

Name: Prashant Suresh Shirgave
Roll No:3 **Batch:**T1
Class:TY(CSE-AIML)

Experiment No. 12

Title: Normalization

Objective: Normalize any database from first normal form to Boyce-Codd Normal Form (BCNF).

Theory:

Normalization:

- Normalization is the process of efficiently organizing data in a database.
- Normalization of data can be considered as a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of
 - (1) minimizing redundancy and
 - (2) minimizing the insertion, deletion, and update anomalies

Normal forms:

1. First normal form (1NF):

- Disallows **multivalued** attributes, **composite** attributes and their combinations.
- Only attribute values permitted by 1NF are single atomic (or indivisible) values.

2. Second normal form (2NF):

A relation schema R is in **2NF** if every nonprime attribute A in R is **fully functionally** dependent on the **primary key** of R.

3. Boyce Codd Normal Form (BCNF): Relation schema R is in BCNF w.r.t. set F if all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$ where α, β are subsets of R, at least one of the following holds:

- $\alpha \rightarrow \beta$ is a trivial functional dependency.
- α is a superkey for schema R.

4. Third Normal Form (3NF): Relation schema R is in 3NF w.r.t. set F if all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$ where α, β are subsets of R, at least one of the following holds: a. $\alpha \rightarrow \beta$ is a trivial functional dependency.

- α is a superkey for schema R.
- Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.


Example: Normalization of ClientRental database from first normal form to 3NF

CREATE TABLE

```
ClientRental ( clientNo
VARCHAR(255), cName
VARCHAR(255),
propertyNo
VARCHAR(255),
pAddress
VARCHAR(255),
rentStart DATE,
rentFinish DATE,
rent INT,
ownerNo
VARCHAR(255), oName
VARCHAR(255)
);
```

INSERT INTO ClientRental (clientNo, cName, propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)
VALUES

```
('CR76', 'Prashant', 'PG4', 'shivaji chowk, mudshingi', '2003-07-01', '2004-08-31', 350, 'CO40', 'Sanika'),
('CR76', 'Prashant', 'PG16', 'Bhagava chowk, mudshingi', '2004-09-01', '2005-09-01', 450, 'CO93', 'Pranali'),
('CR56', 'Rushi', 'PG4', 'shivaji chowk, mudshingi', '2002-09-01', '2003-06-10', 350, 'CO40', 'Sanika'),
('CR56', 'Rushi', 'PG36', 'Nrusih chowk, mudshingi', '2000-10-01', '2001-12-04', 375, 'CO93', 'Pranali'),
('CR56', 'Rushi', 'PG16', 'Bhagava chowk, mudshingi', '2001-11-05', '2002-08-06', 450, 'CO93', 'Pranali');
```

Result Grid									
Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 									
	clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
▶	CR76	Prashant	PG4	shivaji chowk, mudshingi	2003-07-01	2004-08-31	350	CO40	Sanika
	CR76	Prashant	PG16	Bhagava chowk, mudshingi	2004-09-01	2005-09-01	450	CO93	Pranali
	CR56	Rushi	PG4	shivaji chowk, mudshingi	2002-09-01	2003-06-10	350	CO40	Sanika
	CR56	Rushi	PG36	Nrusih chowk, mudshingi	2000-10-01	2001-12-04	375	CO93	Pranali
	CR56	Rushi	PG16	Bhagava chowk, mudshingi	2001-11-05	2002-08-06	450	CO93	Pranali

Step 1: First Normal Form (1NF)

The current table is already in 1NF because:

- Each field has atomic values.
- There are no repeating groups.

Alternate 1NF form of above table is client and propertyRentalOwner Relations: This approach reduces redundancy by separating client information and grouping rental and property details together in a single table, linking back to the client table.

```
CREATE TABLE Client (  
    clientNo VARCHAR(255) PRIMARY KEY,  
    cName VARCHAR(255)  
);
```

```
INSERT INTO Client (clientNo,  
cName) SELECT DISTINCT clientNo,  
cName FROM ClientRental;
```

```
CREATE TABLE PropertyRentalOwner  
( clientNo VARCHAR(255),  
propertyNo  
VARCHAR(255),  
pAddress  
VARCHAR(255),  
rentStart DATE,  
rentFinish DATE,  
rent INT,  
ownerNo  
VARCHAR(255), oName  
VARCHAR(255),  
PRIMARY KEY (clientNo, propertyNo, rentStart),  
FOREIGN KEY (clientNo) REFERENCES  
Client(clientNo)  
);
```

```
INSERT INTO PropertyRentalOwner (clientNo, propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)  
SELECT clientNo, propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName
```

FROM ClientRental;

	clientNo	cName
▶	CR56	Rushi
	CR76	Prashant
•	NULL	NULL

	clientNo	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
▶	CR56	PG16	Bhagava chowk, mudshingi	2001-11-05	2002-08-06	450	CO93	Pranali
	CR56	PG36	Nrusih chowk, mudshingi	2000-10-01	2001-12-04	375	CO93	Pranali
	CR56	PG4	shivaji chowk, mudshingi	2002-09-01	2003-06-10	350	CO40	Sanika
	CR76	PG16	Bhagava chowk, mudshingi	2004-09-01	2005-09-01	450	CO93	Pranali
	CR76	PG4	shivaji chowk, mudshingi	2003-07-01	2004-08-31	350	CO40	Sanika
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Step 2: Second Normal Form (2NF)

2nd Normal Form (2NF) requires that all non-prime attributes (attributes not part of the primary key) depend on the whole primary key, not just a part of it.

