Normalization

-> It is technique to Remove or Reduce.
-> Redundancy from a table.

S.ID	S-name	Age
1	Ram	20
2	varun	25
1	Ram.	20

-> we can remove this by primary key concept.

Row level.

-> Course ID -> Course-Name. -> Faculty ID -> Faculty name

S.ID	S-name	C-ID	C-Name	F.ID	F-Name	Salary
1	Ram	C1	DBMS	F1	John	30000
2	Ravi	C2	Java	F2	Bob	40000
3	Nitin	C1	DBMS	F1	John	30000
4	Amrit	C1	DBMS	F1	John	30000

Coloumn level.

(i) Inseration Anomaly -> Problam occur at special occassion.
(ii) Deletion Anomaly
(iii) Updation Anomaly

Scanned with CamScanner

first Normal form (1NF)

→ Table should Not contain any multivalued Attribute.

↓ Not in 1st NF

Roll No.	Name	Course
1	Sai	C/C++
2	Harsh	Java
3	Onkar	C/DBMS.

→ 1st Solution.

Roll	Name.	Course
1	Sai	C
1	Sai	C++
2	Harsh	Java
3	Onkar	C
3	Onkar	DBMS

↳ Primary key = Course + Roll No.
↓
Composite primary key.

→ 2nd Solution

Roll	Name	Course 1	Course 2
1	Sai	C	C++
2	Harsh	Java	NULL
3	Onkar	C	DBMS

→ Primary key = Roll No.

3rd Solution

Roll No.	Name.
1	Sai
2	Harsh
3	Onkar

← Base table

Roll No.	Course
1	C
1	C++
2	Java
3	C
3	DBMS

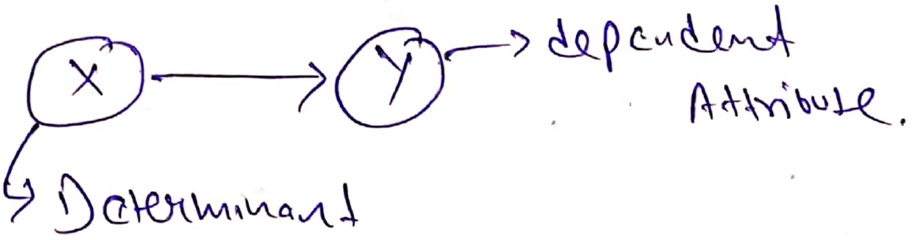
→ foreign key

→ Referntial table.

Scanned with CamScanner

⇒ Closure Method:

* functional dependency



→ X Determine Y or → Y is defined by X.

valid

Sid →	S.name.
1	Ranjit
2	varun.

valid

Sid →	S.name.
1	Ranjit
1	Ranjit.

Invalid

SID →	S-name.
1	Ranjit
1	varun.

valid

Sid →	S-name.
1	Ranjit
2	Ranjit

→ Trival FD : X → Y [Y is subset of X]

↳ They are alway true.

↳ X → Y

LHS ∩ R.H.S ≠ ϕ

→ Non-trivial FD:

x → Y x ∩ Y = ϕ

↳ we have to check the Condition.

Properties of FD :

Sid → Sidname.

→ Reflexivity : if Y is a Subset of X than. $X \rightarrow Y$

→ Augmentation → if $X \rightarrow Y$, than $XZ \rightarrow YZ$.

→ transitive : if $x \rightarrow Y$ and $Y \rightarrow Z$ than $X \rightarrow Z$.

**

Sid → Sname and Sname → City

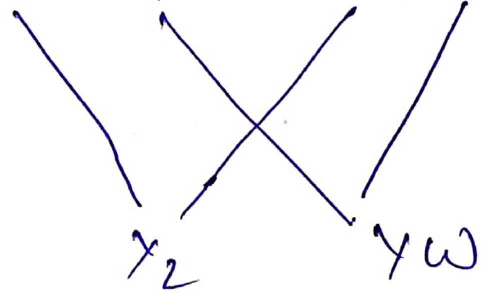
Sid → City.

→ Union : if $X \rightarrow Y$ and $X \rightarrow Z$ than $X \rightarrow YZ$

→ Decomposition : if $X \rightarrow YZ$ than $X \rightarrow Y$ and $X \rightarrow Z$.

