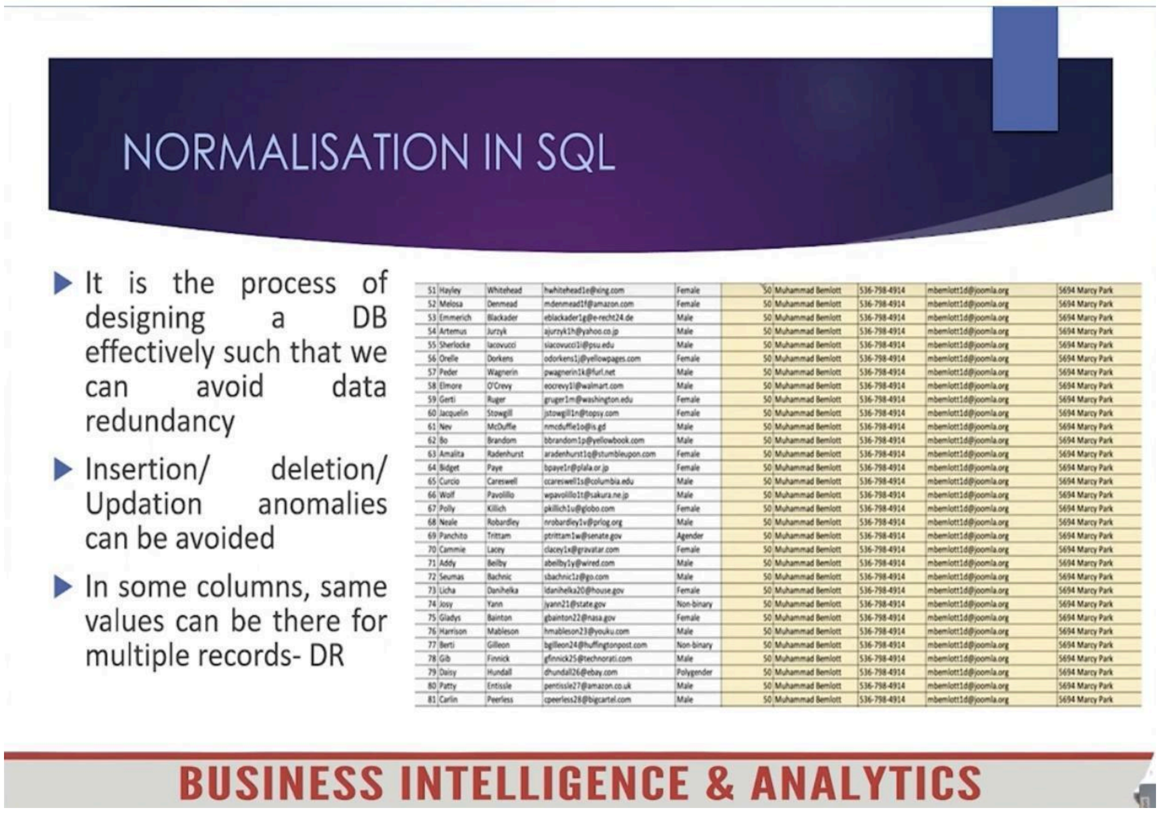
Course Name:Business Intelligence and Analytics
Professor Name:Prof. Saji.K.Mathew
Department Name:Department of Management Studies
Institute Name: Indian Institute of Technology Madras
Week:03
Lecture:10

NORMALISATION

Next, we are moving into a very important topic in SQL which is known as normalization. So, what is normalization? Normalization is a process of designing a database effectively that we can avoid data redundancy. For example, see this table on the right. There are so many information in that, but if you see the yellow part that is there on the right side, you will see that all the data is being repeated unnecessarily. Just because it is related to the part on the left, it is being repeated. So, what we can do is, we can form this yellow part as a separate table so that only it is repeated once or twice and we can just reference it to the white color table, wherever it is needed.



NORMALISATION IN SQL

- It is the process of designing a DB effectively such that we can avoid data redundancy
- Insertion/ deletion/ Updation anomalies can be avoided
- In some columns, same values can be there for multiple records- DR

