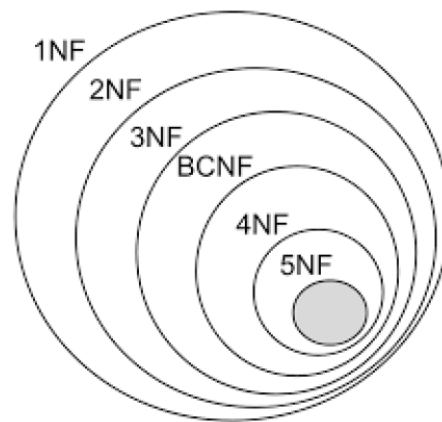
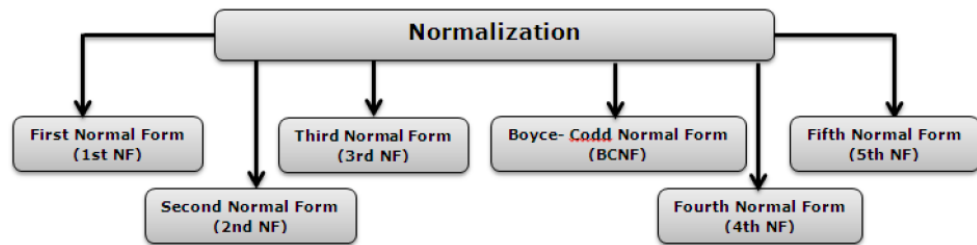


Name	Adwait S Purao
UID no.	2021300101
Experiment No.	10

AIM:	To apply the concepts of Normalization
Program 1	
PROBLEM STATEMENT :	.
Theory :	<p>Normalization</p> <p>A large database defined as a single relation may result in data duplication. This repetition of data may result in:</p> <ul style="list-style-type: none"> ○ Making relations very large. ○ It isn't easy to maintain and update data as it would involve searching many records in relation. ○ Wastage and poor utilization of disk space and resources. ○ The likelihood of errors and inconsistencies increases. <p>So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.</p>



What is Normalization?

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database

grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

Data modification anomalies can be categorized into three types:

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

Following are the various types of Normal forms:

	1NF	2NF	3NF	4NF	5NF
Decomposition of Relation	R	R ₁₁ R ₁₂	R ₂₁ R ₂₂ R ₂₃	R ₃₁ R ₃₂ R ₃₃ R ₃₄	R ₄₁ R ₄₂ R ₄₃ R ₄₄ R ₄₅
Conditions	Eliminate Repeating Groups	Eliminate Partial Functional Dependency	Eliminate Transitive Dependency	Eliminate Multi-values Dependency	Eliminate Join Dependency

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.
4NF	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
5NF	A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional

dependent on the primary key

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency $X \rightarrow Y$.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Example:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Super key in the table above:

{EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_STATE}, {EMP_ID, EMP_STATE, EMP_CITY}, {EMP_ID, EMP_CITY}

Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency $X \rightarrow Y$, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example: Let's assume there is a company where employees work in more than one department.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

EMP_ID \rightarrow EMP_COUNTRY
EMP_DEPT \rightarrow {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

	<p>Functional dependencies:</p> <div> EMP_ID → EMP_COUNTRY EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO} </div> <p>Candidate keys:</p> <p>For the first table: EMP_ID For the second table: EMP_DEPT For the third table: {EMP_ID, EMP_DEPT}</p> <p>Now, this is in BCNF because left side part of both the functional dependencies is a key.</p>
Queries	<p>Query 1: Converting a table into 1 NF form</p> <p>Statement : Remove the multiple values in column Phone_no</p> <p>Code : CREATE database NORMALIZATION; USE NORMALIZATION; CREATE TABLE EMPLOYEES(Emp_ID int NOT NULL, Emp_Name varchar(255), Company_Name varchar(255), Phone_No varchar(255), Emp_Sal float); INSERT INTO EMPLOYEES VALUES (12379,'Adwait Purao','Ethereum','1235467890,3342289346', 168512.34); INSERT INTO EMPLOYEES VALUES (87121,'Pranav Nalawade','Google','9876542130,2617312344', 234601.32); INSERT INTO EMPLOYEES VALUES (87121,'Pranav Nalawade','Rockstar','2617312344', 91312.34); INSERT INTO EMPLOYEES VALUES (98323,'Jay Nadkarni','Apple','8356289346', 689512.34); INSERT INTO EMPLOYEES VALUES (51239,'Vikas Patel','Blockchain Inc','1343289346', 689512.34);</p>

```

CREATE TABLE EMPLOYEESn(
Emp_ID int NOT NULL,
    Emp_Name varchar(255),
    Company_Name varchar(255),
    Phone_No1 bigint,
    Phone_No2 bigint,
    Emp_Sal float
);

INSERT INTO EMPLOYEESn
VALUES (12379,'Adwait Purao','Ethereum',1235467890,3342289346,
168512.34);
INSERT INTO EMPLOYEESn
VALUES (87121,'Pranav Nalawade','Google',9876542130,2617312344,
234601.32);
INSERT INTO EMPLOYEESn
VALUES (87121,'Pranav Nalawade','Rockstar',2617312344,null,
91312.34);
INSERT INTO EMPLOYEESn
VALUES (98323,'Jay Nadkarni','Apple',8356289346,null, 689512.34);
INSERT INTO EMPLOYEESn
VALUES (51239,'Vikas Patel','Blockchain Inc',1343289346,null,
689512.34);

select * from employees;

select * from employeesn;

