**TASK-10**

**Sequences,create user accounts,Index**

**SEQUENCES**

Sequence is a set of integers 1, 2, 3, … that are generated and supported by some database systems to produce unique values on demand.

* A sequence is a user defined schema bound object that generates a sequence of numeric values.
* Sequences are frequently used in many databases because many applications require each row in a table to contain a unique value and sequences provides an easy way to generate them.
* The sequence of numeric values is generated in an a**scending or descending order** at defined intervals and can be configured to restart when exceeds max\_value.

**Syntax:**

CREATE SEQUENCE sequence\_name

START WITH initial\_value

INCREMENT BY increment\_value

MINVALUE minimum value

MAXVALUE maximum value

CYCLE|NOCYCLE ;

**sequence\_name:** Name of the sequence.

**initial\_value:** starting value from where the sequence starts.

Initial\_value should be greater than or equal

to minimum value and less than equal to maximum value.

**increment\_value:** Value by which sequence will increment itself.

Increment\_value can be positive or negative.

**minimum\_value:** Minimum value of the sequence.

**maximum\_value:** Maximum value of the sequence.

**cycle:** When sequence reaches its set\_limit

it starts from beginning.

**nocycle:** An exception will be thrown

if sequence exceeds its max\_value.

**Example**

Following is the sequence query creating sequence in ascending order.

**Example 1:**

CREATE SEQUENCE sequence\_2

start with 1

increment by 1

minvalue 0

maxvalue 100

cycle;

Above query will create a sequence named *sequence\_1*.Sequence will start from 1 and will be incremented by 1 having maximum value 100. Sequence will repeat itself from start value after exceeding 100.

**Example 2:**  
Following is the sequence query creating sequence in descending order.

CREATE SEQUENCE sequence\_2

start with 100

increment by -1

minvalue 1

maxvalue 100

cycle;

Above query will create a sequence named *sequence\_2*.Sequence will start from 100 and should be less than or equal to maximum value and will be incremented by -1 having minimum value 1.

* **Example to use sequence :** create a table named students with columns as id and name.

CREATE TABLE students1

(

ID number(10),

NAME char(20)

);

Now insert values into table

INSERT into students1 VALUES(sequence\_2.nextval,'Ramesh');

INSERT into students VALUES(sequence\_1.nextval,'Suresh');

where *sequence\_1.nextval* will insert id’s in id column in a sequence as defined in sequence\_1.  
**Output:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| ID | NAME |

------------------------

| 1 | Ramesh |

| 2 | Suresh |

----------------------

PRACTICLE STEPS:

STEP-1

CREATE SEQUENCE sequence\_1

start with 1

increment by 1

minvalue 0

maxvalue 100

cycle;

STEP-2

CREATE TABLE students

(

ID number(10),

NAME char(20)

);

STEP-3

INSERT into students VALUES(sequence\_1.nextval,'Ramesh');

INSERT into students VALUES(sequence\_1.nextval,'Suresh');

STEP-4

select \*from students;

**Create User Accounts**

**To Create User Accounts Using MySQL CREATE USER Statement**

## **syntax**

The CREATE USER statement creates a new user in the database server.

Here is the basic syntax of the CREATE USER statement:

CREATE USER [IF NOT EXISTS] account\_name

IDENTIFIED BY 'password';

Code language: SQL (Structured Query Language) (sql)

In this syntax:

First, specify the account name after the CREATE USER keywords. The account name has two parts: username and hostname, separated by the @ sign:

username@hostname

Code language: SQL (Structured Query Language) (sql)

The username is the name of the user. And hostname is the name of the host from which the user connects to the MySQL Server.

The hostname part of the account name is optional. If you omit it, the user can connect from any host.

An account name without a hostname is equivalent to:

username@%

Code language: SQL (Structured Query Language) (sql)

If the username and hostname contains special characters such as space or -, you need to quote the username and hostname separately as follows:

'username'@'hostname'

Code language: SQL (Structured Query Language) (sql)

Besides the single quote ('), you can use backticks ( `) or double quotation mark ( ").

Second, specify the password for the user after the IDENTIFIED BY keywords.

The IF NOT EXISTS option conditionally create a new user only if it does not exist.

Note that the CREATE USER statement creates a new user without any privileges. To grant privileges to the user, you use the GRANT statement.

