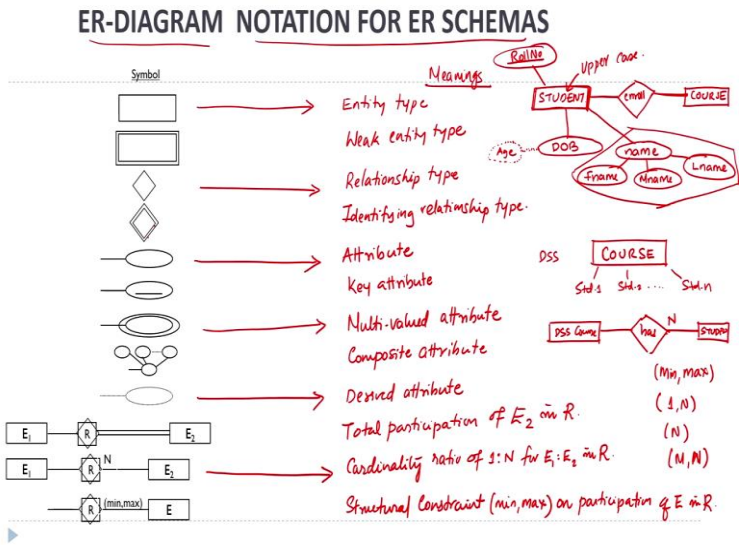
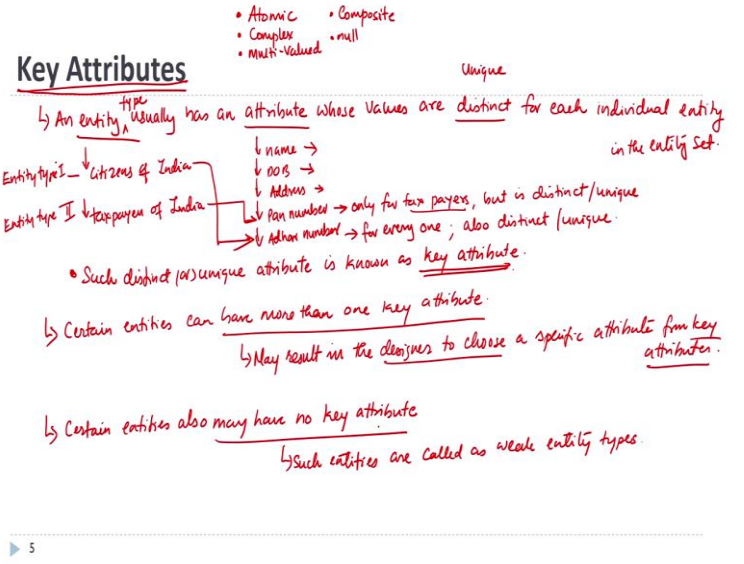
Computer Aided Decision Systems
Industrial practices using Big Analytics
Professor Deepu Philip
Professor Amandeep Singh
Department of Industrial and Management Engineering
Indian Institute of Technology, Kanpur
Lecture 13
Entity Relationship Diagrams

(Refer Slide Time: 00:19)





Welcome back to the lecture on ER diagrams. First, let us understand the ground rules of the ER diagram and the notations for this. So, all these are the major symbols of the ER diagram. And let us talk about what are the meanings of these symbols. So, the first one, this rectangular box. What does this mean? This denotes the Entity type.