To achieve **Second Normal Form (2NF)** from the PropertyRentalOwner structure, we'll separate out the rental details from property and owner details. This will result in three tables:

1. **Client Table** - Contains unique client information.
2. **Rental Table** - Contains rental details (clientNo, propertyNo, rentStart, rentFinish,).
3. **PropertyOwner Table** - Contains the property and owner details (propertyNo, pAddress, Rent,ownerNo, and oName).

1) Client Table

The Client table remains the same as before:

2) Rental Table

The Rental table is populated using data from the PropertyRentalOwner table, which includes the clientNo, propertyNo, rentStart, and rentFinish fields.

CREATE TABLE Rental (

clientNo VARCHAR(255),

propertyNo

VARCHAR(255),

rentStart DATE,

rentFinish DATE,

PRIMARY KEY (clientNo, propertyNo, rentStart),

FOREIGN KEY (clientNo) REFERENCES

Client(clientNo)

);

INSERT INTO Rental (clientNo, propertyNo, rentStart, rentFinish)

SELECT clientNo, propertyNo, rentStart, rentFinish

FROM PropertyRentalOwner;

3) **PropertyOwner Table** : Will contain property details along with owner information.

CREATE TABLE PropertyOwner (

propertyNo VARCHAR(255) PRIMARY KEY,

pAddress

VARCHAR(255), rent

INT,

ownerNo

VARCHAR(255), oName

VARCHAR(255));

INSERT INTO PropertyOwner (propertyNo, pAddress, rent, ownerNo, oName)

SELECT DISTINCT propertyNo, pAddress, rent, ownerNo, oName

FROM

PropertyRentalOwner;

select * from

PropertyOwner;

After 2nd normal form table Becomes:

	clientNo	cName
▶	CR 56	Rushi
	CR 76	Prashant
•	NULL	NULL

	clientNo	propertyNo	rentStart	rentFinish
▶	CR 56	PG16	2001-11-05	2002-08-06
	CR 56	PG36	2000-10-01	2001-12-04
	CR 56	PG4	2002-09-01	2003-06-10
	CR 76	PG16	2004-09-01	2005-09-01
	CR 76	PG4	2003-07-01	2004-08-31
•	NULL	NULL	NULL	NULL

	propertyNo	pAddress	rent	ownerNo	oName
▶	PG16	Bhagava chowk, mudshingi	450	CO93	Pranali
	PG36	Nrusih chowk, mudshingi	375	CO93	Pranali
	PG4	shivaji chowk, mudshingi	350	CO40	Sanika
•	NULL	NULL	NULL	NULL	NULL

Step 3: Third Normal Form (2NF)

To achieve Third Normal Form (3NF) for the provided structure, we will further divide the PropertyOwner table into two separate tables:

- **Client Table:** The Client table remains the same as before:
- **Rental Table:** The Rental table remains the same as before:
- **PropertyForRent Table:** Contains information about properties available for rent, including propertyNo, address, rent, and ownerNo.
- **Owner Table:** Contains unique owner information, including ownerNo and oName.

Owner table:

```
CREATE TABLE Owner (  
    ownerNo VARCHAR(255) PRIMARY  
    KEY, oName VARCHAR(255)  
);
```

```
INSERT INTO Owner (ownerNo,  
oName) SELECT DISTINCT ownerNo,  
oName FROM PropertyOwner;
```

After 3rd normal form table Becomes:

	clientNo	cName
▶	CR56	Rushi
	CR76	Prashant
•	NULL	NULL

	clientNo	propertyNo	rentStart	rentFinish
▶	CR56	PG16	2001-11-05	2002-08-06
	CR56	PG36	2000-10-01	2001-12-04
	CR56	PG4	2002-09-01	2003-06-10
	CR76	PG16	2004-09-01	2005-09-01
	CR76	PG4	2003-07-01	2004-08-31
•	NULL	NULL	NULL	NULL

	ownerNo	oName
▶	CO40	Sanika
	CO93	Pranali
•	NULL	NULL

	propertyNo	pAddress	rent	ownerNo
▶	PG16	Bhagava chowk, mudshingi	450	CO93
	PG36	Nrusih chowk, mudshingi	375	CO93
	PG4	shivaji chowk, mudshingi	350	CO40
•	NULL	NULL	NULL	NULL

Step 4: BCNF

All the tables in the normalized schema—**Client**, **Rental**, **PropertyForRent**, and **Owner**—satisfy BCNF because:

- Each functional dependency has a determinant that is a superkey.
- No non-trivial functional dependencies exist where the determinant is not a superkey.

Outcome: Students are able to normalize any database from 1NF to BCNF.