Setting Up a MySQL User Account

For adding a new user to MySQL, you just need to add a new entry to the **user** table in the database **mysql**.

The following program is an example of adding a new user **guest** with SELECT, INSERT and UPDATE privileges with the password **guest123;** the SQL query is:

*USE mySql;*

*INSERT INTO user*

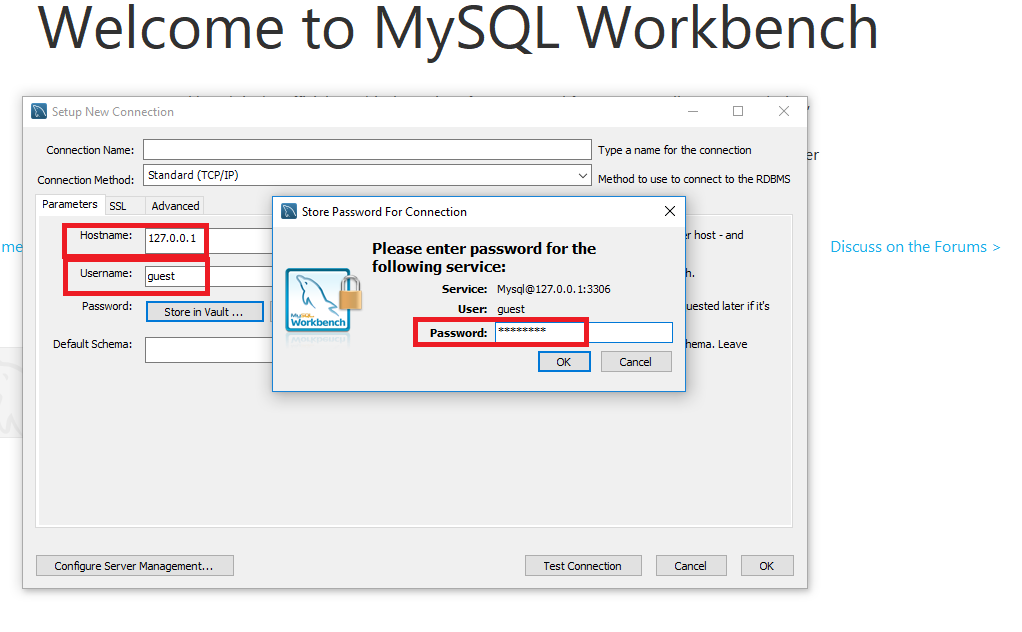
*(host, user, password,*

*select\_priv, insert\_priv, update\_priv)*

*VALUES ('localhost', 'guest',*

*PASSWORD('guest123'), 'Y', 'Y', 'Y');*

Now test the connection with this new user:



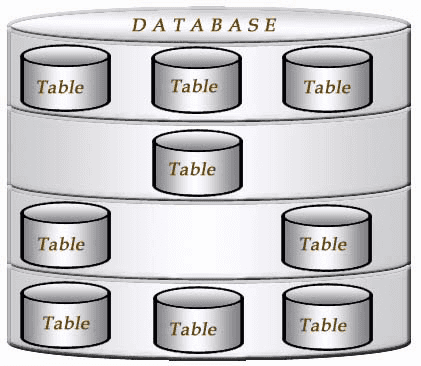
What is a database?

When an amount of data is stored in an organized way, that is called a database.   
In computers, a database is managed by a software called Database Management System.

**What is a table?**

A table is a set of data values. These values are organized using vertical columns and horizontal rows. Columns are identified by their names.

**Pictorial representation of a database with tables**



**MySQL create database**

In MySQL, CREATE DATABASE statement creates a database with the given name.

To use this statement, you must have the CREATE privilege for the database. You will get an error if the database exists and you did not specify IF NOT EXISTS clause.

**Syntax:**

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] database\_name

[create\_specification] ...

create\_specification:

[DEFAULT] CHARACTER SET [=] charset\_name

| [DEFAULT] COLLATE [=] collation\_name

Where

* database\_name is the name of the new database.
* create\_specification options specify database characteristics.
* The CHARACTER SET clause specifies the default database character set.

**Example:**

The following statement creates 'bookinfo' database.

CREATE DATABASE bookinfo;

The database names are case sensitive under Unix but this restriction does not apply in Windows.

This is also true for table names. The best practice is to use same letter case for creating a database as well as a table.

**MySQL: Setting the Default Database**

MySQL **use** statement is used to change the database from default to the given database.

**Syntax:**

use [database\_name];

**MySQL show database**

SHOW statement displays a list of currently existing databases on the server.

**Syntax:**

SHOW [expression];

**Example:**

The following MySQL statement will show the current database.

