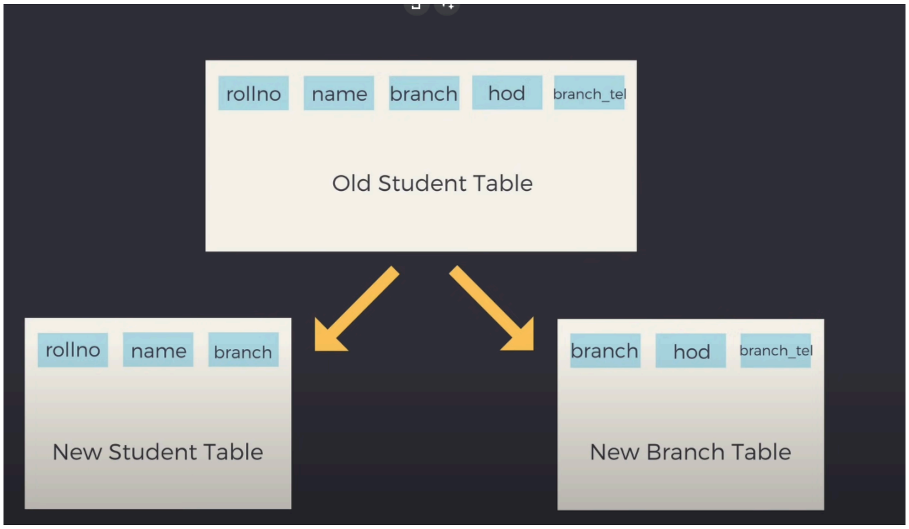
What is normalization :

It is a technique used in relational databases to reduce redundancy of data. It involves the breaking down of a table into multiple tables to achieve minimization of data redundancy.

Data redundancy - repetition of similar data at multiple places.
It increases the size of the database and therefore unnecessary usage of memory

And it makes it difficult for insertion deletion and updating an element.



Not just about eliminating data redundancy but normalization is also about minimization of data redundancy

Because of normalization , insertion, updation and deletion anomaly are also reduced.

This proves that normalization follows Divide and Rule. Logical, Independent but related datas

3 Basic normalization techniques exist :

1st normal
2nd
3rd

And special case boyce codd normal form.

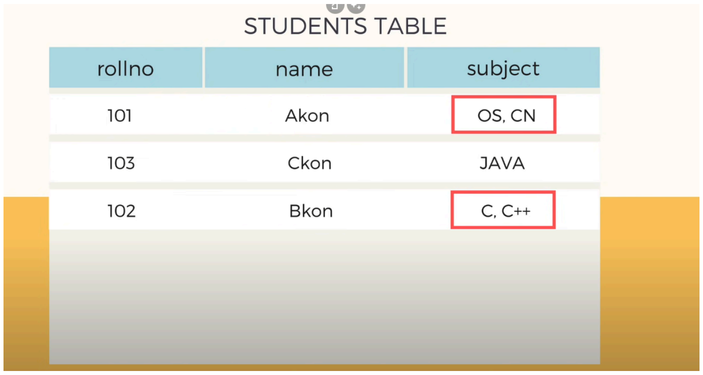
1st Normal form :

1nf expects the user to create a table in such a way that it can easily be extended
And must make sure that it is easy to retrieve data from the table.

A database is said to be poor if a table is not in 1nf. Therefore every table in a db should atleast follow 1nf. This is also the first step in normalization.

There are 4 basic rules that must be followed

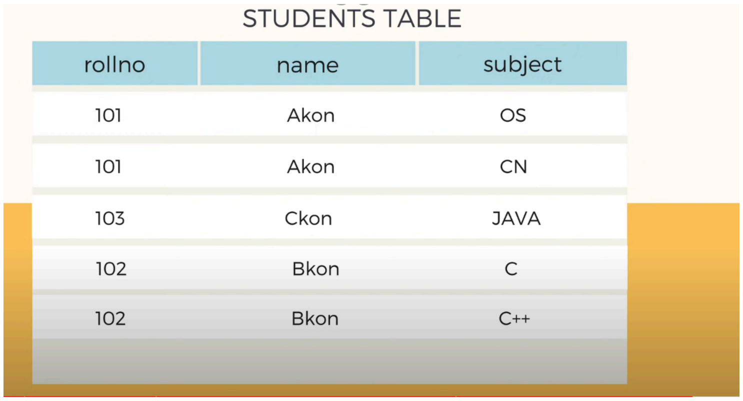
1. Each column of a table should contain only atomic/single values
2. A column should contain values of the same data type.
3. Each column should have a unique name
4. Order in which data is stored in the table does not matter.



