Rule: i) No partial functional dependency.

(i.e. NO non-prime Attribute should depends on partial candidate key.)

Prime Attribute

ii) It should be in 1NF (1st Normal Form).

candidate key : Minimal key that defines all attributes.

Prime Attribute : Any Attribute which is part of any candidate key.

Non-Prime Attribute : Any Attribute which is not the part of any candidate key.

Example: Let's understand via examples.

| User ID | Course ID | Course Fee. |
|---|---|---|
| 1 | CS101 | 5000 |
| 2 | CS102 | 2000 |
| 1 | CS102 | 2000 |
| 3 | CS101 | 5000 |
| 4 | CS102 | 2000 |
| 2 | CS103 | 7000 |

Here, user ID alone is not a candidate key bcz. One user can took many courses.

So (user ID, course ID) = candidate key.

* course Fee is not ~~depen~~ in the candidate key so it is non-prime Attribute.

So, user ID, course ID = candidate key.

course Fee = Non-Prime Attribute.

Here, course fee → course ID (course fee depends upon course ID)

But as the defⁿ says no non prime Attribute should depend upon partial candidate key. So Here the 2NF Rule Breaking.

So this table is not in 2nd Normal Form (i.e. 2NF).

Q How will we convert the table in 2NF ??

A we will create another table. We will seperate the table in two parts. like this

| user ID | Course ID |
|---|---|
| 1 | CS101 |
| 2 | CS102 |
| 1 | CS102 |
| 3 | CS101 |
| 4 | CS102 |
| 2 | CS103 |

| Course ID | Course Fee |
|---|---|
| CS 101 | 5000 |
| CS 102 | 2000 |
| CS 103 | 7000 |

In this setup we haven't store course Fee in multiple places like we previously did. So, this avoid Reduntancy and anomaly in our databases.

So, that's How we convert the table in 2nd NF. (2NF).

---

## 13. Thrid Normal Form (3NF)

Rule: i) NO non prime attribute should Depend upon another non prime Attribute.

ii) It should be in 2NF. (2nd Normal Form).

iii) It should be in 1NF (1st Normal Form).

* Let's see and understand this via help of Example

| Student NO. | Student Name | Student State | Student Country. |
|---|---|---|---|
| 101 | Ram | Haryana | India. |
| 102 | Ramesh | punjab. | India. |
| 103 | Suresh | punjab. | India. |

Now, First we will check that whether the table is in 2NF or not. and 1NF or NOT also.

* No Attribute contain multiple value, so it is in 1NF.

Student ID → Student name, Student state, Student country.

Student State → Student Country.

candidate key

Functional Dependency.

As the Def$^n$ of 2NF says no non prime Attribute should depend upon partial candidate key.

Here Non Prime Attribute = Student name, Student state, Student country.

Here no partial candidate key, candidate key is only one (i.e. student ID) so, Here NO non prime Attribut should depends upon the partial candidate key.

So the following function follows 2NF.

Now we will check Rule No. 1. of 3NF i.e. no non prime Attribute should Depend upon another Non - Prime Attribute.

Here we saw,

Student state → student country.

i.e. Non prime Attribute depends upon another non prime Attributs. So, the 3NF Rule is Breaking.

How to convert the table in 3NF. ??

We Split the table in 2 parts.

table 1 : student ID, Student Name, Student state.

table 2: Student state, Student Country.

**In General:** If a table is not in 3NF we split the table in the following way so that if there is a non prime attributes defining another non prime Attributes.

then we took these two non prime Attribute together in a table and we make a seperate table and that non-prime Attribute which is defining the other non-prime Attribute should be kept in the original table also.

Ex. from our previous Example

Table $t_1$ : Student ID, student Name, Student State.

Table. $t_2$ : Student state, student country.

Example : 02.

| Exam Name | Exam Year | Topper Name | Topper DOB. |
|---|---|---|---|
| ABC | 2016 | Abhishek | 07-08-2000 |
| BCD | 2017. | Anurag | 28-09-2000 |
| EFG | 2017 | Sunil | 03-05-2000 |
| ABC | 2017 | Rahul. | 25-07-2000 |

We have to make the table in 3NF.

**Rule : 1.** First check Whether it is in 1NF ??

Ans Yes the table is in 1NF bcz No attribute contain multiple values.

