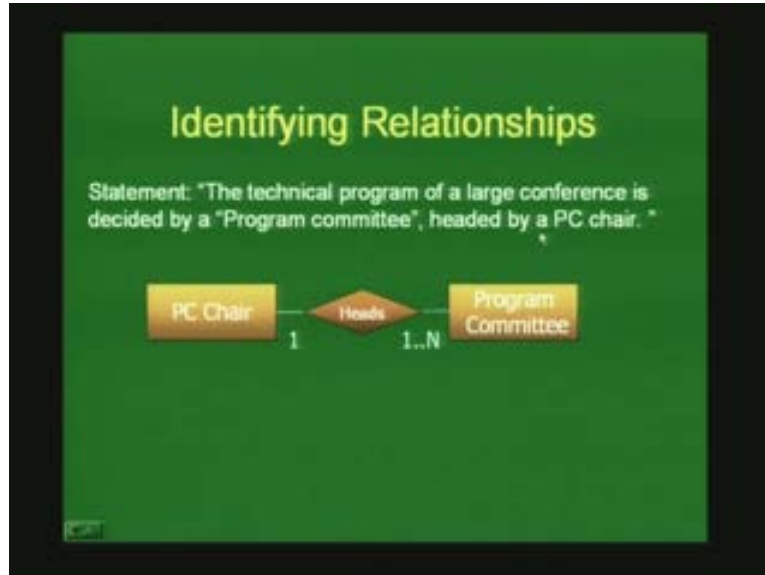
Now look at the verb something is headed by something else and so on, so or handled by and so on. So the verb statements that connect two or more nouns would actually be prime candidates for relationships. So if you look at this statement here, the technical program of a large conference is decided by a program committee headed by a PC chair. So as you can see here, this part already forms a relationship that is or rather the first part is a relationship here that is conference is handled by program committee that is handled by or technical program decided by if I have to make it very explicit.

So conference is handled by a program committee and as you can see the technical program of a large conference is decided by a program committee. So basically it is a one to one relationship that is one conference, one program committee. But then look at this here program committee, we have made program committee into a weak entity here. Why is it a weak entity? First of all what is the weak entity? If you notice carefully, a weak entity is an entity or if you remember your ER modeling classes, a weak entity is an entity which does not have an independent existence of its own, its existence is defined by a relationship.

And the relationship that defines a weak entity is also called a defining relationship. So this is a defining relationship, so shown by double arrows like this (Refer Slide Time: 30:53) and this is what is called as the total participation if you remember ER classes again. So a program committee totally participates in this conference that is the same program committee may not participate in more than one conference entities and this is the defining relationship. So if there is no program committee then there is no conference.

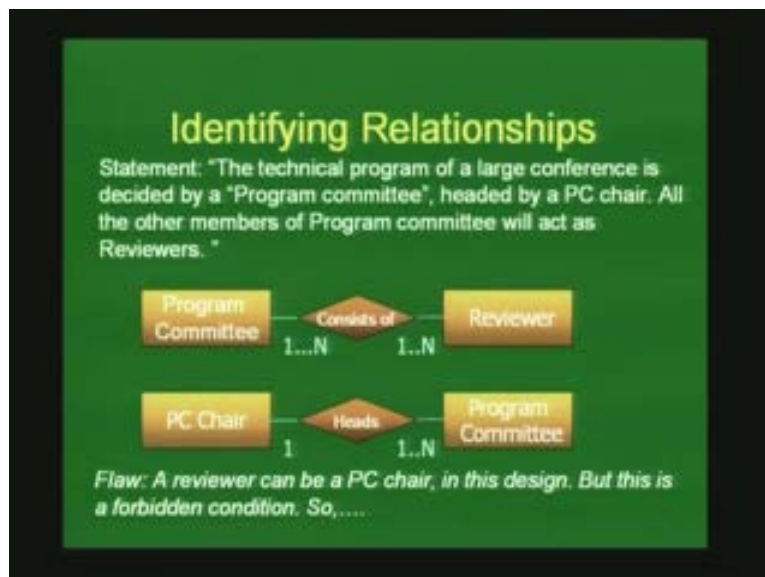
Similarly let us look at another statement. The technical program committee of large conference is decided by a program committee headed by the PC chair, same statement. So again program committee here and PC chair heads program committee that is a PC chair is an entity which we have found and a PC chair heads a program committee.

