



GOBIERNO  
DE ESPAÑA

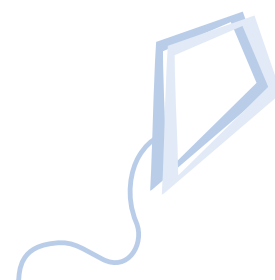
MINISTERIO  
DE EDUCACIÓN,  
POLÍTICA SOCIAL Y DEPORTE

mepsyd

# SQL con MySQL AVANZADO

Javier Robles y Alberto Carrera

PROGRAMACIÓN



Aula  
MENTOR



Nipo: 651-07-401-6

Coordinación del manual:  
Joaquín Gonzalez

# PARTE I. VISTAS

## Tema 1. Vistas

1.1 Antes de comenzar.....	2
1.2 Ventajas de su utilización.....	6
1.3 Creación. Diccionario de Datos.....	8
1.4 Modificación.....	14
1.5 Borrado .....	14
1.6 Operaciones sobre vistas. Restricciones.....	15
1.7 Ejercicios resueltos .....	19

# TEMA 1.

## Vistas

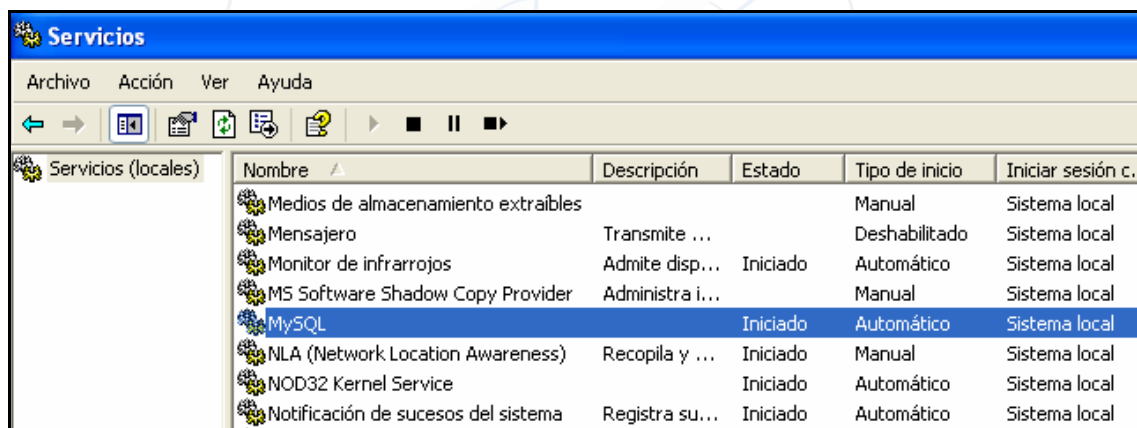
Alberto Carrera Martín

### 1.1 Antes de comenzar

#### a) COMPROBACIÓN DEL ESTADO DEL SERVIDOR

Antes de empezar a trabajar asegúrate que la base de datos está arrancada. Si utilizas Windows XP podrás comprobarlo mediante:

*Botón Inicio / Panel de Control / Rendimiento y Mantenimiento / Herramientas Administrativas / Servicios*




**Ilustración 1. Servicio de MySQL**

En este caso se encuentra iniciado, si no fuera así lo podrías arrancar haciendo clic con el botón derecho sobre el servicio y eligiendo la opción de *Iniciar*.