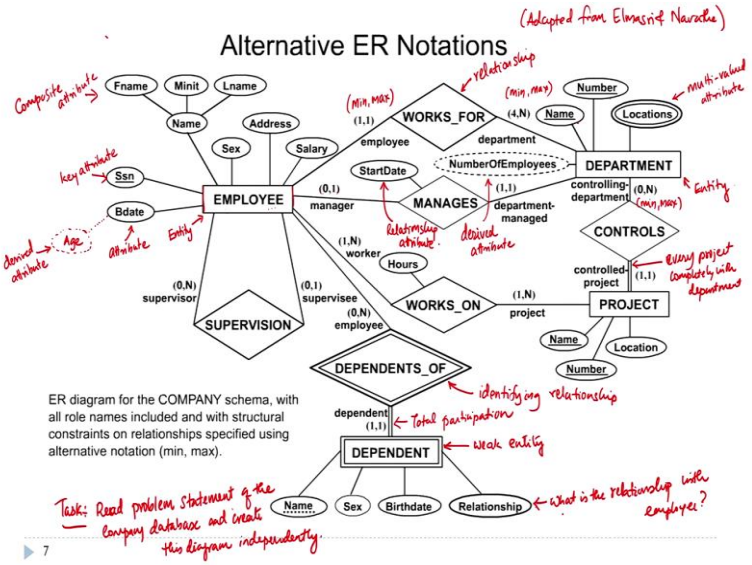
1) If you are trying to write a STUDENT then we write something like this, that means in the diagram this is an Entity type, and then it is the name of the entity is written here in uppercase or capital letters. So, that is the Entity type.
2) The one with the 2, that is the 2 concentrate rectangles. We can call it as the Weak Entity type. Remember, what is Weak Entity type, we already discussed, entities that does not have a Key Attribute, so it is a Weak Entity type.
3) Then, this one, the rhombus or you want to call to the diamond, that is the Relationship type. So, for example, you have STUDENT and then there is another one what we call it as COURSE. So, if you make a relationship between them you can do it like this and we put Enroll right here. So, the students enroll in courses. And this Enroll is the relationship. So, the diamond or the rhombus is the one that actually gives you the Relationship type. And the one with the two concentric one that is the Identifying Relationship type. So, if you have a Weak Entity, so then you need Weak Entity to be identified with the strong entity that has its own so that type of relationship of a Weak Entity to another one is that one that identifies the Weak Entity so, that is why it is called as the Identifying Relationship type.
4) Then comes this oval with the connector with it, that one is typically used to denote what you call as an Attribute. So, oval with the conductor is an Attribute. Then the oval with an Attribute, so if I write STUDENT, if I do something like this Date of Birth (DOB) then, this is the Entity, this is the Attribute of the student. So, if I have an Attribute and I underline that Attribute as the part of it, then that is known as the Key Attribute. So, if I do something like this, roll number and underline this, then that means that is a Key Attribute, both are attributes but roll number is a Key Attribute because it has an underline.
5) Then, where there are two attributes but it does a circle around it or concentric (not circle, oval around it). We call that as the Multi-Valued Attribute. Then you have a nested like an oval with multiple ovals beneath it. So, if I draw something like this, name and I have first name, middle name, last name, if I do it this way. So, entire collection of this, that one is what you can call it as a Composite Attribute. So, now we know how to draw a Composite Attribute.
6) Then comes the one with the dotted lines, that is the Derived Attribute. So, you are using an Attribute from where you are actually deriving it. So, if I do something like this. Let us make it oval like this and I write age. So, age is a Derived Attribute from the date of birth. So, that is the Derived Attribute. So, the major aspects of the Entity and Attributes are completed as part of this.

Then the second one is the Type of Relationship. So, I just drew two lines connecting the students enroll in the course.

1) So, one way of looking at this, the first relationship E1, E2 with an R, where there is two lines as part of it, you can call this as Total Participation of E2 in R. So, in a way to think about it is that, if you do that every student will enroll in a course. So, if I draw a two line here, that means every course is applicable with all students, which is not correct. So, what I am trying to say is that in this relationship or relation, E2 is totally participating in the relationship with respect to E1, that is what the first one says.
2) The second one basically says, Cardinality ratio of 1 is to N for E1 to E2 in R. So, what does this mean is, R is a Relationship. So, one way to think of, I will tell you in a simplest way, let us take a COURSE, so let us take DSS course as an example, you have student 1, student 2, etc., student N. So, the DSS course for every DSS instance course, all the N students participate. So, if I draw like this, if I take an example of DSS course has students. So, this relationship I can say it like this N. So, for one DSS course all the N students participate in that. So, this participation is also known as the Cardinality ratio. Another way to write this 'min max', where we can say is the Structural Constraint, which is the 'min, max' on participation of E is in R.

So, if there are two other courses of DSS or the three different courses of DSS being offered, different students in that, so, then instead of writing this 1 to N we will write something like 'min and max'. So, this 'N, I can also write it as '1,N' or I can just write it as N also, but if I write it as 'M,N,' then that means there are M instances of DSS courses in which N students will participate, which says that the 'min max' criteria is applicable to that particular relationship.

(Refer Slide Time: 09:25)



Now, here is an example of that employee database that we talked about. And this is again adapted from 'Elmasri and Navathe'. So, let us take a quick look on what happens it is. So, we told there are EMPLOYEES, there are DEPARTMENTS and then there is PROJECTS, and then DEPENDENTS. So, if you look into this as an EMPLOYEE there is something called as a Name. So, there is a first name, middle name and last name.

