Assignment

Topic: Normal forms & Join Operation

Name: Aman Yadav

Department: Computer Science & Information Tech.

Course : B.Sc. CS (Hons.) IIIrd Sem

Roll no. - 20207005

Enrollment - GGV/20/05105

Subject - Database Management System Theory.

(1). Assignment No-1

Explain 1NF, 2NF, 3NF & BCNF

1. First Normal form (1NF)

To understand normal forms we have to understand normalisation first.

Normalisation – Normalisation is the process of reducing redundancy. Redundancy in relation may cause insertion, deletion or update anomalies. So it helps to minimize the redundancy in relation. Normal forms are used to remove or reduce redundancy in database table.

First Normal form (1NF) –

If a relation contains a multi-valued attribute, it violates the first normal form or we can say that a relation is in first normal form if it doesnot contain any composite or multi-valued ~~te~~ attribute

A table is in 1NF iff -

1. There are only single valued attribute.
2. Attribute domains doesnot change
3. There will be a unique name for every attribute.
4. The order in which data is stored doesnot matter.

Example :–

Lets take two tables - Table 1 is not in INF because of multi-valued attribute.

Student No	SName	S.Phone	S. STATE
01	Aman Yadav	7000620333, 7223952707	CHHATTISGARH
02	Nilima Sahu	93400 90031	Delhi
03	Prerna Patnayak	7898624630	Maharashtra

Table - 1

(2)

student no	sname	S Phone	S STate
01	Aman Yadav	7000620333	Chhattisgarh
01	Aman Yadav	7223952707	
02	Nilima sahu	9340090031	Delhi
03	Prerna Patnayak	7898624630	Maharashtra

Table - 2. this is in first normal form.

2. Second Normal form (2NF) -

Second normal form is based on the concept of fonitional dependency. Second normal form applies to relation with composite keys. relations with a primary key composed of two or more attributes.

A relation that is in first normal form and every non-primary key attribute is fully functionally dependent on the primary key, then the relation is in second normal form (2NF).

If proper subset of candidate key determines non-prime attribute, then this is known as partial dependency.

The normalisation of 1NF to 2NF involves the removal of partial dependencies. If a partial dependency exist, we remove the partial dependent attribute(s), from relation by placing them in a new relation along with a copy of their dept.

example -

~~eno~~ Student no.	course no.	course fee
1	C_1	1000
2	C_2	1500
1	C_4	2000
4	C_3	1000
4	C_1	1000
2	C_5	2000

(3) In previous table there are many courses having same course fee.

course fee can't alone decide the value of course no. and student no.

also. course fee with student no. can't decide the value of course no.

and course fee with course no. can't decide the value of student no.

Therefore course fee would not be a prime attribute as it doesnot belong to one only candidate key {student no., course no.};

But course fee is dependent on course no which is proper subset of candidate key, Non-prime attribute course fee is dependent on proper subset of the candidate key, which is a partial dependency therefore this table-relation is not in 2NF.

To convert previous table into 2NF, we need to split the table such as; table-1 student no, course no

table-2 course no, course fee.

Student No	course No
1	C_1
2	C_2
1	C_4
4	C_1
4	C_3

course No	course fee
C_1	1000
C_2	1500
C_3	1000
C_4	2000
C_5	2000

2NF tries to reduce the redundant data getting stored in memory, If there are 100 student taking C_1 course, we don't need to store its fee as 1000 for all 100 records, instead once we can store it in the second table as the course fee for C_1 is 1000.

(4) Third Normal form (3NF) :-

A relation is in third form if there is no transitive dependency for non-prime attribute as well as it is in second normal form

A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency $X \rightarrow Y$.

1. X is a super key
2. Y is a prime attribute (each element of y is a part of some candidate key).

In other-words :-

A relation that is in first and second normal form and in which no non-primary key attribute is transistevely dependent on the primary key, then it is in third normal form.

If $A \rightarrow B$ and $B \rightarrow C$ are two FDs then $A \rightarrow C$ is called transitive dependency.

The normalisation of 2NF relations to 3NF involves the removal of transitive dependensive. If a transitive dependency. If a transitive dependency exist, we remove the transitively dependent attributes, from the relation by placing the attribute in a new relation along with a copy of the determinants.

Example :-

in relation student given table - 4.

SNO.	SName	sDist.	s.age	s.country
1	Aman Yadav	Janjgir	20	India
2	Nilima Sahu	Mahasamund	20	India
3	Prerna Patnayak	Bilaspur	20	India

FD set

- s.no → sname
- s.no → s.dist
- s.dist → s.country
- s.no → s.age

candidate key :

s.no

(5) for this relation in table 4.
s.no. → s.dist and s.dist → s.country are true. so s.country is transitively dependent on s.no. it violates the third normal form. To convert it in third normal form we will decompose the relation student as.

Student (s.no, s.name, s.age, s.dist).
dist.country (s.dist, s.country)

consider relation R(A, B, C, D, E)

A → BC, CD → E
B → D, E → A

All possible candidate key in above relation are {A, E, CD, BC} all attribute are on right side of all functional dependencies are prime.

Note— Third normal form (3NF) is considered as adequate for normal relation database design because most of the 3NF tables are free of insertion, update and deletion anomalies. Moreover, 3NF always ensures functional dependency preserving and lossless.

