Normalization

Data Warehousing
08CS

By Ali Asghar Manjotho
Lecturer, Department of Computer Systems Engineering,
MUET, Jamshoro.

Normalization

- It is the process of decomposing a large table or collection of large tables in to set of well structured relations/tables that represent the same data but is free of update anomalies.

- Here we break large tables (tables with more number of columns) in to smaller tables (tables with minimum number of columns).

- Well structured means related attributes are placed in same table.

Normalization

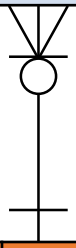
StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software

- Consider above table, It is un-normalized. We will decompose it in to more number of tables.
- All the columns which are related will be moved in to separate table.
- As columns DepartmentID and DepartmentName are related to department so create another table (Department) for them.
- Columns StudentID and StudentName are related to student so place them in Student table and DepartmentID will be placed here as foreign key.

Normalization

Student

StudentID	StudentName	DepartmentID
07CS02	Bilal	Dept01
07CS06	Farhan	Dept01
06ES01	Khalid	Dept02
07SW01	Faheem	Dept03
07SW05	Asif	Dept03



Department

DepartmentID	DepartmentName
Dept01	Computer Systems
Dept02	Electronics
Dept03	Software

Database Structures

1) Un-Normalized Database

2) Normalized Database

3) Denormalized Database

1. Un-Normalized Database

- A database before applying the normalization on to it.
- Un-Normalized database is simply the collection of fields.
- Here we have very large tables (tables with large number of columns).
- It suffers from update anomalies.

2. Normalized Database

- A database after applying the normalization on to it.
- It is the process of decomposing a large table or collection of large tables in to set of well structured.
- Here we have small tables (tables with minimum number of columns).
- It does not suffer from update anomalies.

3. Denormalized Database

- It is the inverse process of normalization.
- It combines number of small relations/tables and forms large tables.
- It contains redundancy.

Goals of Normalization

- Reduce the redundancy.
- Improve the consistency.
- Free the database from update anomalies.

Need for Normalization

- In the early work with relational database theory Dr. Codd discovered that un-normalized relations presented certain problems when we made to insert, deleted or update the data in them.
- He used the term anomalies for these problems.
- The reason we normalize the relations is to remove the anomalies from the data.

Update Anomalies

- The problems/errors caused, when ever we insert, update or delete records in to the database, are called update anomalies.

- There are three types of anomalies:

1) Deletion Anomalies

2) Modification Anomalies

3) Insertion Anomalies

Deletion Anomalies

- The problems/errors caused, when ever we delete one or more records from the database are called deletion anomalies.

- Consider the following database.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software

Deletion Anomalies

- According to the database we have 3 departments.

  1. Computer Systems
  2. Electronics
  3. Software

- Suppose if we delete the record with StudentID=06ES01.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software

Deletion Anomalies

- After deleting the record of 06ES01 our database looks like as,

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software

- Now according to the database we have only 2 departments namely, Computer Systems and Software.

Deletion Anomalies

- 06ES01 was the last record for Electronics department.
- As we have deleted the record of 06ES01 the complete details of Electronics department has been deleted as well.
- We are deleting record of student but along with it the record of department is also being deleted.
- This sort of problem is called deletion anomalies.

Modification Anomalies

- The problems/errors caused, when ever we update one or more records in to the database are called modification anomalies.

- Consider the following database.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software

Modification Anomalies

- Suppose you want to modify the name of Dept03 from *Software* to *Software Engineering*.
- For that you have to update the DepartmentName in all the record where DepartmentID is Dept03.
- Consider the following database.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software

Modification Anomalies

- If you are skipping even a single record without modification then the data will be in inconsistent state.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software Engineering
07SW05	Asif	Dept03	Software

Modification Anomalies

- So if you have to modify the record of one department then you have to modify it at all the records.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software Engineering
07SW05	Asif	Dept03	Software Engineering

Insertion Anomalies

- The problems/errors caused, when ever we insert a record in to the database are called insertion anomalies.

- Consider the following database.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software

Insertion Anomalies

- Suppose we want to insert the new student in to Electronics department then we have to provide the information of corresponding department as well.