SHOW databases;

**Column types**

1. A data type specifies a particular type of data, such as integer, floating-point, Boolean etc.

2. A data type also specifies the possible values for that type, the operations that can be performed on that type and the way the values of that type are stored.

**Integer types**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type** | **Length  in Bytes** | **Minimum Value (Signed)** | **Maximum Value (Signed)** | **Minimum Value (Unsigned)** | **Maximum Value (Unsigned)** |
| TINYINT | 1 | -128 | 127 | 0 | 255 |
| SMALLINT | 2 | -32768 | 32767 | 0 | 65535 |
| MEDIUMINT | 3 | -8388608 | 8388607 to | 0 | 16777215 |
| INT | 4 | -2147483648 | 2147483647 | 0 | 4294967295 |
| BIGINT | 8 | -9223372036854775808 | 92233720368 54775807 | 0 | 184467440737 09551615 |

**Floating-Point Types**

|  |  |
| --- | --- |
| **Types** | **Description** |
| FLOAT | A precision from 0 to 23 results in a four-byte single-precision FLOAT column |
| DOUBLE | A precision from 24 to 53 results in an eight-byte double-precision DOUBLE column. |

MySQL allows a nonstandard syntax: FLOAT(M,D) or REAL(M,D) or DOUBLE PRECISION(M,D). Here values can be stored up to M digits in total where D represents the decimal point. For example, a column defined as FLOAT(8,5) will look like -999.99999. MySQL performs rounding when storing values, so if you insert 999.00009 into a FLOAT(7,4) column, the approximate result is 999.0001.

Following table shows the required storage and range (maximum and minimum value for signed and unsigned integer) for each floating-point type.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type** | **Length in Bytes** | **Minimum Value (Signed)** | **Maximum Value (Signed)** | **Minimum Value (Unsigned)** | **Maximum Value (Unsigned)** |
| FLOAT | 4 | -3.402823466E+38 | -1.175494351E-38 | 1.175494351E-38 | 3.402823466E+38 |
| DOUBLE | 8 | -1.7976931348623 157E+ 308 | -2.22507385850720 14E- 308 | 0, and  2.22507385850720 14E- 308 | 1.797693134862315 7E+ 308 |

**Fixed-Point Types**

Fixed-Point data types are used to preserve exact precision, for example with currency data. In MySQL DECIMAL and NUMERIC types store exact numeric data values. MySQL 5.6 stores DECIMAL values in binary format.

In standard SQL the syntax DECIMAL(5,2)  (where 5 is the precision and 2 is the scale. ) be able to store any value with five digits and two decimals. Therefore the value range will be from -999.99 to 999.99. The syntax DECIMAL(M) is equivalent to DECIMAL(M,0). Similarly, the syntax DECIMAL is equivalent to DECIMAL(M,0). MySQL supports both of these variant forms of DECIMAL syntax. The default value of M is 10. If the scale is 0, DECIMAL values contain no decimal point or fractional part.  
The maximum number of digits for DECIMAL is 65, but the actual range for a given DECIMAL column can be constrained by the precision or scale for a given column.

**Bit Value Types**

The BIT data type is used to store bit-field values. A type of BIT(N) enables storage of N-bit values. N can range from 1 to 64.  
To specify bit values, b'value' notation can be used. value is a binary value written using zeros and ones. For example, b'111' and b'10000000' represent 7 and 128, respectively

**Numeric type attributes**

MySQL supports an extension for optionally specifying the display width of integer data types in parentheses following the base keyword for the type

|  |  |
| --- | --- |
| **Types** | **Description** |
| TYPE(N) | Where N is an integer and display width of the type is upto N digits. |
| ZEROFILL | The default padding of spaces is replaced with zeros. So, for a column INT(3) ZEROFILL, 7 is displayed as 007. |

**MySQL Date and Time Types**

The date and time types represent DATE, TIME, DATETIME, TIMESTAMP, and YEAR. Each type has a range of valid values, as well as a “zero” value.

**DATETIME, DATE, and TIMESTAMP Types**

|  |  |  |  |
| --- | --- | --- | --- |
| **Types** | **Description** | **Display Format** | **Range** |
| DATETIME | Use when you need values containing both date and time information. | YYYY-MM-DD HH:MM:SS | '1000-01-01 00:00:00'  To  '9999-12-31 23:59:59'. |
| DATE | Use when you need only date information. | YYYY-MM-DD | '1000-01-01'  to  '9999-12-31'. |
| TIMESTAMP | Values are converted from the current time zone to UTC while storing and converted back from UTC to the current time zone when retrieved. | YYYY-MM-DD HH:MM:SS | '1970-01-01 00:00:01' UTC  to  '2038-01-19 03:14:07' UTC |

**Time Type**

MySQL fetches and displays TIME values in 'HH:MM:SS' format or 'HHH:MM:SS' format The range of. TIME values from '-838:59:59' to '838:59:59'. The hours part may be rather large because not only the TIME type can be used to represent the time of day, i.e. less than 24 hours, but also the passed time or a time of interval between two events.