STUDENTS TABLE

rollno	name	subject
101	Akon	OS, CN
103	Ckon	JAVA
102	Bkon	C, C++

After 1nf rules :



STUDENTS TABLE

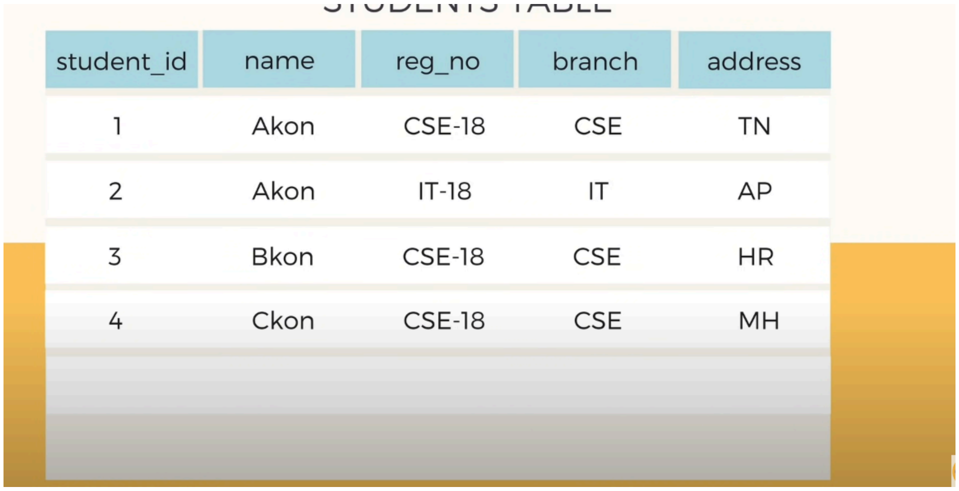
rollno	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	JAVA
102	Bkon	C
102	Bkon	C++

2nd Normal Form(2nf):

Only 2 rules for a table to be in 2nf

1. It should be in 1nf
2. There should not be any partial dependencies

To understand partial dependency ,
We'll look at what dependency is,



STUDENTS TABLE

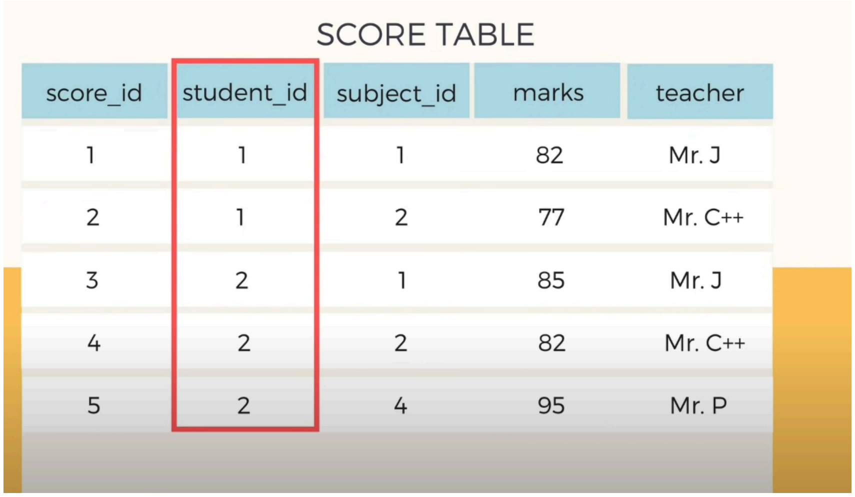
student_id	name	reg_no	branch	address
1	Akon	CSE-18	CSE	TN
2	Akon	IT-18	IT	AP
3	Bkon	CSE-18	CSE	HR
4	Ckon	CSE-18	CSE	MH

In this example we're able to retrieve the information of any student we want we just the student_id which is the primary key of the table. Every information is the table can be retrieved using student_id

This is known as functional dependency.