También lo puedes saber de otras formas y de manera gráfica ejecutando el monitor de sistema de MySQL que tras lanzarlo:

*Botón Inicio / Todos los programas / MySQL / MySQL System Tray Monitor*

aparecerá con el siguiente icono en la barra de estado . Al hacer un clic con el botón derecho sobre él se puede comprobar que la base de datos está “corriendo” o ejecutándose tal y como se aprecia en la primera línea de siguiente ilustración 2. El rectángulo verde del icono ya hace presagiar tal situación; si la figura que nos hubiéramos encontrado dentro del icono hubiera sido un cuadrado rojo, entonces la base de datos estaría parada y por tanto deberíamos arrancarla (opción *Start Instance* en ilustración 3 siguiente)

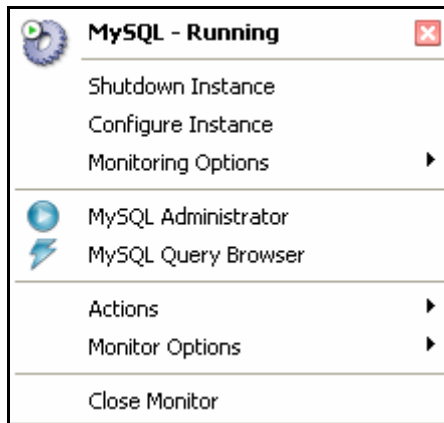


Ilustración 2. Monitor de MySQL. Servicio ejecutándose

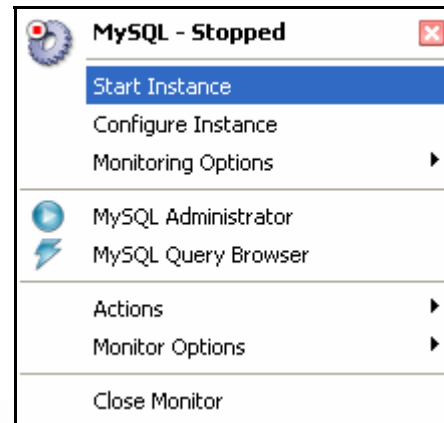


Ilustración 3. Monitor de MySQL. Servicio detenido

## b) CREACIÓN DE LA BASE DE DATOS DE EJEMPLO

La realizaremos de dos maneras de las varias posibles existentes. Lee las dos y después pon en práctica la que desees (o puedes experimentar con las dos)

### b1) Creación de la base de datos ejemplo utilizando una ventana de consola de windows

Los archivos binarios del servidor se encuentran instalados en la carpeta:

`C:\Archivos de programa\MySQL\MySQL Server 5.0\bin`

Nota: La unidad de disco anterior (C:) y la ruta de carpetas puede cambiar en función de donde se realizara la instalación.

Dentro de la carpeta anterior crea a su vez una carpeta denominada *bases* y copia allí el script *tablas1.sql* que encontrarás en la sección de materiales del curso. Una vez hecho, arranca una ventana de consola (*Botón Inicio / Todos los programas / Accesorios / Símbolo de sistema*), sitúate en la carpeta anterior (`C:\Archivos de programa\MySQL\MySQL Server 5.0\bin`) y ejecuta el comando siguiente que te permitirá conectarte a la base de datos como usuario root y al mismo tiempo ejecutar las instrucciones que contiene el script *tablas1.sql* copiado a la carpeta *bases* anterior:

```
mysql -u root -p < bases\tablas1.sql
```

La Instrucción anterior es equivalente a:

```
mysql -u root -p < "C:\Archivos de programa\MySQL\MySQL Server 5.0\bin\bases\tablas1.sql"
```

Tras el mensaje de introducción de la clave del usuario root

```
Enter password: *****
```

El script *tablas1.sql* crea, además de las tablas que puedes ver al final de este apartado, un *usuario1* de clave *usuario1* con todos los permisos de trabajo sobre la base de datos

*practical* que se acaba de crear. Se puede comprobar si en la ventana de consola abres una conexión con el *usuario1*:

```
D:\Archivos de programa\MySQL\MySQL Server 5.0\bin>mysql -u usuario1 -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 41 to server version: 5.0.20a-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| practical |
| test |
+-----+
4 rows in set (0.77 sec)

mysql> USE practical
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_practical |
+-----+
| centros |
| departamentos |
| empleados |
+-----+
3 rows in set (0.01 sec)
```