The TIME values in MySQL can be recognized in many different formats, some of which can include a trailing fractional seconds part in up to 6 digits microseconds precision. The range for TIME values is '-838:59:59.000000' to '838:59:59.000000'.

MySQL explains abbreviated TIME values with colons as the time of the day. Suppose '09:10' means '09:10:00', not '00:09:10'. MySQL understands the abbreviated values without colons as that, the two rightmost digits represent seconds. For example, we think of '0910' and 0910 as meaning '09:10:00', i.e. 10 minutes after 9 o'clock but the reality is MySQL understand them as '00:09:10', i.e. 9 minutes and 10 seconds. So, be careful about using abbreviated time in MySQL.

By default, the values of time that lie outside the TIME are converted to the valid range of time values. For example, '-930:00:00' and '930:00:00' are converted to '-838:59:59' and '838:59:59'. Invalid TIME values are converted to '00:00:00', because '00:00:00' is itself a valid TIME value in MySQL.

**Year Type**

The YEAR type is a 1-byte type used to represent year values. It can be declared as YEAR(2) or YEAR(4) to specify a display width of two or four characters. If no width is given the default is four characters

YEAR(4) and YEAR(2) have different display format but have the same range of values.   
For 4-digit format, MySQL displays YEAR values in YYYY format, with a range of 1901 to 2155, or 0000.   
For 2-digit format, MySQL displays only the last two (least significant) digits; for example, 70 (1970 or 2070) or 69 (2069).

You can specify YEAR values in a variety of formats:

|  |  |
| --- | --- |
| **String length** | **Range** |
| 4-digit string | '1901' to '2155'. |
| 4-digit number | 1901 to 2155. |
| 1- or 2-digit string | '0' to '99'. Values in the ranges '0' to '69' and '70' to '99' are converted to YEAR values in the ranges 2000 to 2069 and 1970 to 1999. |
| 1- or 2-digit number | 1 to 99. Values in the ranges 1 to 69 and 70 to 99 are converted to YEAR values in the ranges 2001 to 2069 and 1970 to 1999. |

**String Types**

The string types are CHAR, VARCHAR, etc…

**CHAR and VARCHAR Types**

The CHAR and VARCHAR types are similar but differ in the way they are stored and retrieved. They also differ in maximum length and in whether trailing spaces are retained.

|  |  |  |  |
| --- | --- | --- | --- |
| **Types** | **Description** | **Display Format** | **Range in characters** |
| CHAR | Contains non-binary strings. Length is fixed as you declare while creating a table. When stored, they are right-padded with spaces to the specified length. | Trailing spaces are removed. | The length can be any value from 0 to 255. |
| VARCHAR | Contains non-binary strings. Columns are variable-length strings. | As stored. | A value from 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in 5.0.3 and later versions. |

CREATE TABLE

MySQL CREATE TABLE is used to create a table within a database.

**Syntax :**

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl\_name

(create\_definition,...)

[table\_options]

[partition\_options]

**index\_col\_name:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| col\_name | Name of the column |
| length | Length of the column |
| ASC | Sorting order in ascending. |
| DESC | Sorting order in descending. |

**reference \_definition:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| REFERENCES | The REFERENCES clauses is used only when specified as part of a separate FOREIGN KEY specification. |

**Examples: MySQL CREATE TABLE**

The following statement will create 'NewPublisher' table which have columns like following:

CREATE TABLE publisher(

pub\_id varchar(8),

pub\_name varchar(50),

pub\_city varchar(25),

country varchar(25),

country\_office varchar(25),

no\_of\_branch int(3),

estd date

);

**MySQL CREATE TABLE when not EXISTS**

The keywords IF NOT EXISTS is used to prevent an error from occurring if the table exists. The keywords IF NOT EXISTS will not verify whether the existing table is of the same structure indicated by the CREATE TABLE statement. The following statement creates a table newauthor if the table 'newauthor' does not exist with the following column name, type, length and default value

CREATE TABLE IF NOT EXISTS newauthor

(aut\_id varchar(8),

aut\_name varchar(50),

country varchar(25),

home\_city varchar(25) NULL);

**MySQL CREATE with LIKE or copy table structure**

MySQL allows you to create a table identical to another by using LIKE. The following MySQL statement will create a table 'author\_copy' whose structure is identical to the table 'author'.

