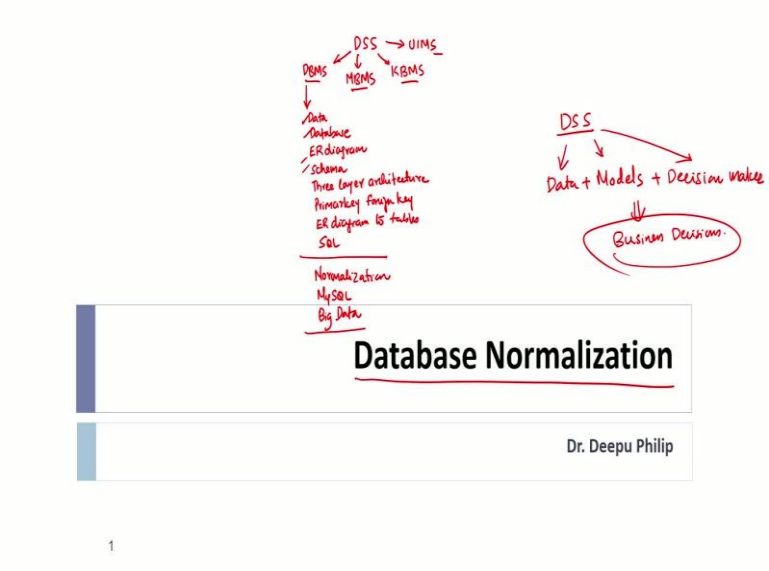
Computer Aided Decision System
Industrial Practices using Big Analytics
Professor Deepu Philip
Department of Industrial & Management Engineering
Indian Institute of Technology, Kanpur
Professor Amandeep Singh
Imagineering Laboratory Indian Institute of Technology Kanpur
Lecture - 15
Database Normalization

Good afternoon everyone. Welcome to at another lecture of the Web Based Decision Support System course for the decision makers and the academicians, practitioners, sub product stuff. So, we have been discussing on different aspects of DSS system.

(Refer Slide Time: 0:35)



So, today if you look at it quickly as a quick recap, we looked into DSS and we have four sub part of DSS, one is Database Management System (DBMS), then Model-Based Management System (MBMS), then Knowledge Base Management System (KBMS) and User Interface Management System (UIMS). And we are taking a brief look into all of them. And now we are looking in depth into Database Management System.

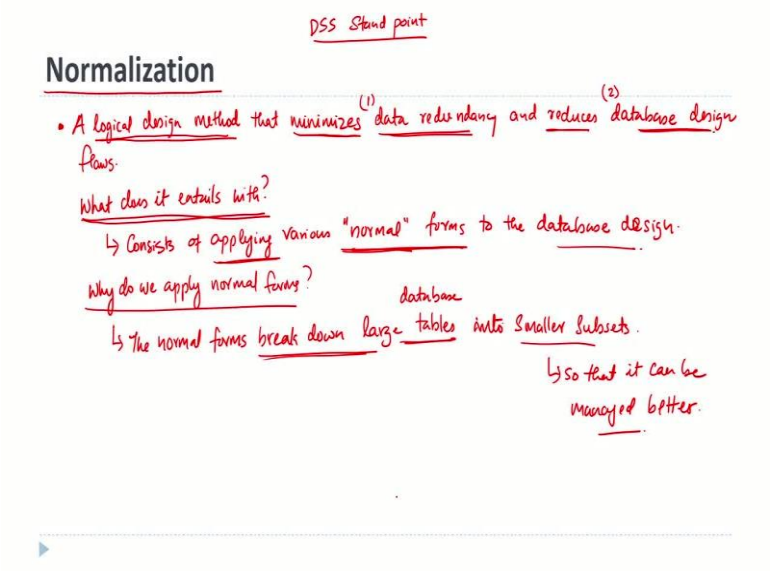
So, we studied what say Data, Database. We also looked into what is called as an Entity Relationship Diagram. We saw what is the Schema. How to translate the three-layer architecture? And then we studied what is the concept of Primary Key and Foreign Key. And, then we also studied what we call as how to translate from an ER diagram to a Tables. And, some brief introduction of SQL, we have gone through these many aspects of it.

So, now all these aspects are being completed from our side. Now, today we are going to look into what they call as a new thing called 'Normalization'. And also, we will try to continue to learn something about 'MySQL' and some commands of MySQL. And finally, we will try to wrap this whole lecture out with introduction to 'Big Data' and the remaining aspects of the Big Data will be taken over by other instructor Dr. Amandeep Singh. And, once this is done I will move towards User Interface Management System or Model-Based Management System, depending upon how much time and other things that we left out.

Remember, all these are supposed to give you the brief introduction to the course and make you familiarize with the ideas and the applied side the putting it into practice, learning it to how to do it on a server side will all be done as part of the advanced course which will be its subsequent to this one.