## **MySQL CREATE USER example**

First, connect to the MySQL Server using the mysql client tool:

mysql -u root -p

Code language: SQL (Structured Query Language) (sql)

Enter the password for the root account and press Enter:

Enter password: \*\*\*\*\*\*\*\*

Code language: SQL (Structured Query Language) (sql)

Second, show users from the current MySQL Server:

mysql> select user from mysql.user;

Code language: SQL (Structured Query Language) (sql)

Here is the output:

+------------------+

| user |

+------------------+

| mysql.infoschema |

| mysql.saession |

| mysql.sys |

| root |

+------------------+

Third, create a new user called bob:

mysql> create user bob@localhost identified by 'Secure1pass!';

Code language: SQL (Structured Query Language) (sql)

Fourth, show all users again:

mysql> select user from mysql.user;

Code language: SQL (Structured Query Language) (sql)

The output will be:

+------------------+

| user |

+------------------+

| bob |

| mysql.infoschema |

| mysql.session |

| mysql.sys |

| root |

+------------------+

5 rows in set (0.00 sec)

Code language: JavaScript (javascript)

The user bob has been created successfully.

Fifth, open a second session and log in to the MySQL as bob:

mysql -u bob -p

Code language: SQL (Structured Query Language) (sql)

Input the password for bob and press Enter:

Enter password: \*\*\*\*\*\*\*\*

Code language: SQL (Structured Query Language) (sql)

Sixth, [show the databases](https://www.mysqltutorial.org/mysql-show-databases/) that bob has access:

mysql> show databases;

Code language: SQL (Structured Query Language) (sql)

Here is the list of databases that bob can access:

+--------------------+

| Database |

+--------------------+

| information\_schema |

+--------------------+

1 row in set (0.01 sec)

Code language: JavaScript (javascript)

Seventh, go to the session of the user root and [create a new database](https://www.mysqltutorial.org/mysql-create-database/) called bobdb:

mysql> create database bobdb;

Code language: SQL (Structured Query Language) (sql)

Eight, [select the database](https://www.mysqltutorial.org/mysql-select-database/) bobdb:

mysql> use bobdb;

Code language: SQL (Structured Query Language) (sql)

Ninth, [create a new table](https://www.mysqltutorial.org/mysql-create-table/) called lists:

mysql> create table lists(

-> id int auto\_increment primary key,

-> todo varchar(100) not null,

-> completed bool default false);

Code language: SQL (Structured Query Language) (sql)

Notice that when you press Enter, instead of showing the mysql> command, the mysql tool shows the -> that accepts new clause of the statement.

Tenth, grant all privileges on the bobdb to bob:

mysql> grant all privileges on bobdb.\* to bob@localhost;

Code language: SQL (Structured Query Language) (sql)

Note that you will learn how to grant privileges to a user in the [GRANT](https://www.mysqltutorial.org/mysql-grant.aspx) tutorial.

Eleventh, go to the bob’s session and show databases:

mysql> show databases;

Code language: SQL (Structured Query Language) (sql)

Now, bob can see the bobdb:

+--------------------+

| Database |

+--------------------+

| bobdb |

| information\_schema |

+--------------------+

2 rows in set (0.00 sec)

Code language: JavaScript (javascript)

Twelfth, select the database bobdb:

mysql> use bobdb;

Code language: SQL (Structured Query Language) (sql)

Thirteenth, [show the tables](https://www.mysqltutorial.org/mysql-show-tables/) from the bobdb database:

mysql> show tables;

Code language: SQL (Structured Query Language) (sql)

The user bob can see the lists table:

+-----------------+

| Tables\_in\_bobdb |

+-----------------+

| lists |

+-----------------+

1 row in set (0.00 sec)

Code language: JavaScript (javascript)

Fourteenth, [insert a row](https://www.mysqltutorial.org/mysql-insert-statement.aspx) into the lists table:

mysql> insert into lists(todo) values('Learn MySQL');

Code language: SQL (Structured Query Language) (sql)

Fifteenth, [query data](https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) from the lists table:

mysql> select \* from lists;

Code language: SQL (Structured Query Language) (sql)

This is the output:

+----+-------------+-----------+

| id | todo | completed |

+----+-------------+-----------+

| 1 | Learn MySQL | 0 |

+----+-------------+-----------+

1 row in set (0.00 sec)

Code language: JavaScript (javascript)

So the user bob can do everything in the bobdb database.

Finally, disconnect from the MySQL Server from both sessions:

mysql> exit

Code language: SQL (Structured Query Language) (sql)

In this task, you have learned how to use the MySQL CREATE USER to create a new user in the database server.

INDEX

# MySQL CREATE INDEX

**Summary**: in this tutorial, you will learn about the index and how to use the MySQL CREATE INDEX statement to add an index to a table.

## **The phone book analogy**

Suppose you have a phone book that contains all the names and phone numbers of people in a city. Let’s say you want to find Bob Cat’s phone number. Knowing that the names are alphabetically ordered, you first look for the page where the last name is Cat, then you look for Bob and his phone number.

Now, if the names in the phone book were not sorted alphabetically, you would need to go through all pages, reading every name on it until you find Bob Cat. This is called sequential searching. You go over all the entries until you find the person with the phone number that you are looking for.