Sample table: author

CREATE TABLE author\_copy LIKE newauthor;

**MySQL CREATE TABLE with SELECT or copy a table**

MySQL AS SELECT allows you to copy the data (either selective or total) or table structure to a new table. The following MySQL statement will create a new table 'author\_copy' with the same structure and data of 'author' table .

CREATE TABLE author\_copy

AS SELECT \*

FROM newauthor;

**MySQL CREATE TABLE with ENUM data type**

While creating a table in MySQL, using ENUM following a column name, you can define that the character set for a column will be enumerated. The following MySQL statement will create a table 'testtable' using the data type ENUM.

CREATE TABLE testtable(book\_lang ENUM('english', 'german','french') );

**MySQL CREATE TABLE with specific data type**

While creating a table in MySQL, using the specific data type name, following a column name, you can define the character set for a column. The following MySQL statement will create a table 'testtable' using the data type as specified.

CREATE TABLE testtable ( string1 VARCHAR(4), string2 CHAR(4));

**MySQL DESCRIBE statement**

MySQL DESCRIBE statement is used to show the structure of the created table.

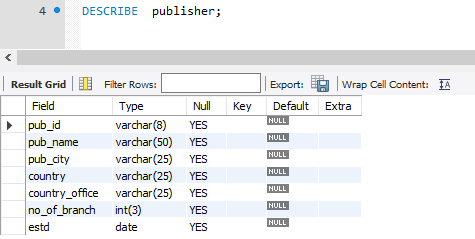
**Syntax:**

DESCRIBE [table\_name];

The following statement will display the structure of a given table.

DESCRIBE publisher;

The result is:



## CONSTRAINT

MySQL CONSTRAINT is used to define rules to allow or restrict what values can be stored in columns. The purpose of inducing constraints is to enforce the integrity of a database.

MySQL CONSTRAINTS are used to limit the type of data that can be inserted into a table.

MySQL CONSTRAINTS can be classified into two types - column level and table level.

The column level constraints can apply only to one column where as table level constraints are applied to the entire table.

MySQL CONSTRAINT is declared at the time of creating a table.

**MySQL CONSTRAINTs are :**

* NOT NULL
* UNIQUE
* PRIMARY KEY
* FOREIGN KEY
* CHECK
* DEFAULT

|  |  |
| --- | --- |
| **CONSTRAINT** | **DESCRIPTION** |
| NOT NULL | In MySQL NOT NULL constraint allows to specify that a column can not contain any NULL value. MySQL NOT NULL can be used to CREATE and ALTER a table. |
| UNIQUE | The UNIQUE constraint in MySQL does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table. |
| PRIMARY KEY | A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint creates a unique index for accessing the table faster. |
| FOREIGN KEY | A FOREIGN KEY in MySQL creates a link between two tables by one specific column of both tables. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY. |
| CHECK | A CHECK constraint controls the values in the associated column. The CHECK constraint determines whether the value is valid or not from a logical expression. |
| DEFAULT | In a MySQL table, each column must contain a value ( including a NULL). While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT. |

**MySQL CREATE TABLE with NULL CONSTRAINT**

Using the default value as NOT NULL, while creating a MySQL table, it can be enforced that a column in a table is not allowed to store NULL values.

**Example**

If you want to create a table 'newauthor' where no columns are allowed to store NULL VALUES the following statement can be used.

CREATE TABLE IF NOT EXISTS newauthor

(aut\_id varchar(8) NOT NULL,

aut\_name varchar(50) NOT NULL,

country varchar(25) NOT NULL,

home\_city varchar(25) NOT NULL );

**MySQL CREATE TABLE to check values with CHECK CONSTRAINT**

Adding a CHECK CONSTRAINT on a column of a table, you can limit the range of values allowed to be stored in that column.

**Example**

If you want to create a table 'newbook\_mast' with a PRIMARY KEY on 'book \_id' column, a unique constraint on 'isbn\_no' column and a set the no\_page in such, that it would hold values more than zero only, the following statement can be used.

CREATE TABLE IF NOT EXISTS

newbook\_mast (book\_id varchar(15) NOT NULL UNIQUE,

book\_name varchar(50) ,

isbn\_no varchar(15) NOT NULL UNIQUE ,

cate\_id varchar(8) ,

aut\_id varchar(8) ,

pub\_id varchar(8) ,

dt\_of\_pub date ,

pub\_lang varchar(15) ,

no\_page decimal(5,0)

CHECK(no\_page>0) ,

book\_price decimal(8,2) ,

PRIMARY KEY (book\_id)

);

Here in the above MySQL statement will create a table 'newbook\_mast' with a PRIMARY KEY on 'book \_id' column, unique constraint on 'isbn\_no' column and adding CHECK(no\_page>0) will set the no\_page in such, that it would hold values more than zero only.