So, today we are going to learn the topic 'Database Normalization' and it is very important topic, especially when it comes to DSS because the DSS uses Data plus Models plus Decision Maker and that translates to Business Decisions. So, DSS support this all the three, and this which we create what we call as a Business Decision, which will have a financial impact on the organization.

(Refer Slide Time: 3:33)



So, let us see what is Normalization, the concept of Normalization. So, there are many definitions available. And again, from a DSS Stand Point is what we are going to talk about, what is the, the Normalization and the concept of Normalization? So, the simplest definition for this:

- A logical design method that minimizes, data redundancy, data redundancy and reduces database design flaws. So, fundamentally, it is a logical design method. The aim is to minimize. It has to minimize two things what is it number one is minimize the data redundancy. And also, number two is to reduce the database design flaws. The second aspect is how to reduce the database design flaws. So, what does it entails to, there the question is, what does it entails with or what are the major steps associated with it? So, the first part is:

  - Consist of applying various "Normal" forms to the database design. We will see what is the "Normal" forms. So, the first thing is you have Multiple "Normal" forms that are available (quote unquote normal forms) and we take these "Normal" forms and applied to the database design in a sequence.

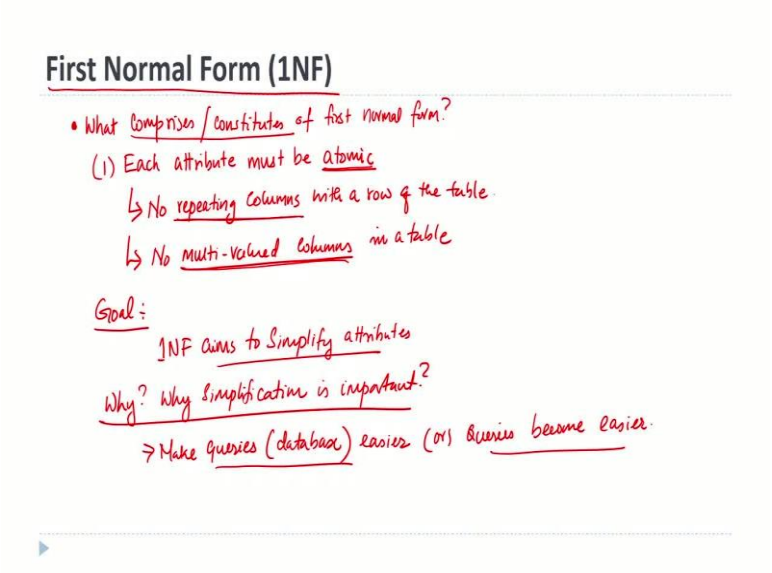
And why do we apply "Normal" forms? The reason is:

- The "Normal" forms break down large database tables into Smaller Subsets. So, as I said earlier, it is a logical design method. And there are two aims, number one is to minimize data redundancy, that is the first goal and number two to reduce the database design flaws. Those are the two goals of the Normalization.

So, what does it consist of, or how do you achieve this? What do you do is, you keep on applying, you consistently apply various normal forms. We will talk through what are different normal form. First normal form, second normal form, that is so many normal forms. We take these normal forms and we apply it to the database design.

And why do we do this? Why do we make these things? Because these normal forms help us in breaking down large database tables into smaller subsets, so that it can be managed better. The aim is to manage this database tables better because once you have a smaller table, then it is easy to manage and better so that you can also reduce redundancy and reduce the design flaws.

(Refer Slide Time: 7:32)



So, without further delay, we go into what we call as the First Normal Form. Usually represented as 1NF.

- What comprises/constitutes of First Normal Form? So, the first one, the initial thing is:

1) Each attribute must be atomic. So, atomic is an interesting word that we need to understand, what does it means? So, it means two things,

- ➢ No repeating columns within a row of the table. So, you cannot have a column repeating. So, first criteria is that, in every row you cannot repeat the same column, thing cannot be repeated.

- ➢ No Multi-Valued Columns in a Table. So, you cannot have repeating columns and you cannot have Multi-Valued Columns. So, this is what it amounts to be. So First Normal Form, each attribute must be atomic. Atomic means there cannot be any Repeating Columns and there cannot be no Multi-Valued Columns.

What is the goal of the 1NF?