```

Original Table:


Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 					
	Emp_ID	Emp_Name	Company_Name	Phone_No	Emp_Sal
▶	12379	Adwait Purao	Ethereum	1235467890,3342289346	168512
	87121	Pranav Nalawade	Google	9876542130,2617312344	234601
	98323	Jay Nadkarni	Apple	8356289346	689512
	51239	Vikas Patel	Blockchain Inc	1343289346	689512

Table in 1NF :

Result Grid						
Filter Rows: <input type="text"/>						
Export:						
Wrap Cell Content:						
	Emp_ID	Emp_Name	Company_Name	Phone_No1	Phone_No2	Emp_Sal
▶	12379	Adwait Purao	Ethereum	1235467890	3342289346	168512
	87121	Pranav Nalawade	Google	9876542130	2617312344	234601
	98323	Jay Nadkarni	Apple	8356289346	NULL	689512
	51239	Vikas Patel	Blockchain Inc	1343289346	NULL	689512

Query 2: Converting a table into 2NF Form

Statement : Split Employee table into two tables Employee List and Details

Code :

```
CREATE TABLE EMPLOYEE_LIST(
Emp_ID int NOT NULL,
    Emp_Name varchar(255)
);
```

```
CREATE TABLE DETAILS(
Emp_ID int NOT NULL,
    Company_Name varchar(255),
    Phone_No1 bigint,
    Phone_No2 bigint,
    Emp_Sal float
);
```

```
INSERT INTO EMPLOYEE_LIST
VALUES (12379,'Adwait Purao');
INSERT INTO EMPLOYEE_LIST
VALUES (87121,'Pranav Nalawade');
INSERT INTO EMPLOYEE_LIST
VALUES (98323,'Jay Nadkarni');
INSERT INTO EMPLOYEE_LIST
VALUES (51239,'Vikas Patel');
```

```
INSERT INTO DETAILS
VALUES (12379,'Ethereum',3342289346,1235467890, 168512.34);
INSERT INTO DETAILS
VALUES (87121,'Google',2617312344,9876542130, 234601.32);
INSERT INTO DETAILS
```

```
VALUES (98323,'Apple',8356289346,null, 689512.34);
INSERT INTO DETAILS
VALUES (51239,'Blockchain Inc',1343289346,null, 689512.34);
```

```
select * from employee_list;
```

```
select * from details;
```

Output :

Original Table:

Result Grid		Filter Rows:		Export:		Wrap Cell Content:	
	Emp_ID	Emp_Name	Company_Name	Phone_No1	Phone_No2	Emp_Sal	
▶	12379	Adwait Purao	Ethereum	1235467890	3342289346	168512	
	87121	Pranav Nalawade	Google	9876542130	2617312344	234601	
	98323	Jay Nadkarni	Apple	8356289346	NULL	689512	
	51239	Vikas Patel	Blockchain Inc	1343289346	NULL	689512	

New Table 1:

Result Grid		Filter Rows:	
	Emp_ID	Emp_Name	
▶	12379	Adwait Purao	
	87121	Pranav Nalawade	
	98323	Jay Nadkarni	
	51239	Vikas Patel	

New Table 2:

