Contenido

[A) INSERCCIÓN DE VALORES EN CAMPOS CON RESTRICCIONES 2](#_Toc65658063)

[B) CREACIÓN DE TABLA E INSERCCIÓN DE DATOS A PARTIR DE CONSULTA 3](#_Toc65658064)

[C) AÑADIR RESTRICCIONES A UNA TABLA CON DATOS 4](#_Toc65658065)

[*D)* CLÁUSULA *AUTO\_INCREMENT* 5](#_Toc65658066)

[E) OPERACIONES CON INTEGRIDAD REFERENCIAL 7](#_Toc65658067)

[F) GESTIÓN DE HOSPITALES 9](#_Toc65658068)

[a. EJERCICIO 1 9](#_Toc65658069)

[b. EJERCICIO 2 10](#_Toc65658070)

# INSERCCIÓN DE VALORES EN CAMPOS CON RESTRICCIONES

Para este caso se creará una tabla en la que se incluirán las siguientes restricciones: *UNIQUE, NOT NULL, DEFAULT* y *CHECK.*

Dicha tabla será la siguiente:

DROP TABLE IF EXISTS RESTRIC;

CREATE TABLE RESTRIC(

    COD INT UNIQUE,

    NOM VARCHAR(10) NOT NULL,

    LET CHAR(2) DEFAULT 'A',

    CANT INT,

    CONSTRAINT CK\_CANT CHECK(CANT > 0 AND CANT < 10)

);

Se probarán los siguientes casos:

**UNIQUE**

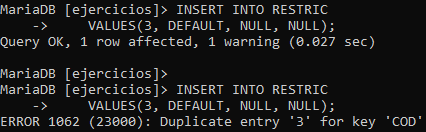
INSERT INTO RESTRIC

    VALUES(3, DEFAULT, NULL, NULL);

INSERT INTO RESTRIC

    VALUES(3, DEFAULT, NULL, NULL);

El primer campo hace referencia a la columna COD, por ende, teniendo la restricción de UNIQUE no puede repetirse.



**NOT NULL**

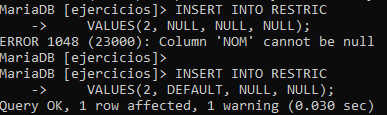
INSERT INTO RESTRIC

    VALUES(2, NULL, NULL, NULL);

INSERT INTO RESTRIC

    VALUES(2, DEFAULT, NULL, NULL);

La columna NOM no puede quedar en NULL como se observa en el primer caso.

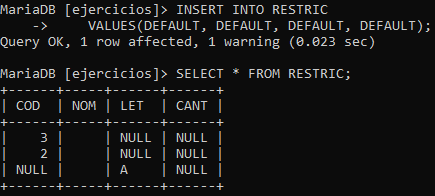


**DEFAULT**

INSERT INTO RESTRIC

    VALUES(DEFAULT, DEFAULT, DEFAULT, DEFAULT);

En todos los casos menos en el tercero (haciendo referencia a la columna LET) se pondrá NULL, ya que no tienen un valor por defecto puesto explícitamente. En el caso de que se use default en un campo que no tenga valor por defecto y no pueda ser NULL, se pondrá una **cadena vacía “”**.



**CHECK**

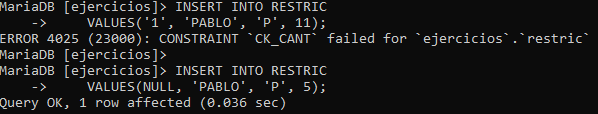
INSERT INTO RESTRIC

    VALUES('1', 'PABLO', 'P', 11);

INSERT INTO RESTRIC

    VALUES(NULL, 'PABLO', 'P', 5);

En estos casos, se confirma que la restricción CHECK funciona con el primer caso, ya que su valor no puede superar los límites.



# CREACIÓN DE TABLA E INSERCCIÓN DE DATOS A PARTIR DE CONSULTA

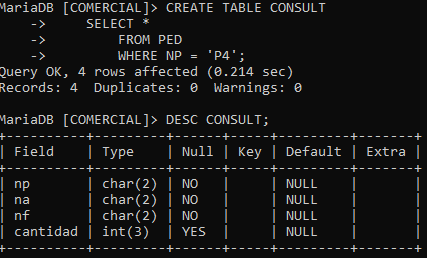
Para la creación de una tabla con consultas se hace de la siguiente manera:

CREATE TABLE CONSULT

    SELECT \*

        FROM PED

        WHERE NP = 'P4';



Para la inserción de valores mediante el resultado de una consulta se puede realizar del siguiente modo:

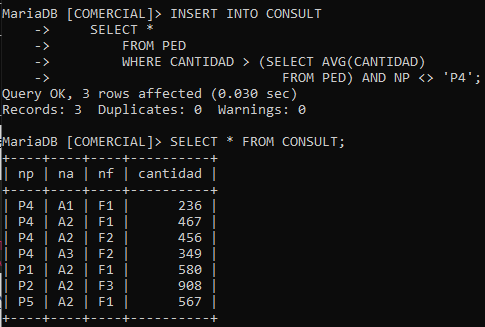
INSERT INTO CONSULT

    SELECT \*

        FROM PED

        WHERE CANTIDAD > (SELECT AVG(CANTIDAD)

                            FROM PED) AND NP <> 'P4';



# AÑADIR RESTRICCIONES A UNA TABLA CON DATOS

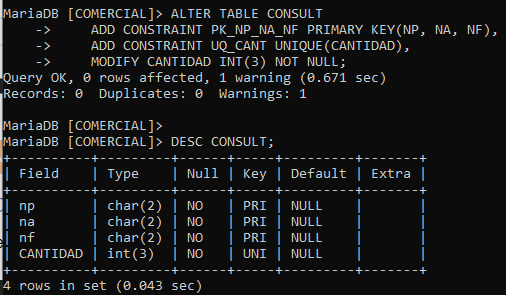
En la siguiente prueba, se ha usado la tabla creada en el apartado anterior, dándonos ningún problema a la hora de modificarla con las siguientes restricciones:

ALTER TABLE CONSULT

    ADD CONSTRAINT PK\_NP\_NA\_NF PRIMARY KEY(NP, NA, NF),

    ADD CONSTRAINT UQ\_CANT UNIQUE(CANTIDAD),

    MODIFY CANTIDAD INT(3) NOT NULL;



# CLÁUSULA *AUTO­\_INCREMENT*

Con esta cláusula se consigue crear un código automático e incremental de la siguiente manera:

DROP TABLE IF EXISTS AUTO\_INC;

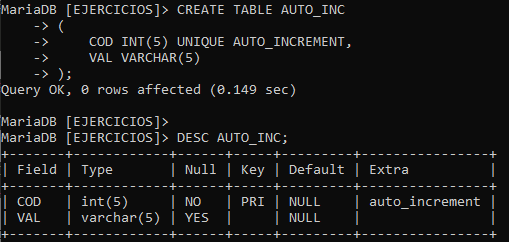
CREATE TABLE AUTO\_INC

(

    COD INT(5) UNIQUE AUTO\_INCREMENT,

    VAL VARCHAR(5)

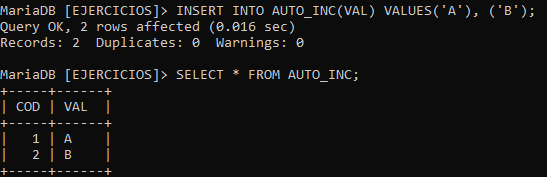
);



Para probar su funcionamiento se insertarán valores:

INSERT INTO AUTO\_INC(VAL) VALUES('A'), ('B');

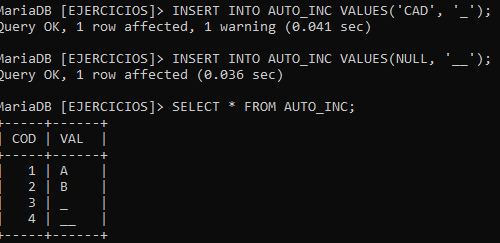
SELECT \* FROM AUTO\_INC;



En caso de que se introduzcan valores erróneos en la columna incremental, se ignorarán sin problemas:

INSERT INTO AUTO\_INC VALUES('CAD', '\_');

INSERT INTO AUTO\_INC VALUES(NULL, '\_\_');

****

**BORRAR VALORES**

Se puede borrar el valor de una columna y a la hora de insertar uno nuevo, se pondrá el siguiente al último valor agregado, esté o no presente en la tabla:

DELETE FROM AUTO\_INC

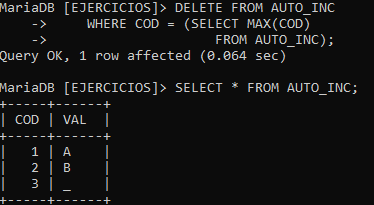
    WHERE COD = (SELECT MAX(COD)

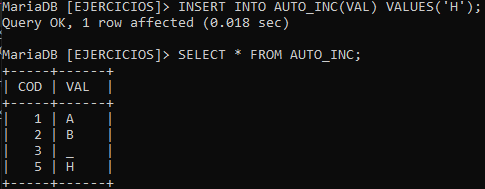
                    FROM AUTO\_INC);

SELECT \* FROM AUTO\_INC;

INSERT INTO AUTO\_INC(VAL) VALUES('H');

SELECT \* FROM AUTO\_INC;





# OPERACIONES CON INTEGRIDAD REFERENCIAL

Para esta ocasión se crearán dos tablas “*Alumno”* “*Profesor”* que estarán relacionadas de tal manera que un alumno pueda estar asociado a un profesor:

DROP TABLE IF EXISTS ALUMNO;

CREATE TABLE ALUMNO

(

    NUM\_EXP INT(10) PRIMARY KEY,

    NOMBRE VARCHAR(15) NOT NULL,

    DNI\_PROFESOR INT(10)

);

DROP TABLE IF EXISTS PROFESOR;

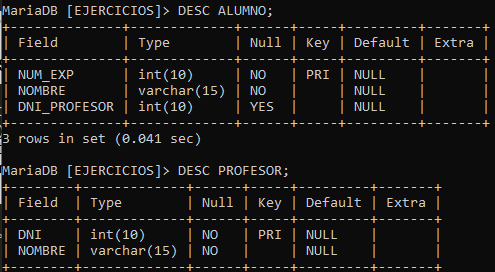
CREATE TABLE PROFESOR

(

    DNI INT(10) PRIMARY KEY,

    NOMBRE VARCHAR(15) NOT NULL

);



Lo siguiente es añadir la restricción de una clave foránea sin cascada primero y con ella después.

**ACTUALIZACIÓN Y BORRADO SIN CASCADA**

ALTER TABLE ALUMNO

    ADD CONSTRAINT FK\_ALUMNO\_PROFESOR FOREIGN KEY(DNI\_PROFESOR)

        REFERENCES PROFESOR(DNI);

Cuando insertemos valores en la tabla Profesor hay que tener en cuenta que después **NO** se podrán **borrar** o **modificar** aquellos que estén relacionados con un Alumno:

INSERT INTO PROFESOR

    VALUES(79979797, 'Alberto'),

    (797979, 'Juan');

INSERT INTO ALUMNO(NUM\_EXP, NOMBRE)

    VALUES(54278, 'Lola');

INSERT INTO ALUMNO

    VALUES(56565, 'Miguel', 797979);

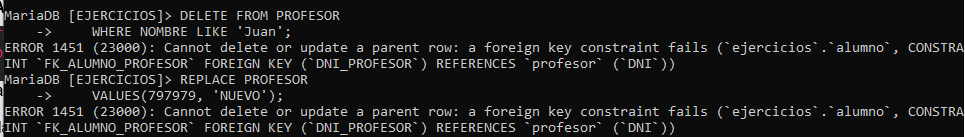
**Estos casos no pueden ser:**

DELETE FROM PROFESOR

    WHERE NOMBRE LIKE 'Juan';

REPLACE PROFESOR

    VALUES(797979, 'NUEVO');

****

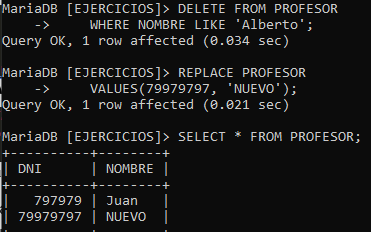
**Pero estos sí:**

DELETE FROM PROFESOR

    WHERE NOMBRE LIKE 'Alberto';

REPLACE PROFESOR

    VALUES(79979797, 'NUEVO');

****

**ACTUALIZACIÓN Y BORRADO CON CASCADA**

ALTER TABLE ALUMNO

    ADD CONSTRAINT FK\_ALUMNO\_PROFESOR FOREIGN KEY(DNI\_PROFESOR)

        REFERENCES PROFESOR(DNI) ON UPDATE CASCADE ON DELETE CASCADE;

En esta situación ya no habrá problemas a la hora de borrar o modificar los valores de los profesores que se encuentren dentro de una relación de integridad referencial:

**Todos los casos son posibles**

DELETE FROM PROFESOR

    WHERE NOMBRE LIKE 'Juan';

DELETE FROM PROFESOR

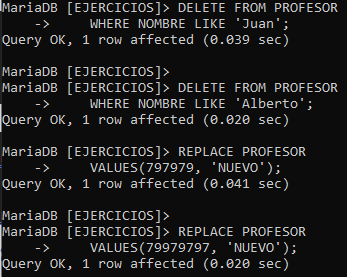
    WHERE NOMBRE LIKE 'Alberto';

REPLACE PROFESOR

    VALUES(797979, 'NUEVO');

REPLACE PROFESOR

    VALUES(79979797, 'NUEVO');



# GESTIÓN DE HOSPITALES

Se usará la BD de HOSPITALES para este ejercicio con el objetivo de hacer modificaciones a la tabla de EMP.

Es **importante recordar** que es recomendable hacer consultas antes de hacer alguna modificación o borrado a los valores de una tabla.

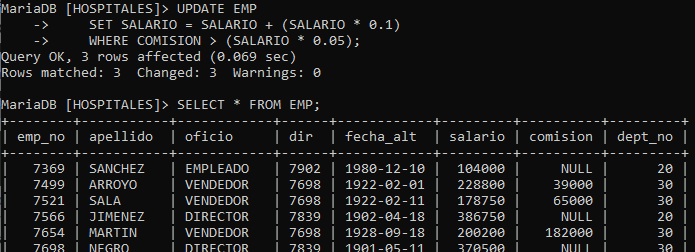
## EJERCICIO 1

**Incrementar el salario un 10% a los empleados con una comisión superior al 5% del salario.**

UPDATE EMP

    SET SALARIO = SALARIO + (SALARIO \* 0.1)

    WHERE COMISION > (SALARIO \* 0.05);



## EJERCICIO 2

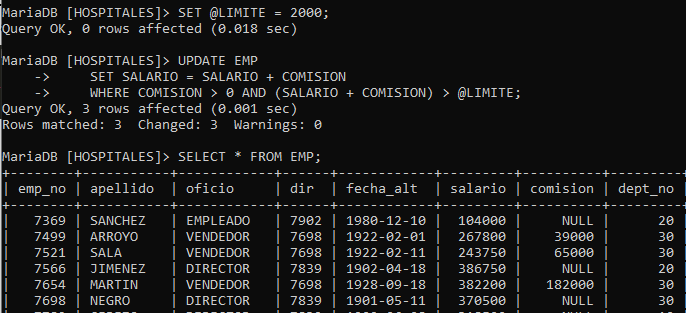
**Buscar todos los empleados que tengan un salario + comisión superior a 2000 y asignarles como nuevo salario esa suma.**

SET @LIMITE = 2000;

UPDATE EMP

    SET SALARIO = SALARIO + COMISION

    WHERE COMISION > 0 AND (SALARIO + COMISION) > @LIMITE;

****