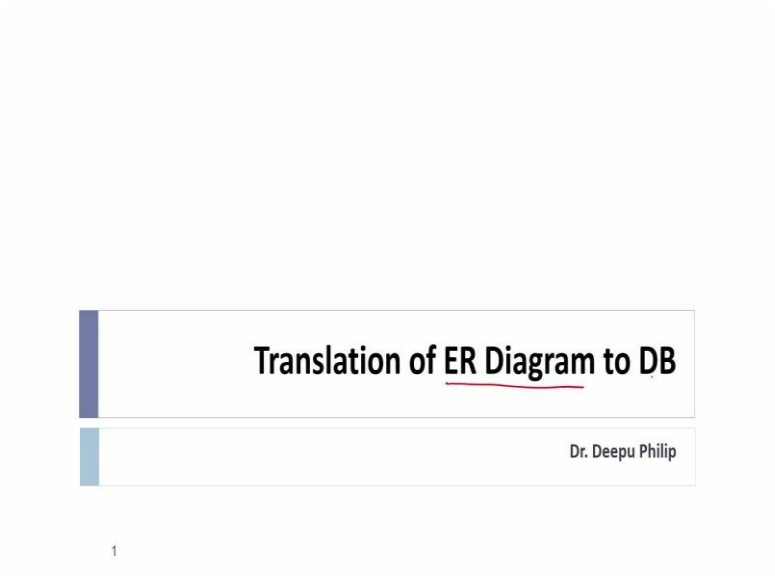
Computer Aided Decision System- Industrial Practices using Big Analytics

Professor Deepu Philip
Department of Industrial & Management Engineering
Indian Institute of Technology, Kanpur
Professor Amandeep Singh
Imagineering Laboratory
Indian Institute of Technology, Kanpur

Lecture - 14

Transition of ER Diagram to DB

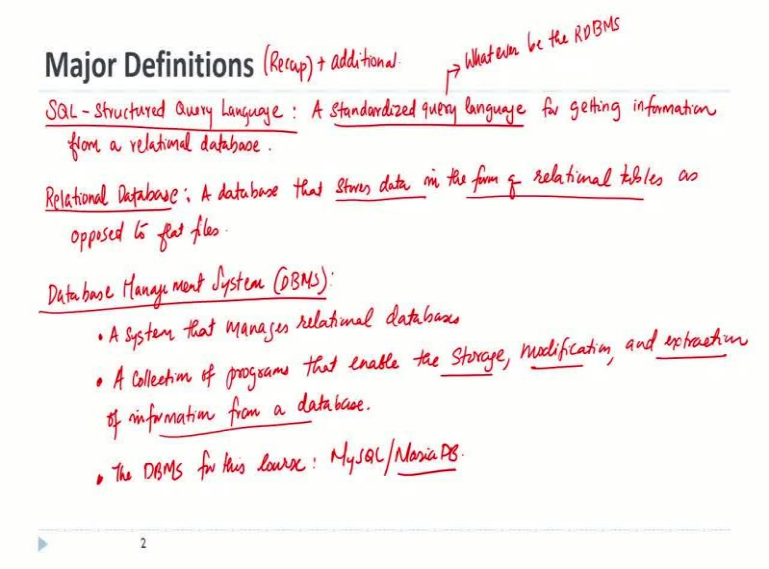
(Refer Slide Time: 00:14)



Good afternoon, everyone. Welcome to at another lecture of the Web Based Decision Support System course, which is intended for both business decision makers, practitioners and academicians. And we have been going through major aspects of it. And, we have assigned quick overview. We have studied that the DSS has four major components. And, we are now going to in slightly in depth into one other major component, the database management system DBMS, and proceeding with that.

So, we have already seen major aspects of it, which includes what is the ER diagram? What is an entity? What is the relationship? How to translate from user specification to ER diagram, etcetera? So, without further delay, we will get into today is topic, which is the 'Translation of ER diagram (Entity Relationship diagram) to DB database'.

(Refer Slide Time: 1:10)



So, let us first start with the Major Definitions. So, this is what they call a (Recap) plus additional. So, we have yesterday discussed something called as:

1) SQL (Structured Query Language): We had seen what it is, actually we talked about what is a query language and high-level stuff, etc. So, what we are telling today by definition:

❖ A standardized query language for getting information from a relational database.

So, it is a standardized query language, SQL is a standardized query language. But the idea is, whatever be the Relational Database Management System (RDBMS), whether it is MySQL, Oracle, Microsoft SQL Server it does not matter. It can get information from the relational database. So, that is the first thing.

2) Relational Database: Let us recap this again, quickly. What is a relational database? So, we say is that,

❖ It is a database that stores data, in the form of relational tables as opposed to flat files. So, when we say a relational database, it is a database that stores data in the form of relational tables. Tables that are related to each other or having data that are related to each other as against flat files. You are not storing it like the flat files.

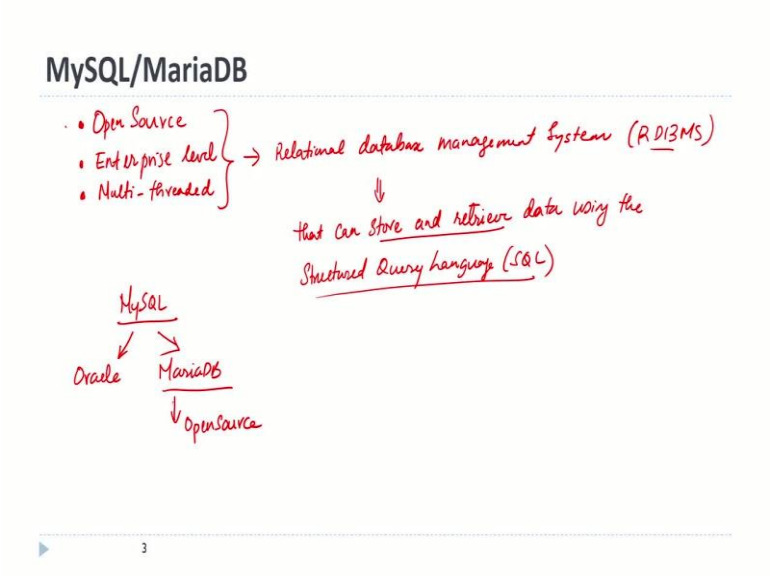
3) Database Management System (DBMS): So, one simple way to do it is,

- A system that manages relational databases.

- A collection of computer programs that enable the storage, modification and extraction of information from a database. So, it is a collection of computer programs that enable the storage, modification and extraction, the main functions of information from a database. So, that is also another definition of the DBMS.

- The DBMS for this course is MySQL, Maria DB. Both are the pretty much the same. Maria DB is an open source portal. MySQL is the paid version out of Oracle. MySQL, which is an open source thing, it is now owned by Oracle.