51	Hayley	Whitehead	hwhitehead1e@xing.com	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
52	Melosa	Denmead	mdenmead1f@amazon.com	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
53	Emmerich	Blackader	eblackader1g@e-recht24.de	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
54	Artemus	Jurzyk	ajurzyk1h@yahoo.co.jp	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
55	Sherlocke	Iacovucci	siacovucci1i@psu.edu	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
56	Orelle	Dorkens	odorkens1j@yellowpages.com	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
57	Peder	Wagnerin	pwagnerin1k@furl.net	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
58	Elmore	O'Crevy	eocrevy1l@walmart.com	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
59	Gerti	Ruger	gruger1m@washington.edu	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
60	Jacquelin	Stowgill	jstowgill1n@topsy.com	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
61	Nev	McDuffie	nmcduffie1o@is.gd	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
62	Bo	Brandom	bbrandom1p@yellowbook.com	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
63	Amalita	Radenhurst	aradenhurst1q@stumbleupon.com	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
64	Bidget	Paye	bpaye1r@plala.or.jp	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
65	Curcio	Careswell	ccareswell1s@columbia.edu	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
66	Wolf	Pavolillo	wpavolillo1t@sakura.ne.jp	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
67	Polly	Killich	pkillich1u@globo.com	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
68	Neale	Robardley	nrobardley1v@prlog.org	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
69	Panchito	Trittam	ptrittam1w@senate.gov	Agender	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
70	Cammie	Lacey	clacey1x@gravatar.com	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
71	Addy	Beilby	abeilby1y@wired.com	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
72	Seumas	Bachnic	sbachnic1z@go.com	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
73	Licha	Danihelka	ldanihelka20@house.gov	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
74	Josy	Yann	jyann21@state.gov	Non-binary	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
75	Gladys	Bainton	gbainton22@nasa.gov	Female	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
76	Harrison	Mableson	hmableson23@youku.com	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
77	Berti	Gilleon	bgilleon24@huffingtonpost.com	Non-binary	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
78	Gib	Finnick	gfinnick25@technorati.com	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
79	Daisy	Hundall	dhundall26@ebay.com	Polygender	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
80	Patty	Entissle	pentissle27@amazon.co.uk	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park
81	Carlin	Peerless	cpeerless28@bigcartel.com	Male	50	Muhammad Bemlott	536-798-4914	mbemlott1d@joomla.org	5694 Marcy Park

BUSINESS INTELLIGENCE & ANALYTICS

In order, instead of writing it, maybe if there are 100 records, we will write it 100 times. Imagine a situation where there are lakhs of records. So, if we are mentioning it lakhs of times, would not it take so much storage? So, that is why we are doing the process of normalization, wherein we split this big table into multiple small tables so that we can just reference it to the original table wherever we need, instead of having this data redundancy. So, the main advantages of doing normalization in SQL is to avoid insertion, many kind of anomalies.

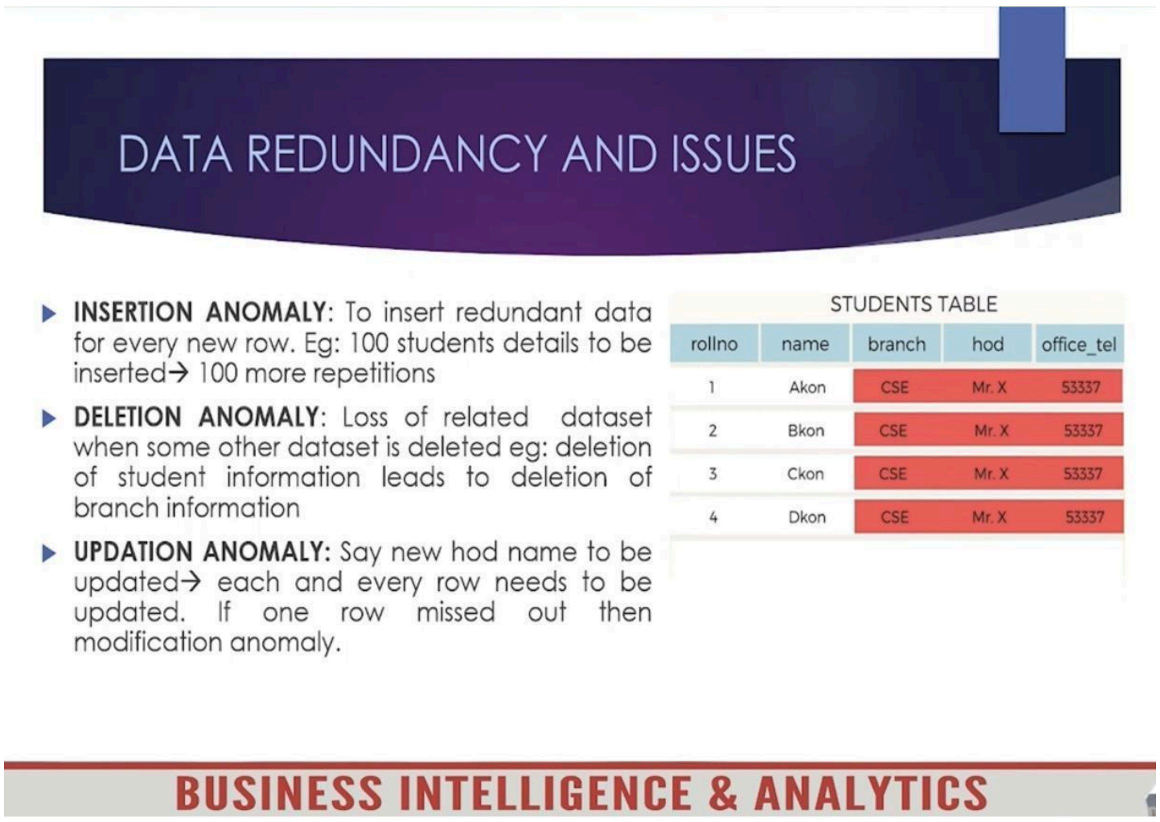
For example, insertion anomaly, deletion anomaly, updation anomaly etc. So, what is data redundancy? In some columns, there are same values for multiple records. That is known as data redundancy and in order to avoid this data redundancy, we perform normalization.