**MySQL CREATE TABLE with CHECK CONSTRAINT using IN operator**

MySQL CHECK CONSTRAINT can be applied to a column of a table, to set a limit for storing values within a range, along with IN operator.

**Example**

If you want to create a table 'newauthor' with a PRIMARY KEY on a combination of two columns (aut\_id,home\_city) and checking a limit value for the column country are 'USA','UK' and 'India', the following statement can be used.

CREATE TABLE IF NOT EXISTS

newauthor(aut\_id varchar(8) NOT NULL ,

aut\_name varchar(50) NOT NULL,

country varchar(25) NOT NULL CHECK (country IN ('USA','UK','India')),

home\_city varchar(25) NOT NULL,

PRIMARY KEY (aut\_id,home\_city));

Here in the above MySQL statement will create a table 'newauthor' with a PRIMARY KEY on a combination of two columns (aut\_id,home\_city) and the value for the column country has been limited by using IN operator.

**MySQL CREATE TABLE with CHECK CONSTRAINT and LIKE operator**

MySQL CHECK CONSTRAINT can be applied to a column of a table, to set a limit for storing values within a range, along with LIKE operator.

**Example**

The MySQL statement stated below will create a table 'newbook\_mast' with a PRIMARY KEY on 'book\_id' column and a CHECK constraint to limit value to be stored for the column dt\_of\_pub along with LIKE operator and another CHECK constraint to column no\_page (without using LIKE operator).

CHECK (dt\_of\_pub LIKE '--/--/----') checks whether the format of the date to be stored is the column dt\_of\_pub is like '--/--/----'.

Here is the statement.

CREATE TABLE IF NOT EXISTS newbook\_mast

( book\_id varchar(15) NOT NULL UNIQUE,

book\_name varchar(50) ,

isbn\_no varchar(15) NOT NULL UNIQUE ,

cate\_id varchar(8) ,

aut\_id varchar(8) ,

pub\_id varchar(8) ,

dt\_of\_pub date CHECK (dt\_of\_pub LIKE '--/--/----'),

pub\_lang varchar(15) ,

no\_page decimal(5,0) CHECK(no\_page>0) ,

book\_price decimal(8,2) ,

PRIMARY KEY (book\_id) );

**MySQL CREATE TABLE with AND and OR operator and CHECK CONSTRAINT**

MySQL CHECK CONSTRAINT can be applied to a column of a table, to set a limit for storing values within a range, along with AND and OR operator.

**Example**

The MySQL statement stated below will create a table 'newpublisher' with a PRIMARY KEY on 'pub\_id' column and CHECK constraint along with AND and OR operator for country and pub\_city columns.

CHECK ((country='India' AND pub\_city='Mumbai') OR (country='India' AND pub\_city='New Delhi')) checks whether (i)country is INDIA and pub\_city is Mumbai OR (ii) country is INDIA and pub\_city is New Delhi.

MySQL Manual says "The CHECK clause is parsed but ignored by all storage engines."

Here is the statement.

CREATE TABLE IF NOT EXISTS newpublisher

(pub\_id varchar(8) ,

pub\_name varchar(50),

pub\_city varchar(25) ,

country varchar(25) ,

country\_office varchar(25) ,

no\_of\_branch int(3),

estd date

CHECK ((country='India' AND pub\_city='Mumbai')

OR (country='India' AND pub\_city='New Delhi')) ,

PRIMARY KEY (pub\_id) );

**MySQL UNIQUE CONSTRAINT**

The UNIQUE constraint creates an index such that, all values in the index column must be unique. An error occurs when any body tries to add a new row with a key value that already exists in that row.

**Example**

The MySQL statement stated below will create a table 'newauthor' with a column 'aut\_id' which will store unique values only since UNIQUE (aut\_id) is used.

CREATE TABLE IF NOT EXISTS

newauthor(aut\_id varchar(8) NOT NULL ,

aut\_name varchar(50)

NOT NULL,

country varchar(25) NOT NULL,

home\_city varchar(25)

NOT NULL,

UNIQUE (aut\_id));

**Example of MySQL UNIQUE CONSTRAINT check unique value**

The MySQL statement stated below will create a table 'newauthor' with a column 'aut\_id' which is meant to store unique values only. Notice that UNIQUE is used within the column definition

CREATE TABLE IF NOT EXISTS

newauthor(aut\_id varchar(8) NOT NULL UNIQUE ,

aut\_name varchar(50) NOT NULL,

country varchar(25)

NOT NULL,

home\_city varchar(25) NOT NULL);

**MySQL CREATE TABLE with DEFAULT CONSTRAINT**

While creating a table, MySQL allows you assign DEFAULT CONSTRAINTS to columns. DEFAULT is used to set a default value for a column and is applied using DEFAULT default\_value; where default\_value is the default value set to the column.

