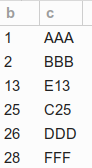
* Índice:

**No se encontraron elementos de tabla de contenido.**



# BDempleados - Consultas INSERT

## Teoría

DROP TABLE IF EXISTS A;

CREATE TABLE A(

b SERIAL,

c VARCHAR(3),

PRIMARY KEY (b));

INSERT into A values

(NULL, 'AAA');

INSERT into A(c) values

('BBB');

INSERT into A values

(25,'C25'),

(NULL,'DDD');

INSERT into A values

(13,'E13'),

(NULL,'FFF');

SELECT \* FROM A;

## Ejercicios

1. Insertar los siguientes departamentos mediante comandos INSERT individuales para cada fila:

**70 MANTENIMIENTO FERROL**

**80 REPARACIONES NARÓN**

**INSERT INTO departamentos**

**VALUES (70, 'MANTENIMIENTO', 'FERROL');**

**INSERT INTO departamentos**

**VALUES (80, 'REPARACIONES', 'NARÓN');**

1. Insertar los siguientes departamentos mediante un único comando INSERT

**90 RECURSOS HUMANOS FERROL**

**100 SANIDAD NARÓN**

**INSERT INTO departamentos**

**VALUES (90,'RECURSOS HUMANOS','FERROL'),**

**(100, 'SANIDAD', 'NARÓN');**

1. A efectos de rapidez, ¿es más rápido alguno de los métodos anteriores cuando se desea insertar muchas filas?

<https://dev.mysql.com/doc/refman/5.7/en/insert-optimization.html>

Insertar muchas filas mediante un único comando INSERT es mucho más rápido que generar un comando INSERT por cada fila.

El motivo es que se reduce la sobrecarga de comunicaciones entre cliente y servidor.

1. ¿Existe algún otro mecanismo para acelerar la inserción de filas en una tabla?

<https://dev.mysql.com/doc/refman/5.7/en/insert-optimization.html>

<https://dev.mysql.com/doc/refman/5.7/en/optimizing-innodb-bulk-data-loading.html>

1. Crear la tabla Mobiliario que tenga 2 atributos:

* **id\_mueble**: PRIMARY KEY *autonomúmerica*
* **descripción**:
* **departamento**: indica el nombre departamento al que pertenece, relacionado con la tabla departamentos.

Introduce filas en la tabla:

* Usando el comando INSERT proporcionando valores a todos los atributos
* Usando el comando INSERT especificando sólo las columnas a las que vamos a dar valor explícitamente.

**CREATE TABLE IF NOT EXISTS mobiliario (**

**id\_mueble BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT,**

**descripcion TEXT,**

**departamento SMALLINT UNSIGNED,**

**PRIMARY KEY (id\_mueble),**

**CONSTRAINT FK\_mobiliario\_departamento**

**FOREIGN KEY (departamento)**

**REFERENCES departamentos (DEP\_NO)**

**ON UPDATE CASCADE**

**);**

**INSERT INTO mobiliario VALUES**

**(NULL, 'mesa delineación',10),**

**(NULL, 'monitor 32 pulgadas',10);**

**INSERT INTO mobiliario (descripcion, departamento) VALUES**

**('mesa de montaje autómatas',10),**

**('proyector 1024x768',10);**

1. Insertar los siguientes empleados mediante comandos INSERT individuales para cada fila. Averigua por qué se produce un error y busca en los apuntes 2 posibles soluciones al problema.

**INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`)**

**VALUES ('9001', '14256821Y', 'Luís', 'Yañez', 'Rioboo', 'ADMINISTRATIVO', '9002', '2016-04-05', 150000, 0, '10', '652986532');**

**INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`)**

**VALUES ('9002', '45325698P', 'Ana', 'Rinlo', 'Rois', 'INGENIERA', '8904', '2016-04-05', 150000, 0, '10', '547852146');**

Solución:

El problema es que la primera fila a insertar incumple la FOREIGN KEY en el atributo JEFE: intentamos insertar un empleado (9001) cuyo jefe (9002) todavía no existe: no existe el empleado 9002 cuando intentamos insertar al empleado 9001

**#Manera de evitarlo 1:**

**#deshabilitar chequeo de las FOREIGN KEY**

**SET FOREIGN\_KEY\_CHECKS=0;**

**#insertardatos**

**INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`)**

**VALUES ('9001', '14256821Y', 'Luís', 'Yañez', 'Rioboo', 'ADMINISTRATIVO', '9002', '2016-04-05', '150000', '0', '10', '652986532');**

**INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`)**

**VALUES ('9002', '45325698P', 'Ana', 'Rinlo', 'Rois', 'INGENIERA', '8904', '2016-04-05', '150000', '0', '10', '547852146');**

**#habilitarchequeo de las FOREIGN KEY**

**SET FOREIGN\_KEY\_CHECKS=1;**

**#el problema es que al volver a habilitar el chequeo de las FOREIGN KEY no se comprueba si las filas introducidas previamente tienen algún error en sus FOREIGN KEY.**

**#Tenemos que comprobar si alguna clave ajena es errónea, para ello buscamos** esas posibles filas erróneas mediante esta consulta: buscar aquellos empleados cuyo jefe no existe.