(Refer Slide Time: 5:48)



So, what is MySQL and Maria DB? It is the main characteristics I am going to write.

- It is an Open Source. Open Source mean it is freely available.

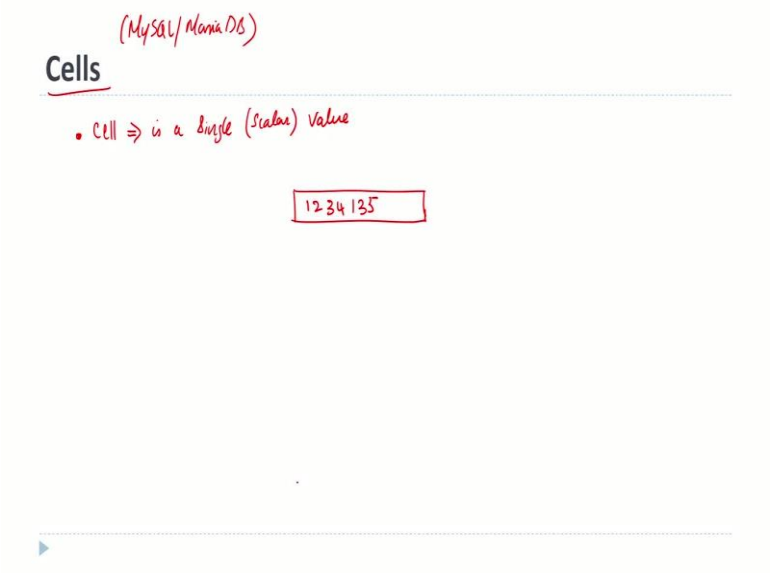
- It is an Enterprise Level. So, you can use it for banks and other critical operations, etc.

- It is a Multi-threaded, means multiple instances can be run at the same time.

These three features are of Relational Database Management System. That is an open source, enterprise level, multi-threaded, relational database management system, that can store and retrieve data using the Structured Query Language (SQL). So, it is downloaded dependent upon the SQL.

So, earlier, MySQL was only one, but then it became two. There is an Oracle version and Maria DB. The Maria DB is the open source version now. I think the Oracle version of the MySQL is now paid. I am not sure about that, but you can figure it out.

(Refer Slide Time: 7:56)

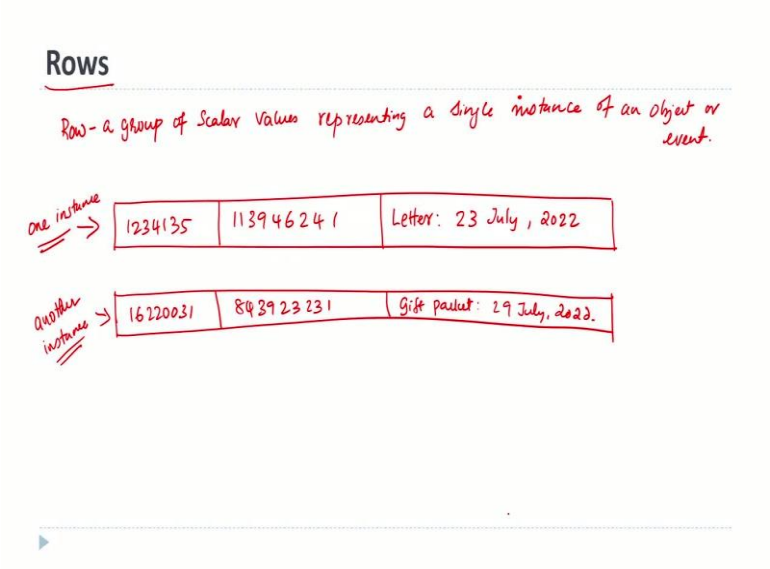


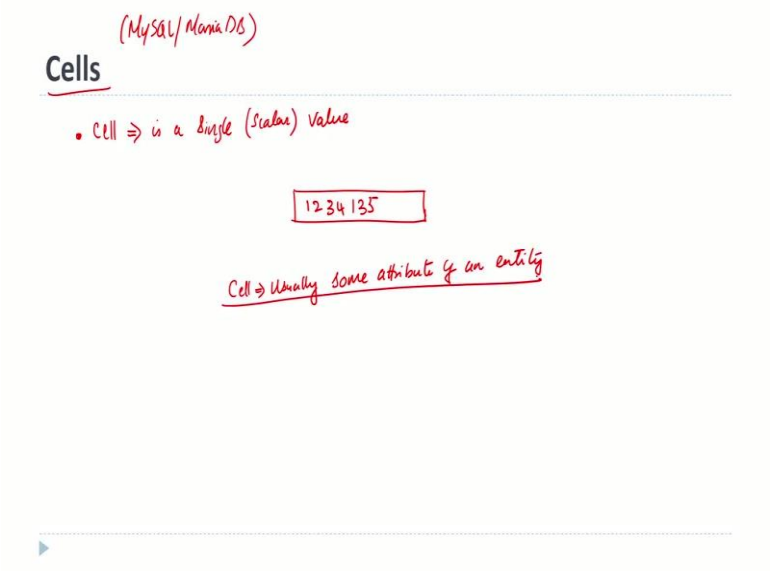
So, now in the database, the MySQL/Maria DB, the building block is the major lowest possible thing is what we call it as 'Cell'. So, how do we define a cell?

- Cell is a single or scalar value.

So, if I say 1234135, some number is being stored there. And this is one single set of value. Then we can call this as I say scalar or single cell, some people call it as a cell, some people call it a scalar value, some people call it as a single value.

(Refer Slide Time: 8:47)



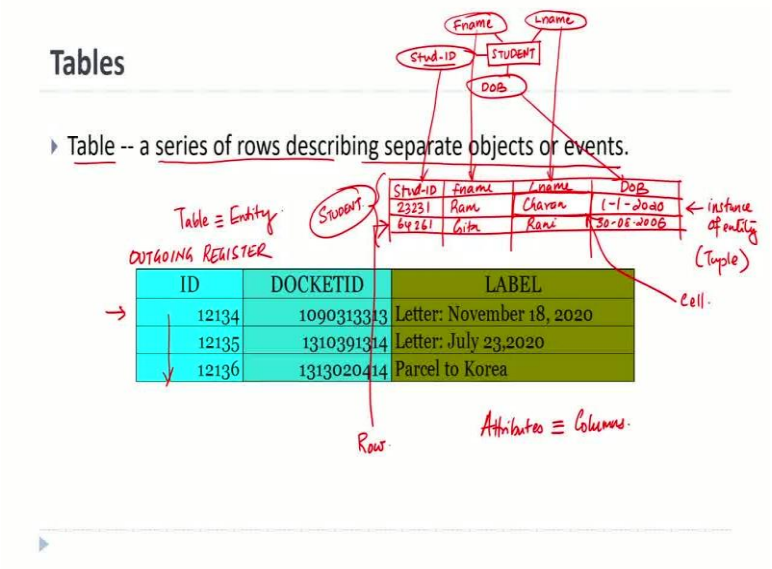


Now, the next one is, what we call a Row. So, what is a Row?