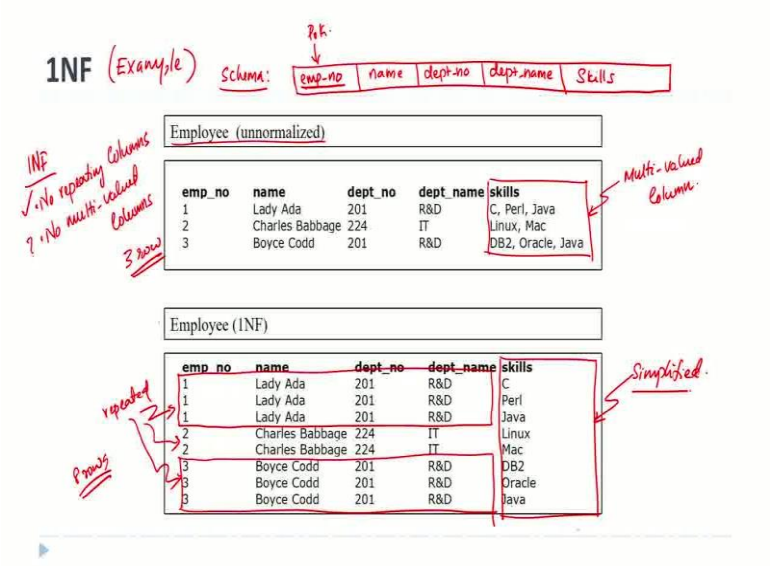
- 1NF (First Normal Form) aims to simplify entity attributes. So, attributes can think about as Columns can simplify attributes.

So, why do you need to simplify attributes? Why Simplification is important or why do we need simplification?

- Make (database) queries easier (or) queries become easier. Our aim is to make the queries easier. So, queries become faster, quicker and you can actually query the database without too much of confusion.

So, that is, why is the need of First Normal Form. Again, as I said earlier what comprises or constitutes are the First Normal Form or fundamental question and the rule is that each attribute mast be atomic. Atomic means No Repeating Columns, No Multi-Valued Columns. If these two conditions satisfy we can call it as atomic. And the aim is at simplifying and why, because to make the queries easy.

(Refer Slide Time: 11:11)



So, let us talk about 1NF example. let us see a Demonstration, how this we can take care of it. So, let us take a Table here, this is an Employee Table, we call this an Un-Normalized Table and we were being discussing this in the previous. So, if I do a Schema of this, the Employee number, so the Schema will be something like this. If I do the Schema, it will be challenged by that Employee number, then there is a Name, then there is a Department number, then Department name and Skills. This is a Schema. And the Employee number with an underlining that means this is the Primary Key (PK), I call it as a PK (not the movie PK), PK stands for Primary Key.

So, then you can see that there is an Employee one named Lady Ada who belongs to Department 201 and the Department name is R&D and the Skills are C, Perl, and Java so, 3 skills. Employee number two Charles Babbage from IT has Linux and Mac are the Skills.

And the Employee three Boyce Codd from Department R&D, again same Department, DB2, Oracle and Java.

So, the rule says No Repeating Columns. So, there are no columns that are repeating. So, 1NF rules are:

- No Repeating Columns.
- No Multi-Valued Columns.

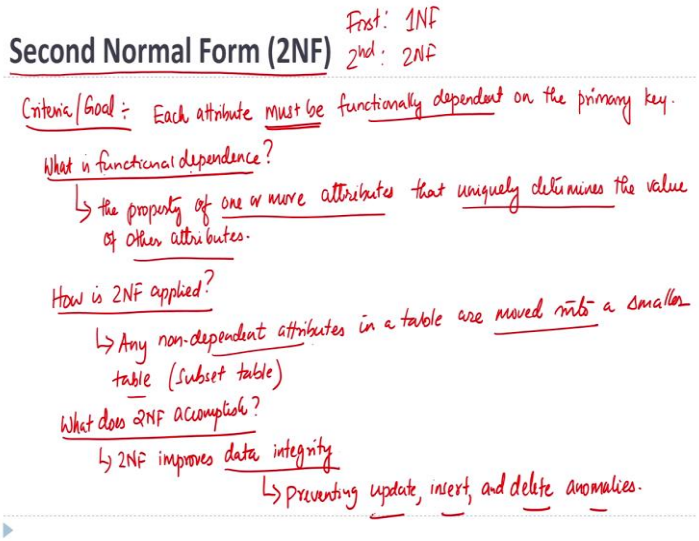
So, now when you look into this, the first part is okay, we do not have any Repeating Columns. But the second part is we have a Multi-Valued, so this is our Multi-Valued Column. So, this means, this Employee Table is currently not in First Normal Form.

How do we make this into First Normal Form? The way to make this into First Normal Form is remove the Multi-Valued Column. So, then the resulting table in the 1NF will come to be something like this- Employee number, Name, Department, Skill. So, you will see Employee number 1, Employee number 1, Employee number 1 Lady Ada. All these details are repeated, where the Skills are different.

So, by Normalizing what happens is, there is only this column got simplified, there is no more Multi-Valued Attributes. We can search how many people in this whole thing knows Java. So, we can basically say Boyce Codd also knows and Lady also knows Java. So, this problem becomes much more easy but you can see that, this much of information is repeated.

And so is this also and so being with Charles Babbage also, but I am just showing this. So, it is repeated and Charles Babbage also repeated. So much of redundancy has now added, so in the row vise you have added redundancy, but in that process, you eliminated the Multi-Valued Columns. But as per the definition of 1NF, so this is a 3-row table. Now, this has become 8-row table. The 8-row table is now in 1NF or 8 rows are created, so that, this Multi-Valued Column (the last column), Skills column got to what we call as a First Normal Form.