(Refer Slide Time: 31:53)



Now here if you can see again, we see that this cardinality that is a program committee is headed by a PC chair is clear that is one program committee should have exactly 1 PC chair. But 1 PC chair can head how many committees? It is neither specified or rather it is neither required nor forbidden, it's not specified in the requirements. So here we have made it into a N cardinality 1 to N or whatever. That is a program, a PC chair can head any number of program committees and so on. So because there is no explicit specification as such in terms of how many program committees can a PC chair head.

(Refer Slide Time: 32:26)



Again some more relationships, so take a look at this statement. The technical program of a large committee whatever headed by a PC chair and so on. All other members of the program committee will act as reviewers. So let us say we already had this one. That is program committee is here and that is headed by a PC chair and 1 PC chair can head 1 to N program committee, program committees and so on.

Now a program committee consists of reviewers which is apparent by the second sentence. That is all other members of program committee will act as reviewers. However it is not exactly correct, it's not exactly what is specified by the requirement. That is what the requirement say all other members of the program committee. So basically what does this means? That is all members who are not PC chair that is who are not acting as PC chairs can be reviewers of this. So if I take two separate relationships like this in isolation, they don't form a consistent set here because it is violating a forbidden condition, it is violating a condition that the PC chair may not be a reviewer.

So how would you rewrite this condition here? So basically we will introduce a new entity called members and basically form, what might be termed as a generalization specialization relationship. So remember the extended ER model allows for a specialization relationship were given a member or given a entity, you can inherit one or more entities from it that is it actually shows the is a relationship.

And in addition to the is a relationship, here we have this circle called D. What is the D specified? D basically specifies that these are disjoint entities that is no entity instance that is part of reviewer can also be PC chair and vice versa.

(Refer Slide Time: 34:07)

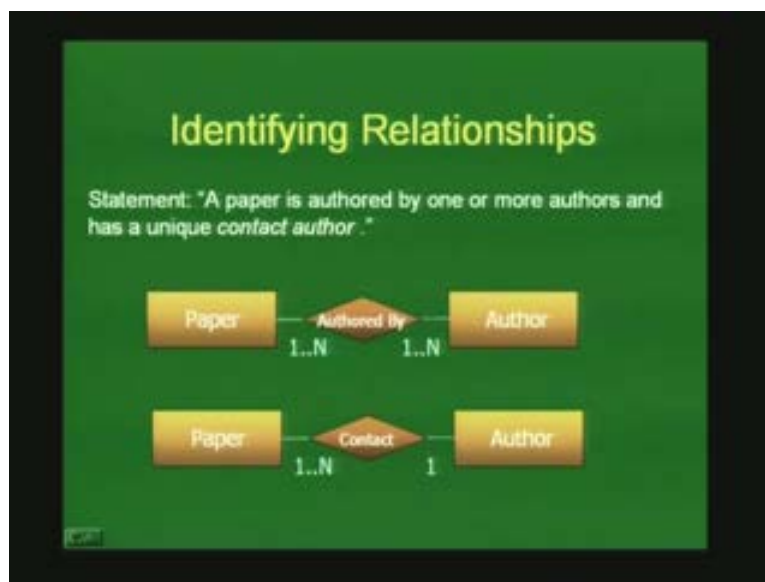


So our committee would now look like this. Committee would consist of members where members would in turn consist of reviewer and PC chair which are disjoint. So committee can consist of 1 to N member, N number of members but there has to be

exactly one PC chair. So basically in addition to this, we have to give a cardinality of 1 here and N here for N reviewers and 1 PC chairs.