SELECT \*

FROM empleados

WHERE jefe NOT IN (

SELECT emp\_no

FROM empleados

);

**#Manera de evitarlo 2:**

#ordenar los INSERT de modo que no se infrinjan las FOREIGN KEY (si fuesen muchas filas sería muy complicado ordenarlas todas, y en algunos casos sería imposible si hubiese referencias cruzadas)

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`) VALUES ('9002', '45325698P', 'Ana', 'Rinlo', 'Rois', 'INGENIERA', '8904', '2016-04-05', '150000', '0', '10', '547852146');

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`) VALUES ('9001', '14256821Y', 'Luís', 'Yañez', 'Rioboo', 'ADMINISTRATIVO', '9002', '2016-04-05', '150000', '0', '10', '652986532');

1. Insertar la siguiente fila, averiguar cuál es el problema y proponer una solución.

**INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`) VALUES ('9003', '85214638P', 'María Ana', 'Pinto', 'Prado', 'PERITA', '9999', '2016-04-05', '150000', '0', '10', '123489745');**

1. Crear una nueva tabla llamada *empleados\_antiguos* a partir de la tabla *empleados* (debe crearse utilizando el resultado de un SELECT), la tabla tendrá los mismos atributos (pero ninguna KEY), e insertar en ella al empleado 8904.

**CREATE TEMPORARY TABLE IF NOT EXISTS *empleados\_antiguos***

**(SELECT \***

**FROM empleados**

**WHERE emp\_no = 8904**

**);**

1. Usando el Workbench copiar al portapapeles la instrucción CREATE TABLE que ha creado la tabla *empleados\_antiguos*del ejercicio anterior y comprobar que diferencias tiene respecto al comando que crea la tabla *empleados*.

CREATE TABLE `empleados\_antiguos` (

`EMP\_NO` smallint(4) NOT NULL,

`DNI` varchar(9) NOT NULL,

`NOMBRE` varchar(65) NOT NULL,

`APELLIDO1` varchar(65) NOT NULL,

`APELLIDO2` varchar(65) DEFAULT NULL,

`OFICIO` varchar(65) DEFAULT NULL,

`JEFE` smallint(4) DEFAULT NULL,

`FECHA\_ALTA` date DEFAULT NULL,

`SALARIO` decimal(8,2) DEFAULT NULL,

`COMISION` decimal(8,2) DEFAULT NULL,

`DEP\_NO` smallint(4) DEFAULT NULL,

`TELEFONO` varchar(15) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `empleados` (

`EMP\_NO` smallint(4) NOT NULL,

`DNI` varchar(9) NOT NULL,

`NOMBRE` varchar(65) NOT NULL,

`APELLIDO1` varchar(65) NOT NULL,

`APELLIDO2` varchar(65) DEFAULT NULL,

`OFICIO` varchar(65) DEFAULT NULL,

`JEFE` smallint(4) DEFAULT NULL,

`FECHA\_ALTA` date DEFAULT NULL,

`SALARIO` decimal(8,2) DEFAULT NULL,

`COMISION` decimal(8,2) DEFAULT NULL,

`DEP\_NO` smallint(4) DEFAULT NULL,

`TELEFONO` varchar(15) DEFAULT NULL,

PRIMARY KEY (`EMP\_NO`),

UNIQUE KEY `UQ\_EMP\_DNI` (`DNI`),

KEY `FK\_EMP\_JEFE` (`JEFE`),

KEY `FK\_EMP\_DEP\_NO` (`DEP\_NO`),

CONSTRAINT `FK\_EMP\_DEP\_NO` FOREIGN KEY (`DEP\_NO`) REFERENCES `departamentos` (`DEP\_NO`) ON UPDATE CASCADE,

CONSTRAINT `FK\_EMP\_JEFE` FOREIGN KEY (`JEFE`) REFERENCES `empleados` (`EMP\_NO`) ON DELETE SET NULL ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#PUEDE COMPROBARSE QUE EN LA TABLA *empleados\_antiguos*faltan las KEY: PRIMARY KEY, UNIQUE KEY, FOREIGN KEY, KEY

1. Crear una nueva tabla llamada *empleados\_antiguos\_reducida* a partir de la tabla *empleados* (debe crearse utilizando el resultado de un SELECT) que incluya sólo estos atributos (emp\_no, apellido1, apellido2 y nombre) y sin KEYs, e insertar en ella al empleado 8904

CREATE TABLE IF NOT EXISTS empleados\_antiguos\_reducida

(SELECT emp\_no, apellido1, apellido2, nombre

FROM empleados

WHERE emp\_no = 8904

);

1. Usando el Workbench copiar al portapapeles la instrucción CREATE TABLE que crea la tabla *empleados\_antiguos\_reducida* del ejercicio anterior

CREATE TABLE `empleados\_antiguos\_reducida` (

`emp\_no` smallint(4) NOT NULL,

`apellido1` varchar(65) NOT NULL,

`apellido2` varchar(65) DEFAULT NULL,

`nombre` varchar(65) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

1. Crear una **tabla temporal** llamada *pedidos\_ano\_2000* que incluya los pedidos del año 2000, la tabla temporal tendrá los mismos atributos de la tabla original, pero sin KEYs.