Result Grid		Filter Rows:		Export:		Wra
	Emp_ID	Company_Name	Phone_No1	Phone_No2	Emp_Sal	
▶	12379	Ethereum	3342289346	1235467890	168512	
	87121	Google	2617312344	9876542130	234601	
	98323	Apple	8356289346	NULL	689512	
	51239	Blockchain Inc	1343289346	NULL	689512	

Query 3: Converting a table into 3NF Form

Statement : Split EmpAdresss table into two tables

Code :

```
create table EmpAddress(  
Emp_ID int,  
Fullname varchar(255),  
zipcode int,  
state varchar(255),  
city varchar(255)  
);
```

```
insert into EmpAddress values(12379,'Adwait  
Purao',200290,"Maharashtra","Mumbai");  
insert into EmpAddress values(87121,'Pranav  
Nalawade',349940,"Karnataka","Bengaluru");  
insert into EmpAddress values(98323,'Jay  
Nadkarni',321200,"Punjab","Amritsar");  
insert into EmpAddress values(51239,'Vikas  
Patel',669294,"Kerala","Palakkad");
```

```
create table address1(  
Emp_ID int,  
Fullname varchar(255),  
zipcode int  
);
```

```
insert into address1 values (12379,'Adwait Purao',200290);  
insert into address1 values (87121,'Pranav Nalawade',349940);  
insert into address1 values (98323,'Jay Nadkarni',321200);  
insert into address1 values (51239,'Vikas Patel',669294);
```

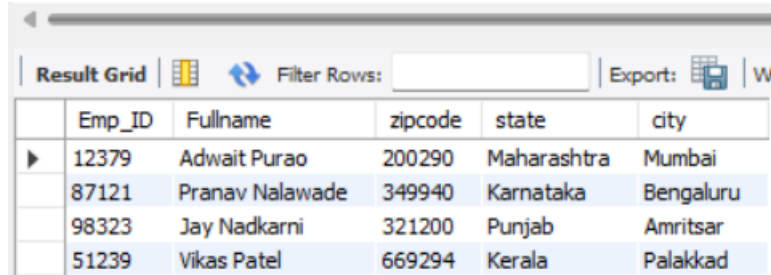
```
create table address2(  
zipcode int,  
state varchar(255),  
city varchar(255)  
);
```

```
insert into address2 values(200290,"Maharashtra","Mumbai");  
insert into address2 values(349940,"Karnataka","Bengaluru");  
insert into address2 values(321200,"Punjab","Amritsar");  
insert into address2 values(669294,"Kerala","Palakkad");
```

```
select * from empaddress;  
select * from address1;  
select * from address2;
```

Output :

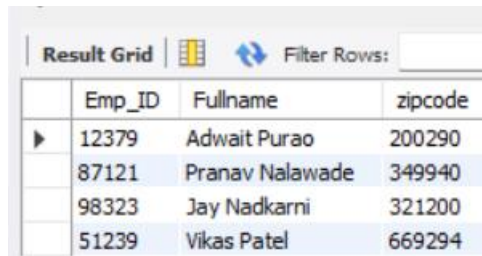
Original Table:



A screenshot of a database result grid interface. It features a toolbar with 'Result Grid', a grid icon, a refresh icon, and a 'Filter Rows:' input field. To the right is an 'Export:' button with a document icon. Below the toolbar is a table with 6 columns: Emp_ID, Fullname, zipcode, state, and city. The table contains 4 rows of data.

	Emp_ID	Fullname	zipcode	state	city
▶	12379	Adwait Purao	200290	Maharashtra	Mumbai
	87121	Pranav Nalawade	349940	Karnataka	Bengaluru
	98323	Jay Nadkarni	321200	Punjab	Amritsar
	51239	Vikas Patel	669294	Kerala	Palakkad

New Table 1:



A screenshot of a database result grid interface showing a table with 4 columns: Emp_ID, Fullname, and zipcode. The table contains 4 rows of data.