Partial Dependency :

To understand what pd is , lets consider the following table score



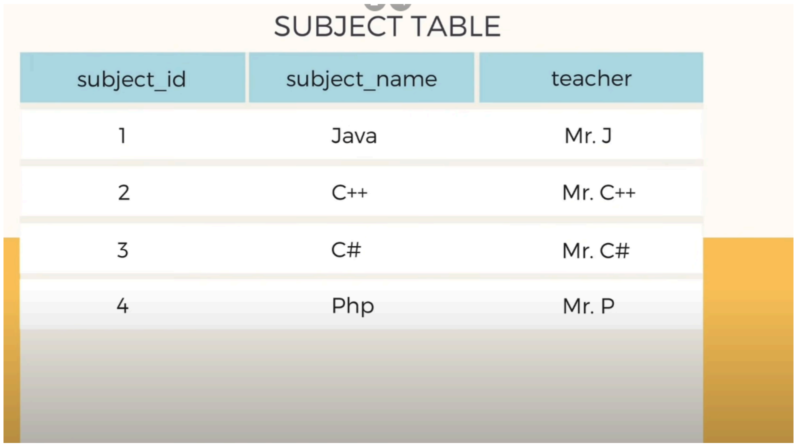
SCORE TABLE

score_id	student_id	subject_id	marks	teacher
1	1	1	82	Mr. J
2	1	2	77	Mr. C++
3	2	1	85	Mr. J
4	2	2	82	Mr. C++
5	2	4	95	Mr. P

We can easily say score_id is the primary key of this table as it uniquely identifies the table. But the student_id and subject_id together makes more sense for a primary key. The student_id or the subject_id alone cannot be the primary key here.

But we can see that the column teacher depends only on the column subj_id and not on student_id

This is known as partial dependency. To eliminate such a dependency , we can add the teacher column to the subject table as follows



SUBJECT TABLE

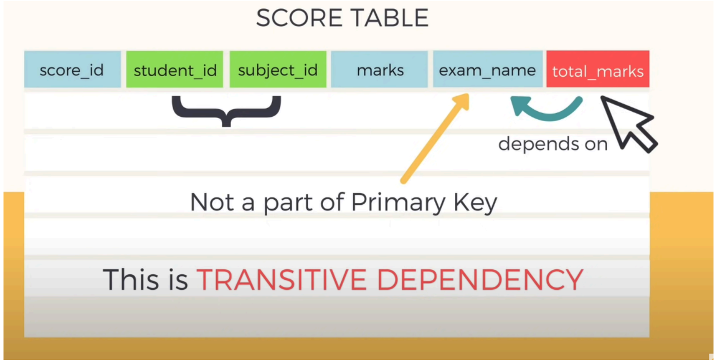
subject_id	subject_name	teacher
1	Java	Mr. J
2	C++	Mr. C++
3	C#	Mr. C#
4	Php	Mr. P

In simpler words : 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.To remove Partial dependency, we can divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.

3NF :

Rules :

1. For cnf , the table should be in 2nf.
2. Should not have transitive dependency



SCORE TABLE

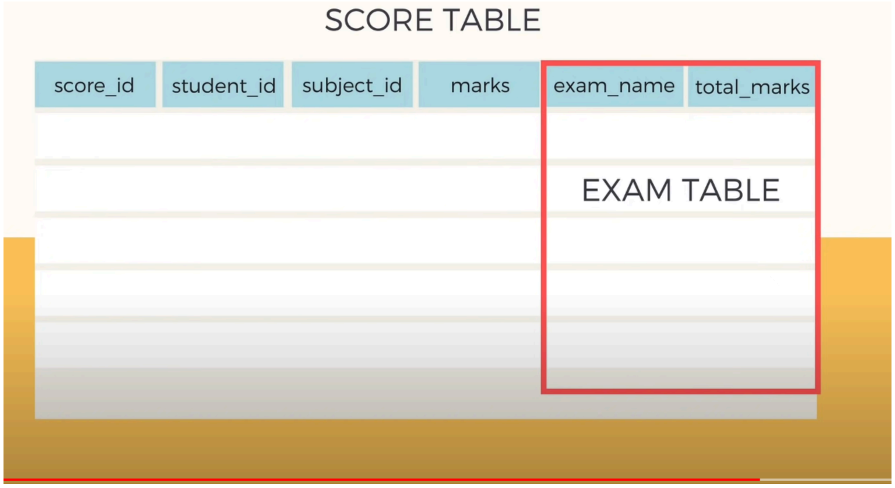
score_id	student_id	subject_id	marks	exam_name	total_marks

depends on

Not a part of Primary Key

This is TRANSITIVE DEPENDENCY

Soln :



SCORE TABLE

score_id	student_id	subject_id	marks	exam_name	total_marks

EXAM TABLE

As the data requirement increases, complexity of database increases and hence the need for normalisation also increases

4. BCNF

1. Should be of 3nf
2. Should have A -> B and A is a superkey

Differences between RDBMS AND NOSQL

RDBMS is more mature, well tested and has been widely implemented all around the world. HSBC was using the ibm db2 earlier. But because they wanted to put their core applications in helping to serve everyone across the globe , they switched to mongodb.