So, see this students table. In this, there is data redundancy. Which is the data redundant part? It is the part in red color. So, there are multiple students, Akon, Bkon, Ckon, Dkon, four students and all of them belong to the same branch. So, is it really necessary to mention their branch along with their names? So, if there are four students, it is not a big matter but if there are lakhs or lakhs and lakhs of students, then mentioning this computer science for each and every student, that is kind of a not a very logical thing to do because it is going to take a huge chunk of data from the storage. So, what we do is, so this data redundancy also has issues, multiple issues that is insertion anomaly. For example, if a new student comes, say name is John, so we are inserting this John's record in this table.

So, what we will do, say he also belongs to computer science department. So, again we have to mention the branches CSE, HOD as Mr. X and also the office telephone number, everything we are mentioning again and again. So, say we have to insert 100 more students, then the repetitions will become 100 times which is known as the insertion anomaly.

Next one is deletion anomaly. It is the same thing wherein you are deleting all students. Say the students have passed out or something and you are deleting the students information, then what happens? The branch, the HOD name, the office telephone numbers which are correspondingly written along with the student names also get deleted, is not it? If there are 100 students and we delete the 100 students names, then all the data, all the records itself get deleted. So, what we are losing is, we are losing all the entire data just because we want to delete a student's names. So, that is what is called as deletion anomaly.

Then there is updation anomaly. Say the HOD is changing for a particular branch, say Mr. X becomes Mr. Y. So, what we have to do? It is not possible to change in one place. We have to update all the students.



DATA REDUNDANCY AND ISSUES

- **INSERTION ANOMALY:** To insert redundant data for every new row. Eg: 100 students details to be inserted→ 100 more repetitions
- **DELETION ANOMALY:** Loss of related dataset when some other dataset is deleted eg: deletion of student information leads to deletion of branch information
- **UPDATION ANOMALY:** Say new hod name to be updated→ each and every row needs to be updated. If one row missed out then modification anomaly.