(Refer Slide Time: 15:21)



Now, let us talk about the following one. So remember, what is Normalization. we sequentially apply a set of normal forms to make the Table (the database design) better. So, the Second Normal Form is:

- Criteria or Goal of the Second Normal Form is that each attribute must be Functionally Dependent on the Primary Key. Each attribute must be (this is a mandatory condition and it should be) Functionally Dependent on the Primary Key.

So, now the question is what is Functional Dependency or what is Functional Dependence?

- The property of one or more attributes (certain attributes) that uniquely determines the value of other attributes. So, what we are saying here is, Functional Dependencies is a property of one or more attributes (certain attributes in a table), it has a capability to uniquely determine the value of other attributes. So, if you know one attribute then you can determine the rest of it.

How is 2NF applied? The application part of 2NF is pretty simple.

- Any non-dependent attributes in a table are moved into a smaller table or a subset table.

What does 2NF accomplish? So, the first thing is,

- 2NF improves data integrity.

So, what is the Data Integrity?

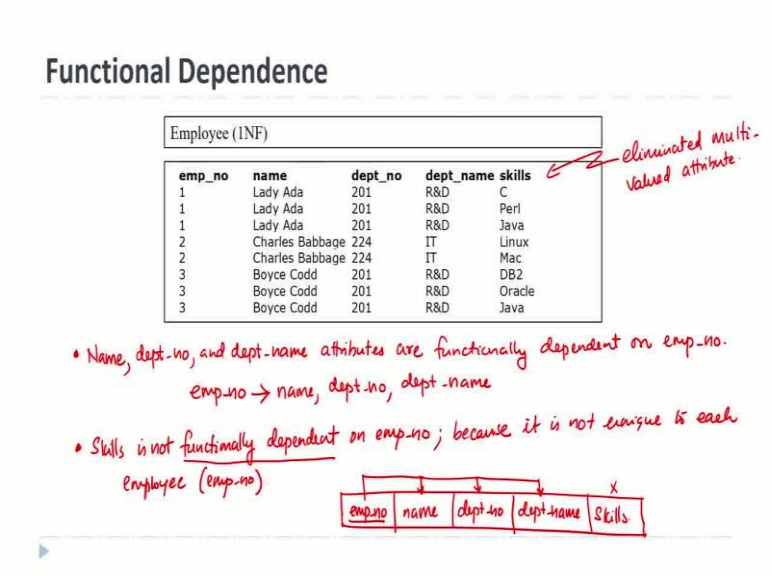
- It is a Preventing update, insert and delete anomalies And what these anomalies? Anomalies, we will see with examples quickly once we go through.

So, the rule is:

1) You first apply 1NF.
2) Second step is to apply 2NF.
3) And after that you do 3NF, 4NF, etcetera like that.

So, we now know what is the Second Normal Form.

(Refer Slide Time: 20:02)



So, now let us take an example of the Functional Dependence. Let us see what Functional Dependence is. So, let us take the Table, that was in the 1NF, the same table that we saw before, where the Skills was a Multi-Valued Attribute. So, we eliminated Multi-Valued Attribute.

So, this Multi-Valued Attribute was eliminated and we got the table in the First Normal Form. So, in this case we can control few things,

- Name, Department number, and Department name, attributes are Functionally Dependent on Employee number. So, in another way to say, it is that Employee number, which is the Key Attribute, determines the Name, Department number, and

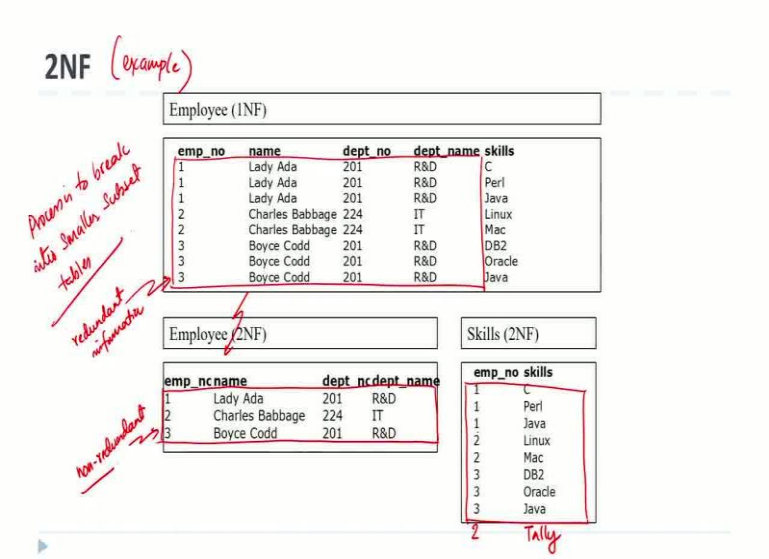
Department name of an Employee. So, if you look into this, that is the Functional Dependent. So, if you know the Employee number you can figure out the Name of the Employee and which Department that person is working for and the number of the Department. So, then the other part is, you should also see that what about the last column?