	Emp_ID	Fullname	zipcode
▶	12379	Adwait Purao	200290
	87121	Pranav Nalawade	349940
	98323	Jay Nadkarni	321200
	51239	Vikas Patel	669294

New Table 2:



A screenshot of a database result grid interface showing a table with 4 columns: zipcode, state, and city. The table contains 4 rows of data.

	zipcode	state	city
▶	200290	Maharashtra	Mumbai
	349940	Karnataka	Bengaluru
	321200	Punjab	Amritsar
	669294	Kerala	Palakkad

Query 4: Converting a table into BCNF Form

Statement : Split Supportstaff table into two tables

Code:

```
create table Supportstaff(  
StaffID int,
```



```

Staffstate varchar(255),
Company varchar(255),
depttype varchar(255),
DeptID int
);

insert into Supportstaff values(1,"Maharahtra","Ethereum","Computer",100);
insert into Supportstaff values(1,"Maharahtra","Google","Electronics",101);
insert into Supportstaff values(2,"Karnataka","Apple","Marketing",365);
insert into Supportstaff values(2,"Karnataka","Blockchain
Inc. ","Computer",729);

create table staff1(
StaffID int,
Staffstate varchar(255)
);

insert into staff1 values (1,"Maharahstra");
insert into staff1 values (2,"Karnataka");

create table staff2(
Company varchar(255),
depttype varchar(255),
DeptID int
);

insert into staff2 values("Ethereum","Computer",100);
insert into staff2 values("Google","Electronics",101);
insert into staff2 values("Apple","Marketing",365);
insert into staff2 values("Blockchain Inc. ","Computer",729);

create table staff3(
StaffID int,
DeptID int
);

insert into staff3 values(1,100);
insert into staff3 values(1,101);
insert into staff3 values(2,365);
insert into staff3 values(2,729);

select * from Supportstaff;
select * from staff1;
select * from staff2;
select * from staff3;

```

Output :

Original Table:

	StaffID	Staffstate	Company	depttype	DeptID
▶	1	Maharashtra	Ethereum	Computer	100
	1	Maharashtra	Google	Electronics	101
	2	Karnataka	Apple	Marketing	365
	2	Karnataka	Blockchain Inc.	Computer	729

New Table 1:

	StaffID	Staffstate
▶	1	Maharashtra
	2	Karnataka

New Table 2:

	Company	depttype	DeptID
▶	Ethereum	Computer	100
	Google	Electronics	101
	Apple	Marketing	365
	Blockchain Inc.	Computer	729

New Table 3:

	StaffID	DeptID
▶	1	100
	1	101
	2	365
	2	729

Query 5: Converting a table into 4 NF Form

Statement : Split CEOTable table into two tables

Code:

```
create table CEOTable(  
FullName varchar(255),  
Company varchar(255),  
Hobby varchar(255)
```

```
);

insert into CEOTable values ("Vitalik Buterin","Ethereum","Reading
Books");
insert into CEOTable values ("Sundar Pichai","Google","Football");
insert into CEOTable values ("Tim Cook","Apple","Singing");
insert into CEOTable values ("Jay Nadkarni","Blockchain","Dancing");

create table CEO2(
FullName varchar(255),
Company varchar(255)
);

insert into CEO2 values ("Vitalik Buterin","Ethereum");
insert into CEO2 values ("Sundar Pichai","Google");
insert into CEO2 values ("Tim Cook","Apple");
insert into CEO2 values ("Jay Nadkarni","Blockchain");

create table CEO3(
FullName varchar(255),
Hobby varchar(255)
);

insert into CEO3 values ("Vitalik Buterin","Reading Books");
insert into CEO3 values ("Sundar Pichai","Football");
insert into CEO3 values ("Tim Cook","Singing");
insert into CEO3 values ("Jay Nadkarni","Dancing");

select * from CEOTable;
select * from CEO2;
select * from CEO3;
```

Output :
Original Table:

Result Grid			
Filter Rows:			
	FullName	Company	Hobby
▶	Vitalik Buterin	Ethereum	Reading Books
	Sundar Pichai	Google	Football
	Tim Cook	Apple	Singing
	Jay Nadkarni	Blockchain	Dancing

New Table 1:

Result Grid		
Filter Rows:		
	FullName	Company
▶	Vitalik Buterin	Ethereum
	Sundar Pichai	Google
	Tim Cook	Apple
	Jay Nadkarni	Blockchain

New Table 2:

Result Grid		
Filter Rows:		
	FullName	Hobby
▶	Vitalik Buterin	Reading Books
	Sundar Pichai	Football
	Tim Cook	Singing
	Jay Nadkarni	Dancing

Conclusion

From this experiment, we learnt about normalisation . We saw the various forms of normalisation such as 1 NF, 2 NF, 3 NF, BC NF, 4 NF.. We saw the advantages of normalisation such as removing redundancy in the tables and also we can minimize issues with data modification . We also learnt about Insertion , Updation and Deletion anomaly.