**SQL Tutorial**

SQL tutorial provides basic and advanced concepts of SQL. Our SQL tutorial is designed for beginners and professionals.

**SQL** (*Structured Query Language*) is used to perform operations on the records stored in database such as updating records, deleting records, creating and modifying tables, views etc.

SQL is just a query language, it is not a database. To perform SQL queries, you need to install any database for example Oracle, MySQL, MongoDB, PostGre SQL, SQL Server, DB2 etc.

**What is SQL**

* SQL stands for **Structured Query Language**.
* It is designed for managing data in a relational database management system (RDBMS).
* It is pronounced as S-Q-L or sometime **See-Qwell**.
* SQL is a database language, it is used for database creation, deletion, fetching rows and modifying rows etc.
* SQL is based on relational algebra and tuple relational calculus.

All DBMS like MySQL, Oracle, MS Access, Sybase, Informix, Postgres and SQL Server use SQL as standard database language.

Why SQL is required

SQL is required:

* To create new databases, tables and views
* To insert records in a database
* To update records in a database
* To delete records from a database
* To retrieve data from a database

What SQL does

* With SQL, we can query our database in a numbers of ways, using English-like statements.
* With SQL, user can access data from relational database management system.
* It allows user to describe the data.
* It allows user to define the data in database and manipulate it when needed.
* It allows user to create and drop database and table.
* It allows user to create view, stored procedure, function in a database.
* It allows user to set permission on tables, procedure and view.

# What is Database

A **database** is an organized collection of data.

**Database handlers** create database in such a way that only one set of software program provide access of data to all the users.

The **main purpose** of database is to operate large amount of information by storing, retrieving and managing.

There are many **dynamic websites** on the world wide web now a days which are handled through databases. For example, a model to checks the availability of rooms in a hotel. It is an example of dynamic website that uses database.

There are many **database available** like MySQL, Sybase, Oracle, Mango DB, Informix, Postgre, SQL Server etc.

**SQL** or Structured Query Language is used to perform operation on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

A cylindrical structure is used to display the image of a database.

What is RDBMS

**RDBMS** stands for *Relational Database Management Systems.*

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.

It is called Relational Data Base Management System (RDBMS) because it is based on relational model introduced by E.F. Codd.

How it works

Data is represented in terms of tuples (rows) in RDBMS.

Relational database is most commonly used database. It contains number of tables and each table has its own primary key.

Due to a collection of organized set of tables, data can be accessed easily in RDBMS.

**Brief History of RDBMS**

During 1970 to 1972, E.F. Codd published a paper to propose the use of relational database model.

RDBMS is originally based on that E.F. Codd's relational model invention.

**What is table**

The RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data.

A table is the simplest example of data storage in RDBMS.

Let's see the example of student table.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **AGE** | **COURSE** |
| 1 | Ajeet | 24 | B.Tech |
| 2 | Aryan | 20 | C.A |
| 3 | Mahesh | 21 | BCA |
| 4 | Ratan | 22 | MCA |
| 5 | Vimal | 26 | BSC |

**What is field**

Field is a smaller entity of the table which contains specific information about every record in the table. In the above example, the field in the student table consist of id, name, age, course.

**What is row or record**

A row of a table is also called record. It contains the specific information of each individual entry in the table. It is a horizontal entity in the table. For example: The above table contains 5 records.

Let's see one record/row in the table.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Ajeet | 24 | B.Tech |

What is column

A column is a vertical entity in the table which contains all information associated with a specific field in a table. For example: "name" is a column in the above table which contains all information about student's name.

|  |
| --- |
| Ajeet |
| Aryan |
| Mahesh |
| Ratan |
| Vimal |

NULL Values

The NULL value of the table specifies that the field has been left blank during record creation. It is totally different from the value filled with zero or a field that contains space.

**Data Integrity**

There are the following categories of data integrity exist with each RDBMS:

**Entity integrity**: It specifies that there should be no duplicate rows in a table.

**Domain integrity**: It enforces valid entries for a given column by restricting the type, the format, or the range of values.