Relating the phone book to the database table, if you have the table phonebooks and you have to find the phone number of Bob Cat, you would perform the following query:

SELECT

phone\_number

FROM

phonebooks

WHERE

first\_name = 'Bob' AND

last\_name = 'Cat';

Code language: SQL (Structured Query Language) (sql)

It is pretty easy. Although the query is fast, the database has to scan all the rows of the table until it finds the row. If the table has millions of rows, without an index, the data retrieval would take a lot of time to return the result.

## **Introduction to index**

An index is a data structure such as B-Tree that improves the speed of data retrieval on a table at the cost of additional writes and storage to maintain it.

The query optimizer may use indexes to quickly locate data without having to scan every row in a table for a given query.

When you [create a table](https://www.mysqltutorial.org/mysql-create-table/) with a [primary key](https://www.mysqltutorial.org/mysql-primary-key/) or[unique key](https://www.mysqltutorial.org/mysql-unique-constraint/), MySQL automatically creates a special index named PRIMARY. This index is called the [clustered index](https://www.mysqltutorial.org/mysql-index/mysql-clustered-index/).

The PRIMARY index is special because the index itself is stored together with the data in the same table. The clustered index enforces the order of rows in the table.

Other indexes other than the PRIMARY index are called secondary indexes or non-clustered indexes.

## **MySQL CREATE INDEX statement**

Typically, you create indexes for a table at the time of creation. For example, the following statement [creates a new table](https://www.mysqltutorial.org/mysql-create-table/) with an index that consists of two columns c2 and c3.

CREATE TABLE t(

c1 INT PRIMARY KEY,

c2 INT NOT NULL,

c3 INT NOT NULL,

c4 VARCHAR(10),

INDEX (c2,c3)

);

Code language: SQL (Structured Query Language) (sql)

To add an index for a column or a set of columns, you use the CREATE INDEX statement as follows:

CREATE INDEX index\_name ON table\_name (column\_list)

Code language: SQL (Structured Query Language) (sql)

To create an index for a column or a list of columns, you specify the index name, the table to which the index belongs, and the column list.

For example, to add a new index for the column c4, you use the following statement:

CREATE INDEX idx\_c4 ON t(c4);

Code language: SQL (Structured Query Language) (sql)

By default, MySQL creates the B-Tree index if you don’t specify the index type. The following shows the permissible index type based on the storage engine of the table:

| **Storage Engine** | **Allowed Index Types** |
| --- | --- |
| InnoDB | BTREE |
| MyISAM | BTREE |
| MEMORY/HEAP | HASH, BTREE |

Notice that the CREATE INDEX statement above is a simplified version of the CREATE INDEX statement introduced by MySQL. We will cover more options in the subsequent tutorials.

## **MySQL CREATE INDEX example**

The following statement finds employees whose job title is Sales Rep:

SELECT

employeeNumber,

lastName,

firstName

FROM

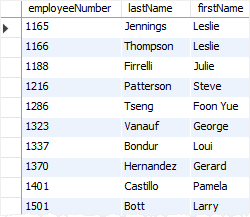
employees

WHERE

jobTitle = 'Sales Rep';

Code language: SQL (Structured Query Language) (sql)

Here is the output:



We have 17 rows indicating that 17 employees whose job title is the Sales Rep.

To see how MySQL internally performed this query, you add the EXPLAIN clause at the beginning of the SELECT statement as follows:

MySQL Index EXPLAIN statement

As you can see, MySQL had to scan the whole table which consists of 23 rows to find the employees with the Sales Rep job title.

Now, let’s create an index for the  jobTitle column by using the CREATE INDEX statement:

CREATE INDEX jobTitle ON employees(jobTitle);

Code language: SQL (Structured Query Language) (sql)

And execute the above statement again:

EXPLAIN SELECT

employeeNumber,

lastName,

firstName

FROM

employees

WHERE

jobTitle = 'Sales Rep';

Code language: SQL (Structured Query Language) (sql)

The output is:



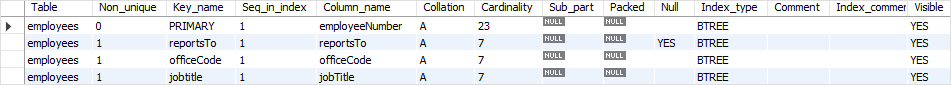
As you can see, MySQL just had to locate 17 rows from the  jobTitle index as indicated in the key column without scanning the whole table.

To show the indexes of a table, you use the [SHOW INDEXES](https://www.mysqltutorial.org/mysql-index/mysql-show-indexes/) statement, for example:

SHOW INDEXES FROM employees;

Code language: SQL (Structured Query Language) (sql)

Here is the output:



Thus, you have learned about MySQL index and how to add an index for a column in a table.