So, this whole thing if you look into the previous diagram, this is a Composite Attribute. So, because of the design this is an attribute (Birthdate). SSN (Social Security Number) this is a Key Attribute, as you can see it is underlined here. Then, if I had Age coming out of it, let us say Age that would have been a Derived Attribute. And there is Sex, Address, Salary, etcetera.

So, an EMPLOYEE, then there is another one which is called as a DEPARTMENT. So, this (EMPLOYEE) is an entity, DEPARTMENT is also an entity. So, an employee is related to department with the relation WORKS_FOR, you can see this. So, in this case, what happens is an EMPLOYEE, there is a 'one to one', this is the 'min max' constraints. So, it is like just to demonstrate, what it is? So, that means an employee will work for a minimum of one department and maximum of one department.

What does that mean? So, the employee can only work for one and only one department, that is it. So, that is what the 'min max' constraint says ‘One to One relationship’. So, an employee will work for one and only one department. Whereas, the other way if you look into it, there is a 'min max' here which is (4,N), which means minimum constraint of the employee for a is 4,

so we have 4 then this one will be called as a DEPARTMENT, can have N EMPLOYEES, its upper limit is an enlarged number.

And there are 2 aspects of the department.

i) You can see that the Name of the department and Number, these both are Key Attributes. So, more than one Key Attribute, name is also unique, number of the department number is also unique.
ii) And location is actually a Multi-Valued Attribute. So, same departments can have multiple locations. So, the Number of EMPLOYEES that you see here, I would say Derived Attribute.

And then what happens is, there is another relationship, you are actually looking into this without knowing the statements. So, I am just kind of show you major important thing. So, you can take a look into it. Now, another part is the DEPARTMENT Controls PROJECT. So, this is another entity, but you can see that, this participating relationship is every project, completely participates in Department or total participation, that is what we remember.

So, project is totally participating in the department with the relation. The department may or may not have Projects. So, it says '0 to N' means this is the 'min max', that means a DEPARTMENT may have 0 PROJECTS (may not have a PROJECT), but there is a PROJECT it has to be associated with the DEPARTMENT, '1 and 1' in that regard. The PROJECT also has Name, Number and Location different in that regard.

And then another thing about this DEPARTMENT is managed by an EMPLOYEE. So, an EMPLOYEE manages a DEPARTMENT, you can look in this way also. So, this relationship is the Manager Relationship. So, it says 0, which means an employee may not manage any department as 0, and at the most can manage one, that is the 'min max' '0, 1'.

And the department is managed by one and only one employee, that is why the 'min max' is '1 and 1'. And you can see there is an Attribute on this Relationship, which is a start date. This is a Relationship Attribute. Same way just like us an entity has an Attribute, a Relationship can also have an Attribute as part of it. So, that is another aspect. Then the another aspect you need to look about it is, this is the Identifying Relationship.

And DEPENDENTS is what we call as a Weak Entity. Why it is called as a Weak Entity? Because the Dependent will not have any existence unless there is an employee in the company.

So, Dependent is there somebody who is a wife or a kid or something of the employee. So, unless the person is employed in the company then that person is Dependent, if the employee not in the company then there is no existence.

So, the existence of the Dependent is consistent or contingent upon the employee, so that is why it is a Weak Entity. So, you can say that there is husband and wife, so the Relationship is the, what is the relationship with employee? It is husband, wife, child, that is what it is. And then here again you can see that the Dependent is 'one to one'. So it is a Total Participation (complete participation in this or what we call as Total Participation).

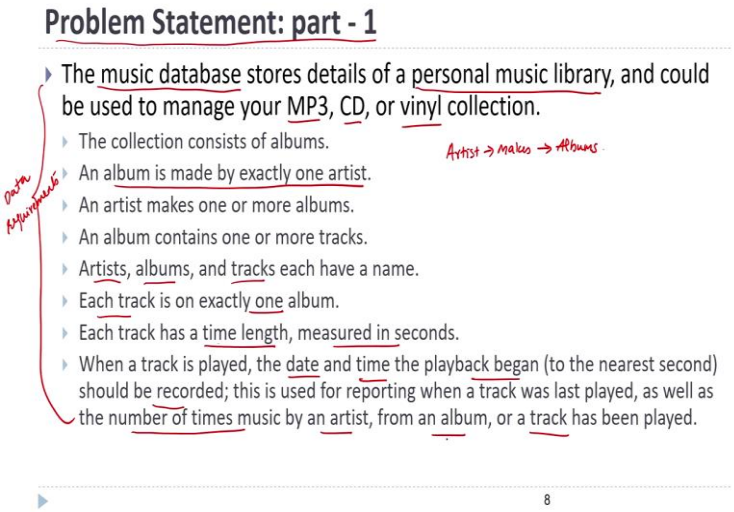
So, it is very clear or you can look into all this kind of Schema and understand that you can have an entity which is by rectangular box and entities can have type of attributes and entities are related to each other. So this is a relationship or relation whatever you want to call it and the relationships can also have an attribute and the derived attributes, the key attributes, multi-valued attributes, all those aspect are basically shown to you as part of this example.