**Rule : 2 :** check whether it is in 2NF ??

Ans (Exam Name, Exam year) → Topper Name, Topper DOB.

(Exam Name, Exam year) ← candidate key.

TOPPER Name → TOPPER DOB.

2NF says it should not happen that a non prime Attribute is derived by part of a candidate key.

and from the above condition 2NF Rule is not violated.

So the table is in 2NF.

**Rule : 3.** Check for 3NF if it is yes then it's OK Otherwise make it in 3NF. (table).

TOPPER Name → TOPPER DOB.

Ans: Topper Name and TOPPER DOB is a non prime attribute and a non prime attribute depends On another non prime Attribute. So Here 3NF Rule is violated.

Q How do we fix it ??

Ans We will make two tables.

Table 1 will contain: Exam name, Exam Year, Topper Name.

Table 2 will contain: TOPPER Name, Topper DOB.

That's How we will fix the table in 3NF.

Abhishek Sharma
Notes

## Alternative Definition of 3NF.

i) Should be in 2NF and 1NF.

ii) Non-Prime Attributes are not transitively dependent on Prime Attribute.

This means

from ~~Armo~~ Armstrom Axioms, Rule of Transitivity is that

> if $A \rightarrow B$ & $B \rightarrow C$
> Then $A \rightarrow C$

means if a ~~Pri~~ prime Attribute —defines a→ non prime Attribute, then Non prime Attribute —defines a→ non-prime Attribute is not allowed.

Same definition in another way,

---

## Summarising Here All 3 Normal Forms.

**1 NF:** Any attribute should ~~Pos~~ contain only single value.

**2 NF:** Any Non Prime Attribute should not be a part of any ~~Pa~~ partial prime Attribute.

Not allowed. ✗ $P \rightarrow NP$ ✗ Not Allowed.

**3NF:** No non prime Attribute should not define any another non prime Attribute.

Not Allowed ✗ $NP \rightarrow NP$ ✗ Not allowed.

~~BCNF~~
BCNF ✗ $(P/NP) \rightarrow P$ ✗ Not Allowed (Study further).

# 14. BCNF (Boyce - codd Normal Form)

**Rule:** i) Both prime and Non-prime attributes they do not define a prime Attribute.

Not Allowed. ✗ | P / NP → P | ⇒ Not Allowed ✗.

ii) The table must follow 1NF, 2NF and 3NF. for BCNF.

**According to BCNF**

for any functional Dependency in your table

Let's suppose $x \rightarrow y$

conditions for BCNF.

i) Trivial OR.

ii) X is a Superkey.

(~~Left~~ hand side (only Superkeys on the left Hand side).

Let's understand through example.

| Student ID | Subject | Proffesor ID. |
|---|---|---|
| 1001 | DBMS | 103 |
| 1001 | OS | 110 |
| 1002 | DBMS | 111 |

Here, Student ID, subject → Proffesor ID.

Or Student ID, Proffesor → subject.

Proffesor ID → subject

In this example:

Candidate Key : i) student ID, Subject

ii) proffesor ID, student ID.

So, Here No non prime Attribute present.

So, 1NF Rule Followed.

So, 2nd NF Rule also followed

bcz. No non prime Attribute Here so we don't care further.

So 3rd NF Rule Also followed.

" " " " "

" "

Now we will check this table follows BCNF or not ??

No, It is not following BCNF bcz

BCNF says Prime or Non-prime attribute they do not define a prime Attribute.

But Here

Professor ID → Subject

↑ Prime ↑ Prime.

So, BCNF Fails.

How to convert the table in BCNF form. ??

We will split the table in 2 tables such that.

table 1 : Student ID , Professor ID.

table 2 : Proffesor ID, Subject.

| Stud. ID | Proff. ID |
|---|---|
| 1001 | 103 |
| 1001 | 110 |
| 1002 | 111 |

| Proff. ID | Subject |
|---|---|
| 103 | DBMS |
| 110 | OS |
| 111 | DBMS. |

Q) What is the Advantage of doing so ??

A) Suppose in the 1st table there are thousands of rows and Subject (string) written in thousand of time. So we have anomaly in Database.

But after BCNF we split the table in 2 table and Subject (String) type written only once.

Remember: BCNF decomposition (split into more table) might not be acceptable. bcz. it may loose functional dependency.