## b2) Creación de la base de datos ejemplo utilizando una herramienta cliente

Otra forma más atractiva de realizar la tarea anterior es utilizar la herramienta gráfica de *MySQL Query Browser* (para este curso se ha utilizado la versión 1.1.2.0 que se encuentra en el fichero *mysql-query-browser-1.1.20-win.msi*). Su instalación es muy sencilla. Tras lanzarla desde el monitor de MySQL visto anteriormente en el apartado 1.1.a (ilustraciones 2 y 3) o también mediante *Botón Inicio / Todos los programas / MySQL / MySQL Query Browser* aparecerá la pantalla inicial de la ilustración 4 de la derecha. Tras introducir el usuario/clave de root, pulsa *OK* para avanzar a la siguiente pantalla.

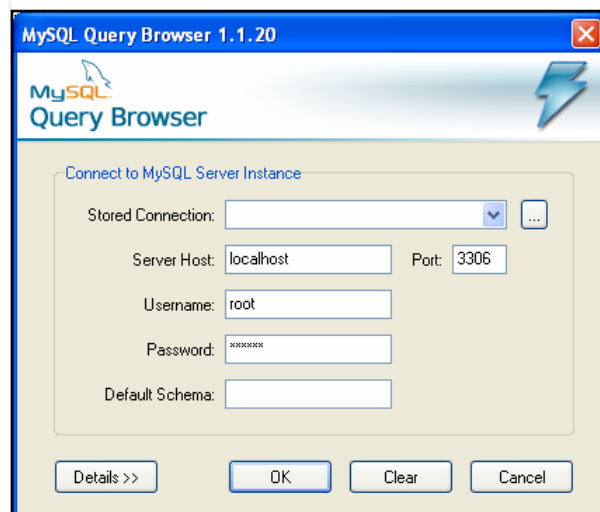
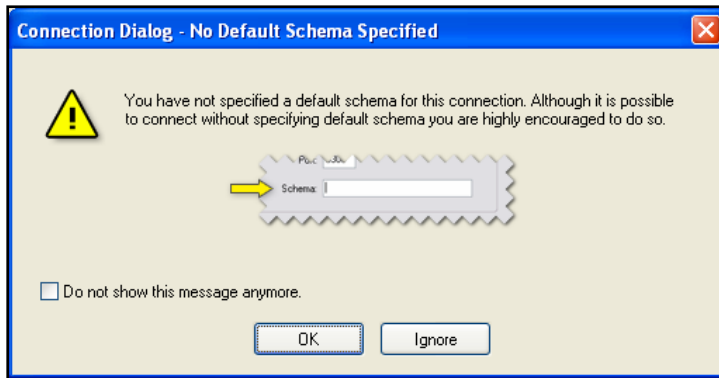


Ilustración 4. MySQL Query Browser



Observa que se ha introducido el usuario y la clave del Administrador *root* pero no la base de datos a la que se conecta (*Default Schema*) por lo que tras pulsar el botón *OK* puede aparecer la siguiente ilustración 5 de la izquierda pantalla para recordárnoslo:

### Ilustración 5. MySQL Query Browser

Pulsaremos el botón *Ignore* para continuar y entrar en la herramienta cliente. Una vez en ella abriremos (opción *File / Open Script...* del menú) el script *tablas1.sql* que podrás encontrar en la sección de materiales del curso (ver ilustración 6)