Again let us look at some more statement when to identify relationships. A paper is authored by one or more authors and has a unique contact author. So again we can see that we can identify relationships straightaway here, paper is authored by authors, one or more authors. And an author can author how many papers. There is no specification, so we just introduced 1 is to N. So we are kind of being liberal model, we are not being very conservative model that is we are allowing for more behaviors than has been required that is an author can submit any number of papers unless it is explicitly forbidden of course.

(Refer Slide Time: 35:31)



And a paper should have a contact author that is a paper here any number of papers should have exactly one contact author. So, one author who acts as the contact author. Again in isolation these two relationships are not sufficient because why do you think they are not sufficient, let me pause for a little while here. Why do you think going back to these two set of entities, why do you think they are not sufficient in themselves. I am sure you would have got the answer.

The thing is while a paper can be authored by one or more authors and a paper can have contact authors, there is no relationship that states that the contact author should be one of the authors here. So one of the authors from here should be taken and be formed as the contact author for a given paper that is so you have to, in the earlier case there was a disjoint relationship, PC chair cannot be or may not be a reviewer. And here there is a membership requirement that is a contact author ought to be one of the authors of the paper.

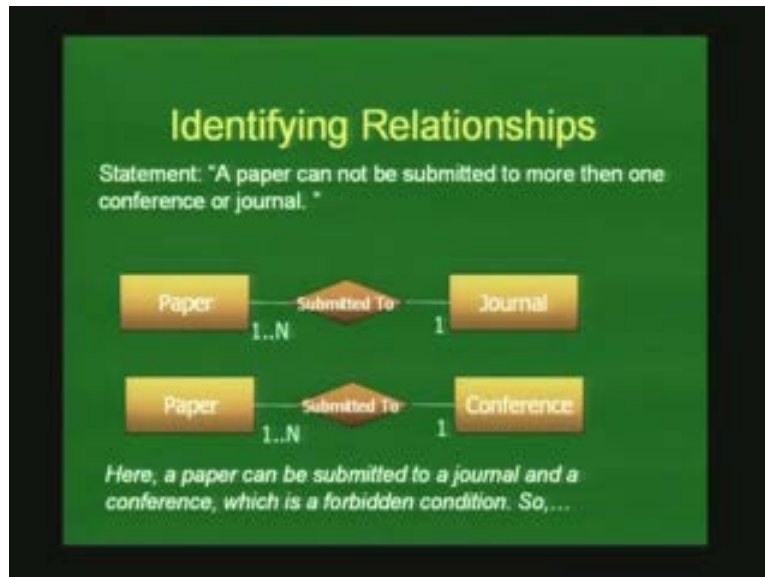
(Refer Slide Time: 37:29)



So how would you go about, let us come back to this again later. So again any person who is a member of the program committee cannot be the author of any paper. So we will come back to that earlier thing after taking this other constraint also into perspective and then draw the entire set of relationships at one goal. So what is this say here? Any person, note that now again person is a noun here so we need another new entity called person.

So any person who is a member of the program committee cannot be the author of any paper. One way to show this is have a person called, have an entity called person and make a disjoint specialization between member. Remember we had an entity called member here, members or whatever. So we had an entity called member and an entity called author. So an author may not be a member or a PC committee member and a member may not be an author and both are persons and so on. So that way you can identify that any person cannot be both an author and a member of the PC committee or the program committee. And a paper may not be submitted even though sometimes when talking in English, we say a paper cannot be submitted to more than one conference or journal. To be more precise, it actually should be a paper may not be submitted to more than one conference or a journal.

(Refer Slide Time: 38:35)



So again here what, first of all what can we imply from that? A paper can be submitted to a journal and a paper can be submitted to a conference. So these two can be implied but what is that we actually need. What we actually need is that while a paper can be submitted to a journal as well as submitted to a conference, it may not be submitted to more than one conference or a journal, the same paper may not be submitted to more than one conference or a journal.