This is because each country has their own application requirements and not just one type of an application requirement will be able to solve the requirements of all countries.

With rdbms, frequent changes had to brought to the application to satisfy the needs of different countries but with the introduction of mongodb’s semistructured document data, they are able to serve countries more faster and more efficiently.

RDBMS : SQL - structured query language

1. Works on tables.
2. Designed using a specific schema
3. Every value in a table is related to each other.
4. Structured AF
5. Main feature of rdbms is that it supports transaction
6. Acid properties are there in dbms therefore used by many companies
7. Consistency of data is >>>>
8. Rdbms supports only vertical scaling which means that rather than improving the db, money well spent on machines to improve the efficiency of the db
9. RDBMS is not cloud ready although it looks like that.

What is Normalization?

Normalization in database design is the process of organizing data to minimize redundancy and improve data integrity. It involves decomposing a table into smaller tables and defining relationships between them. The primary goals are to eliminate redundant data, ensure data dependencies make sense, and make the database more efficient.

There are several normal forms (NF), each with specific requirements. Here are the first three normal forms, which are most commonly used:

First Normal Form (1NF)

A table is in 1NF if:

- All the columns contain atomic (indivisible) values.
- Each column contains values of a single type.
- Each column contains a unique name.

- The order in which data is stored does not matter.

Example: Let's say we have a `Students` table:

StudentID	Name	Subjects
1	John Smith	Math, Science
2	Jane Doe	Math, English
3	Sam Brown	Science, History

This table is not in 1NF because the `Subjects` column contains multiple values. To convert it to 1NF, we need to ensure each column contains atomic values.

1NF Table:

StudentID	Name	Subject
1	John Smith	Math
1	John Smith	Science
2	Jane Doe	Math
2	Jane Doe	English
3	Sam Brown	Science
3	Sam Brown	History

Second Normal Form (2NF)

A table is in 2NF if:

- It is in 1NF.
- All non-key attributes are fully functional dependent on the primary key.

Example: Consider the following `Students` table which is already in 1NF:

StudentID	Name	Subject	Teacher
1	John Smith	Math	Mr. Johnson
1	John Smith	Science	Ms. Clark

2	Jane Doe	Math	Mr. Johnson
2	Jane Doe	English	Mrs. Smith
3	Sam Brown	Science	Ms. Clark
3	Sam Brown	History	Mr. Adams

Here, the table is in 1NF but not in 2NF because `Teacher` depends on `Subject` and not on `StudentID`. We should split this table into two tables to achieve 2NF.

2NF Tables:

`Students` Table:

StudentID	Name
1	John Smith
2	Jane Doe
3	Sam Brown

`Subjects` Table:

SubjectID	Subject	Teacher
1	Math	Mr. Johnson
2	Science	Ms. Clark
3	English	Mrs. Smith
4	History	Mr. Adams

`StudentSubjects` Table:

StudentID	SubjectID
1	1
1	2
2	1
2	3

3	2
3	4

Third Normal Form (3NF)

A table is in 3NF if:

- It is in 2NF.
- There are no transitive dependencies, meaning no non-key attribute depends on another non-key attribute.

Example: Suppose we have the `Subjects` table from 2NF:

SubjectID	Subject	Teacher
1	Math	Mr. Johnson
2	Science	Ms. Clark
3	English	Mrs. Smith
4	History	Mr. Adams

If we include `TeacherPhone` in this table:

SubjectID	Subject	Teacher	TeacherPhone
1	Math	Mr. Johnson	123-456-7890
2	Science	Ms. Clark	234-567-8901
3	English	Mrs. Smith	345-678-9012
4	History	Mr. Adams	456-789-0123

This table is in 2NF but not in 3NF because `TeacherPhone` depends on `Teacher`, not directly on `SubjectID`. To achieve 3NF, we should separate the teacher's information into another table.