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

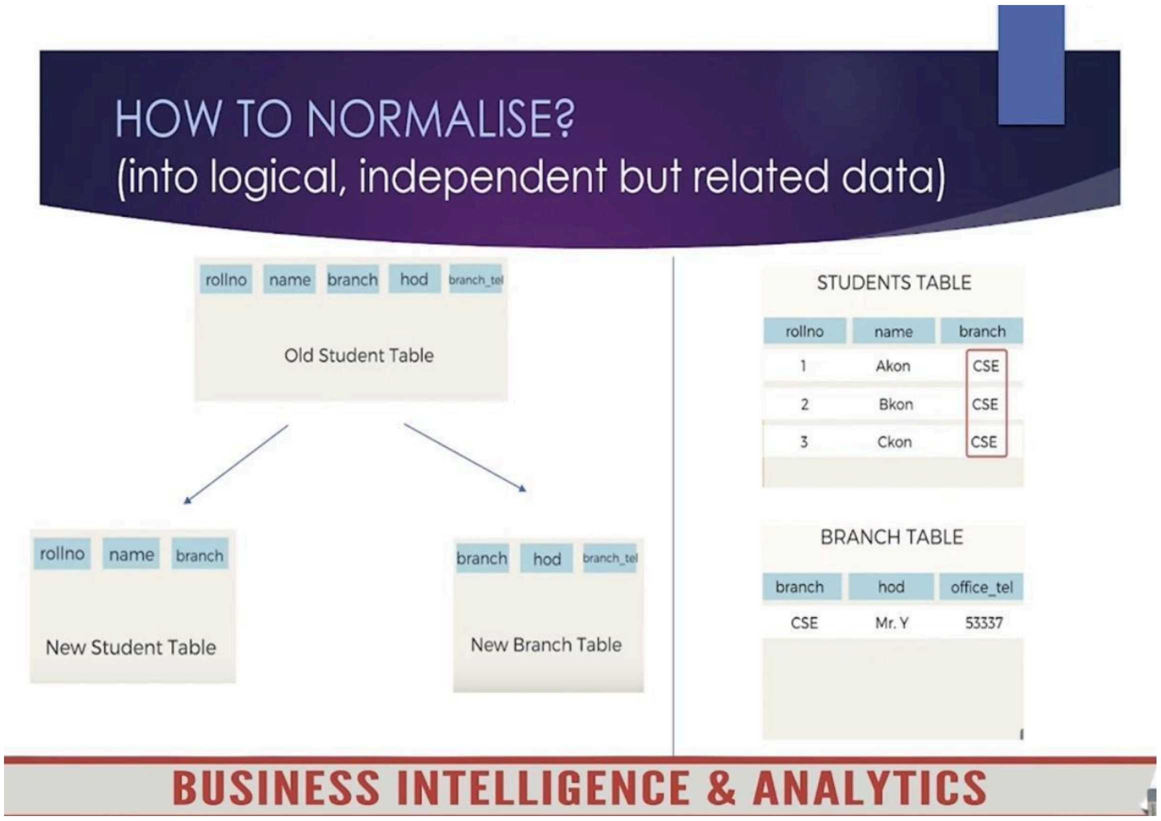
BUSINESS INTELLIGENCE & ANALYTICS

If there are say 1000 students, we have to go and change in each and every record that Mr. X has changed to Mr. Y which is kind of a headache. So, even if one row gets missed out, then it becomes a modification anomaly or updation anomaly. So, what is the solution for this anomalies or data redundancy problems? That is what is known as normalization.

So, normalization is the solution for avoiding all these data anomalies, I mean data redundancy anomalies. So, by normalization what we mean is, we split these initial table into logical independent but related tables or data. For example, we had this old student table wherein the branch HOD name, branch telephone number, everything was getting repeated along with each student's records. So what we do is, we split that old student table to new two tables. First table is student table and next table is branch table.

So if we mention that branch name, HOD name and branch telephone number once in that new branch table itself, we can just reference it to the new student table whenever we need it. So, see the right side. This is our normalized form. So the first table is student's table. In student's table, we have roll number, name and branch.

So say, Akon is from CSE. So what we will do is, CSE will be the primary key in the branch table. So whenever we want to get the branch details of Akon, we will just query using the CSE as the foreign key. So that will go and fetch all the details from the branch table. So suppose if you want to know the HOD's name, what we will do of Akon, what you will do is, you can just perform a simple query wherein you can reference the branch name as CSE and take the information from the branch table, so that you know that is performed very effectively.



HOW TO NORMALISE?
(into logical, independent but related data)

STUDENTS TABLE

rollno	name	branch
1	Akon	CSE
2	Bkon	CSE
3	Ckon	CSE

BRANCH TABLE

branch	hod	office_tel
CSE	Mr. Y	53337

BUSINESS INTELLIGENCE & ANALYTICS

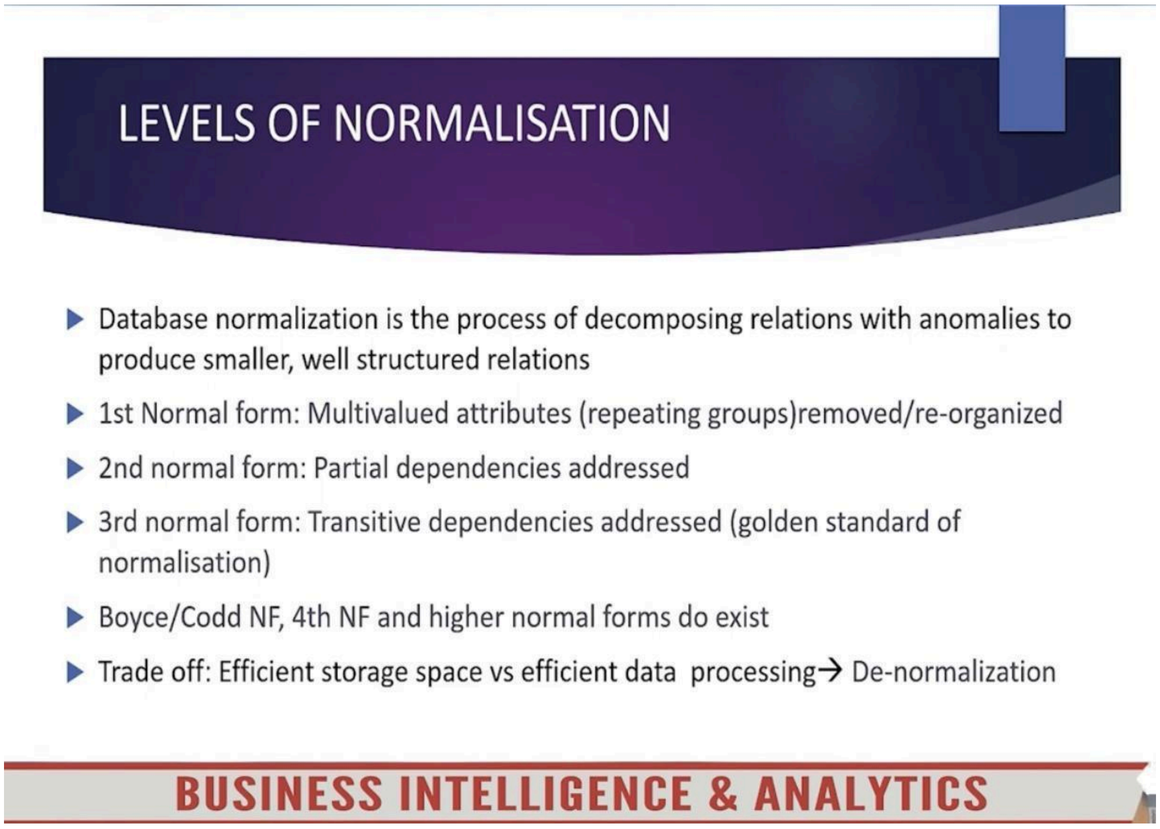
 So in database normalization, what we are doing is, we are splitting the relations or tables in multiple smaller tables with well-structured relations. So what is database normalization? It is a process of decomposing relations with anomalies to produce smaller and well-structured relations. So basically, what we are doing is, decomposing a big table into multiple smaller tables. So there are multiple normal forms as we call, multiple stages of decomposition.