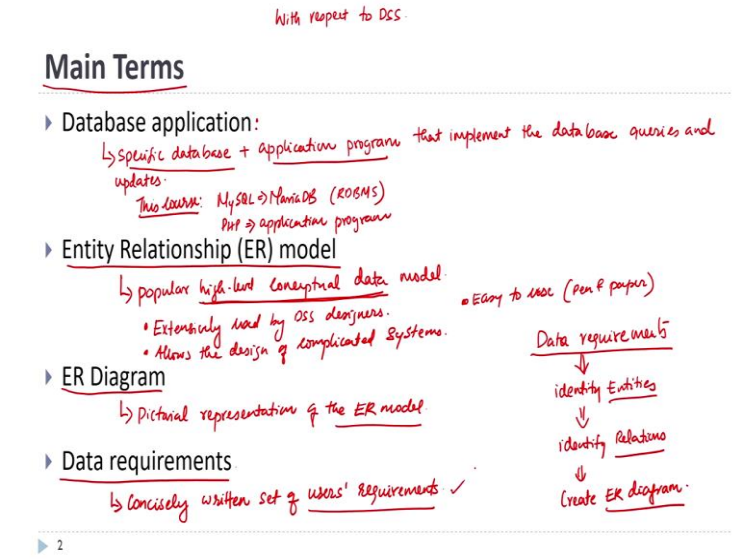
So, now, let us try to see how to create this diagram. So, I have not given you the problem statement. So, one of the task, I am going to give you

- Read problem statement of the company database and create this diagram independently.

Without looking into what the statement problems said, without looking into this diagram, try to create it by yourself and see how much you have come across, what you are missed, what you are not missed. So, you will have a clearer idea. So, but in an easy way that you can see that with a good ER diagram, all aspects of a particular problem can be very well captured.

(Refer Slide Time: 18:04)





So, now, let us move into what we call as the Problem Statement. It is an example where we are trying to create a sample database or a sample ER diagram of this. We are going to work on this one. So, what we are going to do here is a music database, very simple music database. It stores the details of personal music library. So, all these aspects that you see here, we are already discussed.

So, that is something I will take you back. You remember Data Requirements (a concisely written set of user's requirements). So, this kind of a statement, what you see, this whole thing is what we call as Data Requirements. These are concisely written user requirements. So, it is a personal music library, which can be used to manage your mp3 collection or CD collection or vinyl or a big disc collection, whatever it is. So, that is the idea here.

Let us create a simply design a simple personal music database. So, the first thing is:

- The collection consists of albums, that is one statement. Your music collection is consisting of albums.
- An album is made by exactly one artist; this is a condition. So, we can think about it as an artist, makes, albums. So, an album is exactly made by one artist, so it is given to you.
- An artist can make one or more albums. So, while an album can only be made by one artist, an artist can make more than one albums.
- An album contains one or more tracks. So, minimum number of tracks required for an album is 1 and it can have N tracks. So, you can see how the cardinality and all those kinds of things comes out of it.
- Artists, albums and tracks each have a name.
- Each track is on exactly one album, the track is always there in one album, it cannot be there in multiple albums.
- Each track has a time length, measured in seconds. So, that is one attribute of the track, track has a length of the play length that is measured in seconds.
- When a track is played, the date and time the playback began to the nearest second. So, when you play the track, the date and time the playback began to the nearest second should be recorded. So, you should capture when the date and time when it the track was played to the nearest second, so that means hours, minutes, seconds, which is used to report when the track was last played. So, if you want to figure out when it was last played, then you need to capture this information. As well as the number of times music by an artist from an album or a track has been played. How many times you are listening to a particular artist and particular track in an album is also need to be captured as part of it.