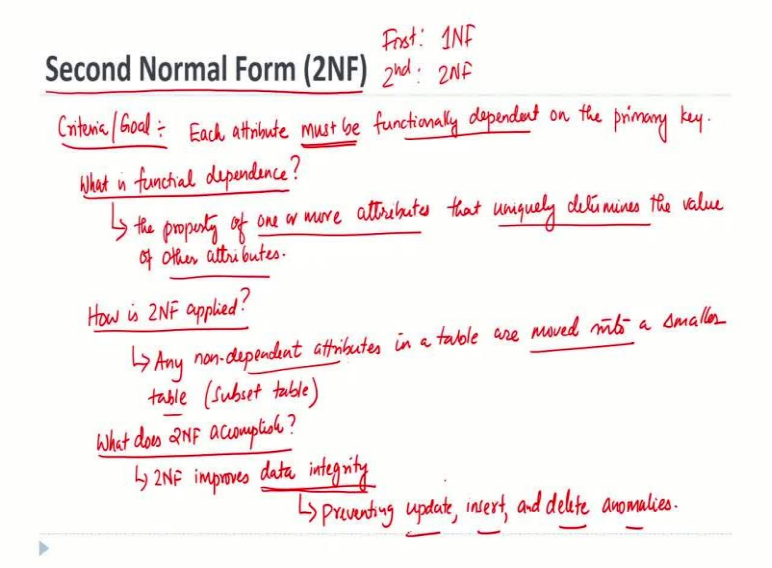
- Skills is not Functionally Dependent on Employee number. Why is it not Functionally Dependent employee number? Because it is not unique to each Employee or Employee number.

So, knowing Java is not unique, other people can also know Java. So, it cannot say that the Employee number three uniquely determines or functionally determines the individual skills of the person.

So, then what happens is, the skills aspect of it is not really something that can be determined by Employee number. So, one other way to denote this, is typically what certain type people do is this. They create a Schema Diagram. There is an Employee number, then there is a Name, then Department number and Department name (this is another approach people do) and Skills. An Employee number you draw something like this, so, that means the Employee number functionally determines all the three attributes and not the Skills attribute. So, that is something, that is outside the Functional Dependency of this Table.

(Refer Slide Time: 23:36)





So, now, how do we get it into this one? How do we convert it do 2NF? So, this is example of 2NF. See the example, the Table that is in the 1NF form the Employee number name with the skills and everything which was reduced from the un-normalized form to 1NF. What we do is, we first create a Table because the:

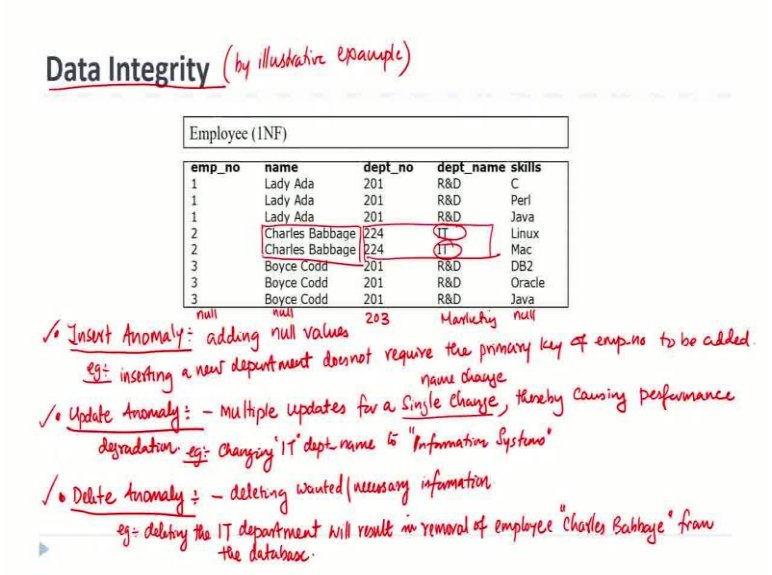
- Process is to break into smaller subset tables so that, the Functional Dependency can be maintained.

So, first we break it into a thing, where these aspects the 2NF Tables, the Employee number uniquely determines the Name, Department number, and Department name of the person. So, this is a new Employee Table (this is a Second Normal Form Table), and we create a second table of 2NF Table called Skills. So, we just keep record of what are the Skill each individual Employee has.

So, if a person acquires a new skill, so the person 2 Charles Babbage acquires a new skill of Accounting or let us say Tally software so, then we have Tally right here. So, that can be added into this and we do not need to change anything in this. So, you can see now that, all that data is being redundant, so these entire were redundant information, that when came here, got reduced. Now, we can see that non-redundant, where the time you implemented the Functional Dependency, this breaking it into two tables, all these, that we typing, the Name of the person, Department number, R&D, everything is gone. And the only time, where you can actually see repeating happens is, just this one. So, now, we can see that the resulting database has become much more efficient in comparison with that of the original database, where all these data, these 8 columns, has been redundantly kept on stored.