So the first normal form says that there should not be any multivalued attributes.We will see all these in details. I am just reading it out as of now. The second normal form states that all partial dependencies should be addressed. Third normal form states that all transitive dependencies should be addressed or there should be no transitive dependencies. So there are multiple higher forms of normalization also, but we are not going into all those.

I think we will stop with Boyce-Codd normalization or which is known as 3.5NF. So it is intermediary between third and fourth normal form. So if you are interested in learning about the higher normal forms, you can go back and read but we will be discussing first, second, third and 3.5 which is also known as Boyce-Codd normal form.

So we will be seeing that in the next slide. Before that, I would like to ask you, what is the trade-off for a normalized data as well as a denormalized data? Why are we actually normalizing the tables? Just take a minute and you know, come with an answer of why we are actually doing the process of normalization. Normalization is actually performed in order to efficiently utilize the storage space. But do not you think referencing multiple tables, it will take so much time and data processing will be becoming very lengthy if we decompose this into multiple tables? You guessed it right.

That will happen. That is why there is a trade-off between normalization and denormalized data. So while normalization efficiently utilizes the storage space, denormalized data, it will perform the data processing more efficiently. So it is just a trade-off and you have to decide how much to normalize and where to stop. So the first form of normalization or what we call is 1NF. That is what we are going to look now.



LEVELS OF NORMALISATION

- Database normalization is the process of decomposing relations with anomalies to produce smaller, well structured relations
- 1st Normal form: Multivalued attributes (repeating groups)removed/re-organized
- 2nd normal form: Partial dependencies addressed
- 3rd normal form: Transitive dependencies addressed (golden standard of normalisation)
- Boyce/Codd NF, 4th NF and higher normal forms do exist
- Trade off: Efficient storage space vs efficient data processing→ De-normalization

BUSINESS INTELLIGENCE & ANALYTICS

So as I already told, in first NF, there should not be any multivalued attributes. So if your database is not even in the form of 1NF, then we call it a poor database design. So at least the minimum criteria for a proper database is, it should be at least normalized in 1NF format. So what is 1NF? 1NF states that there should not be any multivalued attributes. For example, each record or each field, so there will be multiple records under each field.

Say the column 2 in the table that is having 2 values, that is X and Y. So if our table should be in 1NF format, that should not be the case. It should be either X or either Y. There should not be any multivalued attributes. So what we do is, see the table below.

The first one is student's table. So in student's table, the subject column, it has multivalued attributes. For Akon, the subjects he has taken, it is 2 subjects. So both are mentioned in the same field, same record.