- This can cause the data to go in to inconsistence state.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software
07ES05	Muhammad	Dept02	Electronics Engineering

Insertion Anomalies

- Two records containing department information of same department but having different data.
- So data has gone in to inconsistent state.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software
07ES05	Muhammad	Dept02	Electronics Engineering

Insertion Anomalies

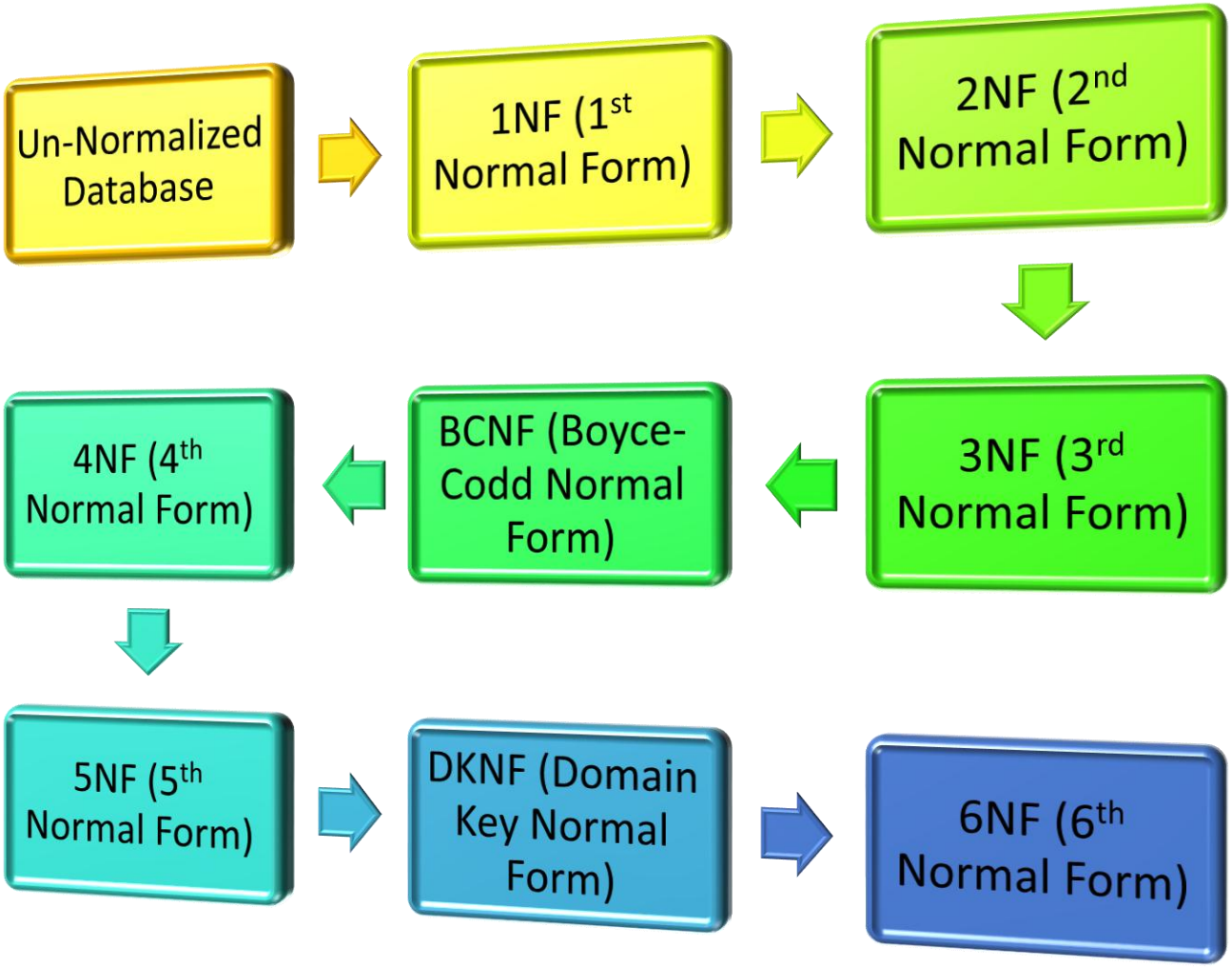
- Suppose we want to insert the new department having no any student registered yet.
- For that we have to place NULL in StudentID and StudentName.