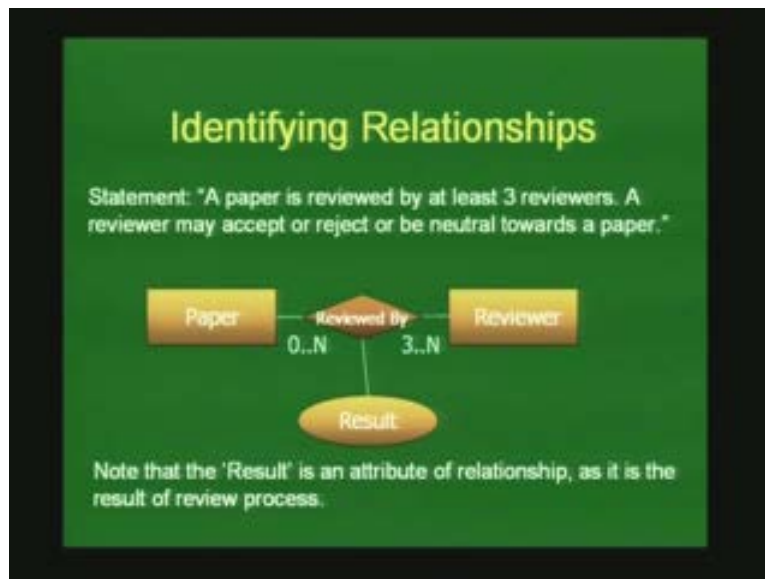
(Refer Slide Time: 39:21)



So here in order to identify that we have used the union or the concept of a union were it is, one might call it the opposite of the specialization condition were you take two or more entities and form a union out of them and form a single entity. So take a journal or a conference and form a union out of them and make an entity called event and the paper is submitted to an event. And how do you say that it has to be submitted to only one conference or a journal at any point in time, only thing is make this, the cardinality of event as one here.

So given paper can be submitted to one paper here or 1 to N papers that is a given event may have N papers and a given paper may be submitted to exactly one event. And what is that event? An event could be a conference or a journal. Calling a journal as an event is not exactly correct sounding in terms of the English definition but anyway we have used this term but you might think of a better term than event to specify or to take the union of journal and a conference.

(Refer Slide Time: 40:50)



Similarly a paper is reviewed by at least 3 reviewers and a reviewer may either accept or reject or be neutral towards the paper. So what is this statement say? A paper is reviewed by reviewers, so N number of papers is reviewed by 3 to N that is at least 3 or anything more than 3.

Now take a look at the second half of the statement. A reviewer may either accept reject or be neutral towards a paper that means the reviewer is going to give a result. The result is either accept, reject or neutral or be neutral. Now, if you see carefully the attribute called result does not belong either to the reviewer nor to the paper because a paper, the result of a paper is actually a combination of the results of three or more reviewers.

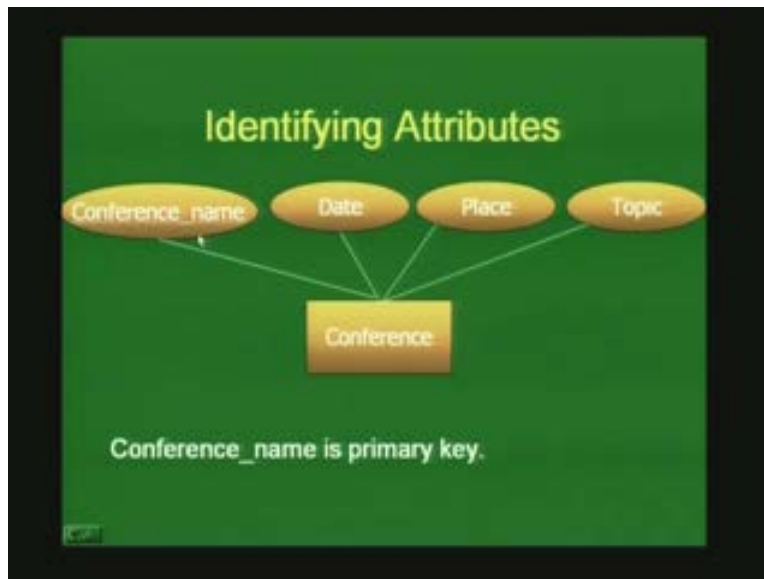
And a reviewer may be reviewing more than one papers, so you can't assign result to a reviewer as well. So the attribute called result is actually an attribute of the relationship