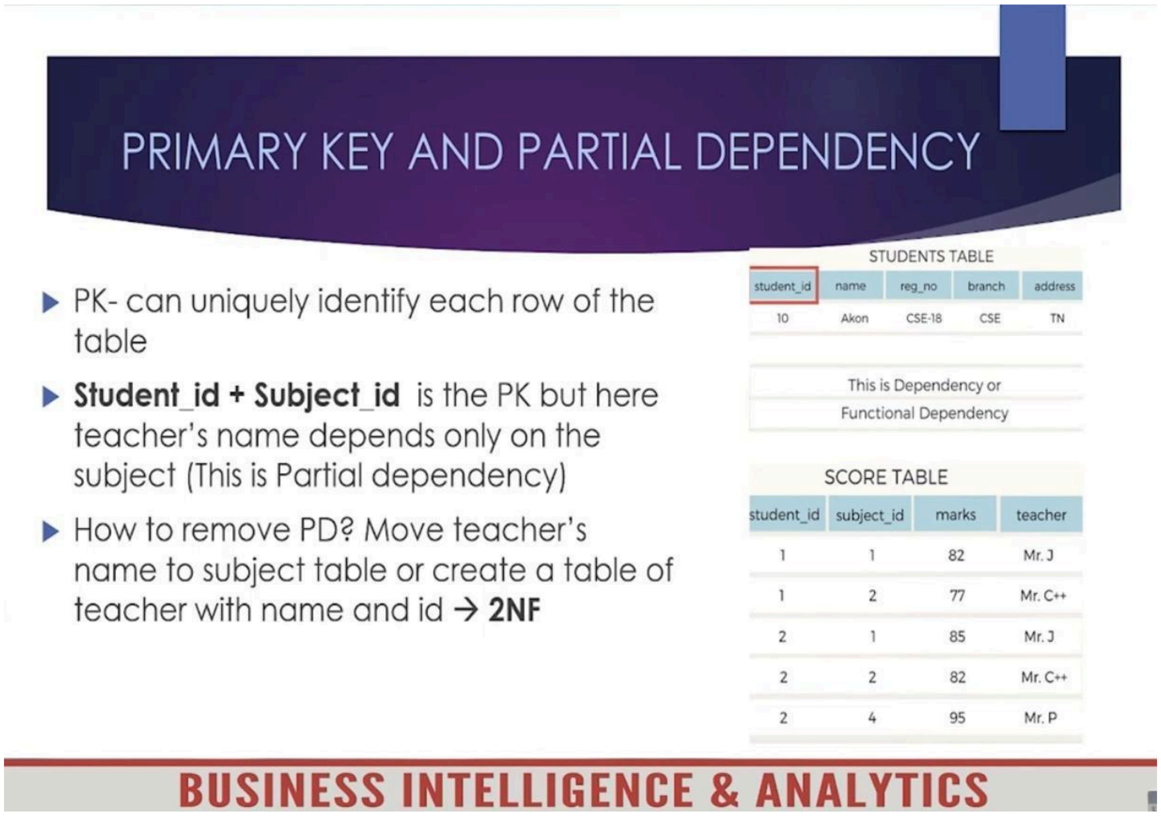
So that should not be the case. We have to split it into 2 records. That is how we normalize in the first normal form. So the normalized table is there in the right side, which is we are splitting 101 Akon OS and the second row is 101 Akon CN. So this is how we normalize a table in 1NF. So if a table is in 1NF, all columns should have values of the same type.Each column should have a unique name and the order in which you store the data, it does not matter. So basically, there should be no multivalued attributes.

We will go into 2NF. So for every advanced NF format that we are going, the previous NF or the previous normal form should already be satisfied. For example, if we are going to normalize in 2NF, the 1NF should already be satisfied.

So for a table to be in second normal form, it should already be in first normal form and the second criteria is that it should not have any partial dependency. You will be wondering, what is a partial dependency. So partial dependency exists when for a composite primary key, any attribute in the table depends only on a part of the primary key and not on the complete primary key. What does this mean? We will see this example in which I will show what is partial dependency. We already discussed what a primary key is.

Primary key is a key which can uniquely identify each row of the table. For example, in the student table, we can say the score table. We can take student ID plus subject ID as the primary key because if you take the student ID, it is having repeating the same values. It cannot uniquely identify the whole table. Similarly, with subject ID, it is also repeating.

There are two 1s and two 2s. So it is also not able to uniquely identify the table. So what we do is we combine both student ID as well as the subject ID and combine it to make it the primary key of the score table. So that is what we have done. But see the teacher's name. Teacher's name is depending only on the subject ID. Teacher is dependent on what subject he or she teaches. It is not dependent on who he teaches. There will be multiple students. So teacher's name is not dependent on student name.



PRIMARY KEY AND PARTIAL DEPENDENCY

- PK- can uniquely identify each row of the table
- **Student_id + Subject_id** is the PK but here teacher's name depends only on the subject (This is Partial dependency)
- How to remove PD? Move teacher's name to subject table or create a table of teacher with name and id → **2NF**

STUDENTS TABLE

student_id	name	reg_no	branch	address
10	Akon	CSE-18	CSE	TN

This is Dependency or
Functional Dependency

SCORE TABLE

student_id	subject_id	marks	teacher
1	1	82	Mr. J
1	2	77	Mr. C++
2	1	85	Mr. J
2	2	82	Mr. C++
2	4	95	Mr. P

BUSINESS INTELLIGENCE & ANALYTICS

It is just dependent on subject ID. So we have the primary key which is subject ID plus student ID and there is the teacher who depends only on the subject ID and not on the student ID. So teacher is depending only on the part of the primary key and not on the whole of the primary key. So what is this called? This is what is known as partial dependency and this partial dependency should not be there if we want to normalize the table in the form of 2NF.