StudentID	StudentName	DepartmentID	DepartmentName
07CS02	Bilal	Dept01	Computer Systems
07CS06	Farhan	Dept01	Computer Systems
06ES01	Khalid	Dept02	Electronics
07SW01	Faheem	Dept03	Software
07SW05	Asif	Dept03	Software
NULL	NULL	Dept04	Civil Engineering

Insertion Anomalies

- But here StudentID is the primary key hence the primary key column can not contain the NULL values.
- So this sort of insertion can cause problems.

Normalization Process



Normalization Example (Un-Normalized Structure)

Client NO	Client Name	PropertyNo	PropertyAddress	RentStart	RentFinish	Rent	Owner No	OwnerName
C01	Ali	P01	Jamshoro	1-Jan-2004	30-May-2009	350	O01	Farhan
		P02	Karachi	1-Jun-2009	31-Dec-2010	450	O02	Ismail
C02	Bilal	P01	Jamshoro	1-Jun-2009	31-Dec-2010	350	O01	Farhan
		P02	Karachi	1-Jan-2011	31-Jan-2011	450	O02	Ismail
		P03	Qasimabad	1-Jan-1996	28-Feb-2003	700	O03	Asif

It is un-Normalized structure because it contains multiple values in single cell.

Normalization Example (convert to 1NF)

A relation is said to be in 1NF (1st Normal Form) if at the intersection of each row and column there is one and only one value.

Normalization Example (convert to 1NF)

Client NO	Client Name	PropertyNo	PropertyAddress	RentStart	RentFinish	Rent	OwnerNo	OwnerName
C01	Ali	P01	Jamshoro	1-Jan-2004	30-May-2009	350	O01	Farhan
C01	Ali	P02	Karachi	1-Jun-2009	31-Dec-2010	450	O02	Ismail
C02	Bilal	P01	Jamshoro	1-Jun-2009	31-Dec-2010	350	O01	Farhan
C02	Bilal	P02	Karachi	1-Jan-2011	31-Jan-2011	450	O02	Ismail
C02	Bilal	P03	Qasimabad	1-Jan-1996	28-Feb-2003	700	O03	Asif

Now it is in 1NF because at the intersection of every row and column we have only one value.

Normalization Example (convert to 1NF)

- Here we have composite primary key (ClientNo and PropertyNo).

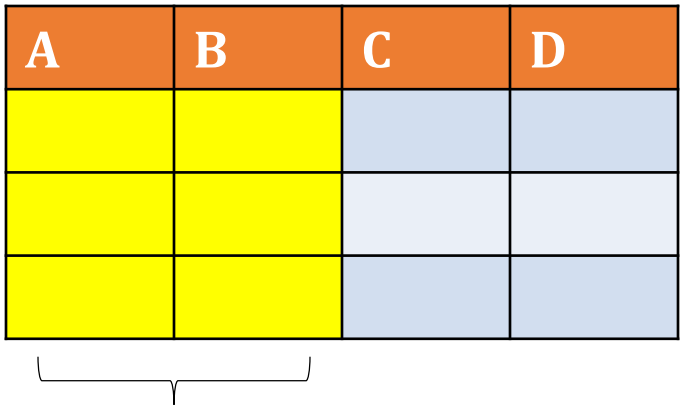
ClientNO	PropertyNo
C01	P01
C01	P02
C02	P01
C02	P02
C02	P03

Normalization Example (convert to 2NF)

- A relation is said to be in 2NF (2nd Normal Form) if it is in 1NF and every non primary key column is fully-functionally dependent on primary key columns.

- Here we have to remove partial dependencies.

Normalization Example (convert to 2NF)



Composite Primary Key

- If A, B, C and D are the attributes (columns) of a table.
- A and B are the composite primary key.
- C and D are non-primary key columns.