- ❖ It is a group of scalar values representing a single instance of an object or event.

It is complicated definition, but it is very easy to understand. If you think about this, cell is usually some attribute of an entity. So, some attribute of the entity usually gets stored in the form of a cell, whereas in a Row, what happens is, so an example of a Row is something like this, 1234135, and you have 113946241 or something like this. And then, there is something like a Letter: 23 July, 2022 something. So, assume that, this is something that is register, that we take care of a place. So, this is one instance. So, if this table contains another row, that may be something like this here later which is 16220031 843923231. And you can say that as a Gift Packet: 29 July, 2022, this is another instance. These are instances of an object or an event. Lot of times this could be an instance of an entity also. So, a lot of the times entities can be instances.

(Refer Slide Time: 11:15)



So, now then comes what we call as the Table. So, what is a Table?

- We define a table as a series of row describing separate objects or events.

So, let us say you have ER entity diagram, you have something called a STUDENT, and you have something like Student ID, and you have First name, Last name, Date of Birth. So, then, one of the way to think about it is, this can be translated to a table like this, where Student ID, First name, Last name, Date of Birth. So, you can say 23231, Ram, Charan, 01-01-2020 and 64261 Gita, Rani, 30-08-2006, etc.

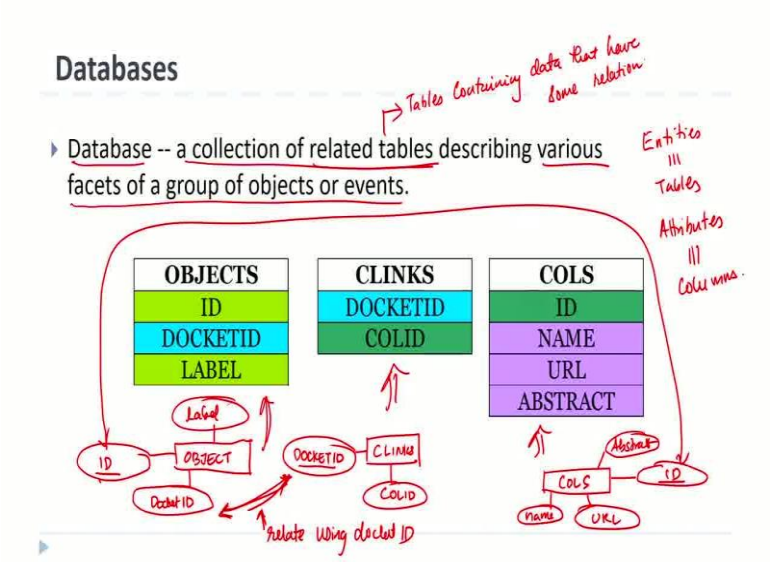
So, the entire table is a representation of student. So, it is entity typically. So, table, you can think about it as equivalent to an entity. And each row of the table is an instance of entity. Sometimes, it is also known as 'Tuple' in some ways. So, same way, what we saw previous in that is the same thing, we have an ID, where these numbers are returned 1234, 1235 like this, these are consecutive IDs. And there is some docket ID, some other ID is going on. And this docket ID is one of the Labels says Letter with a number November 18, 2020.

There is another Letter: July 23, 2020. then there is a Parcel to Korea. So, it might be a register, so this could be an entity which is like, Outgoing Register or something like this. And each one of them is an entry in the particular register. So, it is an instance of the register.

So, that is what table is, typically all about in this regard. But also remember, each one of these the student ID, you can see the first name, last name, date of birth, each one of them. So, the Attributes, usually translate to what we call as columns in a table. So, that is how the

attributes are considered most of the time. And each one of them, for example, you take this Charan, we call as a Cell. And, this is what we call as a Row. So, I hope now you guys have a mapping of how does an entity translates to what we call as a table and what are the different aspects of it.

(Refer Slide Time: 14:57)



Then, Databases, what is databases? Now, we have seen what is a Cell, we have seen what is a Row, we have seen what is a Table and we are seeing the connection between the cell the columns, the rows, entities, attributes, etc. Then, what is the Database? So, the definition is:

- Database in simply It is a collection of related tables. The critical word is ‘related tables’.

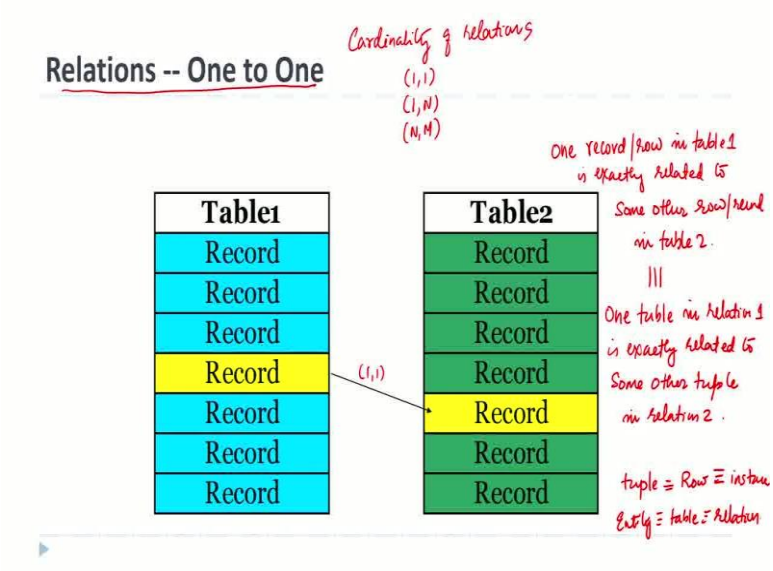
So, tables containing data that have some relation or we can identify some relation as part of this. And what does this related Tables to do? They describe various phases of a group of objects or events. So, that describe various aspects of it. So, here is an example, you have an object this is an object database or object table. This object table, which has three attributes so, you can draw this as like this OBJECT entity. You have ID, you have Docket ID and you have Label. So, this entity object gets translated to a table.

So, then there is another entity called CLINKS. Which has something called Docket ID and there is something called COLID. So, let us assume Docket ID is the unique identifier. So, then that is connected here. Now, you can see that between these two docket IDs, you can connect you can relate these two databases using Docket ID. So, now the other one COLS, is

basically whatever that entity is, you have ID as unique identifier, there is Name, URL and Abstract.

So, that gets translated to another table and this ID is related to this ID. So, you can see that things can be connected or related using another relation. So, the Entities translated to Tables, Attributes to Columns. And, this is another way to think about it. And, then relationships into the form of case, we will see in an example.

(Refer Slide Time: 18:04)



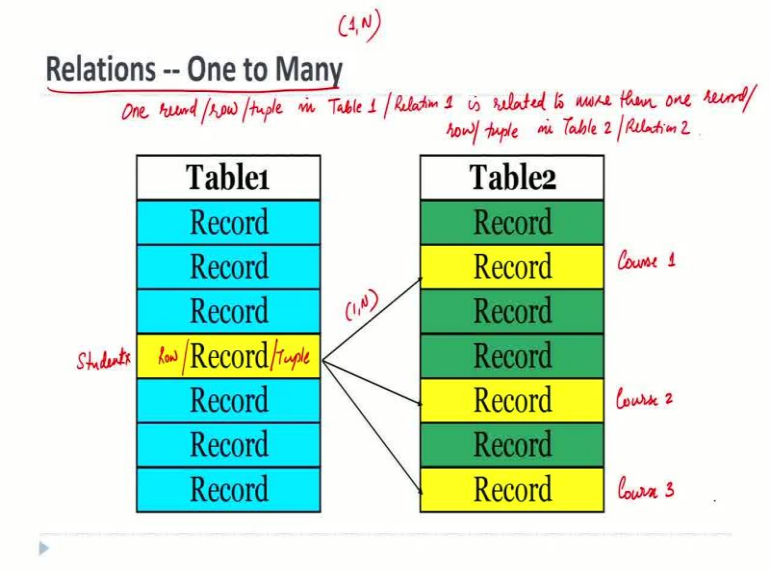
So, now let us talk about what are the major type of Relations. The first relation is called ‘One to One’. We have also seen this is also related to what we call as the ‘Cardinality of Relations’. We have seen one to one, 1, N and N, N. So, we will be trying to figure these out pretty much, all the three. So, this one what we are saying is one to one that means, one record. 1, 1 means,

- One record or row in table 1 is exactly related to some other row or record in table 2.

You can also rewrite this. Another way of saying this is:

- One tuple in relation 1 is exactly related to some other tuples in relation 2. So, when you hear the word ‘Tuple’, Tuple literally means a ‘Row’. Tuple is equivalent to a row, or an instance, which is equal to an instance of an entity. And then, entity equivalent to a Table, which is equivalent to also known as a Relation. So, these are the terms that you will see very popular in the DBMS.

(Refer Slide Time: 20:17)

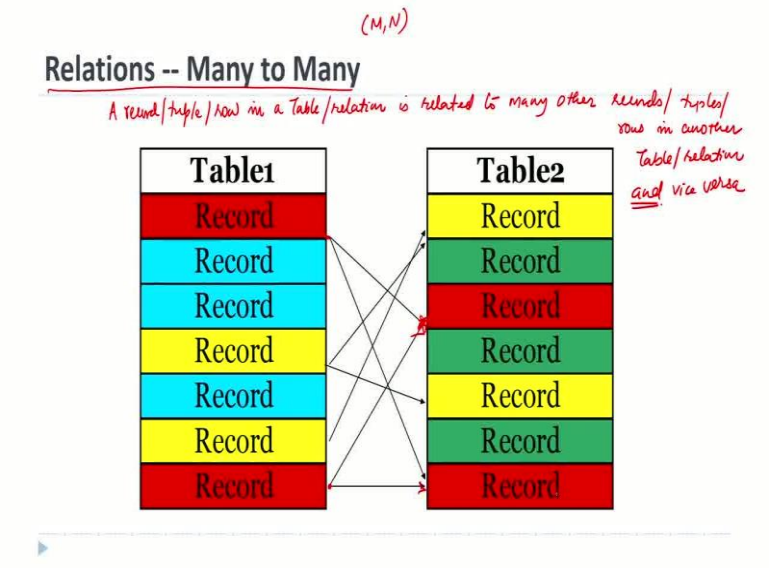


Now, comes the second one, ‘One to Many’. This is the scenario, what we call it this in the 1, N. So, how do we say this, let us say this way,

- One record/row/tuple in Table 1/Relation 1 is related to more than one record/row/tuple in Table 2 or Relation 2.

So, we will see multiple example of this but this is the 1 common scenario, One Record. this is One Record or you can think about it as 1 row or 1 tuple whatever you want to call, it is related to multiple records in another table, and somehow related. So, if you take an example of here I say, it is a Student and these are the Courses like course 1, course 2 and course 3 for the student. This particular student is Student X is taking course 1, 2 and 3. So, that student 1 is connected or related to these three courses.

(Refer Slide Time: 21:57)

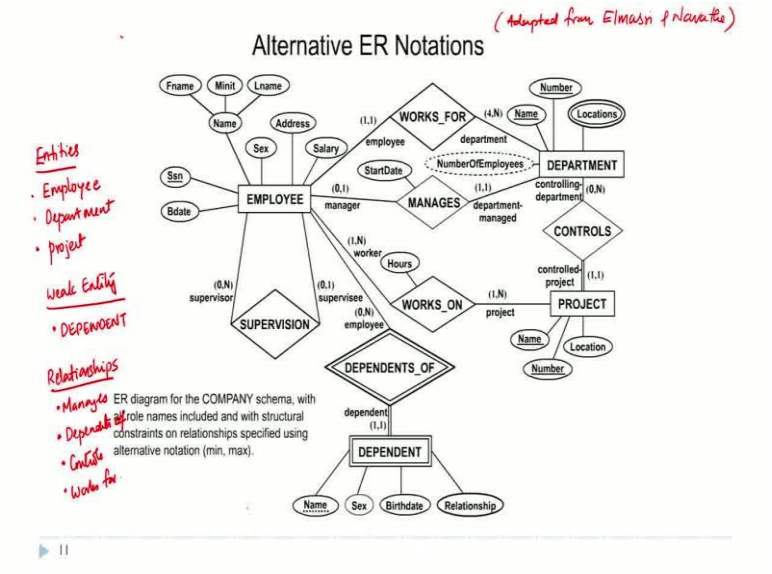


Now, let us talk to what we called as ‘Many to Many’. So, this relation is many to many, where we call as M, N in the cardinality thing. So, the one way to write about this is the following the previous argument. We can write it as,

- A record, tuple, row in a Table or Relation is related to many other records, tuples, rows in another Table or Relation and vice versa.

For example, if you take this particular record, it is connected to this record and this record. But this record is also connected to another record in this Table, we can see that there are two connections possible as part of this. So, the same record can be connected to different records in the Table, both sides. So, it is a Many to Many. So, M to N relationship. It is more complicated to manage, but we can manage that.

(Refer Slide Time: 23:35)



So, let us look in this example of the what we call the Company Database, the ER diagram. We call it as this is (Adapted from Elmasri and Navathe). So, let us see that this is the ER diagram, where we have

i) EMPLOYEE entity,

ii) DEPARTMENT entity,

iii) PROJECT entity,

iv) DEPENDENT is a Weak Entity.

And relationships were

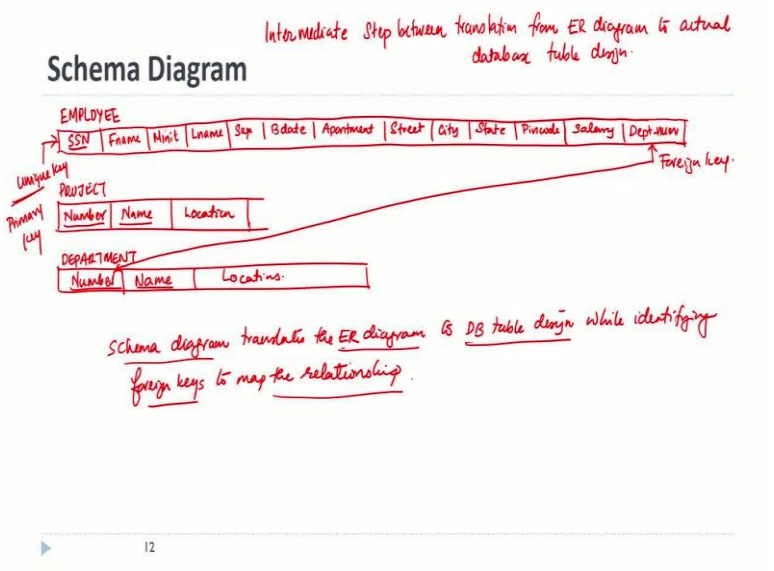
i) Manages,

ii) Dependent of Controls,

iii) Works for, etcetera.

There is so many attributes so, this was the broader idea. So, now with this our fundamental attempt at this point is to translate this into what we call as a Schema or Schema Diagram.

(Refer Slide Time: 25:16)



We already seen the previous class,

- It is an intermediate step between translation from ER diagram to actual database table design.

So, it is an intermediary step. So, one of the way to look into this, the one of the thing is Employee. So, let us take how do we draw the Schema Diagram of the Employee. So, the Schema Diagram of the Employee will most probably be something like this, just going to draw this slightly big. This is the EMPLOYEE and the Attributes of this. So, it is the SSN (Social Security Number), which is a Key Attribute is written there, then the birth date, first name, middle name, last name. So, you have First name, Middle (initial), Last name. Then, sex, birthdate. Then you have address and salary. So, we will call it as apartment, street, city, state, pin code or zip code whatever you want to call it. And then what we have is the salary.

So, this whole thing, what we just drew here is, what we call as the Schema of the Table. So, now let us do one more, let us take the Project as other one. Let us take the next entity as Project. So, Project has a name and a number and a location. So, we will draw something like this, it will be a PROJECT. PROJECT has a Number, it has a Name and the Location. So, if you look into this, then we know that the name and the number are underlined. So, that means there are Key Attributes. So, then we are also underlined the name and number, kind of it.

So, then the next one is the DEPARTMENT. DEPARTMENT has a name, number and multivalued locations. So, it is Department Number, Department Name and Locations but these locations are Multi-Valued Attributes but these are same way. So. this Representation

of the translation of an ER diagram, what we just wrote here, we call as the Schema, or basically the design of the Table, is what we call as the Schema Diagram.

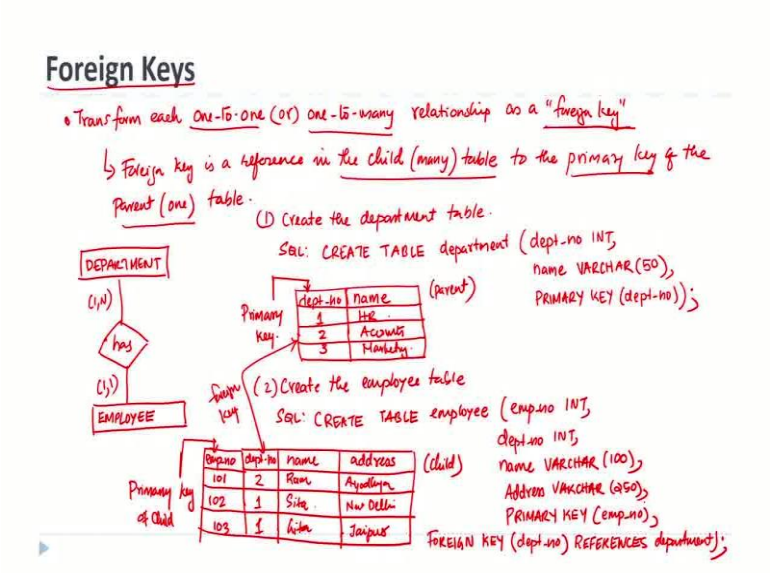
So, now there is one thing that you can think about is, there is an Employee that works for a department, whereas the Department can have multiple employees. So, one way to think about it is, there is a Department or Department as a Number and so if you think about this Employee Table. Let us call this as the Department Number. So, now what happens is, this Department Number is something, which department the employees working for. That Department Number is linked to this Department Number.

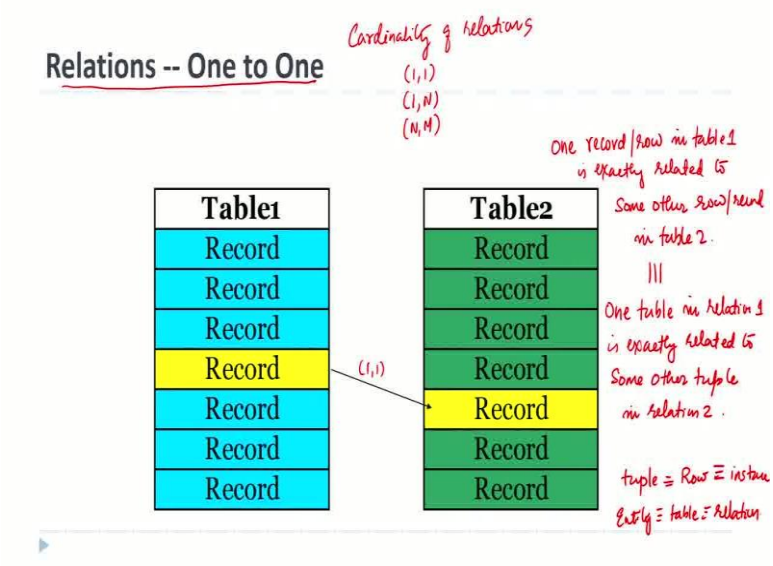
So, now you can see this Relationship what we see here, the Employee works for the Department is pretty much created by putting the Department Number in the Employee Table. So, the SSN is the Unique Key. Each Employee, each Row will have the unique value. So this kind of thing is called a Primary Key, whereas the Department for which the Employee is working for, which is a Primary Key of another table, this is sometimes called as a Foreign Key. We will explain what a Foreign Key in a minute now. But this in a Schema, it also allows you or the

