**EXPERIMENT NO: 05**

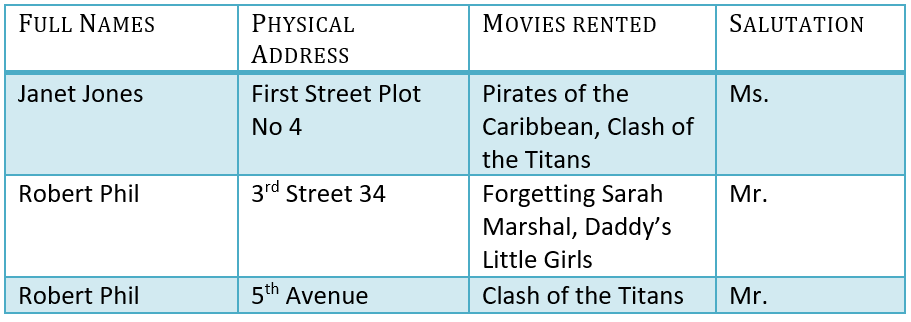
**Title**: Normalize any database from first normal form to Boyce-Codd Normal Form (BCNF).

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

**Theory:**

## Database Normalization With Examples

Database **Normalization Example** can be easily understood with the help of a case study. Assume, a video library maintains a database of movies rented out. Without any normalization in database, all information is stored in one table as shown below. Let’s understand Normalization database with normalization example with solution:



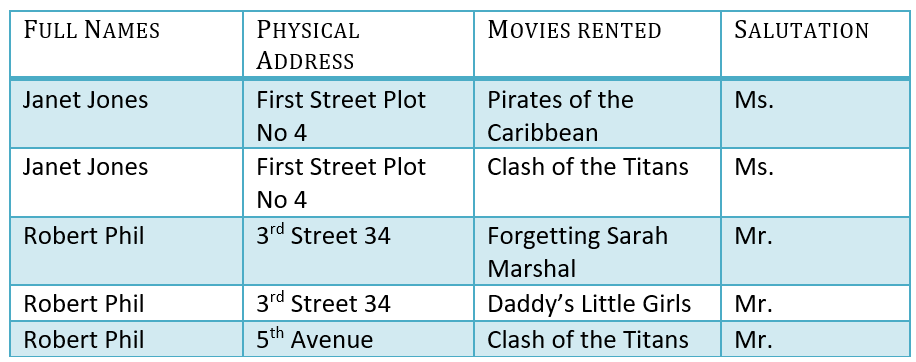
Here you see **Movies Rented column has multiple values.** Now let’s move into 1st Normal Forms:

## ****1NF (First Normal Form) Rules****

* Each table cell should contain a single value.
* Each record needs to be unique.

The above table in 1NF-

1NF Example



Example of 1NF in DBMS

Before we proceed let’s understand a few things —

## What is a KEY in SQL?

A **KEY in SQL** is a value used to identify records in a table uniquely. An SQL KEY is a single column or combination of multiple columns used to uniquely identify rows or tuples in the table. SQL Key is used to identify duplicate information, and it also helps establish a relationship between multiple tables in the database.

Note: Columns in a table that are NOT used to identify a record uniquely are called non-key columns.

### What is a Primary Key?

Primary Key in DBMS

A primary is a single column value used to identify a database record uniquely.

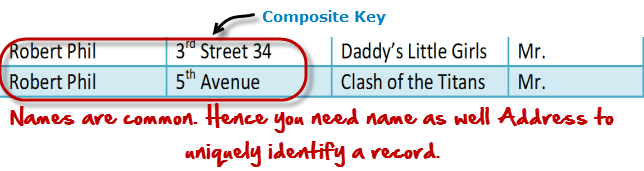
It has following attributes

* A [primary key](https://www.guru99.com/dbms-keys.html) cannot be NULL
* A primary key value must be unique
* The primary key values should rarely be changed
* The primary key must be given a value when a new record is inserted.

### What is Composite Key?

A composite key is a primary key composed of multiple columns used to identify a record uniquely

In our database, we have two people with the same name Robert Phil, but they live in different places.



Composite key in Database

Hence, we require both Full Name and Address to identify a record uniquely. That is a composite key.

Let’s move into second normal form 2NF

## 2NF (Second Normal Form) Rules

* Rule 1- Be in 1NF
* Rule 2- Single Column Primary Key that does not functionally dependant on any subset of candidate key relation

It is clear that we can’t move forward to make our simple database in 2nd Normalization form unless we partition the table above.



