






Introducción a SQL y MariaDB

Esta guía presenta una introducción al gestor de bases de datos MaríaDB y el lenguaje de consultas SQL.

Descargue la última versión de este documento de:
<https://github.com/jdvelasq/fundamentos-de-analitica/blob/master/02-intro-sql.pdf>

JUAN DAVID VELÁSQUEZ HENAO, MSc, PhD
Profesor Titular
Departamento de Ciencias de la Computación y la Decisión
Facultad de Minas
Universidad Nacional de Colombia, Sede Medellín

 jdvelasq@unal.edu.co
 [@jdvelasquezh](https://twitter.com/jdvelasquezh)
 <https://github.com/jdvelasq>
 <https://goo.gl/prkJAq>
 <https://goo.gl/vXH8jy>

- Es un gestor de bases de datos relacionales distribuido bajo licencia GLP, cuya primera versión fue liberada en 2009.
- Es la siguiente generación de la base de datos MySQL.
- Es ampliamente utilizada por organizaciones.
- Existen implementaciones disponibles para Windows, MacOS X, Ubuntu, Mint, Red Hat
- Es altamente compatible con MySQL y comparte su misma estructura.
- Todos los ejemplos presentados son compatibles con MySQL.
- Existe software de terceras partes que puede ser usado para gestionar MariaDB como HeidiSQL, phpMyAdmin, DBEdit, Navicat, ...

SHOW DATABASES;

Lista las bases de datos en el servidor.

EJERCICIO.

¿Qué bases de datos hay instaladas por defecto en el servidor?

CREATE DATABASE *database_name*;

CREATE DATABASE IF NOT EXISTS *database_name*;

Crea la base de datos llamada *database_name*.

EJERCICIO.

Cree las bases de datos llamadas *db1* y *db2*.

USE *my_database*;

Se conecta a la base de datos llamada *my_database*.

EJERCICIO.

Active la base de datos llamada *db1*. Y luego la base llamada *db2*.

DROP DATABASE *database_name*;

DROP DATABASE IF EXISTS *database_name*;

Borra la base de datos *database_name*.

EJERCICIO.

Borre la base de datos llamada *db2*.

```
CREATE TABLE table_name (<column_definitions>);
```

```
    <column_definitions> ::
```

```
        <column_name> <data_type>
```

```
        [NOT NULL | NULL]
```

```
        [DEFAULT <default_value>] [AUTO_INCREMENT]
```

```
        [UNIQUE [KEY] | [PRIMARY] KEY] [COMMENT '<string>']
```

EJERCICIO.

Cree la tabla *employees*.

```
CREATE TABLE employees (  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    surname VARCHAR(100),  
    givenname VARCHAR(100),  
    pref_name VARCHAR(50),  
    birthday DATE  
);
```

EJERCICIO.

Qué devuelve el comando:

SHOW CREATE TABLE tb1;

EJERCICIO.

Qué devuelve el comando:

DESCRIBE tb1;

DROP TABLE *table_name*;

DROP TABLE IF EXISTS *table_name*;

Elimina la tabla llamada *table_name*.

```
ALTER TABLE table_name <alter_definition>[, alter_definition] ...;
```

```
<alter_definition> ::
```

```
    ADD <column_name> <column_definition> [FIRST | AFTER <column_name>]
```

```
    MODIFY <column_name> <column_definition>
```

```
    DROP <column_name>
```

EJEMPLO.

```
ALTER TABLE employees ADD username varchar(20) AFTER pref_name;
```

EJEMPLO.

```
ALTER TABLE employees MODIFY pref_name varchar(25);
```

EJEMPLO.

```
ALTER TABLE employees DROP username;
```

```
INSERT [INTO] <table_name> [( <column_name>[, column_name,...])]
{VALUES | VALUE}
({expression|DEFAULT},...)[, (...),...];
```

EJEMPLO.

```
INSERT INTO employees VALUES
    (NULL, "Perry", "Lowell Tom", "Tom", "1988-08-05");
```

EJEMPLO.

```
INSERT INTO employees VALUES
    (NULL, "Pratt", "Parley", NULL, NULL),
    (NULL, "Snow", "Eliza", NULL, NULL);
```

EJEMPLO.

```
INSERT INTO employees (surname,givenname) VALUES
    ("Taylor", "John"),
    ("Woodruff", "Wilford"),
    ("Snow", "Lorenzo");
```

EJEMPLO.

```
INSERT INTO employees (pref_name,givenname,surname,birthday)
    VALUES ("George", "George Albert", "Smith", "1970-04-04");
```

EJEMPLO.

```
INSERT employees (surname) VALUE ("McKay");
```

EJEMPLO.

```
INSERT INTO employees VALUES
    (NULL, "Kimball", "Spencer", NULL, NULL);
```

```
UPDATE <table_name>  
SET column_name1={expression|DEFAULT} [, column_name2={expression|  
DEFAULT}] ...  
[WHERE <where_conditions>];
```

EJEMPLO.

```
UPDATE employees SET  
    pref_name = "John", birthday = "1958-11-01"  
    WHERE surname = "Taylor" AND givenname = "John";
```

EJEMPLO.

```
UPDATE employees SET  
    pref_name = "Will", birthday = "1957-03-01"  
    WHERE surname="Woodruff";
```

EJEMPLO.

```
UPDATE employees SET birthday = "1964-04-03" WHERE surname = "Snow";
```

EJEMPLO.

```
UPDATE employees SET birthday = "1975-04-12" WHERE id = 2;
```

```
DELETE FROM <table_name> [WHERE <where_conditions>];
```

EJEMPLO.

```
DELETE FROM employees  
WHERE givenname="Spencer" AND surname="Kimball"
```



```
LOAD DATA [LOCAL] INFILE '<filename>'
  INTO TABLE <tablename>
  [(<column_name>[, <column_name>,...]];
```

EJEMPLO.

```
CREATE TABLE phone (
  id serial PRIMARY KEY,
  emp_id int,
  type char(3),
  cc int(4),
  number bigint,
  ext int);
```

```
LOAD DATA LOCAL INFILE
  '/Volumes/JetDrive/GitHub/data-science-docs/phone.txt'
  INTO TABLE phone
  FIELDS TERMINATED BY ',' IGNORE 1 LINES (emp_id,type,cc,number,ext);
```

```
emp_id,type,cc,number,ext
1,wrk,1,1235551212,23
1,hom,1,1235559876,NULL
1,mob,1,1235553434,NULL
2,wrk,1,1235551212,32
3,wrk,1,1235551212,11
4,mob,1,3215559821,NULL
4,hom,1,3215551234,NULL
```

```
SELECT <what> FROM <table_name>
    [WHERE <where-conditions>]
    [ORDER BY <column_name>];
```

EJEMPLO.

```
SELECT * FROM employees;
```

EJEMPLO.

```
SELECT * FROM employees LIMIT 3;
```

EJEMPLO.

```
SELECT givenname,surname FROM employees;
```

EJEMPLO.

```
SELECT * FROM employees WHERE birthday >= '1970-01-01';
```

EJEMPLO.

```
SELECT * FROM employees WHERE surname LIKE "McK%";
```

EJEMPLO.

```
SELECT * FROM employees WHERE givenname = 'Neil' OR givenname = 'John';
```

EJEMPLO.

```
SELECT * FROM employees WHERE surname = 'Snow' AND givenname LIKE 'Eli%';
```

EJEMPLO.

```
SELECT * FROM employees ORDER BY surname LIMIT 10;
```

EJEMPLO.

```
SELECT * FROM employees WHERE surname IN ('Snow', 'Smith', 'Pratt');
```

EJEMPLO.

```
SELECT * FROM employees WHERE surname NOT IN ('Snow', 'Smith', 'Pratt');
```

EJEMPLO.

```
SELECT * FROM employees WHERE birthday >= '1970-01-01' ORDER BY surname;
```

EJEMPLO.

```
SELECT surname,givenname,type,cc,number,ext  
FROM employees JOIN phone  
ON employees.id = phone.emp_id;
```

EJEMPLO.

```
SELECT surname,givenname,type,cc,number,ext  
FROM employees LEFT JOIN phone  
ON employees.id = phone.emp_id;
```

EJEMPLO.

```
SELECT AVG(TIMESTAMPDIFF(YEAR,birthday,CURDATE()))  
FROM employees;
```

EJEMPLO.

```
SELECT COUNT(*) FROM employees;
```

EJEMPLO.

```
SELECT COUNT(pref_name) FROM employees;
```

EJEMPLO.

```
SELECT * FROM employees  
WHERE birthday = (SELECT MIN(birthday) FROM employees);
```