4. Boyce – Codd Normal Form (BCNF)

Boyce-codd normal form is based on functional dependencies that take into account all candidate key in a relation; however, BCNF also has additional constraints compared with the general defination of 3NF

A relation is in BCNF iff, X is a superkey for every functional dependency X → Y in given relation.

in otherword we can say that a relation is in BCNF, if and only if every determinant is a form (BCNF) candidate key.

(6) find the highest normal form

BCNF decomposition may always not possible with dependency preserving however, it always satisfies loseless join condition. for example. Relation R (V, W, X, Y, Z) with functional dependencies.

$V, W \rightarrow X$

$Y, Z \rightarrow X$

$W \rightarrow Y$

It would not satisfy dependency preserving BCNF decomposition.

Redundancies are sometime still present in a BCNF relation as it is not always possible to eliminate them completely.

(7)

Assignment No -2

SQL Join operation with examples.

SQL join operation is used to combine data or row from two or more table based on a common field betn. them.

Different types of join operation are —

1. Inner join
2. left join
3. Right join
4. full join.

Student

roll no.	name	address	age.
1	Aman Yadav	Janjgir	20
2	Nilima Sahu	Mahasamund	20
3	Prerna Patnayak	Bilaspur	21
4	Ebenezer k. Praveen	Bilaspur	21
5	Abhay Mansar	Champa	22
6	Ashutosh V.	Bilaspur	20

Student Course

course id	roll no.
1	1
2	2
2	3
3	4
1	5

1. Inner Join - the inner join keyword select all rows from both the table as long as the condition satisfies. This keyword will create the result set by combining all the rows from both the table where the condition satisfies i.e. value of common field will be same.

Syntax: select table1.column1, table1.column2 . . .
from table1 inner join table2 on table1.matching_column = table2.matching_column;

(8)

table1 : first table
table2 : second table
matching_column - column common to both tables.

we can also write join instead of inner join.

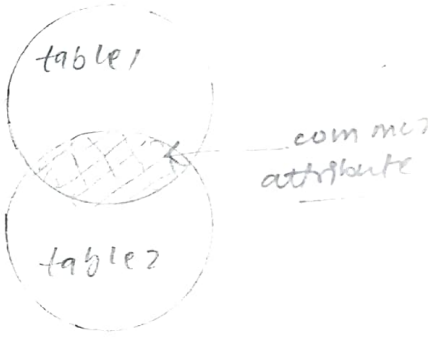
Example queries (inner join).

This query will show the names and age of student enrolled in different courses.

```
select studentCourse.courseId , Student.Name, student.age from student inner join studentCourse on student.rollno = studenCourse.rollno ;
```

output:

courseId	name	age.
1	Aman Yadav	20
2	Nilima Sahu	20
2.	Prerna Patnayak	21
3.	Ebenezer k. Praveen	21
1.	Abhay Mansar	22

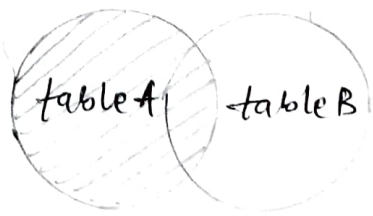


2. **Left join** :- This join will return all the ~~table~~ row from the table on the left side of join and matching rows of the right table of join. The non-matching rows will give resultset as null.

Syntax :-

```
select table1.column1, table1.column2... from table1
left join table2 on table1.matching_column = table2.matching-
column ;
select student.name, studentCourse.courseId from student
left join studentCourse on studentCourse.rollno = student.
rollno;
```

name	course Id
Aman Yadav	1
Nilima Sahu	2
Prerna Pat.	2.
Ashutosh V	null



(9)

3. Right join — Right join is similar to left join but this join will ~~right~~ return all the row from right table of join and matching rows of left table of join. the rows for which there is no matching row result set will contain null.

Syntax:-

```
select table1.column1, table1.column2.... from table1
right join table2 on table1.matching_column =
table2.matching_column;
```

```
select student.name, studentCourse.courseId from student
right join studentCourse on studentCourse.rollno =
student.rollno;
```

output.

name	course id
Aman Yadav	1
Niling Sahu	2
Prerna Patnayak	2
null	4
null	5

4. full join :- full join creates the result set by combining result of both left join and right join. the result set will contain all the row from both tables. for non-matching rows set will contain null.

```
select table1.column1, table1.column2.... from table1 full
join table2 on table1.matching_col = table2.matching_col;
```

```
select student.name, studentCourse.courseId from student
full join studentCourse on studentCourse.rollno = student.roll
no;
```

(10) output:

name	course id.
Aman Yadav	1
Nilima Sahu	2
Prerna Patnayak	2
Ebenezer K.P.	3
Abhay Manwar	1
Ashutosh Vaishnav	null

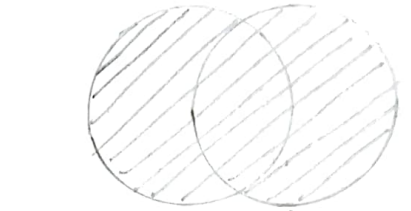


table 1 ↗ ↖ table 2

full join