**Referential integrity**: It specifies that rows cannot be deleted, which are used by other records.

**User-defined integrity**: It enforces some specific business rules that are defined by users. These rules are different from entity, domain or referential integrity.

**Difference between DBMS and RDBMS**

Although DBMS and RDBMS both are used to store information in physical database but there are some remarkable differences between them.

The main differences between DBMS and RDBMS are given below:

|  |  |  |
| --- | --- | --- |
| **No.** | **DBMS** | **RDBMS** |
| 1) | DBMS applications store **data as file**. | RDBMS applications store **data in a tabular form**. |
| 2) | In DBMS, data is generally stored in either a hierarchical form or a navigational form. | In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables. |
| 3) | **Normalization is not** present in DBMS. | **Normalization is** present in RDBMS. |
| 4) | DBMS does **not apply any security** with regards to data manipulation. | RDBMS **defines the integrity constraint** for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property. |
| 5) | DBMS uses file system to store data, so there will be **no relation between the tables**. | in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well. |
| 6) | DBMS has to provide some uniform methods to access the stored information. | RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information. |
| 7) | DBMS **does not support distributed database**. | RDBMS **supports distributed database**. |
| 8) | DBMS is meant to be for small organization and **deal with small data**. it supports **single user**. | RDBMS is designed to **handle large amount of data**. it supports **multiple users**. |
| 9) | Examples of DBMS are file systems, **xml** etc. | Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc. |

After observing the differences between DBMS and RDBMS, you can say that RDBMS is an extension of DBMS. There are many software products in the market today who are compatible for both DBMS and RDBMS. Means today a RDBMS application is DBMS application and vice-versa.

SQL Syntax

SQL follows some unique set of rules and guidelines called syntax. Here, we are providing all the basic SQL syntax.

* **SQL** is not case sensitive. Generally SQL keywords are written in uppercase.
* SQL statements are dependent on text lines. We can place a single SQL statement on one or multiple text lines.
* You can perform most of the action in a database with SQL statements.
* SQL depends on relational algebra and tuple relational calculus.

SQL statement

SQL statements are started with any of the SQL commands/keywords like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP etc. and the statement ends with a semicolon (;).

Example of SQL statement:

1. **SELECT** "column\_name" **FROM** "table\_name";

Why semicolon is used after SQL statements:

Semicolon is used to separate SQL statements. It is a standard way to separate SQL statements in a database system in which more than one SQL statements are used in the same call.

In this tutorial, we will use semicolon at the end of each SQL statement.

SQL Commands

These are the some important SQL command:

**SELECT**: it extracts data from a database.

**UPDATE**: it updates data in database.

**DELETE**: it deletes data from database.

**CREATE TABLE**: it creates a new table.

**ALTER TABLE**: it is used to modify the table.

**DROP TABLE**: it deletes a table.

**CREATE DATABASE**: it creates a new database.

**ALTER DATABASE**: It is used to modify a database.

**INSERT INTO**: it inserts new data into a database.

**CREATE INDEX**: it is used to create an index (search key).

**DROP INDEX**: it deletes an index.

[**next →**](https://www.javatpoint.com/sql-operators)[**← prev**](https://www.javatpoint.com/sql-syntax)

SQL Data Types

The SQL data type defines a kind of value that a column can contain.

In a database table, every column is required to have a name and a data type.