(Refer Slide Time: 21:17)

Problem Statement: part - 2

data requirements

- There's no requirement to capture composers, group members or sidemen, recording date or location, the source media, or any other details of artists, albums, or tracks.
- Because this database is for a personal collection, it's relatively simple and stores only the relationships between artists, albums, and tracks. It ignores the requirements of many music genres, making it most useful for storing popular music.

Problem Statement: part - 1

data requirements

- The music database stores details of a personal music library, and could be used to manage your MP3, CD, or vinyl collection.
  - The collection consists of albums.
  - An album is made by exactly one artist.
  - An artist makes one or more albums.
  - An album contains one or more tracks.
  - Artists, albums, and tracks each have a name.
  - Each track is on exactly one album.
  - Each track has a time length, measured in seconds.
  - When a track is played, the date and time the playback began (to the nearest second) should be recorded; this is used for reporting when a track was last played, as well as the number of times music by an artist, from an album, or a track has been played.

Artist → makes → Albums

Entities
(1) ALBUM
(2) ARTIST
(3) TRACK
(4) PLAYBACK / PLAYED

Entity

academic program : B.Tech Program | Civil Engineering / Chemical Stream } Dual degree
M.Tech | Computer Science

- Basic object of the ER model
  - ↳ usually it is a "thing" in the real world that has an independent existence.
    - ↳ recognizable object / concept / item, etc.

  Eg: Student of IIT Kanpur (🚶)
- Each entity has attributes
  - ↳ particular properties that describes the entity.

  Eg: Student attributes → Roll no (1234), first name (Ram), middle name (Kumar), last name (Krishna), date of birth (1-1-1985), address, academic program, etc..
- There are many types of attributes.
  - ↳ Simple attribute (atomic) vs. Composite attribute (sub-divisible)
  - ↳ Single valued (single for a particular entity) vs. Multi-valued
  - ↳ Stored vs derived

  Composite → name: Ram Kumar Krishna (Ram Kumar Krishna Fname. Mname. Lname.) Atomic

  Date of Birth : 01-01-1980 ← Stored attribute
  Age : (as of 01-01-2023) : 40 years ← derived attribute

Now, second part of the Problem Statement is continuation. This is again what do we call as Data Requirements, a concisely written user statement. And this says basically:

- There is no requirement to capture composers, group members or sidemen, recording date or location. So, you do not want to capture who is the composer, you do not want to capture who is the group member, who was the sidemen or chorus when it was recorded, and the location of recording, the source media or any other details of artists, albums or tracks. So, this information you do not want to store, ignore, so these information is available, but do not store. So, it is also important to note what to store and it is also important to note what not to store as well. Now, why is it this written here?
- Because this database is for personal collection, it is relatively simple and stores only relationship between artists, albums and tracks. So, the three things we are interested in part of the artists, albums and tracks. It ignores the requirements of many music genres. So, whether it is classical, pop, rock, all those kind of things, not matters. So, at the point is it is basically just the main focus is storing the popular music of a particular individual. So, that is what is called Problem Statement. So if you take a look into it, you can see that we are already discussed, there is an artist and an album, then there is a track and we also need to keep track of when the thing was played.

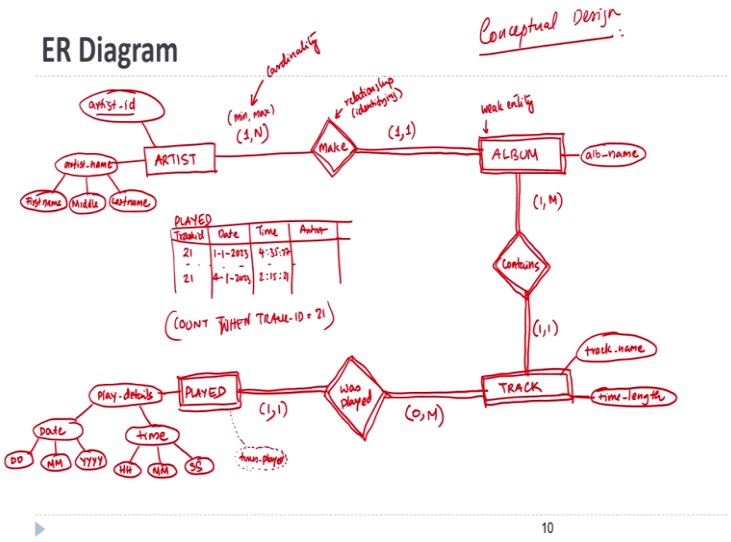
So, in a broad sense, the first thing I will do is I will try to identify the entities, which are the major entities as part of this. So, from the right up, we say that an album is made by, so a collection it consists of albums.