itself. So remember that we had talked about attributes which belong to relationships. So as long as there is an instance of this relationship existing in the system, an instance of this attribute may also exist in the system. Whenever the relationship instance does not exist, when can a relationship instance not exist? For example when there is a paper which is not assigned to any reviewers for example then there is no instance of the relationship at all that is reviewed by and so on.

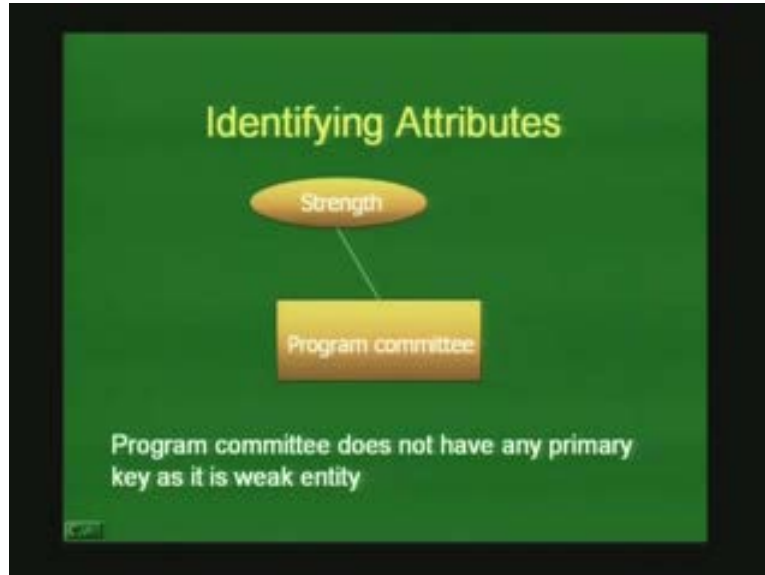
So there is no question of a result existing in this or even when a paper is assigned to just 2 reviewers. So if you look at the relationship here, the relationship requires that a paper be submitted to at least 3 reviewers. So therefore there is no instance of such a relationship existing and therefore there is no instance of the result in the databases.

Similarly coming back to attributes. Now let us say conference. Now the requirements doesn't say anything as such but as application designers, it is our responsibility to identify some of the major attributes of a particular entity and also identify key attributes. So here in this case a conference name and the date and the place, topic and all of those things would be attributes of the conference and usually something like the conference name would be the primary key or would be the key attribute of the conference.

(Refer Slide Time: 43:21)



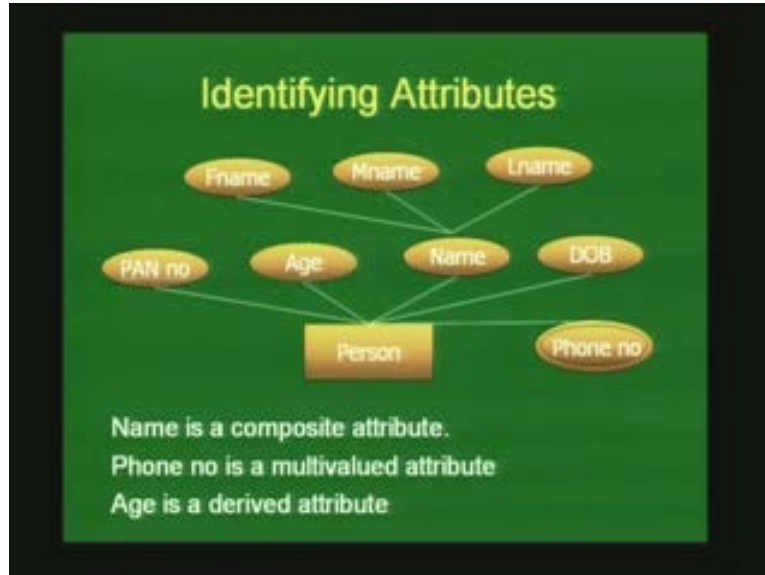
(Refer Slide Time: 43:37)