→ Psedutransitivity : if $X \rightarrow Y$ and $W \rightarrow Z$.
than. $WX \rightarrow Z$

→ Composition : if $X \rightarrow Y$ and $Z \rightarrow W$ than $ZX \rightarrow YW$



→ Second Normal form:

→ (i) table or relation must be in 1st Normal form.

(ii) All the non-prime attribute should be fully funcional dependent on Candidate key.

Customber.

Cust. ID	Store ID	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Banglore
4	3	mumbai

Cadidate. key : Customber ID + Store ID.

Non- prime : Location.

2 NF

Store ID	Location
1	Delhi
2	Banglore
3	Mumbai

2 NF

Cust. ID	Store
1	1
1	3
2	1
3	2
4	3

(iii) There should be No partial dependency.

Scanned with CamScanner

* Third Normal form:

→ table or relation must be in 2nd Normal form

and

→ there should be No transitive dependency in table.

Roll No.	State	City.
1	Punjab	Mohali
2	Haryana	Ambala.
3	Punjab	Mohali
4	Haryana	Ambala.
5	Bihar	Patna.

F.D → Rool No. → State

sate → city.

* BCNF (Boyne. Codd Normal form):

→ table should be in 3rd Normal form.

→ L.H.S of each FD shood be Candiadte key or Super key.

Rollno	Name	voter ID	age.
1	Ravi	K0123	20
2	varun	M034	21
3	Ravi	K786	23
4	Rahul	J286	21

CK : ⟨ Voter ID : Rol No ⟩

~~CR~~ : { Roll No → name

F.D   Roll No → Vote ID

voter ID → age

voter id → Roll No. }

Scanned with CamScanner

NORMALIZATION & TYPES OF NORMALIZATION

1) DEFINE NORMALIZATION

Normalization can be defined as :-

- A process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.
- A process of organizing data into tables in such a way that the results of using the database are always unambiguous and as intended. Such normalization is intrinsic to relational database theory. It may have the effect of duplicating data within the database and often results in the creation of additional tables.

Types of normalization

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)

First Normal Form (1NF)

First normal form enforces these criteria:

- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary key

First Normal Form

Table_Product

Product Id	Colour	Price
1	Black, red	Rs.210
2	Green	Rs.150
3	Red	Rs. 110
4	Green, blue	Rs.260
5	Black	Rs.100

This table is not in first normal form because the “Colour” column contains multiple Values.

After decomposing it into first normal form it looks like:

Product_id	Price
1	Rs.210
2	Rs.150
3	Rs. 110
4	Rs.260
5	Rs.100

Product_id	Colour
1	Black
1	Red
2	Green
3	Red
4	Green
4	Blue
5	Black

Second Normal Form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

SECOND NORMAL FORM

Table purchase detail		
Customer_id	Store_id	Location
1	1	Patna
1	3	Noida
2	1	Patna
3	2	Delhi
4	3	Noida

- This table has a composite primary key i.e. customer id, store id. The non key attribute is location. In this case location depends on store id, which is part of the primary key.

After decomposing it into second normal form it looks like:

Table Purchase	
Customer_id	Store_id
1	1
1	3
2	1
3	2
4	3

Table Store	
Store_id	Location
1	Patna
2	Delhi
3	Noida

Third Normal Form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

THIRD NORMAL FORM

Table Student Details

Book_id	Genre_id	Genre type	Price
1	1	Fiction	100
2	2	Sports	110
3	1	Fiction	120
4	3	Travel	130
5	2	sports	140

- In the table, book_id determines genre_id and genre_id determines genre type. Therefore book_idd determines genre type via genre_id and we have transitive functional dependency.

After decomposing it into third normal form it looks like:

TABLE BOOK		
Book_id	Genre_id	Price
1	1	100
2	2	110
3	1	120
4	3	130
5	2	140

TABLE GENRE	
Genre_id	Genre type
1	Fiction
2	Sports
3	Travel

Boyce-Codd Normal Form (BCNF)

- It is an advance version of 3NF that’s why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Boyce-Codd Normal Form

Student	Course	Teacher
Aman	DBMS	AYUSH
Aditya	DBMS	RAJ
Abhinav	E-COMM	RAHUL
Aman	E-COMM	RAHUL
abhinav	DBMS	RAJ

- KEY: {Student, Course}
- Functional dependency {student, course} -> Teacher

Teacher-> Course

- Problem: teacher is not superkey but determines course.