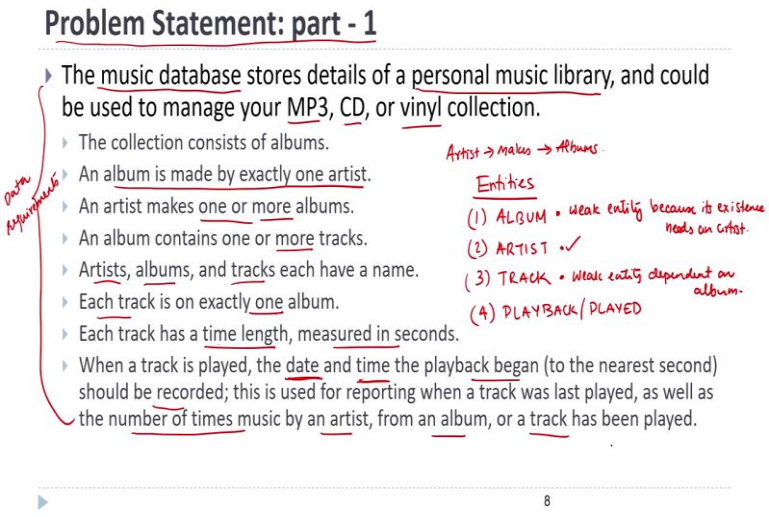
1) ALBUM is an entity. An album is made by exactly one artist an artist can make more than one album. So, there is another entity, so album is a physical entity that is in a CD disc or mp3 or vinyl collection, whatever it is.
2) An ARTIST (an individual, a human being, he or she) is also an entity. So, remember I told earlier also what is an entity, it is ‘a thing’ in the real world that has an independent existence. So, album is a thing, artist is a human being or has an independent existence. And then we said an album contains one or more tracks and artist, album and track have a Name.
3) TRACK, this is the third entity that comes out of it. And each track is exactly in one album. So, track, album, etcetera, is the relationship as part of it. And each track has a time length,

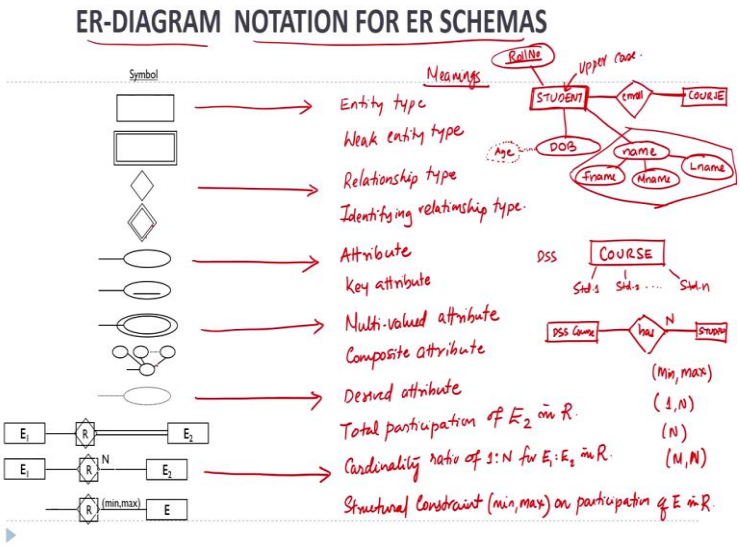
measured in seconds. So, when the track is played, the date and time on the playback began need to be recorded. So, you have to keep track of that.