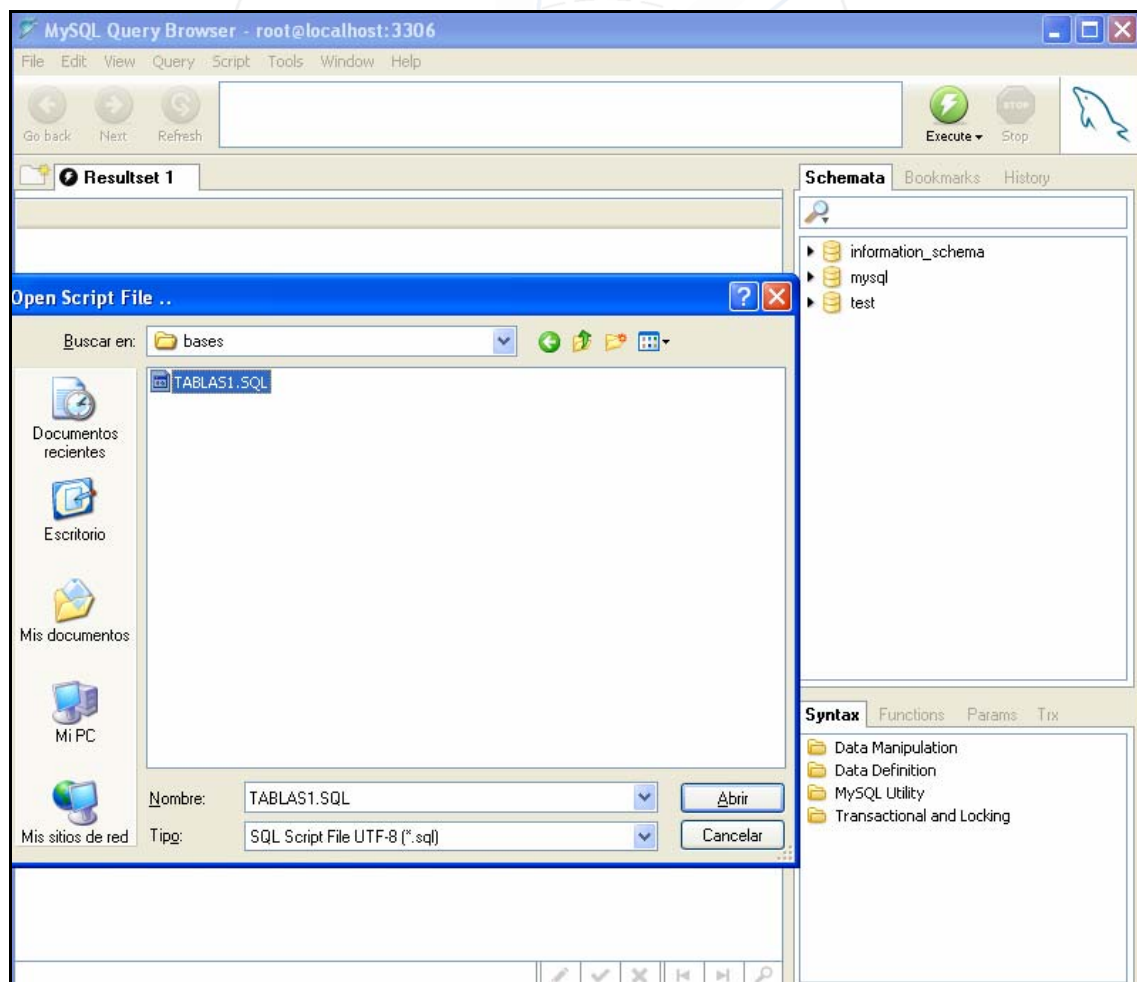


Ilustración 6. Cargando un script en MySQL Query Browser

A continuación puedes ejecutar el script mediante el botón *Execute* (parte superior derecha como se puede apreciar en la ilustración 6 anterior) con lo que habrás creado el

usuario *usuario1* de clave *usuario1* y la base de datos *práctica1* con las 3 tablas que la componen (haciendo doble clic en cada una de ellas podrás ver sus registros)