So, now, let us look at what we called as (I mentioned a term before what we called as) Data Integrity. So, this is the data integrity, preventing update, insert and delete anomalies.

(Refer Slide Time: 26:11)



So, let us talk about what is Data Integrity by illustrative examples. So, we are going to look into this by each individual example. So, the first one we are going to see is:

- Insert Anomaly: So, what is an Insert Anomaly is, that in a simplest thing, it is called as adding Null Values. So, here is an example for this- Inserting a new Department does not require the Primary Key of Employee number to be added.

So, one example is this so, I am creating a new Department call 203 and it is called as Marketing. So, if I add something called 203, Marketing right here. Then this value will be Null and can be Null and this could also be Null. So, this is called as an Insert Anomaly because this information, by just adding a new Department into this, I will have to put Null values for the remaining name because having an employee information is not really mandatory as part of this.

- Update Anomaly- Multiple updates for a single change thereby causing performance degradation. Let us take an example for this- Changing IT department name to Information Systems department. Assume that we changed IT department name to Information System. Somebody decided to change the name of the department. So, what do you do? You end up going here, you change this and you change this, if I change the department R&D to I will say innovative design or something like that. So,

then 123456 rows are to go there and type, type, type, type, type, and make the changes. So, when you make a single change, like for example, in this case the single change, we can talk about as a Name change. So, then it will result in, you have to change it in many places, wherever this Department name shows up, and then all data has to be removed and then the new data has to be protected. And when you have so many rows in it, now only we have 8 rows. Assume there are 8000 rows, then you go through and change wherever rows this department name is appearing twice so, that becomes the Update Anomaly.

- Delete Anomaly- This is also another anomaly and the simplest way to say about this is, Deleting wanted or necessary information. Now, let us take an example out of this- If I delete the IT department will result in removal of Employee. Who will be removed out of this, if I remove the IT department? Charles Babbage will be removed from here, employee Charles Babbage from the database. So, let us say the top person decides that IT department is useless, we need to remove this department so they deleted it. And in this kind of a Table, where you are not taken care of stuff like this and you deleted the IT department, then along with the deletion of IT department, Charles Babbage's information also, the employee whose necessary, who was the only one with the Linux and the Mac experience in the organization also, gets deleted out.

So, these three things, what we call as Data Integrity as part of this is, basically driven by the Insert Anomaly (number one), Update Anomaly, Delete Anomaly. And the aim of doing 2NF is to prevent these anomalies.

(Refer Slide Time: 32:00)

Third Normal Form (3NF)

First : Apply 1NF
Second : Apply 2NF
Third : Apply 3NF

Goal/Aim: Remove transitive dependencies

What is transitive dependence?
↳ two separate entities exist within one database table.

How do we address transitive dependence?
↳ Any transitive dependencies are moved into a smaller (subset) table.

What this results in?
3NF further improves data integrity
↳ prevents update, insert, and delete anomalies.

Data Integrity (by illustrative example)

Employee (1NF)

emp_no	name	dept_no	dept_name	skills
1	Lady Ada	201	R&D	C
1	Lady Ada	201	R&D	Perl
1	Lady Ada	201	R&D	Java
2	Charles Babbage	224	IT	Linux
2	Charles Babbage	224	IT	Mac
3	Boyce Codd	201	R&D	DB2
3	Boyce Codd	201	R&D	Oracle
3	Boyce Codd	201	R&D	Java

null null 203 Manuchy null

- Insert Anomaly : adding null values
  eg: inserting a new department does not require the primary key of emp_no to be added.
- Update Anomaly : - multiple updates for a single change, thereby causing performance degradation. eg: changing 'IT' dept_name to "Information Systems" name change
- Delete Anomaly : - deleting wanted/necessary information
  eg: deleting the IT department will result in removal of employee "Charles Babbage" from the database.

2NF (example)

Employee (1NF)

emp_no	name	dept_no	dept_name	skills
1	Lady Ada	201	R&D	C
1	Lady Ada	201	R&D	Perl
1	Lady Ada	201	R&D	Java
2	Charles Babbage	224	IT	Linux
2	Charles Babbage	224	IT	Mac
3	Boyce Codd	201	R&D	DB2
3	Boyce Codd	201	R&D	Oracle
3	Boyce Codd	201	R&D	Java

Process to break into smaller subset tables

redundant information

Employee (2NF)

emp_no	name	dept_no	dept_name
1	Lady Ada	201	R&D
2	Charles Babbage	224	IT
3	Boyce Codd	201	R&D

non-redundant

Skills (2NF)

emp_no	skills
1	C
1	Perl
1	Java
2	Linux
2	Mac
3	DB2
3	Oracle
3	Java

2 Tally

So, now we move to the Third Normal Form (3NF). So, our first step is:

1) Apply 1NF
2) Apply 2NF
3) Apply 3NF

So, Third Normal Form the Goal or Aim. What do we gain to achieve by doing a Third Normal Form? The goal is to Remove Transitive Dependencies. To Second Normal Form was to make sure the Functional Dependency, here you are trying to Remove Transitive Dependencies.