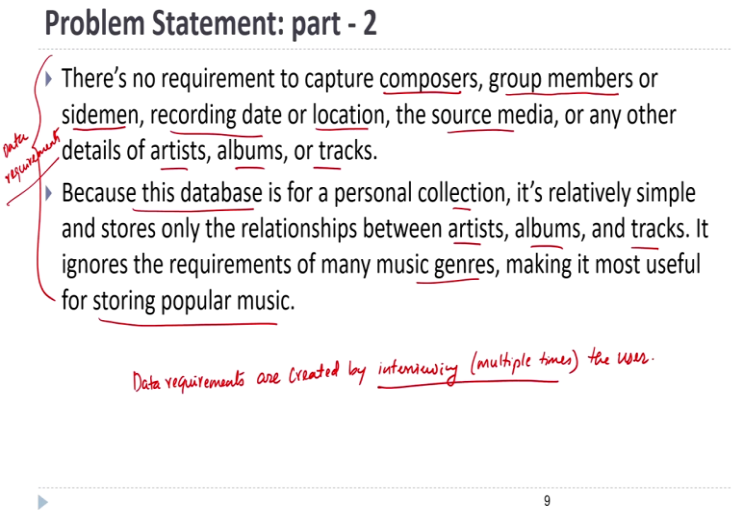
4) So that gives you the fourth entity, this is pretty much saying that played or PLAYBACK or PLAYED, you can use whichever one you want to do, when the track was played nearest to the second you need to record that. So, from this description, I can identify four entities as part of it. So, once you identify that, then we can make the ER diagram.

(Refer Slide Time: 24:52)









So, let us start with the entity that we identified first. So, there is an album and an artist. Usually, when I start it is easy for me to start with the human, so in this we will start with the artist.

- So, artist is an entity, so I am going to draw a rectangle and I will write it as ARTIST. So, that is an entity, artist is an entity as we keep track of it. And album, artist and tracks have a name. So, in this case, what we will do is, we will have an attribute called, you can write it as Artist Name. And it is not given to you whether the first name, last name, middle name is there, but you can assume that, that is true. So, you can make it something like first name, middle, last name, you can do it that way. So, now you have what is this one, you remember this, this is a Composite Attribute, you can see this part. So, now we have created an attribute which has a Composite Attribute. Then if you look into this album, artists and tracks have a name. Each track has exactly one album.

Track has a length and etcetera, there is nothing more mentioned about the details of the artist it is just there. So, one of the ways to do this is we can assume that there is an attribute, for each artist there is a sequential number or ID so you can create something called as an Artist ID and then say that that is an attribute and underline it, basically saying that each artist will have exactly unique ID, that is Key Attribute as an Artist ID.

- Now, the artist what do they do, we have seen in the relationship that artists makes one or more albums, an album is made by exactly one artist. So, now I will take the next entity, I will kind of draw it right here, which is the ALBUM. So, we have seen previously, that an artist makes one or more albums so I can use that and identify the relationship of Make, an artist makes an album. So, artists can make one or more albums, which is we are basically told about this. So, you can take it either way. The artist name will be there, minimum is 1 and can make more than 1. So, the relationship cardinality here is '1, N'. So this is the 'min max' we talked about. Artists can make minimum of one, maximum of N, in this regard. Now, if you look into the previous one, it says an album is made by exactly one artist, so that means every album has a total participation in the make relationship because exactly one artist makes it, so we do it like this. So, that means it is an identifying relationship, for album it is an identifying relationship. And also, other part is, let us capture the aspects of album. An artist, album and track have a name. Let us call it as the album name for part of this, Album Name. If you want to do you can make it as a composite, you do not want to make it as a composite, that is up to you, that does not matter. But album does not have an ID.

The only thing is album is exactly made by one artist. So, without an artist album does not have an existence. So, that gives you the idea that album is a Weak Entity. So, we write here album weak entity because its existence needs an artist, it does not have a unique identification, standing of its own. So, then that means this entire relationship with the album, or with an artist is a weak entity and it is an identifying relationship.

- Now, what is the 'min max' for this, an album requires a minimum of 1 artist and the maximum is also 1 because we know that no 2 people, so it is very clearly written here album is made by exactly one artist, so the 'min max' is '1, 1'. So, the cardinality is, so we can think about this as a Weak Entity. And, as we said this is a relationship, but it is also an identifying relationship, we mentioned that very clearly. And this is what we call as the Cardinality of the participating relationship.

- So, now there is a third entity that we want to go back is the track, now let us take the TRACK. So, the track is associated with albums. An album contains one or more tracks. So, now let us take what is the track. So, I am going to just put it here because it is easy to draw that way, TRACK.

So, now the question is the relationship between album and track is an album contains one or more tracks, so let us use that as a relationship (album contains track). Now, album is a Weak Entity. So, because the album is a Weak Entity, without an album, the existence of the track is also not there. So, track definitely will also be a Weak Entity. But also remember each track has a time length, measured in seconds, and each track also have a name. Artist, album and track each have a name.

