VARIABLES DE USUARIO

En MySQL no hay que declararlas o indicar el tipo como en otros lenguajes, se usa directamente. Para indicar que es una variable, hay que poner delante un @. Ej.: @vUsu.

Las variables tienen **ámbito de sesión**.

CARGAR EL VALOR DE UNA VARIABLE

-- ASIGNAR VALOR

SET @vUsu = 10;

VISUALIZAR EL VALOR DE UNA VARIABLE

-- VISUALIZAR CONTENIDO

SELECT @vUsu;

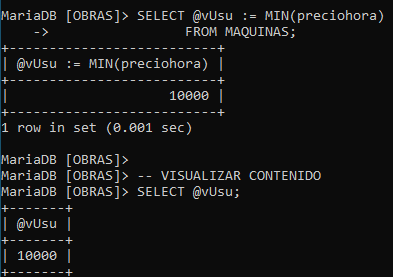
Si no se asigna ningún valor a una variable, se instancia con NULL.

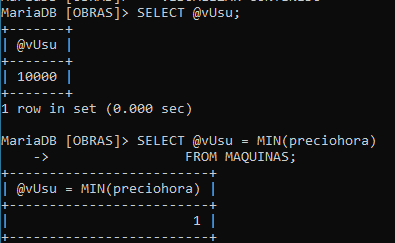
Se puede asignar como valor el resultado de una consulta:

SELECT @vUsu := MIN(preciohora)

                FROM MAQUINAS;

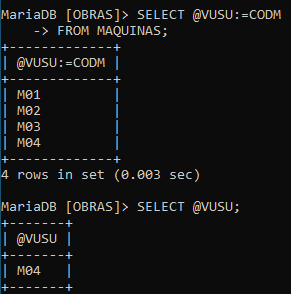
**IMPORTANTE PONER :=** Si solo se pone un =, se transforma en una comparación



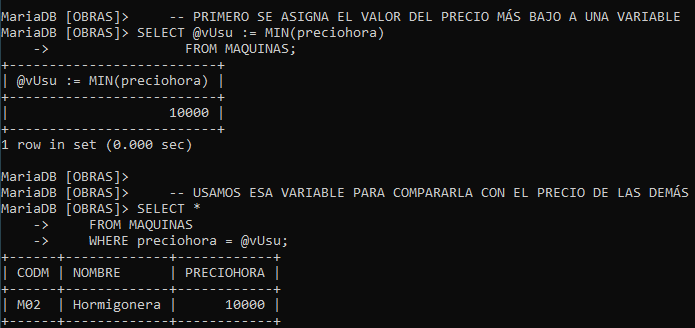


El resultado es 1 porque es cierto que @vUsu (que vale 10000) es igual que el resultado de la consulta.

**Las consultas deberían devolver un solo valor, sin embargo, si devuelven varios se muestran, pero solo se almacena el último.**

****

CASO REAL EJEMPLO



Sacar la información de todas las máquinas cuyo precio es el más bajo.

COMANDOS TRANSACCIONALES

Conjunto de operaciones sobre una BD con la característica de que se hacen todas o ninguna.

Dos formas:

* SET AUTOCOMMIT = 0: El sistema lo tiene a 1 por defecto, confirmando todo lo que se hace. A partir de esta instrucción, todo lo que se haga queda “en el aire” que se podrá confirmar o anular con COMMIT o ROLLBACK correspondientemente.
* START TRANSACTION: Como el anterior. Más dedicado a solo una serie pequeña de instrucciones.

-- INSERTAR UN PROVEEDOR Y LUEGO MODIFICAR SU CIUDAD

SET AUTOCOMMIT = 0;

INSERT INTO PRO

    VALUES('P14', 'Luis', 'Salamanca');

SELECT \* FROM PRO;

COMMIT;

SELECT \* FROM PRO;

UPDATE PRO

    SET CIUDADP = 'Zamora'

    WHERE NP = 'P14';

SELECT \* FROM PRO;

ROLLBACK;

SELECT \* FROM PRO;

Si en otra sesión se hace una consulta antes del COMMIT, observaremos que no hay cambios, pero a la hora de hacer el COMMIT, sí se podrán ver. Conclusión: Los cambios temporales tienen ámbito de sesión.

SAVEPOINT

**IMPORTANTE:** Usar SET AUTOCOMMIT = 0 antes de hacer un SAVEPOINT, si no, no dejará volver a dicho punto de guardado.

AUTOCOMMIT = 1



AUTOCOMMIT = 0



Crea un punto de transacción con un nombre identificativo. Se puede usar ROLLBACK TO SAVEPOINT para deshacer todos los cambios hasta el punto de guardado. Si se hace un COMMIT o un ROLLBACK, se borrarán estos puntos de control.

Ejemplo, qué sale en cada SELECT:

1. 1
2. Empty set
3. 2
4. 2
5. 2 y 3
6. 2, 3 y 4
7. 2 y 3
8. 2

BLOQUEO DE TABLAS (LOCK Y UNLOCK)

LOCK TABLES

TABLA\_NOMBRE <READ | WRITE>

Bloquea tablas de la sesión actual. Estos bloqueos impiden queo otros clientes escriban o lean en estas tablas hasta que finalice el bloqueo.

Esta acción realiza un COMMIT implícito de cualquier transacción activa antes de bloquear las tablas.

El bloqueo WRITE está por encima o es más potente que el de READ

**READ**

* La sesión que tiene dicho bloqueo puede leer, pero no escribir en la tabla.
* Puede haber varias sesiones que tengan este bloqueo.
* Otras sesiones pueden leer sin darle un READ explícitamente.

**WRITE**

* Puede leer y escribir.
* Solo la sesión que tiene el bloqueo puede acceder a la tabla.
* Las solicitudes de actividad de otras sesiones quedan bloqueadas mientras este bloqueo se mantenga.

El bloqueo se tiene que hacer en una sola operación.

Mientras tenga bloqueadas las tablas, solo se podrá trabajar con dichas tablas.

UNLOCK TABLES

Desbloquea todas las tablas.

DEMOSTRACIÓN READ

        -- ROOT --

-- 1) BLOQUEO DE LECTURA A ART

LOCK TABLES ART READ;

-- 4) ESCRIBIR ART

INSERT INTO ART (NA) VALUES ('A5');

-- 6) LEER PED

SELECT \* FROM PED;

        -- DAM1 --

-- 2) LEER ART

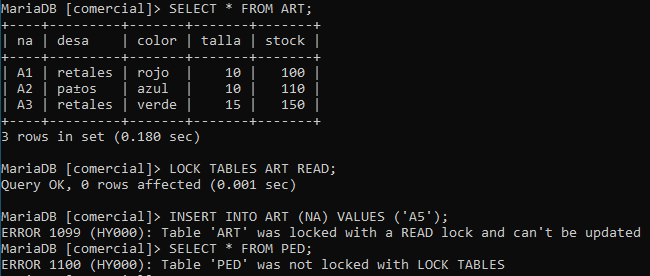
SELECT \* FROM ART;

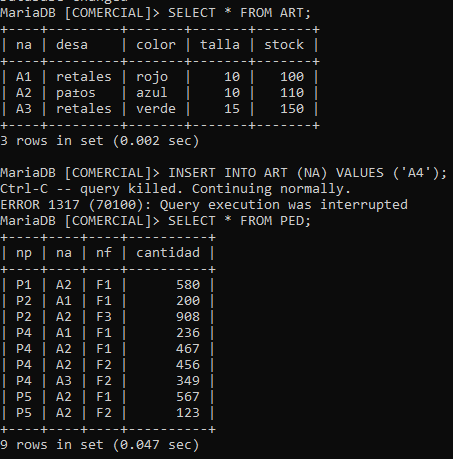
-- 3) ESCRIBIR ART

INSERT INTO ART (NA) VALUES ('A4');

-- 5) LEER PED

SELECT \* FROM PED;





DEMOSTRACIÓN WRITE

        -- ROOT --

-- 1) BLOQUEO DE LECTURA A ART

LOCK TABLES ART WRITE;

-- 3) LEER PED

SELECT \* FROM PED;

-- 5) LEER ART

SELECT \* FROM ART;

-- 7) ESCRIBIR ART

INSERT INTO ART (NA) VALUES ('A7');

        -- DAM1 --

-- 2) LEER ART

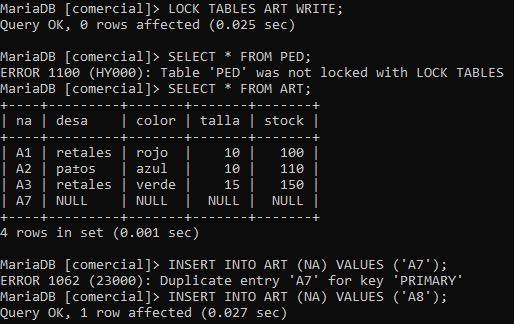
SELECT \* FROM PED;

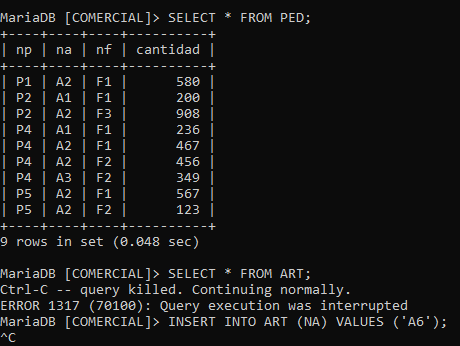
-- 4) LEER PED

SELECT \* FROM ART;

-- 6) ESCRIBIR ART

INSERT INTO ART (NA) VALUES ('A6');





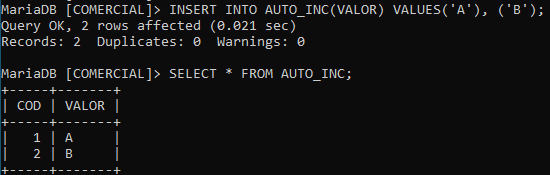
CONCLUSIÓN

Si una tabla está bloqueada en lectura, el usuario que está trabajando con ella solo puede ver esa tabla y solo esa, además no puede modificarla. Para otro usuario ajeno a esa sesión, solo podrá ver los datos y si quiere editarla, tendrá que esperar a que se desbloqué dicha tabla.

En caso de que bloqué con WRITE, el usuario que trabaja con ella puede ver y editar solo con ella. Sin embargo, para los otros usuarios no podrán ni ver ni editar los datos de esa.

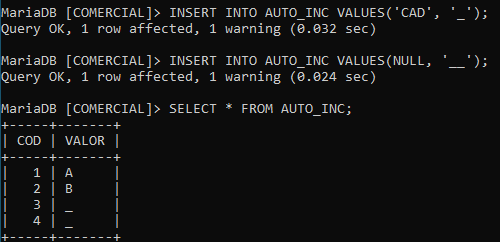
AUTOINCREMENT

Incrementa el valor de una columna de uno en uno en cada nueva fila.



A la hora de borrar el mayor valor incremental y volver a insertar un nuevo valor, se asignará a esta fila el valor incremental del último + 1, a pesar de que ya no exista dicho valor. Tiene un contador interno.

Si metemos un valor erróneo o NULL:



Es decir, se ignoran los valores de la columna incremental.