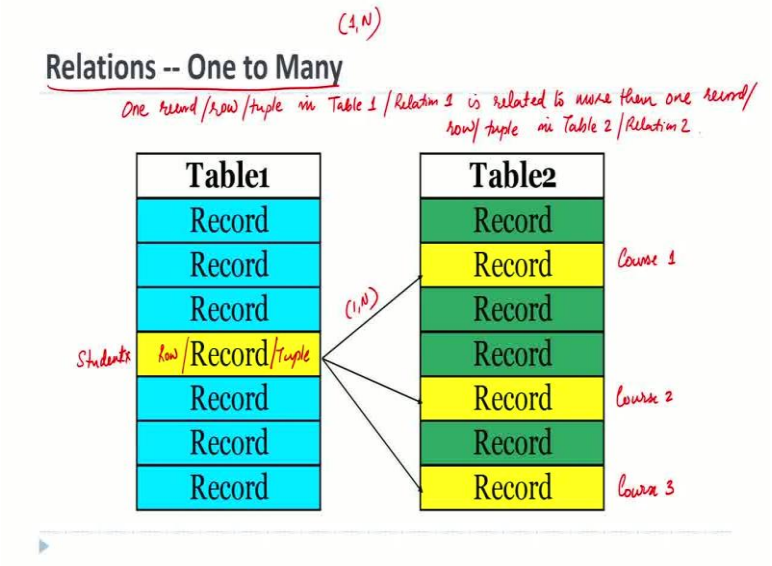
- Schema diagram translates the ER diagram to DB Table design, while identifying Foreign Keys to map the Relationship.

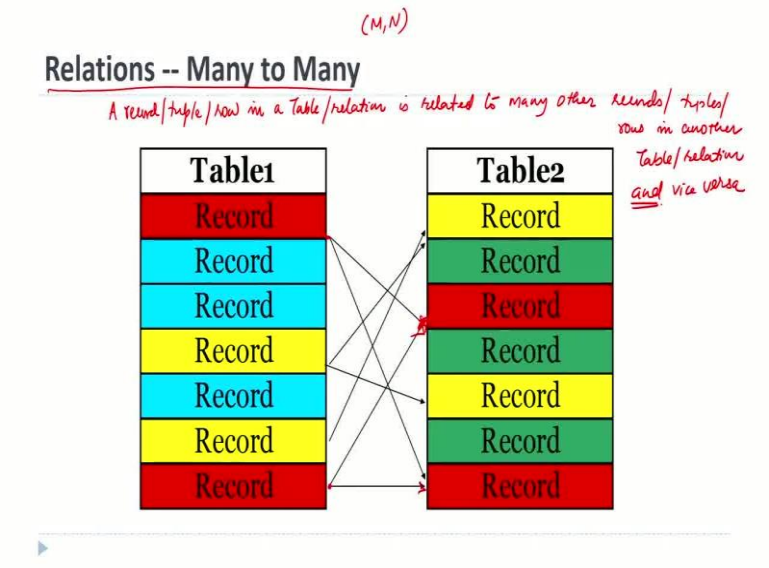
The reason why we use a Schema Diagram is to convert the ER diagram the Entity concept, whatever we do in this. Translate thus into a DB Table Design, you want to design the Database Table, while you want to also identify the Foreign Keys. How do you map? How do you embed these Relationships works for Managers, Controls, Works On etcetera? These Relationships into different Tables by linking various Columns. That is what the main advantage of creating a Schema Diagram. This is an intermediary translation part.

(Refer Slide Time: 31:35)









Now, that I just mentioned Foreign Keys here, so, let us now talk about what is Foreign Keys in a second. So, Foreign Key is an important concept and again we are not going to go into too much of details. But the fundamental aspect is,

- Transform each ‘One to One’ (or) ‘One to Many’ relationships as a “foreign key”.

The main idea here is that use either One to One or One to Many relationship, you translate or transform each one of them into a Foreign Key. So, what is a One to One relationship? This is a One to One relationship, exactly one record and table 1 is related to another record in table 2, One to Many is One record or one tuple is related to Many other records in table 2. Table 1 one record is related to many records in table 2. So, in both cases, the one to one or one to many, we map the relationship as a Foreign Key.

So, another way to think about it is,

- ❖ Foreign Key is a reference in the child table or many table to the Primary Key of the parent or one table.

So, what we are saying is, if you think about it, this table is what we call as the parent, this is the child. And same thing, this is the parent and this is the child.

So, you think in that way, what we are talking here is Foreign Key is the reference in the child (in the many table) to that of the Primary Key of the parent (the one table), table from which it comes in. So now, let us take a look into this, Department and Employee. One department has multiple employees or employee works for one department or department has many employees. So, let us draw that as an example. So, let us take this, DEPARTMENT has

EMPLOYEE. So, Department 1 N, that is how we do here. So, each employee will work for exactly one department, whereas one department will have multiple employees.

So, if you want to really draw the min max here, the employee going this way. Employee will work for a minimum of one department, maximum of one department, a Department will have a minimum of one employee, maximum of N employees. So, that is another way to draw it.

So, then when you have a relationship like this, how do you create or how do you translate that to tables? So, I am going to write two queries here. So, the first one is, I am going to create a table. My first step is:

1) Create the Department Table. So, I will write the SQL command. So, let us write the SQL for the time being. Create Table Department, so I am going to say it as (department_number) as I am going to say is as INT, name is VARCHAR (variable character) as (50). And then I am saying that Primary Key is (department_number). So, this query will create a Table with two Columns. So, how would you call as a Table like this.

If you think about it. Department underscore number, and name. So, Department number 1- HR, Department number 2- Accounts, Department number 3- Marketing, etcetera. So, each one of the Row is an example of a particular department. So, this department number 1, 2, 3, these are the Primary Key. So, one department, each department, how many employees so, in this case, the One Table or the Primary Table or the Parent Table will become the department.

2) Create the Employee Table. Now, comes the Child Table in this, because our department can have many employees. So, I am going to write the SQL for creating the employee, that is Create Table Employee with EMP number (Employee number), INT, I am just putting as INT the stands for Integer. So, then, I am going to say as department underscore number as also as INT, name VARCHAR (100), as the name of the employee in this case.