So what is a dependency? I just forgot to tell before this. So what is a dependency? Before discussing dependency, we will see what are primary and non-primary attributes or what we call as prime and not prime attributes.

Prime attributes are those attributes which are present in the primary key and non-prime attributes are the attributes that are present in the, that are not present in the primary key. For example, see the score table. What is the primary key? Student ID plus subject ID is the primary key. So student ID is a prime attribute. Subject ID is a prime attribute. And what about marks and teacher? Both are non-prime attributes. So a functional dependency is when a non-prime attribute completely depends on the primary key. That is known as a full functional dependency. But the teacher is depending here only on the subject ID and not on the student ID. So the teacher field is partially dependent on primary key and not fully dependent on primary key. So this is known as partial dependency.

So as I discussed, partial dependency should not be there if we want to normalize our tables in the form of 2NF. What we will do? How do we normalize usually? You guessed it right. We will move this teacher's name and create it as a separate table. So that is what we do. So to remove the partial dependency, we can divide the table, remove the attribute which is causing the partial dependency and move it to some other table where it is fitting well. So that is what we do. So that is the case of partial dependency and 2NF.

Now we will move on to 3NF. So just repeat in your mind what I have taught till now. First one was 1NF. So in 1NF, there should not be any multivalued attributes. In 2NF, there should not be any partial dependency. Now we have come to 3NF. 3NF means what? It is an advanced form of 2NF. So what did I say earlier? It should already be in 2NF in order to achieve 3NF. So the first criteria for 3NF is, it should already be in 2NF. Second criteria is that it should not have any transitive dependency.

So we will see what is a transitive dependency. Transitive dependency is when there is an attribute in a table which depends on a non-prime attribute and not on a prime attribute. So I already discussed what a prime attribute is. Prime attribute is nothing but those attributes that are part of a primary key. So non-prime attributes, they are not part of the primary key.

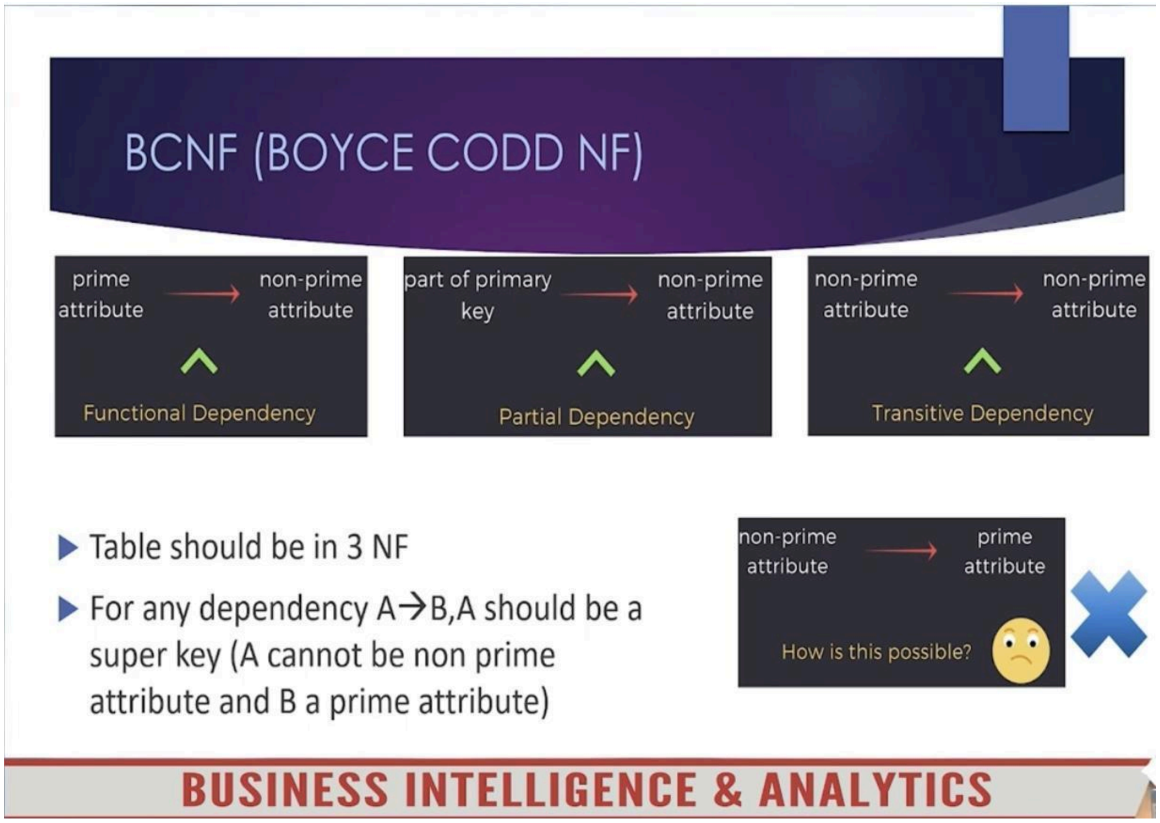
So see the score table in the left side. What are the primary keys? Primary key is a combination of student ID plus subject ID. We have combined both because each one was not able to uniquely identify any row in that table. So we have combined both. And see the total marks.