So, question is what is Transitive Dependence?

- So Transitive Dependence means, the simplest way to answer to this question is, Two separate entities exist within one Table. So, there is one Database Table, you can think about as one database table. One single database table, in which two separate entities or two separate items are physical exist within the same Table.

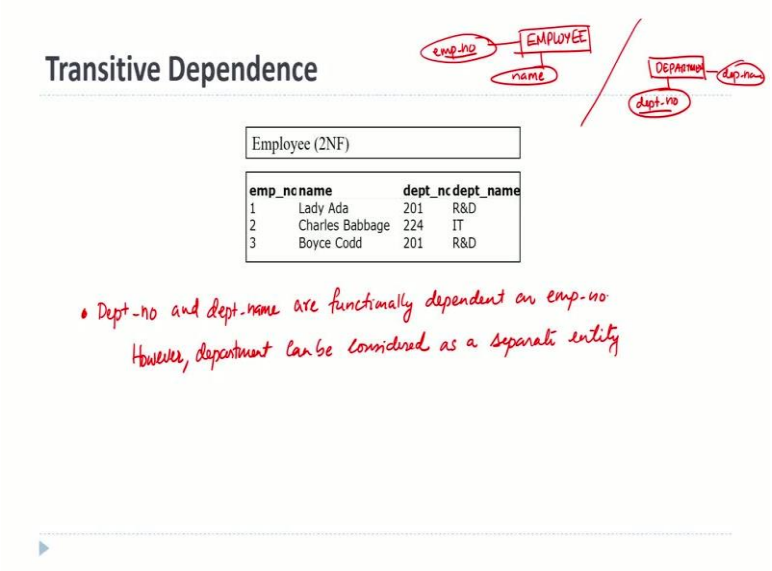
So, how do we address Transitive Dependence?

- Any Transitive Dependencies are moved into smaller or a subset of Tables. Same way, as we do the Functional Dependency here also, we move into a subset table.

So, what this result in? What is the Third Normal Form?

- 3NF further improves more data integrity. The aim is again it improves data integrity more, which means it prevents update, insert and delete anomalies. So, in this table we have seen that, this Department number and Department name. Department is also another entity. An Employee is another entity. They are co-existing in one table. Though Functionally it is but Transitive Dependency which means two separate entities existing in one table, does not really help in this regard.

(Refer Slide Time: 35:44)



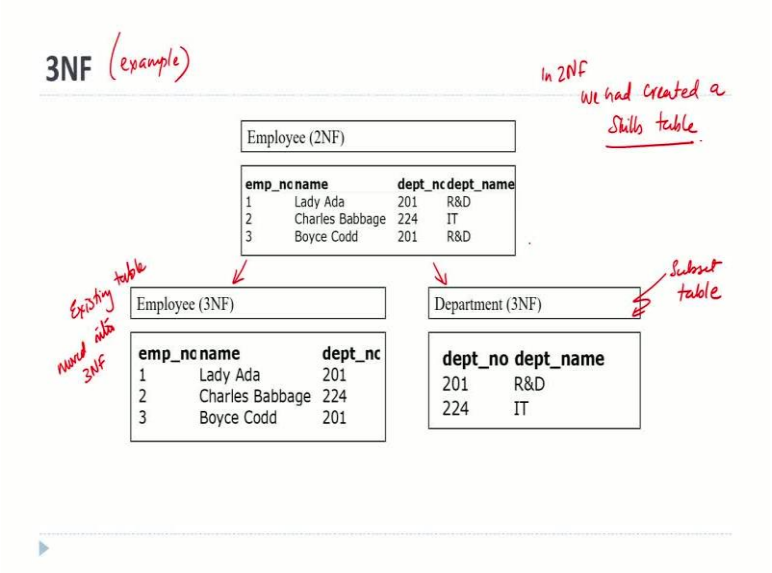
So, then how do we solve that problem? So, as we said earlier, we will take the Table, that is in this 2NF, which is the Employee name. So, we have one entity which is the Employee, which has an Employee number to it. And then there is a Name of the Employee, as part of this. And then there is a Department entity, that has a Department number, and Department name.

So, both of these are two separate entities, but they co-exist in one table. So, that is the one, what we call as Transitive Dependency. So, how do we solve that problem or how do we address this problem? So, the way to think about it is:

- Department number and Department name are Functionally Dependent on Employee number. They are Functionally Dependent because the Employee tells you what Department the person is belong.

- However, Department can be considered as a separate entity. So, as I shown you here, the two separate entities. You can consider them as two separate entities. So, then how do we tackle this problem?