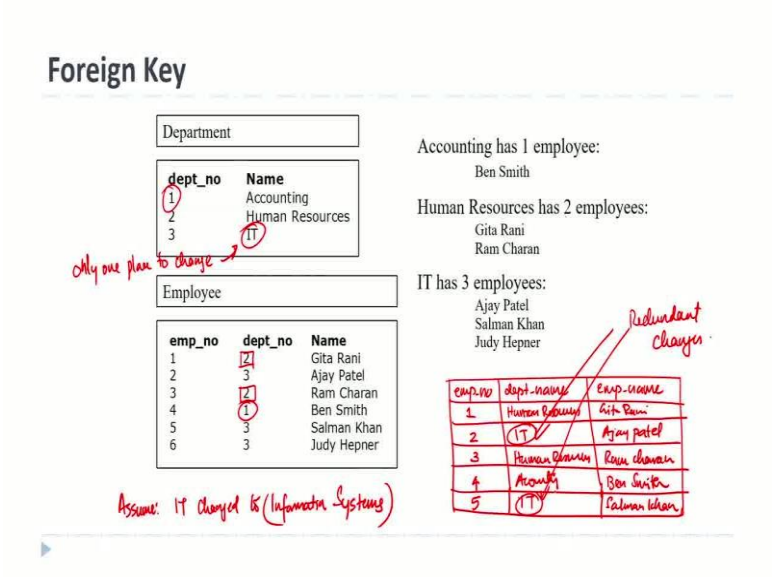
Then, what you call as Address for the time being. Address VARCHAR (250), something like this. Then, I will say Primary Key, I will say thus EMP (Employee number) that is the Primary Key. And then I also have something called Foreign Key, where I will say thus department no. REFRENCES Department. So, now this command tells the DBMS, that now I

have another Table, that is the second Table, where I have EMP number, the employee number. Then, there is something called Department number and there is a Name and Address.

So, I have Employee number as 101. Then there is an Employee number as 102, Employee number 103 etcetera and I have a Ram, Sita, Gita, like this and have Department numbers as 2, 1 and 1 address is Ayodhya, something like that. Sita is New Delhi, Gita is Jaipur etcetera. So, you can see that Department 1, 1 which means these Sita and Gita belongs to the Department number 1, which is the HR.

So, this Employee number is the Primary Key of the Child Table. This is the Child Table, whereas the Department number, which is the Primary Key of this is connected as the Foreign Key. So, the idea is, both the parent and the child when especially when it comes to what we call as the One to One or One to Many relationships, we create using the Foreign Key. we connect these tables using the Foreign Key.

(Refer Slide Time: 41:22)



So, I have an example. So, here is the Department. As department number 1, 2, 3, Department is the Relationship or Department Table. I have Department Number and Department Name as we wrote earlier, because this is the query that we created Department Name and Number. And then, we have Employee and Employee Number, Department Number.

So, we have Gita Rani, Ajay Patel, Ram Charan, Ben Smith, Salman Khan, Judi Hepner, as different thing. So, you can see that from here Employee number 1 Gita Rani works for

Department number 2. So, which is Department number 2, you go to Department Table 1, find the Primary Key, which is Department number 2 says Human Resource. So, Gita Rani works for Human Resources.

So, by instead of typing the Department Name all here, this Primary Key can be put in here, so that we can cross reference each department. So now, what are the advantages of doing this? So, let us look into this quickly. Accounting has only one employee because this the Ben Smith one employee of Accounting. This is connected to each other.

Then Human Resources has 2 employees how do you know there are 2 employees because Human Resource department is number 2. So, you have 2 and 2 so, 2 employees like this. And IT has 3 employees all the 3 in the department numbers related to IT. Instead of this, let us think about a scenario where you had something like this. Employee number, Department name and Employee name, assume it this way.

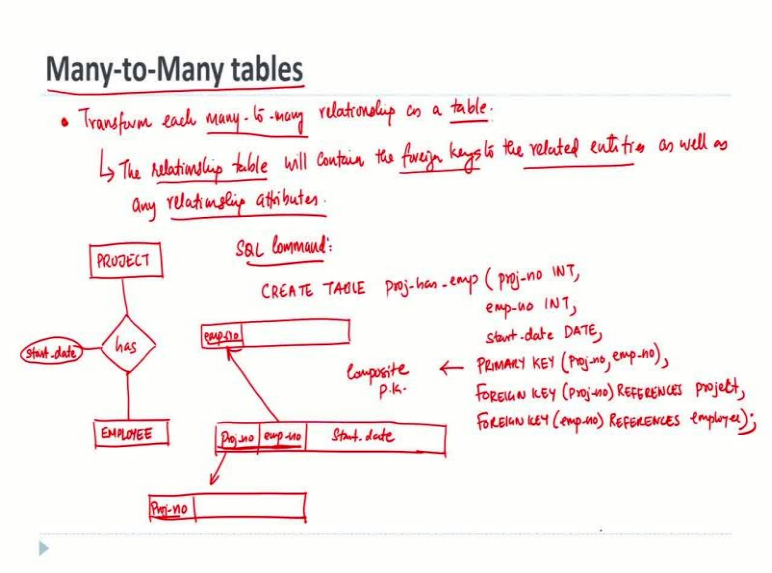
And we had something like this, and Employee number 1. This is Human Resources. Employee name is Gita Rani. Employee number 2 is IT Ajay Patel, Employee number 3 is Human Resources Ram Charan, Employee number 4 is Accounting Ben Smith. Let us just take one more employee. Employee number 5 is IT Salman Khan.

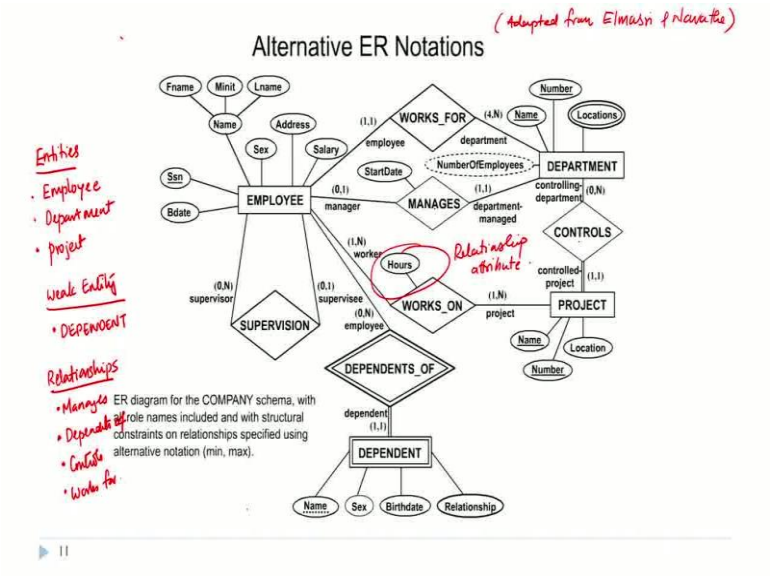
So, assume that somebody decided to change the name of IT department to Information System. Assume: IT changed to Information Systems.