Similarly program committee does not have any primary key because it's a weak entity type which we actually saw earlier. So a program committee does not have a key attribute but it may have other attributes like what is the strength of the program committee, how many people are there in the program committee as of now and so on. So a program committee has a is a weak entity type having no key attribute but it has its own other attributes.

And have a look at the entity called person. For person again you can think of lot of different attributes, what is the name of the person. Now when you say name, usually in several cultures, you actually divide a actually name into first name, last name, middle name and so on, the initials and title and so on and so forth. So name could actually be a composite attribute here which in turn has multiple other attributes, many other attributes say first name, middle name, last name, title, initials and so on and so forth.

(Refer Slide Time: 44:11)

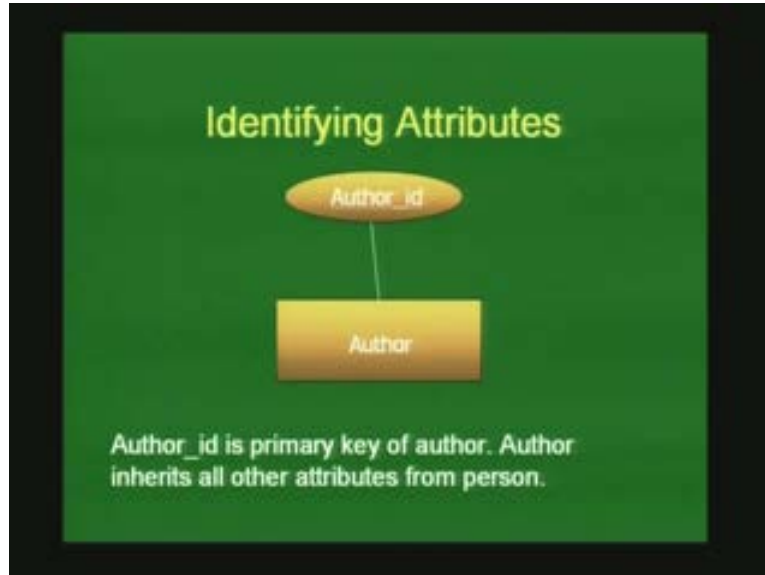


Then there could be age or date of birth address and usually you need to have a unique identity to unique way of identifying a person. This was actually created by some of my students whose pan number has a key attribute for a person but usually in a, it is quite unlikely that in a conference setting, you would ask for a person's pan number. Usually it would be the email address of this person which or the contact email address of this person which would be the key attribute.

Similarly they could have something like phone numbers and phone number here is treated as a multi valued attribute which means that this attribute can have multiple values. So what is that mean? That a person can have multiple phone numbers. And I hope you know the difference between a multi valued attribute and a composite attribute. A composite attribute is also made of multiple attributes but each of these different attributes may belong to different domains.