Normalization Example (convert to 2NF)

Attribute C is said to be fully functionally dependent on primary key if C is dependent on A and is also dependent on B.

$A + B \longrightarrow C$ (C is fully functionally dependent on primary key)

$A \longrightarrow C$ (C is partially dependent on primary key)

Or

$B \longrightarrow C$ (C is partially dependent on primary key)

Normalization Example (convert to 2NF)

A ⟶ C

C is functionally dependent on A if there exists 1-1 left to right relationship between A and C.

Student ⟶ Department

For every student there is only one department. Hence department is functionally dependent on Student.

Normalization Example (convert to 2NF)

Student ⟶ **Phone Number**

For every student there may be more than one telephone numbers. Hence Phone number is not functionally dependent on Student.

Normalization Example (convert to 2NF)

Client NO	Client Name	PropertyNo	PropertyAddress	RentStart	RentFinish	Rent	OwnerNo	OwnerName
C01	Ali	P01	Jamshoro	1-Jan-2004	30-May-2009	350	O01	Farhan
C01	Ali	P02	Karachi	1-Jun-2009	31-Dec-2010	450	O02	Ismail
C02	Bilal	P01	Jamshoro	1-Jun-2009	31-Dec-2010	350	O01	Farhan
C02	Bilal	P02	Karachi	1-Jan-2011	31-Jan-2011	450	O02	Ismail
C02	Bilal	P03	Qasimabad	1-Jan-1996	28-Feb-2003	700	O03	Asif

- Consider the above table, we want to convert it in to 2NF.
- ClientNO + PropertyNO is the composite primary key.
- ClientName, PropertyAddress, RentStart, RentFinish, Rent, OwnerNO and OwnerName are non-primary key columns.

Normalization Example (convert to 2NF)

- Now we will check each non-primary key column with the composite primary ley column.
- If any non-primary key column is fully functionally dependent on primary key column, we will leave it in the same table.
- If any non-primary key column is partially dependent on primary key column, we will remove it from the current table and place it in another table.

Normalization Example (convert to 2NF)

Check ClientName

ClientNo ⟶ **ClientName**

One ClientNo has only 1 ClientName so ClientName is dependent.

PropertyNo ⟶ **ClientName**

One PropertyNo has many ClientNames so ClientName is not dependent.

Normalization Example (convert to 2NF)

- ClientName is dependent on ClientNo but not dependent on PropertyNo
- So ClientName is partially dependent on composite primary key.
- Hence we will remove it from current table and move it in to another table (Client)

Normalization Example (convert to 2NF)

ClientNo	PropertyNo	PropertyAddress	RentStart	RentFinish	Rent	OwnerNo	OwnerName
C01	P01	Jamshoro	1-Jan-2004	30-May-2009	350	O01	Farhan
C01	P02	Karachi	1-Jun-2009	31-Dec-2010	450	O02	Ismail
C02	P01	Jamshoro	1-Jun-2009	31-Dec-2010	350	O01	Farhan
C02	P02	Karachi	1-Jan-2011	31-Jan-2011	450	O02	Ismail
C02	P03	Qasimabad	1-Jan-1996	28-Feb-2003	700	O03	Asif

ClientNo	ClientName
C01	Ali
C02	Bilal

Normalization Example (convert to 2NF)

Check PropertyAddress

ClientNo ⟶ **PropertyAddress**

One ClientNo has many PropertyAddresses so PropertyAddress is not dependent.

PropertyNo ⟶ **PropertyAddress**

One PropertyNo has only 1 PropertyAddress so PropertyAddress is dependent.

Normalization Example (convert to 2NF)

- PropertyAddress is not dependent on ClientNo but is dependent on PropertyNo.
- So PropertyAddress is partially dependent on composite primary key.
- Hence we will remove it from current table and move it in to another table (PropertyOwner)

Normalization Example (convert to 2NF)

ClientNo	PropertyNo	RentStart	RentFinish	Rent	OwnerNo	OwnerName
C01	P01	1-Jan-2004	30-May-2009	350	O01	Farhan
C01	P02	1-Jun-2009	31-Dec-2010	450	O02	Ismail
C02	P01	1-Jun-2009	31-Dec-2010	350	O01	Farhan
C02	P02	1-Jan-2011	31-Jan-2011	450	O02	Ismail
C02	P03	1-Jan-1996	28-Feb-2003	700	O03	Asif

ClientNo	ClientName
C01	Ali
C02	Bilal

PropertyNo	PropertyAddress
P01	Jamshoro
P02	Karachi
P03	Qasimabad

Normalization Example (convert to 2NF)

Check RentStart

ClientNo ⟶ **RentStart**

One ClientNo has many RentStarts so RentStart is not dependent.