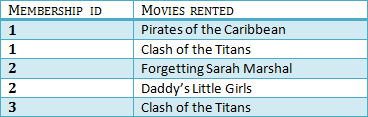


We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

We have introduced a new column called Membership\_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

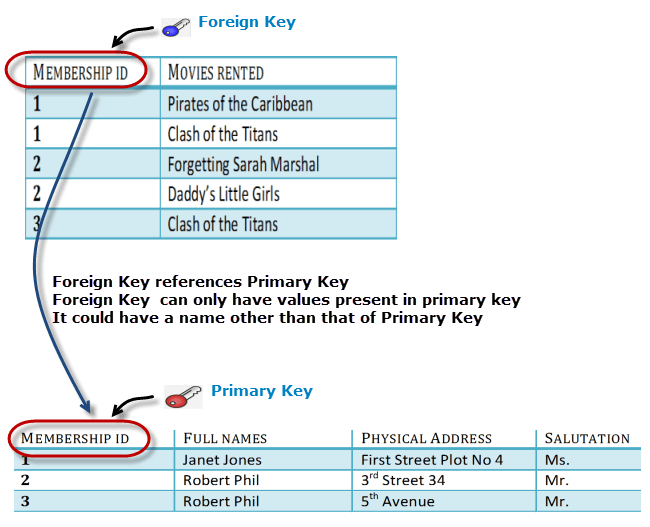
## Database – Foreign Key

In Table 2, Membership\_ID is the Foreign Key



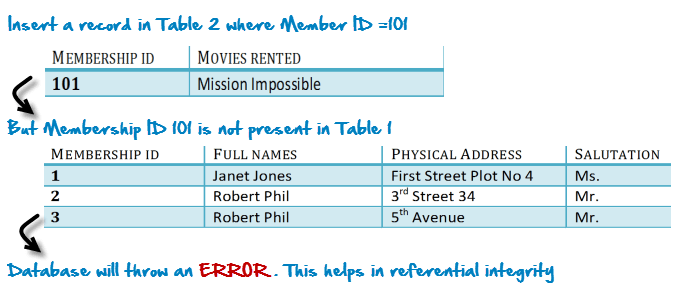
Foreign Key in DBMS

Foreign Key references the primary key of another Table! It helps connect your Tables

* A foreign key can have a different name from its primary key
* It ensures rows in one table have corresponding rows in another
* Unlike the Primary key, they do not have to be unique. Most often they aren’t
* Foreign keys can be null even though primary keys can not

## Why do you need a foreign key?

Suppose, a novice inserts a record in Table B such as



You will only be able to insert values into your foreign key that exist in the unique key in the parent table. This helps in referential integrity.

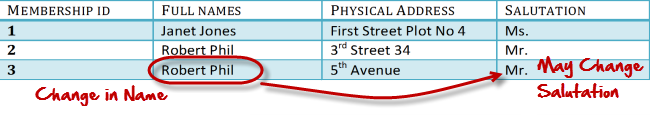
The above problem can be overcome by declaring membership id from Table2 as foreign key of membership id from Table1

Now, if somebody tries to insert a value in the membership id field that does not exist in the parent table, an error will be shown!

## What are transitive functional dependencies?

A transitive [functional dependency](https://www.guru99.com/dbms-functional-dependency.html) is when changing a non-key column, might cause any of the other non-key columns to change

Consider the table 1. Changing the non-key column Full Name may change Salutation.



Let’s move into 3NF

## 3NF (Third Normal Form) Rules

* Rule 1- Be in 2NF
* Rule 2- Has no transitive functional dependencies

To move our 2NF table into 3NF, we again need to again divide our table.

### 3NF Example

Below is a 3NF example in SQL database:







We have again divided our tables and created a new table which stores Salutations.

There are no transitive functional dependencies, and hence our table is in 3NF

In Table 3 Salutation ID is primary key, and in Table 1 Salutation ID is foreign to primary key in Table 3

Now our little example is at a level that cannot further be decomposed to attain higher normal form types of normalization in DBMS. In fact, it is already in higher normalization forms. Separate efforts for moving into next levels of normalizing data are normally needed in complex databases. However, we will be discussing next levels of normalisation in DBMS in brief in the following.

## BCNF (Boyce-Codd Normal Form)

Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate**Key.

Sometimes is BCNF is also referred as **3.5 Normal Form.**

## 4NF (Fourth Normal Form) Rules

If no database table instance contains two or more, independent and multivalued data describing the relevant entity, then it is in 4th Normal Form.

## 5NF (Fifth Normal Form) Rules

A table is in 5th Normal Form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

**Conclusion :-**

Thus we have Normalize any database from first normal form to Boyce-Codd Normal Form (BCNF).

.

**Sample Questions**

1. What is normalization in a database?
2. What are the four 4 types of database normalization?
3. What is normalization in database with example?
4. How do you normalize a database?
5. Why is 2NF used?
6. What is normalization and its types?
7. What is normalization?
8. Why do we use normalization?
9. What is normalization and its advantages?