So decomposition of table to get the BCNF is not a good idea.

---

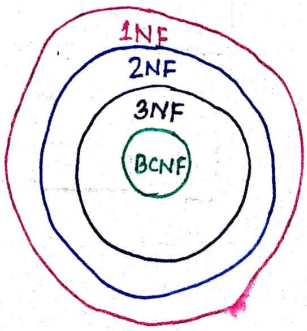
* BCNF is the most strict form

then 3NF then 2NF then 1NF

* Whatever in BCNF will also be in 3NF

Whatever in 3NF will also be in 2NF

Whatever in 2NF will also be in 1NF



# 15. Indexing in Database.

The whole idea of indexes is to improve performance of your Query.

## Let's Learn Indexing through an example:-

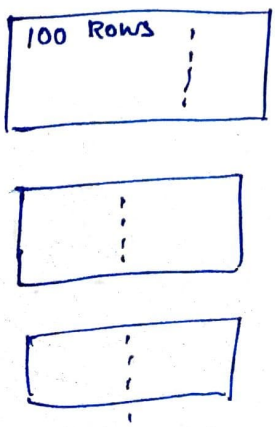
Suppose we have a E-commerse website Database. like this :—

| Order ID | order date | cost | Customer ID. |
|---|---|---|---|
| 101 | 15-06-19 | 2000 | 102 |
| ⋮ | ⋮ | ⋮ | ⋮ |

If we have minimum amount of order then we have a small database and we can handle easily but if we have a thousands of hundereds of orders then it will be really difficult to maintain it.

* All the data is saved in our Hard disk. like this in hard Disk we have blocks of rows. Like these are many blocks and each block can contain only 100 rows.

Disk block.




and so on.

If we search a particular customer ID. then we don't know where it is. So, we have to traverse through all the Disk block.

So Basic idea of indexing is to put an order in the data in your harddisk. and this particular indexing is called clustered indexing.

Clustered indexing :- To put order in your database storage in files.
↳ Data stored in increasing order in your disk block.
Examp: if I have to store data in the table and we have n orders then it will store like that.

The smallest order id will apper first like that.

| Order ID. |
|---|
| 101 |
| 102 |
| 103 |
| ⋮ |
| L |

This will give some Advantage. if I want to search any order id then we will do Binary search and find it.

Example.

| Order ID | order date | Cost | customer ID. |
|---|---|---|---|
| 101 | 15-06-19 | 2000 | 102 |
| 102 | 15-06-19 | 5000 | 103 |
| 105 | 15-06-19 | 6000 | 101 |
| 101 | 15-06-19 | 8000 | 102 |

↳ clustered indexing.

* If we don't do clustered index in our table then these all order. id can be stored in any order in table. and This type of organisation of your data is called Heap organisation. bcz. the tables are act like Heap.

Abhishek Bhaiya's Notes

* Most of the DATABASE serverses they Bydefault create clustered index on your primary key.

Prime Index: When primary key is used as an index

* prime index is also a clustered index.

* Remember we can create only one clustered index. bcz. there is only one way to ~~order~~ D. organise data in physical order.

Let's see one more example.

| Order ID | order Dat | cost | customer ID. |
|---|---|---|---|
| 101 | 15-06-2021 | 2000 | 102 |
| 102 | 15-06-2021 | 5000 | 103 |
| 105 | 15-06-2021 | 6000 | 101 |
| 106 | 15-06-2021 | 8000 | 102 |

This table is in clustered index. along order id.

But if we want to Arraye this table along customr id also then we can't do via clustered index.

Bcz. a table has only one culustered index.

So How can we arraye customr id table also.

This arraymment is called Non-Clustered indexing. Or Secondary indexing.

* The idea of clustered index and non clustered index is understood via example of Book.

## Book Example.

We saw the book. in book

i) All the chapters are assigned in the increasing order

like : chapter 1, chapter 2 ----- chapter n.

This is clustered indexing.

ii) But If we want to index a particular item of the book or topic of the book. we found it in the end of the book with their respective page no.

This is secondary index or non - clustered index.

So, chapter in the book are example of clustered indexing. and particular topic that is present in the book's last page with their page no. is the example of secondary indexing or non - clustered indexing.

That's How indexing Helps us to find a particular item in the table.

---

Day-3 Notes ended.

Abhishek Sharma.

02/09/2021