**Example**

The MySQL statement stated below will create a table 'newpublisher' with a PRIMARY KEY on 'pub\_id' column, a CHECK constraint with logical operators for country and pub-city columns and a default value for pub\_id, pub\_name, pub\_city and country columns.

The MySQL statement also sets the default value white space for pub\_id, pub\_name, pub\_city columns and 'India' as a default value for a country column.

Here is the statement below.

CREATE TABLE IF NOT EXISTS newpublisher

(pub\_id varchar(8) NOT NULL UNIQUE DEFAULT '' ,

pub\_name varchar(50) NOT NULL DEFAULT '' ,

pub\_city varchar(25) NOT NULL DEFAULT '' ,

country varchar(25) NOT NULL DEFAULT 'India',

country\_office varchar(25) ,

no\_of\_branch int(3),

estd date

CHECK ((country='India' AND pub\_city='Mumbai')

OR (country='India' AND pub\_city='New Delhi')) ,

PRIMARY KEY (pub\_id));

**MySQL CREATE TABLE with AUTO INCREMENT**

MySQL allows you to set AUTO\_INCREMENT to a column. Doing so will increase the value of that column by 1 automatically, each time a new record is added.

**Example**

The MySQL statement stated below will create a table 'newauthor' with a PRIMARY KEY on 'id' column and the 'id' column is an auto incremented field.

CREATE TABLE IF NOT EXISTS newauthor

(id int NOT NULL AUTO\_INCREMENT,

aut\_id varchar(8),

aut\_name varchar(50),

country varchar(25),

home\_city varchar(25) NOT NULL,

PRIMARY KEY (id));

**MySQL PRIMARY KEY CONSTRAINT**

Usually, a table has a column or combination of columns that contain values used to uniquely identify each row in the table.This column or combination of columns is called PRIMARY KEY and can be created by defining a PRIMARY KEY CONSTRAINT while creating a table. A table can have only one PRIMARY KEY. A PRIMARY KEY column cannot contain NULL values.

**Example**

The MySQL statement stated below will create a table 'newauthor' in which PRIMARY KEY set to the column aut\_id.

CREATE TABLE IF NOT EXISTS

newauthor(aut\_id varchar(8) NOT NULL ,

aut\_name varchar(50) NOT NULL,

country varchar(25) NOT NULL,

home\_city varchar(25) NOT NULL,

PRIMARY KEY (aut\_id));

**MySQL CREATE TABLE PRIMARY KEY CONSTRAINT on single column**

In this topic, we have discussed how to set a PRIMARY KEY CONSTRAINT on a column of a table.

**Example**

The MySQL statement stated below will create a table 'newauthor' in which PRIMARY KEY set to the aut\_id column. Notice that here PRIMARY KEY keyword is used within the column definition.

CREATE TABLE IF NOT EXISTS

newauthor(aut\_id varchar(8) NOT NULL PRIMARY KEY,

aut\_name varchar(50) NOT NULL,

country varchar(25)

NOT NULL,

home\_city varchar(25) NOT NULL);

**MySQL CREATE TABLE PRIMARY KEY UNIQUE CONSTRAINT**

In this topic, we have discussed how to set PRIMARY KEY as well as UNIQUE constraints on columns of a table while creating a table with CREATE TABLE command.

**Example**

The MySQL statement stated below will create a table 'newauthor' in which PRIMARY KEY is set to the aut\_id column and UNIQUE is set to the home\_city column.

CREATE TABLE IF NOT EXISTS

newauthor(aut\_id varchar(8) NOT NULL PRIMARY KEY,

aut\_name varchar(50) NOT NULL,

country varchar(25) NOT NULL,

home\_city varchar(25) NOT NULL UNIQUE);

**MySQL CREATE TABLE PRIMARY KEY on multiple columns**

MySQL allows you to set PRIMARY KEY on multiple columns of a table. Doing this allows you to work on multiple columns as a single entity set as PRIMARY KEY for a table.

**Example**

The MySQL statement stated below will create a table 'newauthor' in which PRIMARY KEY is set with the combination of aut\_id and home\_city columns.

CREATE TABLE IF NOT EXISTS

newauthor(aut\_id varchar(8) NOT NULL ,

aut\_name varchar(50) NOT NULL,

country varchar(25) NOT NULL,

home\_city varchar(25) NOT NULL,

PRIMARY KEY (aut\_id, home\_city));

**MySQL creating table with FOREIGN KEY CONSTRAINT**

While creating (or modifying) a MySQL table, you can set a FOREIGN KEY CONSTRAINT to a column of the table. A foreign key is a column or combination of columns which can be used to set a link between the data in two tables. PRIMARY KEY of a table is linked to the FOREIGN KEY of another table to enhance data integrity.