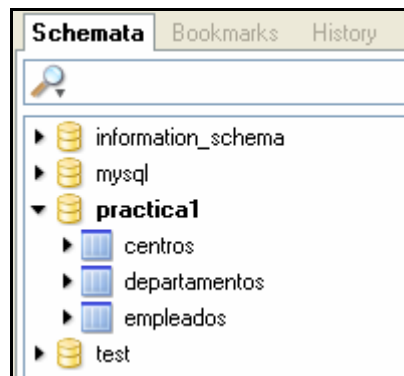


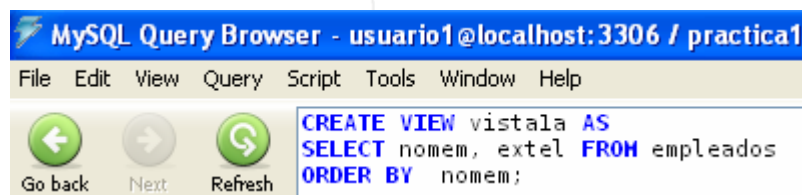
Ilustración 7. Base de datos *practica1* creada

## 1.2 Ventajas de su utilización

Nota: Para probar los ejemplos siguientes puedes hacerlo conectándote a la base de datos (*default schema*) *práctica1* como usuario *usuario1* de clave *usuario1* mediante las dos formas vistas en el apartado anterior. Por su sencillez y claridad utilizaremos la segunda de ellas, la herramienta gráfica *MySQL Query Browser*. Después de introducir cada sentencia recuerda pulsar el botón *Execute*.

Antes de definir lo que es una vista vamos a ver las ventajas que nos aportan.

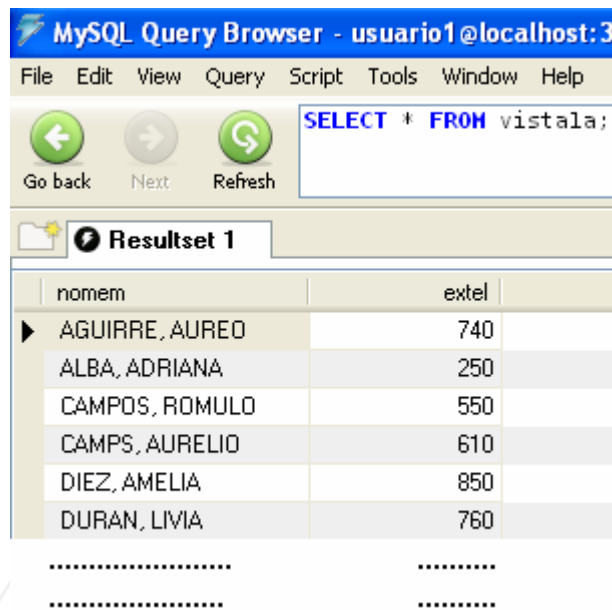
- Seguridad y confidencialidad: Si un empleado de la centralita telefónica necesita para su trabajo conocer exclusivamente el nombre y extensión telefónica de todos sus Compañeros, el resto de columnas con datos privados como el número de hijos, sueldo... de la tabla debe ser “invisible” para él. Por tanto el Administrador de la base de datos creará una “visión parcial” (*vistala* en el ejemplo) de la tabla:



Instrucción 1

y sólo le permitirá consultar dicha vista y no la tabla original:





### Instrucción 2

Desde el punto de vista de la administración, es más fácil administrar permisos sobre una vista que asignar privilegios a las columnas de las tablas.

- Facilidad de uso para consultas "complejas". Si un Jefe desea consultar los Centros junto con sus Departamentos y nombres de los Directores:

Centro nº	Nombre Centro	Departamento nº	Nombre Departamento	Director
10	SEDE CENTRAL	130	FINANZAS	GARCIA, AUGUSTO
10	SEDE CENTRAL	120	ORGANIZACION	PEREZ, JULIO
10	SEDE CENTRAL	121	PERSONAL	PEREZ, JULIO
10	SEDE CENTRAL	122	PROCESO DE DATOS	CAMPS, AURELIO
10	SEDE CENTRAL	100	DIRECCION GENERAL	LOPEZ, ANTONIO
20	RELACION CON CLIENTES	110	DIRECCION COMERCIAL	PEREZ, MARCOS
20	RELACION CON CLIENTES	111	SECTOR INDUSTRIAL	PEREZ, MARCOS
20	RELACION CON CLIENTES	112	SECTOR SERVICIOS	GARCIA, OCTAVIO