**EXPERIMENT NO: 06**

**Title**:Implementation of Database connectivity with any object oriented language.

**Aim:**Implementation of Database connectivity with any object oriented language.

**Theory:**

**Java Database Connectivity**

JDBC means access to the Java Database. It's a step forward for ODBC (Open Database Connectivity). JDBC is a standard API specification for moving data from the frontend to the backend. This API consists of classes and interfaces written in Java.

This simply serves as an interface between your Java system and databases (not the one we use in Java) or network, i.e. this provides a connection between the two so that a developer can send Java code information and store it in the database for future use.    
  
The Java JDBC API allows Java applications to connect to relational databases such as MySQL, PostgreSQL, MS SQL Server, Oracle, H2 Database, etc. The JDBC API allows querying and updating relational databases, as well as calling stored procedures, and obtaining the database meta data.The Java JDBC API is part of the Java SE SDK core, making JDBC usable to all Java applications wishing to use it. Here is a diagram of a Java program connecting to a relational database using JDBC:

**JDBC is independent from SQL**

JDBC is not standardizing the SQL sent to the server. You, the JDBC API client, are writing the SQL. The SQL dialect used by the various databases varies slightly, so to be 100% independent of the database, you also need to be 100% independent of the database (i.e. use commands that are understood by all databases).

**JDBC is not for databases that are not relational**

The Java JDBC API is built to communicate with relational databases, meaning that you use standard SQL to connect with databases. The JDBC API is not intended for non-related servers like Mongo DB, Cassandra, Dynamo and so on. From a Java application you can use such databases, but you should see what drivers such databases provide for Java itself.

**JDBC is independent of the type of database**

The Java JDBC API standardizes how to connect to a database, how to execute queries against it, how to access a request output, how to execute database changes, how to call stored procedures, and how to get meta data from the server. Through "standardizing," I mean the repository's software looks the same across various products.Therefore, if your project needs this in the future, it will be much easier to switch to another database.

**Steps for Java program and database connectivity mentioned below:**

* **Load JDBC driver**

First of all, you need to load or register the driver before you use it in the program. You must register once in your program. In one of the two ways listed below, you can register a driver:

* **Class.forName()**

Here at runtime we load the class file of the driver into memory. No need to use fresh or create object. The instance below uses Class.forName() to load the Oracle driver.–

Class.forName("org.h2.Driver");

* **DriverManager.registerDriver():**

DriverManager is an integrated Java class with a register of static members. Here we call the driver class constructor at the moment of compilation. DriverManager.registerDriver() is used in the following example/instance to register the Oracle driver -

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver())

* **Connectionsestablisment**

We used code below after loading the driver to create connections:

 Connection connection = DriverManager.getConnection(DBurl,username,password)

* **username :** it is a username from which your sql command prompt can be accessed.
* **password :** It is a password from which your sql command prompt can be accessed.
* **connection :**It is an element of connection, i.e. it is a communication interface reference.
* **DBurl :** It is a Uniform Resource Locator. It can be created as follows:

String url = “ jdbc:oracle:thin:@localhost:1521:xe”

Where Oracle is the database used, the driver used is tiny, where the database is located, @localhost is the IP address, 1521 is the port number, and xe is the service provider. The 3 parameters above are String sort, which the programmer must declare before calling the function. You may refer to the use of this from the final code.

* **Statement creation**

 Whether the server needs to be modified or queried, you will need to build a JDBC Statement or JDBC PreparedStatement through which the update or request will occur.

Statement statement = connection.createStatement();

* **Query execution**

 Here comes the most important part, i.e. the query execution. Here's a query from the SQL. We here understand that we can have various kinds of questions. Some of them are the following:

* **Request to delete / modify / insert a table in a database.**
* **Request to collect or retrieve information from the database.**

The Statement interface's executeQuery) (method is used to perform queries to extract server values. This method returns the ResultSet item which can be used to obtain from the table all data / records.

The Statement interface executeUpdate(sql query) method is used to perform update / insert.

**Example:**

int result = stmt.executeUpdate(sql);

if (result == 1)

System.out.println("inserted successfully : "+sql);

else

System.out.println("insertion failed");

Here sql is sql query of the type String

* **Close database connections**

You have to open the connection again when you're finished with the JDBC server connection. In the request, but also within the database server, a JDBC connection could take up a large amount of sources. Therefore, after use it is important to close the connection to the database again. You close a JDBC relation through the method of closing). Here is an example of closing a JDBC connection:

connection.close();

**Conclusion :-**

Thus we have implement Database connectivity with any object oriented language.

**Sample Questions**

* 1. How does JDBC connect to database?
  2. How does JDBC communicate with database?
  3. What do you mean by database connectivity?
  4. What are the steps for database connectivity?