**Syntax :**

FOREIGN KEY [column list] REFERENCES [primary key table] ([column list]);

**Arguments**

|  |  |
| --- | --- |
| **Name** | **Description** |
| column list | A list of the columns on which FOREIGN KEY is to be set. |
| REFERENCES | Keyword. |
| primary key table | Table name which contains the PRIMARY KEY. |
| column list | A list of the columns on which PRIMARY KEY is set in the primary key table. |

**Example**

**If you want to do the following tasks:**

A new table 'newbook\_mast' will be created.

The PRIMARY KEY for that table 'newbook\_mast' is 'book\_id'.

The FOREIGN KEY for the table 'newbook\_mast' is 'aut\_id'.

The 'aut\_id' is the PRIMARY KEY for the table 'newauthor'.

The FOREIGN KEY 'aut\_id' for the table 'newbook\_mast' points to the PRIMARY KEY 'aut\_id' of the table 'newauthor'.

That means the 'aut\_id's which are present in the 'newauthor' table, only those authors will come to the 'newbook\_mast' table.

Here is the MySQL statement below for the above tasks.

CREATE TABLE IF NOT EXISTS newbook\_mast

(book\_id varchar(15) NOT NULL PRIMARY KEY,

book\_name varchar(50) ,

isbn\_no varchar(15) NOT NULL ,

cate\_id varchar(8) ,

aut\_id varchar(8) ,

pub\_id varchar(8) ,

dt\_of\_pub date ,

pub\_lang varchar(15) ,

no\_page decimal(5,0) ,

book\_price decimal(8,2) ,

FOREIGN KEY (aut\_id) REFERENCES newauthor(aut\_id));

**MySQL CREATE TABLE with FOREIGN KEY CONSTRAINT on multiple columns**

MySQL allows assigning FOREIGN KEY CONSTRAINTS on multiple columns of a table. Doing this, more than one columns of a table is set with a FOREIGN KEY CONSTRAINT referenced PRIMARY KEYs belonging to different tables.

**Example**

**If you want to do the following tasks:**

A new table 'newpurchase' will be created.   
The PRIMARY KEY for that table 'newpurchase' is 'invoice\_no'.  
The one FOREIGN KEY for the table 'newpurchase' is a combination of 'ord\_no' and 'book\_id'.   
The another FOREIGN KEY for the table 'newpurchase' is 'cate\_id'.  
The 'ord\_no' and 'book\_id' combination is the PRIMARY KEY for the table 'neworder'.  
The 'cate\_id' is the PRIMARY KEY for the table 'category'.  
The FOREIGN KEY 'ord\_no' and 'book\_id' combination for the table 'newpurchase', which points to the PRIMARY KEY 'ord\_no' and 'book\_id' combination of the table 'neworder'.  
That means the distinct ('ord\_no' and 'book\_id') combination which are present in the in the 'neworder' table only those unique 'order number' and 'book id' combination will come in the 'newpurchase' table.

The another FOREIGN KEY 'cate\_id' for the table 'newpurchase', which points to the PRIMARY KEY 'cate\_id' of the table 'category'. That means the 'cate\_id' which are present in the 'category' table only those 'category' will come in the 'newpurchase' table.

Here is the MySQL statement below for the above tasks.

CREATE TABLE IF NOT EXISTS newpurchase

(invoice\_no varchar(12) NOT NULL UNIQUE PRIMARY KEY,

invoice\_dt date ,

ord\_no varchar(25) ,

ord\_date date ,

receive\_dt date ,

book\_id varchar(8) ,

book\_name varchar(50) ,

pub\_lang varchar(8) ,

cate\_id varchar(8) ,

receive\_qty int(5) ,

purch\_price decimal(12,2) ,

total\_cost decimal(12,2) ,

INDEX (ord\_no,book\_id),

FOREIGN KEY(ord\_no,book\_id) REFERENCES neworder(ord\_no,book\_id),

INDEX (cate\_id),

FOREIGN KEY(cate\_id) REFERENCES category(cate\_id));

**MySQL CREATE TABLE with FOREIGN KEY CONSTRAINT on multiple tables**

MySQL allows assigning FOREIGN KEY CONSTRAINTS on multiple tables. Doing this, more than one columns of a table is set with a FOREIGN KEY CONSTRAINT referenced to PRIMARY KEYs belonging to different tables.