After decomposing it into Boyce-Codd normal form it looks like:

Student	Course
Aman	DBMS
Aditya	DBMS
Abhinav	E-COMM
Aman	E-COMM
Abhinav	DBMS

Course	Teacher
DBMS	AYUSH
DBMS	RAJ
E-COMM	RAHUL

Fourth Normal Form (4NF)

- Fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key.

It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce-Codd Normal Form (BCNF). It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

FOURTH NORMAL FORM

Student	Major	Hobby
Aman	Management	Football
Aman	Management	Cricket
Raj	Management	Football
Raj	Medical	Football
Ram	Management	Cricket
Aditya	Btech	Football
Abhinav	Btech	Cricket

- Key: {students, major, hobby}
- MVD: ->-> Major, hobby

After decomposing it into fourth normal form it looks like:

Student	Major
Aman	Management
Raj	Management
Raj	Medical
Ram	Management
Aditya	Btech
Abhinav	Btech

Student	Hobby
Aman	Football
Aman	Cricket
Raj	Football
Ram	Cricket
Aditya	Football
Abhinav	Cricket

Fifth Normal Form (5NF)

A database is said to be in 5NF, if and only if,

- It's in 4NF.
- If we can decompose table further to eliminate redundancy and anomaly, and when we re-join the decomposed tables by means of candidate keys, we should not be losing the original data or any new record set should not arise. In simple words, joining two or more decomposed table should not lose records nor create new records.

FIFTH NORMAL FORM

Seller	Company	Product
Aman	Coca cola company	Thumps Up
Aditya	Unilever	Ponds
Aditya	Unilever	Axe
Aditya	Uniliver	Lakme
Abhinav	P&G	Vicks
Abhinav	Pepsico	Pepsi

- Key: {seller, company, product}
- MVD: Seller ->-> Company,

product Product is related to

company.

After decomposing it into fifth normal form it looks like:

Seller	Product
Aman	Thumps Up
Aditya	Ponds
Aditya	Axe
Aditya	Lakme
Abhinav	Vicks
Abhinav	Pepsi

Seller	Company
Aman	Coca cola company
Aditya	Unilever
Abhinav	P&G
Abhinav	Pepsico

Continued in next slide…

Company	Product
Coca cola company	Thumps Up
Unilever	Ponds
Unilever	Axe
Unilever	Lakme
Pepsico	Pepsi
P&G	Vicks

Thank You

Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. Let’s discuss about anomalies first then we will discuss normal forms with examples.

Anomalies in DBMS

There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly. Let’s take an example to understand this.

Example: Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp_id for storing employee’s id, emp_name for storing employee’s name, emp_address for storing employee’s address and emp_dept for storing the department details in which the employee works. At some point of time the table looks like this:

emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890

166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

The above table is not normalized. We will see the problems that we face when a table is not normalized.

Update anomaly: In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert anomaly: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

Delete anomaly: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

To overcome these anomalies we need to normalize the data. In the next section we will discuss about normalization.

Normalization

Here are the most commonly used normal forms:

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

First normal form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

This table is **not in 1NF** as the rule says “each attribute of a table must have atomic (single) values”, the emp_mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we should have the data like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	Subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}
Non prime attribute: teacher_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says “**no** non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table complies with 2NF we can break it in two tables like this:
teacher_details table:

teacher_id	teacher_age
111	38
222	38
333	40

teacher_subject table:

teacher_id	subject

111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form (2NF).

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan

1201	Steve	222999	MP	Gwalior	Ratan

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on
Candidate Keys: {emp_id}
Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.

Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

employee table:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008

1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam

292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

Boyce Codd normal form (BCNF)

It is an advance version of 3NF that’s why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100

1002	American	Purchasing department	D134	600

Functional dependencies in the table above:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: {emp_id, emp_dept}

The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:
emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning

1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate keys:
For first table: emp_id
For second table: emp_dept
For third table: {emp_id, emp_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.

Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology

59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Fifth normal form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Example

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen