






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.
- 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.

En la base de datos db1, cree la tabla tb1 que contiene:

- El SSN de la persona.
- El nombre de la franquicia de la tarjeta de crédito (CCNETWORK).
- El número de la tarjeta de crédito como cadena de texto (CCN).
- La clave de 4 dígitos (KEY).
- La fecha de vencimiento (VALIDTHRU).
- El cupo (MAXL).

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;
```

EJERCICIO.

Agregue la fecha de expedición de la tarjeta de crédito antes de la fecha de vencimiento.

EJERCICIO.

Agregue al final de la tabla el nombre del banco emisor.

```
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");
```

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

EJERCICIO.

Inserte un registro con la siguiente información a la tabla *tb1*.

- SSN: 216-51-1025
- CCNETWORK: Visa
- CCN: 3608-2181-5988-1718
- CLAVE: 1763

EJERCICIO.

Inserte las siguientes tarjetas de crédito en la tabla *tb1*.

- 3608-2067-5394-1306
- 3608-2588-4236-1129
- 3608-1682-5551-1055
- 3608-2181-5691-1038


```
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;
```

EJERCICIO.

Para todas las tarjetas de crédito en la tabla *tb1* fije como fecha de expiración sep-2019.

```
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>, ...]);
```

EJERCICIO.

Para la tabla *tb1* agregue el campo PIN que es un número de seguridad de tres dígitos.

EJERCICIO.

Cargue la información del archivo *ccdata.txt* a la tabla *tb1*.

```
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;
```

EJERCICIO.

Agregue franquicias (CCNETWORK) a los registros de la tabla *tb1*. Las franquicias validas son: AMEX, MASTERCARD, VISA.

EJERCICIO.

Seleccione los registros con franquicia VISA.

EJERCICIO.

Seleccione los registros que no sean franquicia VISA.

EJERCICIO.

Seleccione los registros PIN mayor de 350.

EJEMPLO.

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

```
INSERT INTO phone (emp_id,type,cc,number,ext) VALUES  
    (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 surname,givenname,type,cc,number,ext  
FROM employees JOIN phone  
ON employees.id = phone.emp_id;
```

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

EJERCICIO.

Genere una tabla llamada tb2, y cárguela con los siguientes datos:

key	F1	F2
1	11	12
2	21	22
4	41	42

Genere una tabla llamada tb3, y cárguela con los siguientes datos:

key	F3	F4
1	13	14
2	23	24
3	33	34

Realice un join por el campo key.

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 * 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;
```

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;
```

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>