**Example**

**If you want to do the following tasks :**

Aa new table 'newbook\_mast' will be created.   
The PRIMARY KEY for that table 'newbook\_mast' is 'book\_id'.  
The one FOREIGN KEY for the table 'newbook\_mast' is 'aut\_id'.  
The another FOREIGN KEY for the table 'newbook\_mast' is 'pub\_id'.   
The 'aut\_id' is the PRIMARY KEY for the table 'newauthor'.  
The 'pub\_id' is the PRIMARY KEY for the table 'newpublisher'.   
The FOREIGN KEY 'aut\_id' for the table 'newbook\_mast', which points to the PRIMARY KEY 'aut\_id' of the table 'newauthor'.  
That means the 'aut\_id' which are present in the in the 'nuwauthor' table only those authors will come to the 'newbook\_mast' table.  
The another FOREIGN KEY 'pub\_id' for the table 'newbook\_mast' , which points to the PRIMARY KEY 'pub\_id' of the table 'newpublisher'.   
That means the 'pub\_id' which are present in the in the 'newpublisher' table only those publishers will come to the 'newbook\_mast' table.

Here is the MySQL statement below for the above tasks.

CREATE TABLE IF NOT EXISTS

newbook\_mast (book\_id varchar(15) NOT NULL PRIMARY KEY,

book\_name varchar(50) ,

isbn\_no varchar(15) NOT NULL ,

cate\_id varchar(8),

aut\_id varchar(8) ,

pub\_id varchar(8) ,

dt\_of\_pub date ,

pub\_lang varchar(15) ,

no\_page decimal(5,0) ,

book\_price decimal(8,2) ,

INDEX (aut\_id),

FOREIGN KEY(aut\_id) REFERENCES newauthor(aut\_id),

INDEX(pub\_id),

FOREIGN KEY(pub\_id) REFERENCES newpublisher(pub\_id) );

**MySQL CREATE TABLE with CASCADE and RESTRICT**

MySQL allows creating a table with CASCADE and RESTRICT options.

CASCADE option deletes or updates the row from the parent table (containing PRIMARY KEYs), and automatically delete or update the matching rows in the child table (containing FOREIGN KEYs).

RESTRICT option bars the removal (i.e. using delete) or modification (i..e using an update) of rows from the parent table.

**Example**

**If you want to do the following tasks:**

A new table 'newpurchase' will be created.   
The PRIMARY KEY for that table 'newpurchase' is 'invoice\_no'.  
The one FOREIGN KEY for the table 'newpurchase' is a combination of 'ord\_no' and 'book\_id'.  
The another FOREIGN KEY for the table 'newpurchase' is 'cate\_id'.  
The 'ord\_no' and 'book\_id' combination is the PRIMARY KEY for the table 'neworder'.  
The 'cate\_id' is the PRIMARY KEY for the table 'category'.  
The FOREIGN KEY 'ord\_no' and 'book\_id' combination for the table 'newpurchase', which points to the PRIMARY KEY 'ord\_no' and 'book\_id' combination of the table 'neworder'. That means the distinct ('ord\_no' and 'book\_id') combination which are present in the 'neworder' table only those unique 'order number' and 'book id' combination will come in the 'newpurchase' table.   
The another FOREIGN KEY 'cate\_id' for the table 'newpurchase' , which points to the PRIMARY KEY 'cate\_id' of the table 'category'. That means the 'cate\_id' which are present in the 'category' table only those 'category' will come in the 'newpurchase' table.   
The ON UPDATE CASCADE ensures that the records inside the child table 'newpurchase' always points to the PRIMARY KEY inside the parent table 'neworder'.   
If any record gets deleted/updated from the 'neworder' table MySQL handles the deletion/updating of the records from 'newpurchase' table.  
ON DELETE RESTRICT prevents a record in a parent table 'neworder' being deleted or altered when it is still referenced from a child table 'newpurchase'.

Here is the MySQL statement below for the above tasks.

CREATE TABLE IF NOT EXISTS newpurchase

(invoice\_no varchar(12) NOT NULL UNIQUE PRIMARY KEY,

invoice\_dt date ,

ord\_no varchar(25) ,

ord\_date date ,

receive\_dt date ,

book\_id varchar(8) ,

book\_name varchar(50) ,

pub\_lang varchar(8) ,

cate\_id varchar(8) ,

receive\_qty int(5) ,

purch\_price decimal(12,2) ,

total\_cost decimal(12,2) ,

INDEX (ord\_no,book\_id),

FOREIGN KEY(ord\_no,book\_id) REFERENCES

neworder(ord\_no,book\_id)

ON UPDATE CASCADE ON DELETE RESTRICT,

INDEX (cate\_id),

FOREIGN KEY(cate\_id) REFERENCES category(cate\_id))