(Refer Slide Time: 37:39)



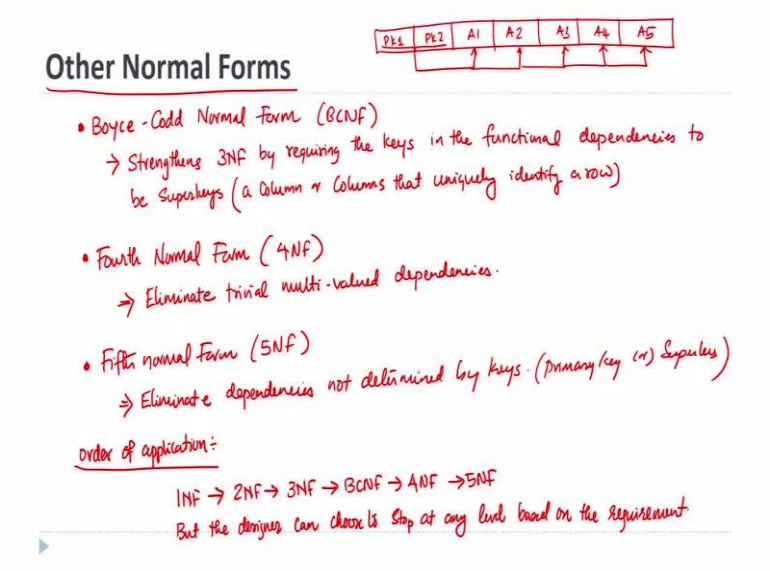
So, the simplest thing to do it is, let us work this example out. So, the Employee (2NF Table), Employee number, Name and with the two separate entities existing in it. So, to move it into the 3NF. So, the first part is,

- We break the Employee number, the Employee name and the Department number only into one table. And this is the Existing Table moved into 3NF
- And we have the Department a newly created Table, this is the subset table, that is created.

So, the Employee Table got reduced. So, the Department Name got removed, but the Department Table 3NF, the new subset table that is created, has Department number with the Department name as part of it. Now, when you look into this table, that we have new 3NF Table created. Earlier also we have splitted:

- In 2NF we had created a Skills Table, that is another subset. So, as we go through, we keep on applying Normal Forms, we break tables into smaller, smaller things, to make sure that the Table becomes, the you can easily manage the database and update, delete, and insert anomalies can be managed.

(Refer Slide Time: 39:18)



So, then comes the Other Normal Forms. So, we have gone to first second and third normal forms. Now, there is three more Other Normal Forms. So, the first one we go into talk about is:

- Boyce-Codd Normal Form (It is also known as BCNF)- We will talk about the order of application. What is this aims to? The main aim is,
  - It strengthens Third Normal Form by requiring the keys in the Functional Dependencies to be super keys that means it is a Column or Columns that uniquely identify a row. So, the Boyce Codd Normal Form literally means that you have a table like this and you have, let us say for example, PK1 and PK2 the two Composite Primary Keys. And we have attribute1, attribute2, attribute3, attribute4 and attribute5, then either this one or two, they should all uniquely identify together (identify the remaining five columns). So, the entire row is uniquely identified by this. This is a stricter form of the Third Normal Form.
- Fourth Normal Form (4NF):
  - Eliminate trivial Multi-Valued Dependencies. These things we will discuss more down in the remaining lectures because these are outside the scope of this introductory class.
- Fifth Normal Form (5NF):

- Eliminate Dependencies not determined by keys. So, whether it is a Primary Key or a Super Key. If there are any dependencies that cannot be determined by the keys then eliminate those dependencies.

So, the Order of Application for us in this regard:

1NF first followed, then Second Normal Form, then Third Normal Form, then Boyce Codd Normal Form, then Fourth Normal Form, then Fifth Normal Form. But the designer can choose to stop at any level based on the requirement. There is no rule that says you have to go all the way to the Fifth Normal Form. You are free to stop at Second Normal Form, you can stop at BCNF, you can stop at Fourth Normal Form. It is all depending upon the type of the database that you have, the application of the database and how the DSS is supposed to utilize the database.

So, many of the time I have seen people just applying First Normal Form and calling it good, but that may not be the good idea because many times you may also end up getting redundancies in the row. So, it may be better to go to Second Normal Form, Third Normal Form. I usually in complicated databases take it to Third or Boyce Codd Normal Form. But 4NF, 5NF usually when you require really good speed, extremely good retrieval and write speed then you need to look into the higher normal forms.

So, with this, we complete this portion of the lecture, especially of looking into Normal Forms, where we are now seen, how do we take the Database Design, the Tables that you designed and you break them into smaller ones so that your queries and other kinds of things are efficient and cute to process. So, we are not seen in details what are the major queries which we will cover in the next class or the major queries are.

And then after that we will take a quick look into how these queries pertaining to MySQL or Maria DB. And after that we will look into what is Big Data and then we will call it good after that and that will probably cover the basic requirements of DBMS (Database Management System) and the Component of the DSS. Thank you very much.