**CREATE TEMPORARY TABLE IF NOT EXISTS pedidos\_ano\_2000**

**(SELECT \***

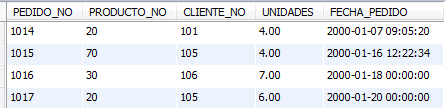
**FROM pedidos**

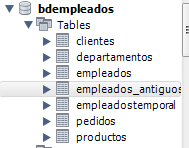
**WHERE YEAR(FECHA\_PEDIDO) = '2000'**

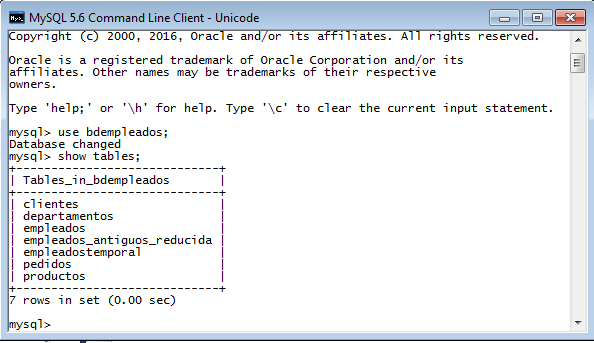
**);**

1. Mostar con un SELECT la tabla anterior *pedidos\_ano\_2000*.

* Comprobar que, aunque la tabla existe en memoria RAM, la tabla no aparece en el listado de tablas del Workbench,
* Comprobar que tampoco aparece si lo consultamos desde una consola con el comando *showtables*







1. Crear una segunda conexión al servidor usando Workbench y comprobar que:
2. La tabla temporal no se muestra en el listado que muestra Workbench.
3. Esta segunda conexión no puede utilizar la tabla temporal *pedidos\_ano\_2000,* (si intenta realizar una consulta sobre la tabla, da error, porque no existe para esa conexión)y sólo puede utilizarla la conexión que creo la tabla temporal, es decir, la tabla temporal sólo existe para la conexión que la crea, y desaparece cuando se cierra esa conexión.

#una consulta para comprobar que podemos acceder a los datos sería ésta:

SELECT \* FROM pedidos\_ano\_2000 ;

#si tenemos 2 conexiones activas podemos comprobar que sólo la conexión que ha creado la tabla temporal puede realizar la consulta.

1. **Exportar datos a un fichero en el servidor**: Conectarse al servidor remotamente (desde un equipo cliente distinto al servidor) y Exportar los pedidos del año 2000 a un fichero llamado *pedidos2000servidor.txt*situado en el servidor

**#volcamos el SELECT**

**SELECT \* INTO OUTFILE '*pedidos2000servidor.txt*'**

**FROM bdempleados.pedidos**

**WHERE YEAR(FECHA\_PEDIDO) = '2000'**

**;**

**#En Linux máquina virtual funciona**

**#Pero en Windows de clase produce este error:**

***Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement***

**en el documento** de apuntes se explica cómo solucionarlo, para lo que son necesarias credenciales de administrador, imposible en el ordenador real de clase.

1. **Exportar datos a un fichero en el cliente**: Conectarse al servidor remotamente (desde un equipo cliente distinto al servidor) y exportar los pedidos del año 2000 a un fichero llamado *pedidos2000cliente.txt* situado en el cliente

**#Desde la consola del Cliente Windows, contra un Servidor Linux (máquina virtual):**

mysql -u root -pAbcd1234. bdempleados-h 172.20.8.58 –e " SELECT \* FROM bdempleados.pedidos WHERE YEAR(FECHA\_PEDIDO) = '2000'" > pedidos2000cliente.txt

**#Desde la consola del Cliente Linux (máquina virtual) contra unServidor Windows:**

mysql -u root -pAbcd1234. bdempleados-h 172.20.8.52 -e "SELECT \* FROM bdempleados.pedidos WHERE YEAR(FECHA\_PEDIDO) = '2000'" > pedidos2000cliente.txt

1. Crear una tabla vacía llamada *pedidos2000*que tenga exactamente la misma estructura de la tabla *pedidos* (incluidas las KEYs) (pero sin datos).

* Desde la consola importar a la nueva tabla los datos del fichero creado anteriormente.

CREATE TABLE *pedidos2000*LIKE*pedidos*;

LOAD DATA INFILE 'pedidos2000cliente.txt' INTO TABLE bdempleados.*pedidos2000*;