EJEMPLO.

```
SELECT SUM(TIMESTAMPDIFF(YEAR,birthday,CURDATE())) FROM employees;
```

EJEMPLO.

```
SELECT surname, COUNT(*) FROM employees GROUP BY surname;
```

EJEMPLO.

```
SELECT surname, COUNT(*) FROM employees GROUP BY surname HAVING COUNT(*) > 1;
```

```
WITH emp_raleigh AS (  
    SELECT * FROM employees  
        WHERE office='Raleigh'  
)  
SELECT * FROM emp_raleigh  
    WHERE title != 'salesperson'  
    ORDER BY title;
```

```
SELECT * FROM (  
    SELECT * FROM employees  
        WHERE office='Raleigh'  
) AS emp_raleigh  
WHERE title != 'salesperson'  
    ORDER BY title;
```

```
WITH emp_raleigh AS (  
    SELECT * FROM employees  
        WHERE office='Raleigh'  
) ,  
emp_raleigh_dbas AS (  
    SELECT * from emp_raleigh  
        WHERE title='dba'  
)  
SELECT * FROM emp_raleigh_dbas;
```

```
CREATE TABLE commissions (  
    id serial primary key,  
    salesperson_id BIGINT(20) NOT NULL,  
    commission_id BIGINT(20) NOT NULL,  
    commission_amount DECIMAL(12,2) NOT NULL,  
    commission_date DATE NOT NULL  
);
```

```
SELECT  
    salesperson_id,  
    YEAR(commission_date) AS year,  
    SUM(commission_amount) AS total  
FROM  
    commissions  
GROUP BY  
    salesperson_id, year;
```

```
WITH commissions_year AS (  
    SELECT  
        salesperson_id,  
        YEAR(commission_date) AS year,  
        SUM(commission_amount) AS total  
    FROM  
        commissions  
    GROUP BY  
        salesperson_id, year  
)  
SELECT *  
FROM  
    commissions_year CUR,  
    commissions_year PREV  
WHERE  
    CUR.salesperson_id=PREV.salesperson_id AND  
    CUR.year=PREV.year + 1;
```

TALLER PRACTICO.






1. Usando el servicio web generatedata.com, genere una tabla con 50 registros; la tabla debe tener los campos descritos a continuación y debe guardarla con formato CSV:
 - Nombre del cliente.
 - Nombre de la compañía.
 - Número de la tarjeta de crédito separado por '-' en el formato de MasterCard.
 - Fecha de nacimiento del cliente.
 - Fecha de vencimiento de la tarjeta.
 - Clave de la tarjeta.
 - Dígitos de seguridad de la tarjeta.
2. Cree la tabla usando SQL.
3. Cargue los datos a la tabla usando el comando load.
4. Agregue un nuevo campo que sea el banco emisor de la tarjeta.
5. Actualice el nombre del banco emisor a CityBank para los primeros 10 registros y Bank of America para el resto.
6. Cuente la cantidad de tarjetas MasterCard.
7. Agregue 40 nuevos registros aleatorios con tarjetas de crédito VISA, 20 de ellas emitidas por CityBank y las restantes por Bank of America.
8. Cuál es el cliente más viejo que tiene tarjeta VISA?
9. Cuál es el cliente más joven que tiene tarjeta VISA?
10. Cuál cliente con tarjeta MasterCard tiene el mayor valor como clave de la tarjeta de crédito.

Introducción a SQL y MariaDB

Esta guía presenta una introducción al gestor de bases de datos MaríaDB y el lenguaje de consultas SQL.

Descargue la última versión de este documento de:
<https://github.com/jdvelasq/fundamentos-de-analitica/blob/master/02-intro-sql.pdf>

JUAN DAVID VELÁSQUEZ HENAO, MSc, PhD
Profesor Titular
Departamento de Ciencias de la Computación y la Decisión
Facultad de Minas
Universidad Nacional de Colombia, Sede Medellín

 jdvelasq@unal.edu.co
 [@jdvelasquezh](https://twitter.com/jdvelasquezh)
 <https://github.com/jdvelasq>
 <https://goo.gl/prkJAq>
 <https://goo.gl/vXH8jy>