You changed the name of two Information Systems. In this design you only need to change this here, only one place to change. Just change this IT. In this case you will have to change many places, wherever you type IT, you have to go there and change.

So, what happens is, this violates the principle of DBMS, because you want to eliminate redundancy, these kind of multiple changes, Redundant Changes or what we called as Update Anomaly. We will study about this later. But that is one of the other reasons why Foreign Keys are important rather than typing models information in a database.

(Refer Slide Time: 45:03)





Now, let us take the next one, what we call as the Many to Many Tables. So, the Many to Many Tables is, in this case, the primary rule or the fundamental guiding principle is,

- ❖ Transform each many to many relationship as a table.

Create a new Table. Each Many to Many relationship or is now translated to a Table. So now, how this happens?

- ➢ The relationship table will contain the guideline, the Foreign Keys to the related entities as well as any relationship attributes.

So, the relationship table, the new table that you will create, will contain the Foreign Keys to both of the related entities. So, both the Foreign Keys from both related tables. Many to

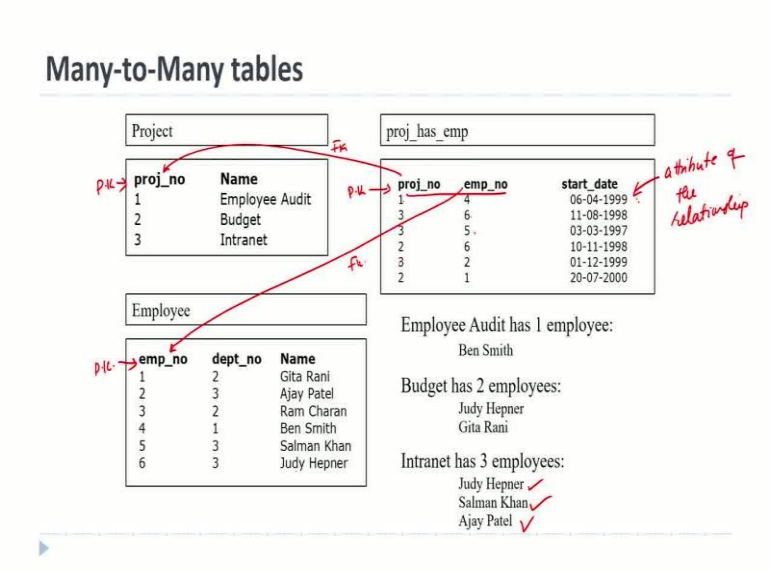
Many is related to both tables. So, in that regard, we are creating a Foreign Keys from both table will come into this picture. And also, that any of the relationship attributes. What is the relationship attribute? Hoursis a relationship attribute.

Let us take an example of the Project and Employee. The Employee works for a Project and let us take start date of something, that you need to capture. So, let us create a relationship here. So, we have PROJECT. And, PROJECT has EMPLOYEE and the Start Date of the Employee on that particular Project is an aspect of the relationship. If that is a case, then what we can do is, we can write a SQL command for this. Let us create this SQL command for creating this relationship table.

- So, we write it as CREATE TABLE, I am going to name it as proj. has emp. I am saying that the Project has Employees. So, new relationship table that I am creating at this point. And you have something called Project number as the INT for the time being, then I have Employee number that is also Integer, then start date as DATE. Then I have Primary Key, I have Project number, Employee number. So, I have two of them, two of the Rows together, now identifying the Primary Key. So, these kind of Primary Keys are also known as Composite Primary Key, two of them are there. And then I am saying FOREIGN KEY, Project number is a Foreign Key, REFERENCES Project that is the references the Project table. And I am also going to say FOREIGN KEY Employee number REFERENCES Employee.

So, what we are saying is the Primary Key in this regard, if I create a Schema Table, I will have Project number. I have another one called as emp number and then I have start date but both of these attributes together will be my Primary Key for this Project. And the Project number will refer to the Project Table. It will refer to this one, whereas the Employee number will refer to the Employee table. That will be the Primary Key of the Employee Table. So, that is how the relationship gets mapped out in this regard.

(Refer Slide Time: 50:57)



So, that is the case, then let us look at an example quickly. So, you have, I said earlier that Project Table has the Project number. Project number is the Primary Key for this one. Each Project is uniquely identified by Number in this regard. And then I have an Employee Table. As we were talking earlier, we have a Project Table. So, the here is your PROJECT and you have your EMPLOYEE. And this table is your Project has Employee Table. So, this is a new Table that captures the relationship. The Project has Employee.

So, there is a Project Table and here is the Employee Table. So, there are three Projects, Employee Audit, Budget and intranet, and the employee number is 123456. So, from there, we create a new system that has a Table, that is called as Project has Employees. So, the Project number is now the Primary Key here, for this table. But this is a Foreign Key relationship. The Employee number, this is the Primary Key here, for the Employee table. But the Employee, this one is a Foreign Key relationship. Both of these put together as the Primary Key for this table. So, when you say that for Project number 1, which is Employee Audit, Employee number 4, 4 is Ben Smith, started working on it from the date is 06-04-1999.

So, if you look into the details, then we can say that the Employee Audit has one Employee that is Ben Smith, we are able to connect them using this. You take the Project number 3, Project number 3 is Intranet. Employee Number 6 is working on it. So, 6 is we look into this Judy Hepner. So, you can see Intranet Project has Judy Hepner. Then 3 has 5, Employee number 5 is Salman Khan, that is also there. Then Project number 3, which is again Intranet has Employee number 2. Employee number 2 is Ajay Patel.

So, using this approach, you can map which Employee is working on which Project and the Start Date information which is a thing, that is connected to your relationship, as we shown here, is captured as part of the Table, in this regard.

So, the Start Date this is an Attribute of the relationship. So, with this I hope the Translation, the Concept of Primary Key, Foreign Key etcetera are clear with you. And what we will do now is, after this, we will move to the next concept of how to form this table design, Primary Key, Foreign Key etcetera. We will talk about what is Database Normalization, so that the queries and other things can be done efficiently.

And after that, we will look into some of the major SQL commands and then where to look to learn well, how can you learn some internal resources to learn MySQL and MySQL, Maria DB etc. And once that is completed, then I will give a quick introduction to Big Data and then, that will pretty much complete my portion of the Database aspect for this particular course.

But also understand we are not gone into details of step by step or breaking it down. I am just giving you an overview. These kinds of steps of how to conceptually design from a User specification, come up with an ER diagram, from the ER diagram to a complete Schema, from Schema to a Table, identify Primary Key, Foreign Key create it, normalize it. All those continuous examples will be taught as part of the advanced component of this course. But this is to give you a fair idea what is going on? So, thank you for your patient listening and we will continue the remaining in the next class. Thank you.