3NF Tables:

`Subjects` Table:

SubjectID	Subject	TeacherID
1	Math	1
2	Science	2
3	English	3
4	History	4

`Teachers` Table:

TeacherID	Teacher	TeacherPhone
1	Mr. Johnson	123-456-7890
2	Ms. Clark	234-567-8901
3	Mrs. Smith	345-678-9012
4	Mr. Adams	456-789-0123

These are the basics of normalization up to the third normal form, ensuring that data is logically stored without redundancy and maintaining data integrity.

4o
so 2nf should not have partial dependencies ?
ChatGPT

That's correct. In 2NF, there should be no partial dependencies. This means that all non-key attributes must be fully functionally dependent on the entire primary key. If a table has a composite primary key (a primary key consisting of multiple columns), then every non-key attribute must depend on all parts of this composite key, not just some of them.

Here's a clearer example to illustrate this concept:

Example: Partial Dependencies

Consider a table `Orders`:

OrderID	ProductID	ProductName	Quantity	UnitPrice
1	101	Pen	10	1.00
1	102	Notebook	5	2.00

2	101	Pen	20	1.00
2	103	Pencil	15	0.50

In this table:

- `OrderID` and `ProductID` together form the composite primary key.
- `ProductName` and `UnitPrice` depend only on `ProductID`, not on the combination of `OrderID` and `ProductID`.

This situation is an example of partial dependency. To eliminate partial dependencies and bring the table to 2NF, we need to separate the `ProductName` and `UnitPrice` into a different table.

2NF Tables

`Orders` Table:

OrderID	ProductID	Quantity
1	101	10
1	102	5
2	101	20
2	103	15

`Products` Table:

ProductID	ProductName	UnitPrice
101	Pen	1.00
102	Notebook	2.00
103	Pencil	0.50

Now:

- The `Orders` table contains information about orders and the quantity of each product in the order.
- The `Products` table contains information about products and their prices.

This separation ensures that non-key attributes (like `ProductName` and `UnitPrice`) are fully dependent on the primary key of their respective tables (`ProductID` in the `Products` table) and there are no partial dependencies in the `Orders` table. Thus, the tables are now in 2NF.

1/2
4o
can you explain about bcnf and 4nf
ChatGPT

Boyce-Codd Normal Form (BCNF)

BCNF is a stricter version of the Third Normal Form (3NF). A table is in BCNF if:

- It is in 3NF.
- For every functional dependency X→YX \rightarrow YX→Y, XXX should be a super key.

A super key is a set of one or more columns that can uniquely identify a row in a table.

Example: Consider a `Course` table:

CourseID	Instructor	Department
C1	Prof. A	CS
C2	Prof. B	EE
C3	Prof. C	ME
C4	Prof. A	CS

Functional dependencies:

1. CourseID→InstructorCourseID \rightarrow InstructorCourseID→Instructor
2. Instructor→DepartmentInstructor \rightarrow DepartmentInstructor→Department

In this table, the primary key is `CourseID`. However, `Instructor` also determines `Department`, which means `Instructor` should also be a candidate key. To bring this table to BCNF, we need to decompose it.

BCNF Tables:

`Courses` Table:

CourseID	Instructor

C1	Prof. A
C2	Prof. B
C3	Prof. C
C4	Prof. A

`Instructors` Table:

Instructor	Department
Prof. A	CS
Prof. B	EE
Prof. C	ME

Now, both tables are in BCNF because there are no non-trivial functional dependencies where the determinant is not a super key.

Fourth Normal Form (4NF)

A table is in 4NF if:

- It is in BCNF.
- It has no multi-valued dependencies (MVDs).

A multi-valued dependency occurs when one attribute in a table uniquely determines another attribute, independently of all other attributes.

Example: Consider a `Students` table:

StudentID	Course	Hobby
1	Math	Chess
1	Math	Football
1	Science	Chess
1	Science	Football

2	Math	Reading
2	Science	Reading

In this table:

- `StudentID` determines both `Course` and `Hobby` independently.
- There is a multi-valued dependency: StudentID→→CourseStudentID \rightarrow \rightarrow CourseStudentID→→Course and StudentID→→HobbyStudentID \rightarrow \rightarrow HobbyStudentID→→Hobby.

To bring this table to 4NF, we should split it into two tables.

4NF Tables:

`StudentCourses` Table:

StudentID	Course
1	Math
1	Science
2	Math
2	Science

`StudentHobbies` Table:

StudentID	Hobby
1	Chess
1	Football
2	Reading

Now, both tables are in 4NF because there are no multi-valued dependencies in either table. Each table only has functional dependencies.