These are the general data types in SQL.

|  |  |  |
| --- | --- | --- |
| **Data-type** | **Syntax** | **Explanation** |
| Integer | INTEGER | The integer data type is used to specify an integer value. |
| Smallint | SMALLINT | The smallint data type is used to specify small integer value. |
| Numeric | NUMERIC(P,S) | It specifies a numeric value. Here 'p' is precision value and 's' is scale value. |
| Real | REAL | The real integer is used to specify a single precision floating point number. |
| Decimal | DECIMAL(P,S) | It specifies a decimal value. Here 'p' is precision value and 's' is scale value. |
| Double precision | DOUBLE PRECISION | It specifies double precision floating point number. |
| Float | FLOAT(P) | It specifies floating-point value e.g. 12.3, 4.5 etc. Here, 'p' is precision value. |
| Character | CHAR(X) | Here, 'x' is the character's number to store. |
| Character varying | VARCHAR2(X) | Here, 'x' is the character's number to store |
| Bit | BIT(X) | Here, 'x' is the number of bits to store |
| Bit varying | BIT VARYING(X) | Here, 'x' is the number of bits to store (length can vary up to x). |
| Date | DATE | It stores year, month and days values. |
| Time | TIME | It stores hour, minute and second values |
| Timestamp | TIMESTAMP | The timestamp data type is used to store year, month, day, hour, minute and second values. |
| Time with time zone | TIME WITH TIME ZONE | It is exactly same as time but also store an offset from UTC of the time specified. |
| Timestamp with time zone | TIMESTAMP with TIME ZONE | It is same as timestamp but also stores an offset from UTC of the time specified. |

SQL Operators

SQL statements generally contain some reserved words or characters that are used to perform operations such as comparison and arithmetical operations etc. These reserved words or characters are known as operators.

Generally there are three types of operators in SQL:

1. SQL Arithmetic Operators
2. SQL Comparison Operators
3. SQL Logical Operators

SQL Arithmetic Operators:

Let's assume two variables "a" and "b". Here "a" is valued 50 and "b" valued 100.

**Example:**

|  |  |  |
| --- | --- | --- |
| **Operators** | **Descriptions** | **Examples** |
| + | It is used to add containing values of both operands | a+b will give 150 |
| - | It subtracts right hand operand from left hand operand | a-b will give -50 |
| \* | It multiply both operand?s values | a\*b will give 5000 |
| / | It divides left hand operand by right hand operand | b/a will give 2 |
| % | It divides left hand operand by right hand operand and returns reminder | b%a will give 0 |

SQL Comparison Operators:

Let's take two variables "a" and "b" that are valued 50 and 100.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Examine both operands value that are equal or not,if yes condition become true. | (a=b) is not true |
| != | This is used to check the value of both operands equal or not,if not condition become true. | (a!=b) is true |
| < > | Examines the operand?s value equal or not, if values are not equal condition is true | (a<>b) is true |
| > | Examine the left operand value is greater than right Operand, if yes condition becomes true | (a>b) is not true |
| < | Examines the left operand value is less than right Operand, if yes condition becomes true | (a<="" td=""> |
| >= | Examines that the value of left operand is greater than or equal to the value of right operand or not,if yes condition become true | (a>=b) is not true |
| <= | Examines that the value of left operand is less than or equal to the value of right operand or not, if yes condition becomes true | (a<=b) is true |
| !< | Examines that the left operand value is not less than the right operand value | (a!<="" td=""> |
| !> | Examines that the value of left operand is not greater than the value of right operand | (a!>b) is true |

SQL Logical Operators:

This is the list of logical operators used in SQL.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| ALL | this is used to compare a value to all values in another value set. |
| AND | this operator allows the existence of multiple conditions in an SQL statement. |
| ANY | this operator is used to compare the value in list according to the condition. |
| BETWEEN | this operator is used to search for values, that are within a set of values |
| IN | this operator is used to compare a value to that specified list value |
| NOT | the NOT operator reverse the meaning of any logical operator |
| OR | this operator is used to combine multiple conditions in SQL statements |
| EXISTS | the EXISTS operator is used to search for the presence of a row in a specified table |
| LIKE | this operator is used to compare a value to similar values using wildcard operator |

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

These rules can be applied on any database system that manages stored data using only its relational capabilities. This is a foundation rule, which acts as a base for all the other rules.

## Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

## Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

## Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

## Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

## Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

## Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

## Rule 7: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

## Rule 8: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

## Rule 9: Logical Data Independence

The logical data in a database must be independent of its user’s view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

## Rule 10: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

## Rule 11: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

## Rule 12: Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

**ACID Properties**

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain **A**tomicity, **C**onsistency, **I**solation, and **D**urability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

* **Atomicity** − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
* **Consistency** − The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
* **Durability** − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
* **Isolation** − In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

**What is Normalization?**

* Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.
* It divides larger tables to smaller tables and links them using relationships.

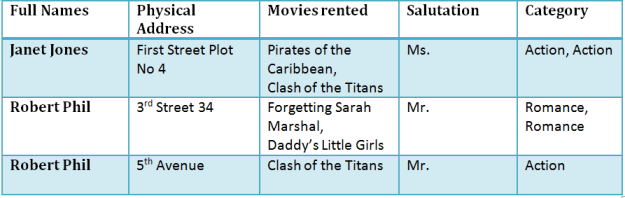
The inventor of the relational model Edgar Codd proposed the theory of normalization with the introduction of First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined with Raymond F. Boyce to develop the theory of Boyce-Codd Normal Form.

Theory of Data Normalization in SQL is still being developed further. For example, there are discussions even on 6th Normal Form. **However, in most practical applications, normalization achieves its best in 3rd Normal Form**. The evolution of Normalization theories is illustrated below-

[What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples](https://www.guru99.com/images/NormalizationProcess(1).png)

### Database Normalization Examples -

Assume a video library maintains a database of movies rented out. Without any normalization, all information is stored in one table as shown below.

[[](https://www.guru99.com/images/NormalizationTable1.png)](https://www.guru99.com/images/NormalizationTable1.png)

[Table 1](https://www.guru99.com/images/NormalizationTable1.png)

Here you see **Movies Rented column has multiple values**.

## Database Normal Forms

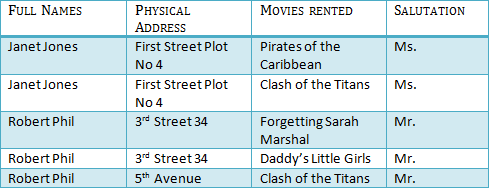
Now let's move into 1st Normal Forms

## ****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

[[](https://www.guru99.com/images/1NF.png)](https://www.guru99.com/images/1NF.png)

[Table 1: In 1NF Form](https://www.guru99.com/images/1NF.png)

Before we proceed let's understand a few things --

## What is a KEY?

A KEY is a value used to identify a record in a table uniquely. A KEY could be a single column or combination of multiple columns

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

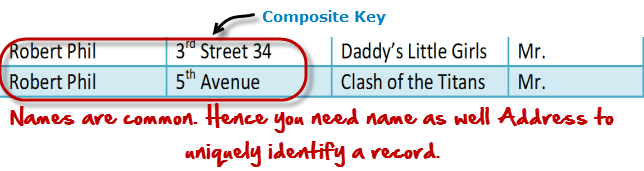
What is a Primary Key?

|  |  |
| --- | --- |
| [What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples](https://www.guru99.com/images/PrimaryKey.png) | A primary is a single column value used to identify a database record uniquely.  It has following attributes   * A primary key cannot be NULL * A primary key value must be unique * The primary key values cannot 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.

[](https://www.guru99.com/images/CompositeKey.png)

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

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

[[](https://www.guru99.com/images/Table2.png)](https://www.guru99.com/images/Table2.png)

[Table 1](https://www.guru99.com/images/Table2.png)

[[](https://www.guru99.com/images/Table1.png)](https://www.guru99.com/images/Table1.png)

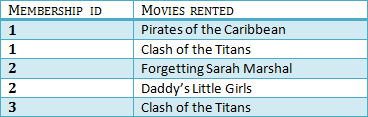
[Table 2](https://www.guru99.com/images/Table1.png)

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.