PropertyNo ⟶ **RentStart**

One PropertyNo has many RentStarts so RentStart is not dependent.

Normalization Example (convert to 2NF)

- RentStart is not dependent on ClientNo and is also not dependent on PropertyNo.
- So PropertyAddress is not partially dependent on composite primary key.
- Hence we will leave it in the same table.

Normalization Example (convert to 2NF)

Check RentFinish

ClientNo → RentFinish

One ClientNo has many RentFinishes so RentFinish is not dependent.

PropertyNo → RentFinish

One PropertyNo has many RentFinishes so RentFinish is not dependent.

Normalization Example (convert to 2NF)

- RentFinish is not dependent on ClientNo and is also not dependent on PropertyNo.
- So RentFinish is not partially dependent on composite primary key.
- Hence we will leave it in the same table.

Normalization Example (convert to 2NF)

Check Rent

ClientNo ⟶ **Rent**

One ClientNo has many Rents so Rent is not dependent.

PropertyNo ⟶ **Rent**

One PropertyNo has only 1 Rent so Rent is dependent.

Normalization Example (convert to 2NF)

- Rent is not dependent on ClientNo but is dependent on PropertyNo.
- So Rent is partially dependent on composite primary key.
- Hence we will remove it from current table and move it in to another table (PropertyOwner)

Normalization Example (convert to 2NF)

ClientNo	PropertyNo	RentStart	RentFinish	OwnerNo	OwnerName
C01	P01	1-Jan-2004	30-May-2009	O01	Farhan
C01	P02	1-Jun-2009	31-Dec-2010	O02	Ismail
C02	P01	1-Jun-2009	31-Dec-2010	O01	Farhan
C02	P02	1-Jan-2011	31-Jan-2011	O02	Ismail
C02	P03	1-Jan-1996	28-Feb-2003	O03	Asif

ClientNo	ClientName
C01	Ali
C02	Bilal

PropertyNo	PropertyAddress	Rent
P01	Jamshoro	350
P02	Karachi	450
P03	Qasimabad	700

Normalization Example (convert to 2NF)

Check OwnerNo

ClientNo ⟶ **OwnerNo**

One ClientNo has many OwnerNos so OwnerNo is not dependent.

PropertyNo ⟶ **OwnerNo**

One PropertyNo has only 1 OwnerNo so OwnerNo is dependent.

Normalization Example (convert to 2NF)

- OwnerNo is not dependent on ClientNo but is dependent on PropertyNo.
- So OwnerNo is partially dependent on composite primary key.
- Hence we will remove it from current table and move it in to another table (PropertyOwner)

Normalization Example (convert to 2NF)

ClientNo	PropertyNo	RentStart	RentFinish	OwnerName
C01	P01	1-Jan-2004	30-May-2009	Farhan
C01	P02	1-Jun-2009	31-Dec-2010	Ismail
C02	P01	1-Jun-2009	31-Dec-2010	Farhan
C02	P02	1-Jan-2011	31-Jan-2011	Ismail
C02	P03	1-Jan-1996	28-Feb-2003	Asif

ClientNo	ClientName
C01	Ali
C02	Bilal

PropertyNo	PropertyAddress	Rent	OwnerNo
P01	Jamshoro	350	O01
P02	Karachi	450	O02
P03	Qasimabad	700	O03

Normalization Example (convert to 2NF)

Check OwnerName

ClientNo ⟶ **OwnerName**

One ClientNo has many OwnerNames so OwnerName is not dependent.

PropertyNo ⟶ **OwnerName**

One PropertyNo has only 1 OwnerName so OwnerName is dependent.