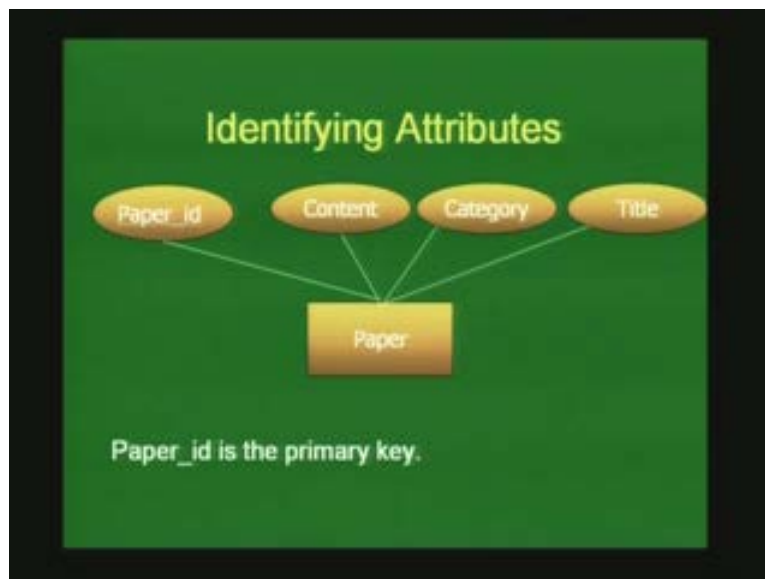
So name can have first name, middle name, last name were a middle name can be constraint to be a single letter, if the middle name is an initial whereas first name and last name can be varchar or strings and so on. But when I say phone number, when there are multiple values for that phone number, all of these values belong to the same domain or of the same type. So that's the difference between a multi valued attribute and composite attribute.

(Refer Slide Time: 46:25)



So similarly other these thing when I say that, when I say author you can give an Author_id for each author, a login id or whatever and every other attributes of a person would be inherited by author because author is a person and so an author is supposed to have a pan number and date of birth and phone number and so on and so forth.

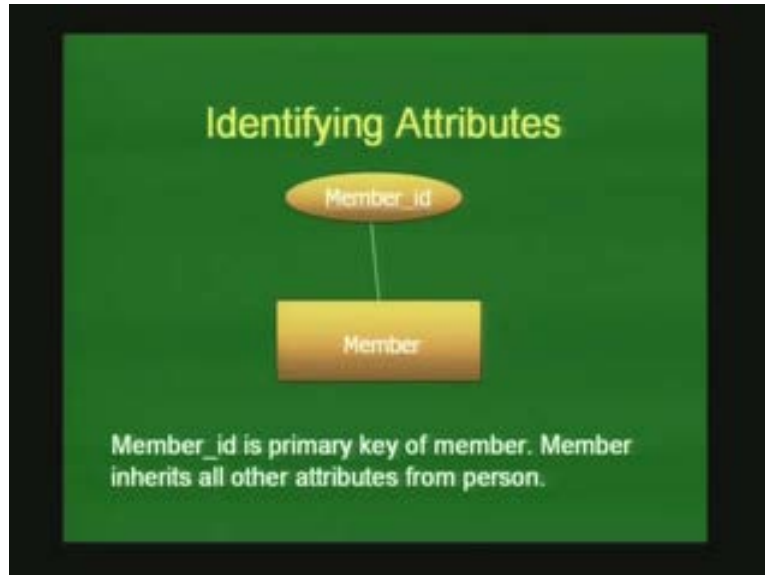
(Refer Slide Time: 46:45)



Similarly for paper, its already specified there that each paper would have a unique identification or a unique key. So for paper, paper_id would be the key and several other things what is the title, what is the category, the classification of those attributes, the