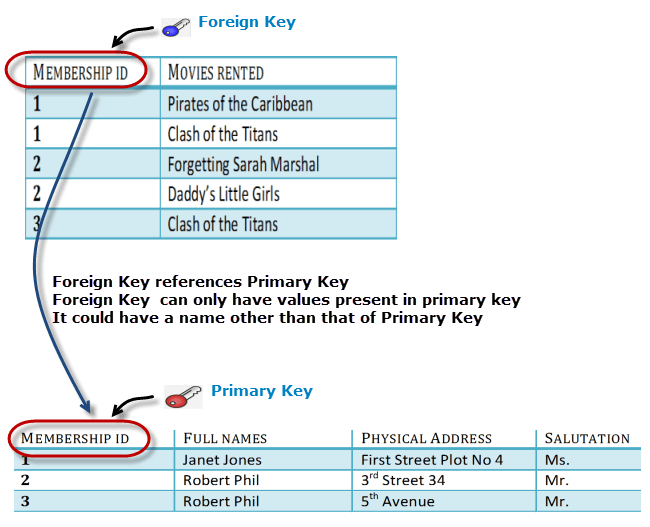
We have introduced a new column called Membership\_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

## Database - Foreign Key

In Table 2, Membership\_ID is the Foreign Key

[](https://www.guru99.com/images/foreign_key_table.png)

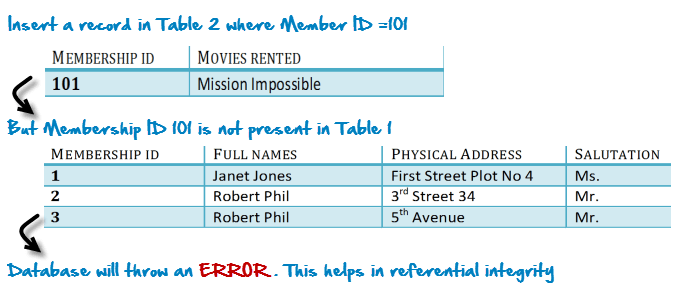
|  |  |
| --- | --- |
| [What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples](https://www.guru99.com/images/ForeignKey.png) | Foreign Key references the primary key of another Table! It helps connect your Tables   * A foreign key can have a different name from its primary key * It ensures rows 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 |

[](https://www.guru99.com/images/ForeignKeyRelationWithPrimary.png)

Why do you need a foreign key?

Suppose an idiot 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.

[](https://www.guru99.com/images/WhyDataBaseIsImportant.png)

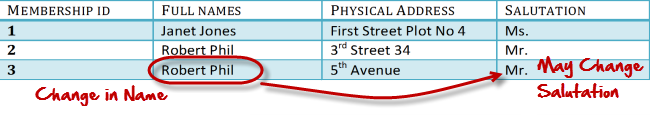
The above problem can be overcome by declaring membership id  from Table2  as foreign key of membership id from Table1

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 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.

[](https://www.guru99.com/images/transitive_functional_dependencies.png)

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

[[](https://www.guru99.com/images/2NFTable1.png)](https://www.guru99.com/images/2NFTable1.png)

[TABLE 1](https://www.guru99.com/images/2NFTable1.png)

[[](https://www.guru99.com/images/2NFTable2.png)](https://www.guru99.com/images/2NFTable2.png)

[Table 2](https://www.guru99.com/images/2NFTable2.png)

[[](https://www.guru99.com/images/2NFTable3.png)](https://www.guru99.com/images/2NFTable3.png)

[Table 3](https://www.guru99.com/images/2NFTable3.png)

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 forms of normalization. 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 normalizations in brief in the following.

## Boyce-Codd Normal Form (BCNF)

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.

### 6NF (Sixth Normal Form) Proposed

6th Normal Form is not standardized, yet however, it is being discussed by database experts for some time. Hopefully, we would have a clear & standardized definition for 6th Normal Form in the near future...

That's all to Normalization!!!