So, let us use one entity, one attribute called as Track Name. And then, there is also another one it says each track has a time length, measured in seconds. So, we call the next one as Time Length, that is measured in seconds. Each track is exactly on one album. So, we know that album is a Weak Entity. Track cannot have an existence without the album, so track is also a Weak Entity. And it is a total participation in the relationship. It is an identifying relationship.

- Now, the thing is, we see each track is on exactly one album, so we can write the cardinality of it is like an album is made by exactly one artist same thing, '1, 1'. And now that it is a Weak Entity, because an album dependent on an artist, album can contain one or more tracks. So, this cardinality here is '1, M' we use N there let us use M for the time being, and this relationship is also totally participating because without a track the album will also not happen, so that is the other aspect of it.
- Now, what happens is track is also taken, which is also a Weak Entity dependent on album. Now, let us take the last entity which is the PLAYBACK. So, now the playback or played entity is taken and I am going to take the PLAYED one. I am just using played or you want to call it playback you can use playback. And the important information here is when a track is played, the date and time or the playback begins. So, this is to be stored.

So, one way to think about it is, how to store the Play Details. So, the play detail is, what does it stored, it says when a track is played the date and the time of the playback began to the nearest second should be recorded. So, you need to record the date and time. So, fine, we can do that, we have one which is the Date and there is one which is the Time. So, this is a Composite Attribute.

Now, it is also said that the time of the playback should be stored to the nearest second. So, I am going to assume at this point that the date we are going to store it as DD-MM-YYYY, so the date of the playback is stored in that way and time is stored as HH (hour), MM (minutes) and SS (seconds). So, this is the play details of the track or when the track was played. Now, the question is, the relationship between track and this one was, you can think about the relationship between the mass was played.

So, the track was played, some point of time. So, now the track is a Weak Entity that means the played will only happen if the track exists. So, played is also a Weak Entity. Now, since it is played as a Weak Entity, this relationship is also an identifying relationship because played has no existence without a track and track has no existence without an album and album has no existence without an artist.

When a track is played, the date and time the playback began it should be recorded and it is used for reporting when the track was last played. And then as well as the number of times the music by an artist from an album or a track has been played. So, now you also need to keep track of how many times album was played. So, you have something like Times Played, how many times you have played this one that also need to be kept track of others. So, the minimum number of times a track can be played is actually 0, you do not need to play, actually track can be played 0, there are certain tracks that may not be played. And let us use the maximum number of times, a track can be played more than once. So, it can be played more than once. But when you say the Played, this data will only happen when the track is played. So, the minimum is once. So, that is what happens. And every time you play it, the Played will keep on recording and updating this. So, the maximum cardinality in this is '1, 1'. So, the track you can play minimum of 0 maximum of N, but the played data, whenever the track is played, you need to keep track of the date and time aspects of how many times a track was played. Makes sense? So, when this happens, then you can say that this is not a good idea.

So, you can move that to the Times Played right here, we can do it that way also. So, how many times the track was played that can be kept track that way. Another way to do it is, you do not have to do it also, you can basically say that, this can be a Derived Attribute Times Played. This also is possible, because in the database, so if you think about it, the one way to understand this complexity is this way.

- Let us call it as the, I am just going to say Track ID, date, time, then artists, I do not know I just have other info's, like etcetera. So, track id of let us say 21, 1-1-2023,

4:35:27, then you have some other entries here then there is 21, 4-1-2023, 2:15:21 like this. So, then by counting from the table, how many times the Times Played can also be derived, so this is the played entity, which appears as a table in the database.

- So, by counting the number of times the track is played, when track id is equal to 21. So, this will give you how many times a track has played So, that also is another way, you can think about as a derived attribute also. So, these are multiple ways you can think about the concept being added into the database. So, this information gives you an overall idea of how to do a ER diagram or use ER diagram to design a database from the user stuff. So, remember, these data requirements are typically given to you in writing, or you do an interview. So, how do you create data requirements?
- Data requirements are created by interviewing (multiple times) the user or the data generator. So, you keep on interviewing, you meet them, discussions, etcetera. From there you come up with these kinds of concise statements. From there you identify the entities. From the entities, you identify the relationships, and then use the constructs of the ER diagram to create the ER, the database, the Conceptual Design, so this whole thing is the conceptual design of your music database.

So, I hope this clear the main aspects of the database design to you. We will continue the other aspects of the normalization, other aspects of the database along with how do you do SQL and etcetera, how to translate this conceptual design into tables or actual database. All of those we will discuss in the coming class. Thank you very much.