In our school time, there will be so many exams. There will be practicals, there will be sessionals, there will be end term exams. So the total marks, say end term would be 100 marks, practical would be 20 marks. So the marks dependent on the exams name. So there is a column in the score table called marks which is dependent on exam name alone. Is the marks dependent on student ID or the subject ID? No, right.

So total marks which is a field in the score table, it is a non-prime attribute and it is dependent on another non-prime attribute. It is not even depending on prime attribute.

So that is what is known as transitive dependency because total marks is depending on exam name and exam name is depending on the primary key. So it is kind of a transitive dependency that is happening and it is not directly depending on the primary key. So again what we do? How do we normalize it? We will just cut it off from the, you know, we just form a new table with just exam name and total marks and then reference it to the main tables or score table.

So that is how we usually do. So we form a separate exam table with the columns exam name and total marks. So the main idea is that all non-prime attributes should be depending only on prime attributes. So in transitive dependency, it is all non-prime attributes are depending on a non-prime attribute, which should not be the case. That is why we are normalizing it.



BCNF (BOYCE CODD NF)

prime attribute → non-prime attribute
Functional Dependency
part of primary key → non-prime attribute
Partial Dependency
non-prime attribute → non-prime attribute
Transitive Dependency
non-prime attribute → prime attribute
How is this possible?

- Table should be in 3 NF
- For any dependency A→B,A should be a super key (A cannot be non prime attribute and B a prime attribute)

BUSINESS INTELLIGENCE & ANALYTICS

I hope you are not that much confused because we are going to 3.5 normal form or what is known as Boyce-Codd normal form. Before going into Boyce-Codd normal form, I will just like to revisit what I have already discussed before this. So I taught what is functional dependency. What was a functional dependency? Functional dependency is when a non-prime attribute is dependent on prime attribute alone. So that is what is known as functional dependency.

What is a partial dependency? Partial dependency is a case where a non-prime attribute is dependent on only a part of primary key and not on the whole of primary key. So it is just depending on a very small part of the primary key. So that is what is partial dependency and in which normal form was partial dependency to be removed? Yes, it is second normal form. Then the third one that we discussed is transitive dependency in which a non-prime attribute is dependent on another non-prime attribute. So a non-prime attribute is deriving another non-prime attribute which should not be the case.

So every non-prime attribute should be dependent on primary key or the prime attribute. So that was the problem with the transitive dependency and in third normal form, transitive dependency we have normalized and it was not allowed. So what is Boyce-Codd normal form or what is known as 3.5NF? What are the criteria for that? Basically the first criteria is that the table should already be in 3NF and the next one is for any dependency A derives B, A should be a super key.

That is A cannot be non-prime attribute and B being a prime attribute. Now you will be getting confused. How can a prime attribute depend on a non-prime attribute? It is always the opposite case. How can a prime primary key or a prime attribute be dependent on some other non-prime attribute? So that is what is to be avoided in BCNF normal form.

So let us take an example in this. Here we have a college enrollment table with 3 fields which are student ID, subject and professor. So what is the primary key here? It is student ID plus subject which is the primary key here. Suppose in this example I want you to assume that one professor is teaching only one subject. Let us say that Saji Mathew sir is teaching only business intelligence and analytics in this class. So professor is dependent on, I mean subject is dependent on what? If I tell you I want to know this professor who is teaching BIA, then you would obviously answer it, it is Saji sir.

So it does not need any additional information. That means that subject can be derived from the professor name itself. So subject is a prime attribute but this prime attribute can be derived from the non-prime attribute which is the professor. So that is when the Boyce-Codd normal form kicks in. So it should not be the case. There should not be a case where a prime attribute derives itself from the non-prime attribute.

So as you guessed, what we are going to do? We are going to normalize it such that such kind of discrepancies are avoided. So we split this college enrollment table into two tables which is student table and professor table. So in student table, there will be student ID as well as professor ID which is a foreign key because professor ID is the primary key in the professor table. So there will be two tables, student table and professor table. Student ID is the primary key in student table and PID is the primary key in professor table.

So whenever we want to know the professor name, we can just do that using student ID itself. Say we can retrieve the professor name of the student whose ID is 3. So what we do is we see the corresponding professor ID and go and get the data from the professor table itself.

So that is how we normalize in BCNF form or what is known as 3.5 normal form. The next topic is we will be seeing a case, which is a Shopsense retail case and then we will be implementing some of the things that we have already discussed. Thank you.