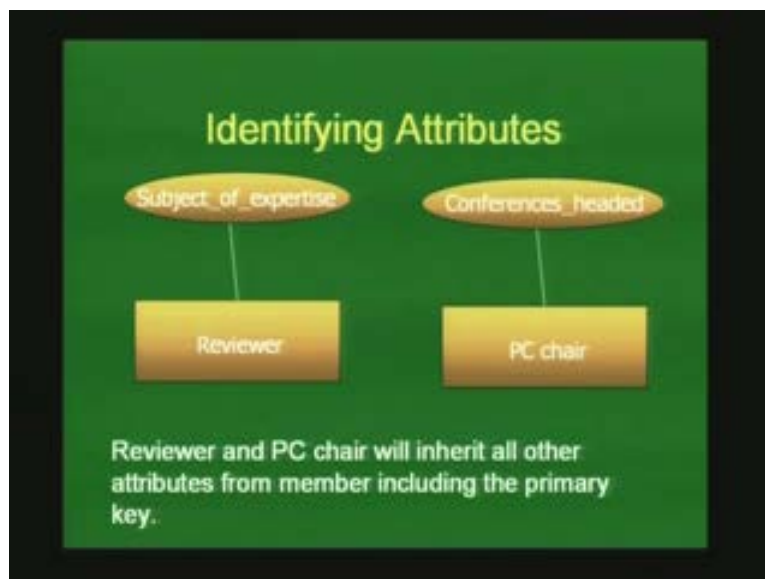
paper content itself, the keywords that are given for the paper and so on, all of them could be attributes of a paper.

(Refer Slide Time: 47:16)



And when I say PC member, you can again give member_id for each members, again some kind of a login id or something which would form the primary key for each member.

(Refer Slide Time: 47:32)



And again several other this thing reviewers and PC chair. So reviewer would have something called subject of expertise and PC chair would have something called

So conference is handled by a program committee and program committee is a weak entity type, so it has no existence without a conference. And program committee consists of different members, so among the members there are reviewers and a PC chair and there is one PC chair, there is exactly one here. And this is a disjoint relationship that is a PC chair may not be a reviewer and a reviewer may not be a PC chair and a PC chair heads program committee. So PC chair may head one or more program committees like this.

And similarly you have a conference and a journal forming an event. That is a given conference or a journal may be forming an event to which a paper is submitted. So a paper may be submitted to 0 or 1 event. So you may not submit a paper at all or you may submit it to at most one event. And an event should have at least one paper or it may have any number of papers and so on. And a reviewer reviews a paper or paper is reviewed by reviewer and there is a constraint here that is a paper is reviewed by at least 3 reviewers.

Now the reviewer, the review of a paper, the process of review of a paper will result in a result being assigned or will create a new attribute called result which the reviewer assigns for this paper. So this result is actually a attribute of the relationship itself. Now again a paper is authored by an author and there is a contact author. So there is exactly one contact author and it is authored by one or more authors and so on. And both author and members or persons, why do we need this persons? Because we are having attributes of a person separately that is a person should have a pan number and address and telephone number, email and so on.

So, all of those attributes of a person are inherited by both members and authors. Similarly all of the attributes of or a combination of the attributes of conference and journals is inherited by comes to event and all attributes of members are inherited by reviewers and PC chairs. So a member should have certain privileges or benefits or whatever, all of those are inherited by both reviewers and the PC chair. And because PC chair is a separate member, a PC chair may also have some attributes which are not shared by reviewer or which may not exist for a particular reviewer and so on.

So what we saw today is we have taken a fairly complicated example, I mean it's not a and this is a realistic example a conference management system in fact you can search the internet for something called confman which is a freely available I guess open source conference management system which uses a back end database management system in order to manage activities like this or you might going to MSRCMT which is the Microsoft research conference management tool which is actually used by major conferences around the world and which also has something like this. That is there are reviewers, there are authors, there are papers, there are PC chairs, there are committees and so on and so forth.

And there are little bit or rather significantly more complicated than this but the level of complication to which or the level of detail to which we have seen in this is fairly representative enough because we have seen some of the major kinds of conceptual requirements that arrive. For example a PC chair is a member of the program committee

but may not be a reviewer, but may not be any other reviewer and contact author should be part of the author list and so on and so forth.

So, all of this form tricky details which manifest themselves during your conceptual design. So what will do in the next class is to take this idea forward and take up individual chunks or pieces of this ER diagram and try to convert them to the lower relational schema and see what kinds of tricky situations arise when we convert them to a relational schema. So let us finish this class here and see you all in the next class. So this brings us to the end of this session.