Normalization Example (convert to 2NF)

- OwnerName is not dependent on ClientNo but is dependent on PropertyNo.
- So OwnerName is partially dependent on composite primary key.
- Hence we will remove it from current table and move it in to another table (PropertyOwner)

Normalization Example (convert to 2NF)

ClientNo	PropertyNo	RentStart	RentFinish
C01	P01	1-Jan-2004	30-May-2009
C01	P02	1-Jun-2009	31-Dec-2010
C02	P01	1-Jun-2009	31-Dec-2010
C02	P02	1-Jan-2011	31-Jan-2011
C02	P03	1-Jan-1996	28-Feb-2003

ClientNo	ClientName
C01	Ali
C02	Bilal

PropertyNo	PropertyAddress	Rent	OwnerNo	OwnerName
P01	Jamshoro	350	O01	Farhan
P02	Karachi	450	O02	Ismail
P03	Qasimabad	700	O03	Asif

Normalization Example (convert to 2NF)

Rental

ClientNo	PropertyNo	RentStart	RentFinish
C01	P01	1-Jan-2004	30-May-2009
C01	P02	1-Jun-2009	31-Dec-2010
C02	P01	1-Jun-2009	31-Dec-2010
C02	P02	1-Jan-2011	31-Jan-2011
C02	P03	1-Jan-1996	28-Feb-2003

Client

ClientNo	ClientName
C01	Ali
C02	Bilal

PropertyOwner

PropertyNo	PropertyAddress	Rent	OwnerNo	OwnerName
P01	Jamshoro	350	O01	Farhan
P02	Karachi	450	O02	Ismail
P03	Qasimabad	700	O03	Asif

Normalization Example (convert to 3NF)

- A relation is said to be in 3NF (3rd Normal Form) if it is in 1NF and 2NF and no any non-primary key column is transitively dependent on primary key columns.

- Here we have to remove transitive dependencies.

Normalization Example (convert to 3NF)

- If A, B and C are the attributes of a table.
- If B is functionally dependent on A and C is functionally dependent on B, then C is transitively dependent on A.

If $A \longrightarrow B$ (B is functionally dependent on A)

And $B \longrightarrow C$ (C is functionally dependent on B)

Then $A \longrightarrow C$ (C is transitively dependent on A)

Normalization Example (convert to 3NF)

PropertyOwner

PropertyNo	PropertyAddress	Rent	OwnerNo	OwnerName
P01	Jamshoro	350	O01	Farhan
P02	Karachi	450	O02	Ismail
P03	Qasimabad	700	O03	Asif

Normalization Example (convert to 2NF)

PropertyNo ⟶ **OwnerNo**

One PropertyNo has only 1 OwnerNo so OwnerNo is dependent.

OwnerNo ⟶ **OwnerName**

One OwnerNo has only 1 OwnerName so OwnerName is dependent.

- Here OwnerName is transitively dependent on PropertyNo.
- So we will remove OwnerName from the current table and place it in to another table (Owner).

Normalization Example (convert to 3NF)

PropertyOwner

PropertyNo	PropertyAddress	Rent	OwnerNo
P01	Jamshoro	350	O01
P02	Karachi	450	O02
P03	Qasimabad	700	O03

OwnerNo	OwnerName
O01	Farhan
O02	Ismail
O03	Asif

Normalization Example (convert to 3NF)

Rent

ClientNO	PropertyNo	RentStart	RentFinish
C01	P01	1-Jan-2004	30-May-2009
C01	P02	1-Jun-2009	31-Dec-2010
C02	P01	1-Jun-2009	31-Dec-2010
C02	P02	1-Jan-2011	31-Jan-2011
C02	P03	1-Jan-1996	28-Feb-2003

Client

ClientNO	ClientName
C01	Ali
C02	Bilal

Property

PropertyNo	PropertyAddress	Rent	OwnerNo
P01	Jamshoro	350	O01
P02	Karachi	450	O02
P03	Qasimabad	700	O03

Owner

OwnerNo	OwnerName
O01	Farhan
O02	Ismail
O03	Asif