**Ilustración 8. Centros, Departamentos y Directores**

en lugar de tener que realizar la siguiente consulta:

```
SELECT c.numce AS "Centro nº", c.nomce AS "Nombre Centro",
       d.numde AS "Departamento nº",
       d.nomde AS "Nombre Departamento", e.nomem AS "Director"
FROM centros c
JOIN departamentos d ON c.numce= d.numce
JOIN empleados e ON d.dirrec=e.numem
ORDER BY c.numce;
```

### Instrucción 3

el Administrador puede preparar la siguiente vista:

```
CREATE VIEW vista1b AS
SELECT c.numce AS "Centro nº", c.nomce AS "Nombre Centro",
       d.numde AS "Departamento nº",
       d.nomde AS "Nombre Departamento", e.nomem AS "Director"
FROM centros c
JOIN departamentos d ON c.numce= d.numce
JOIN empleados e ON d.direc=e.numem
ORDER BY c.numce;
```

**Instrucción 4**

para que el Jefe sólo tenga que consultar:

```
SELECT * FROM vista1b
```

**Instrucción 5**

y con ello obtendrá las 8 filas y 5 columnas anteriores con toda la información sobre centros, departamentos y sus directores.

Nota: Si no estás acostumbrado a la sintaxis anterior en la que se utiliza la cláusula JOIN ... ON para unir tablas puedes utilizar esta otra:

```
CREATE OR REPLACE VIEW vista1b AS
SELECT c.numce AS "Centro nº", c.nomce AS "Nombre Centro",
       d.numde AS "Departamento nº",
       d.nomde AS "Nombre Departamento", e.nomem AS "Director"
FROM centros c, departamentos d, empleados e
WHERE c.numce=d.numce AND d.direc=e.numem
ORDER BY c.numce;
```

**Instrucción 6**

Hay que advertir que aunque el aspecto de una vista sea el de una tabla, una vista no es nada más que una instrucción SELECT del SQL almacenada con un nombre (vista1a, vista1b...). Por tanto no contiene datos (filas) permanentemente como una tabla y de esta manera no hay ficheros asociados a ellas. De todas formas, como se puede comprobar con la última instrucción lanzada, la forma de manejar vistas y tablas es muy similar y podemos estar consultando un origen de datos sin saber si éste es una tabla o una vista.

## 1.3 Creación. Diccionario de Datos

Pueden utilizarse las vistas a partir de la versión de MySQL 5.0.1.

Para poder crear vistas se requiere el privilegio CREATE VIEW así como los privilegios para poder seleccionar las columnas que forman parte de la vista. Sintaxis básica:

```
CREATE [OR REPLACE]
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
[DEFINER = {nombreusuario} | CURRENT_USER]
[SQL SECURITY {DEFINER | INVOKER}]
VIEW nombre_vista [(lista_columnas)]
  AS sentencia_select
  [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Recuerda que las palabras reservadas que aparecen entre corchetes [] son opcionales. La | indica elegir una opción entre las posibles alternativas.

La sentencia anterior crea una nueva vista o reemplaza una existente (cláusula OR REPLACE) si esta ya existiera.

El creador de la vista, en este caso *usuariol*, debe estar autorizado para poder crear vistas, es decir, poseer el privilegio CREATE VIEW. Si observas las líneas 4 y 5 del script *tablas1.sql* con el que estás trabajando y que aparecen a continuación, el Administrador *root* se lo ha concedido, tanto este privilegio como todos los posibles:

```
4 GRANT ALL ON practical.* TO usuariol IDENTIFIED BY 'usuariol';
5 FLUSH PRIVILEGES;
```

#### Instrucción 7

El resto de cláusulas de creación de vistas:

- ALGORITHM permite elegir entre dos modos de ejecución: MERGE o TEMPTABLE. Si se utiliza MERGE, las consultas que se ejecutan sobre la vista se combinan con la consulta de definición de la vista para crear una nueva consulta. Si se usa TEMPTABLE, se utiliza la definición de la vista para crear una tabla temporal donde se almacenan los datos (y por tanto las consultas se ejecutan sobre esta tabla temporal). Es una cláusula opcional. Si no se especifica o se define como UNDEFINED, MySQL analizará la mejor opción posible, generalmente MERGE pues el inconveniente que presenta la opción TEMPTABLE es que las tablas temporales sobre las que trabaja no tienen índices y por tanto el rendimiento de las consultas puede ser inferior si se utiliza esta opción.

- DEFINER (a partir de MySQL 5.1.2) especifica el propietario de la vista. Por defecto es el usuario actual (CURRENT\_USER).

- SQL SECURITY (a partir de MySQL 5.1.2) especifica el modo de interactuar con la vista. Se puede precisar que las consultas que se efectúen sobre la vista se traten con los privilegios del propietario de la vista (DEFINER) o con los privilegios de quien ejecute las consultas sobre la vista (INVOKER); la primera de las dos anteriores es la opción por defecto.

- *nombre\_vista* es el nombre que se da a la vista, no debe ser el mismo que el utilizado para una tabla. Si se desea crear la vista en otra base de datos distinta de la actual entonces deberá especificarse la base de datos mediante la siguiente notación: *base\_datos.nombre\_vista*.

- *lista\_columnas* son los nombres de columnas que va a contener la vista. Si no se ponen, se entenderá que serán todas las columnas que devuelva la sentencia SELECT de consulta; esta última determinará las columnas y tablas que aparecerán en la vista; si se indica la lista de columnas deberá hacerse separada por comas y el número de nombres debe ser el mismo que el número de columnas devuelto por la sentencia SELECT. Al igual que las tablas, las vistas no pueden tener nombres de columnas duplicados. Las columnas devueltas por la sentencia SELECT pueden ser simples referencias a columnas de la tabla, pero también pueden ser expresiones conteniendo funciones, constantes, operadores, etc.

- WITH CHECK OPTION añade una opción de seguridad a la hora de realizar operaciones de manipulación de datos utilizando vistas (en el apartado 1.6 veremos por su importancia algún ejemplo). Utilizando LOCAL los datos que se inserten a través de la vista deben respetar solamente las restricciones de la vista. Si se especifica CASCADED en el momento de la creación de la vista entonces deben respetarse las restricciones de la vista fuente o “padre” a partir de la cual fue creada. La opción por defecto es CASCADED.

Ejemplos de utilización.

Notas:

- El nombre que le damos a las vistas en este tema, para su fácil localización, es *vista1a*, *vista1b*... Realmente debemos utilizar nombres mucho más significativos que indiquen qué información va a presentar (como p.ej. los utilizados en los ejemplos del apartado 6).
- Todos los ejemplos se ejecutan desde el browser teniendo seleccionada la base de datos *practical* (ver ilustración 7 anterior). De esta manera se omite la selección de la misma (sentencia *USE practical*) o preceder al nombre de cada tabla o vista el nombre de la base de datos (*SELECT \* FROM practical.vista1a*);).

Ejemplo1. Nombre y presupuesto de todos los departamentos del centro número 10:

```
CREATE OR REPLACE VIEW vista1c AS
SELECT nomde, presu
FROM departamentos
WHERE numce=10;
```

**Instrucción 8**

Si consultamos la vista anterior:



nomde	presu
DIRECCION GENERAL	12000000
ORGANIZACION	3000000
PERSONAL	2000000
PROCESO DE DATOS	6000000
FINANZAS	2000000

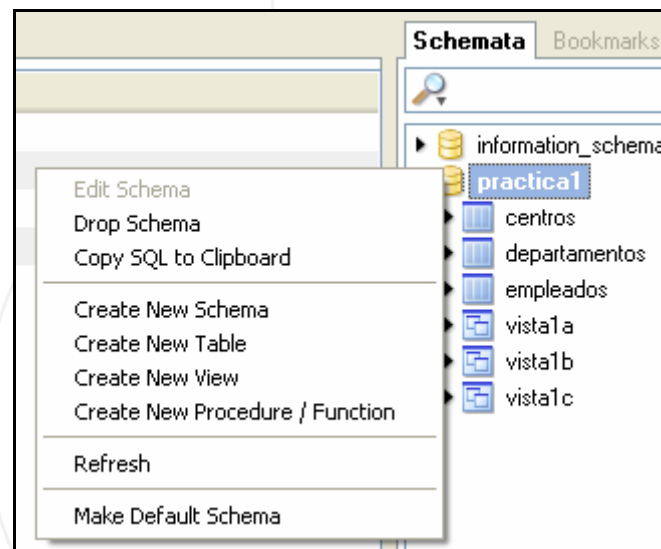
**Instrucción 9**

Como comentábamos anteriormente, a efectos de consulta y manipulación, una vista puede ser considerada como una tabla. A continuación seleccionamos de la vista anterior aquellos departamentos que superan los 5000000 de presupuesto:

```
SELECT *
FROM
vistalc
WHERE presu > 5000000
```

**Instrucción 10**

Posiblemente una vez creada no puedas ver la vista en el panel del esquema a no ser que cierres el programa cliente gráfico y vuelvas a entrar. Para evitar esto, selecciona con el botón derecho la base de datos *practical* y elige la opción de refrescar (*Refresh*) tal y como aparece en la ilustración 9:

**Ilustración 9. Refrescando la base de datos practical1**Ejemplo 2.

Nombre, salario y número de departamento de los empleados que superan los 400 de salario:

```
CREATE OR REPLACE
VIEW vistald AS
SELECT nomem, salario, numde
FROM empleados
WHERE salario > 400;
```

**Instrucción 11**Ejemplo 3.

A quién debemos felicitar el 12 de octubre:

```
CREATE OR REPLACE VIEW vistale
AS
SELECT * FROM empleados
WHERE nomem LIKE '%PILAR'
```

**Instrucción 12**Ejemplo 4

Nombre, número de departamento y antigüedad en años en la empresa de los empleados (recordad que la columna fecin contiene la fecha de ingreso en la empresa):

```
CREATE OR REPLACE VIEW vistalf
AS
SELECT nomem, numde, YEAR(CURRENT_DATE) - YEAR(fecin) AS ANTIGÜEDAD
FROM empleados
```

### Instrucción 13

#### Ejemplo 5

Total de empleados y media de sus salarios

```
CREATE OR REPLACE VIEW vistalg
AS
SELECT COUNT(*) AS "Total de empleados", AVG (salario) AS "Media de salarios"
FROM empleados;

SELECT * FROM vistalg;
```

Total de empleados	Media de salarios
34	302.9412

### Instrucción 14

¿Dónde se guarda la definición de la vista?

Si miras el panel *Schemata* (ilustraciones 9 y 10) en la parte superior derecha del browser encontrarás información sobre las base de datos. Una de ellas, la *information\_schema*, contiene datos acerca de los datos, es decir información sobre todos los objetos de las bases de datos existentes en el servidor, es lo que se conoce como **DICCIONARIO DE DATOS** o **CATALOGO del SISTEMA**. La información que contienen sus tablas es de sólo lectura; realmente no se trata de tablas sino de vistas. Haciendo doble clic en cada una de sus entradas verás toda la información relativa a la misma. Si lo haces sobre el objeto **VIEWS** podrás ver todas las características de las vistas existentes en las bases de datos del servidor MySQL (ilustración 11 siguiente):

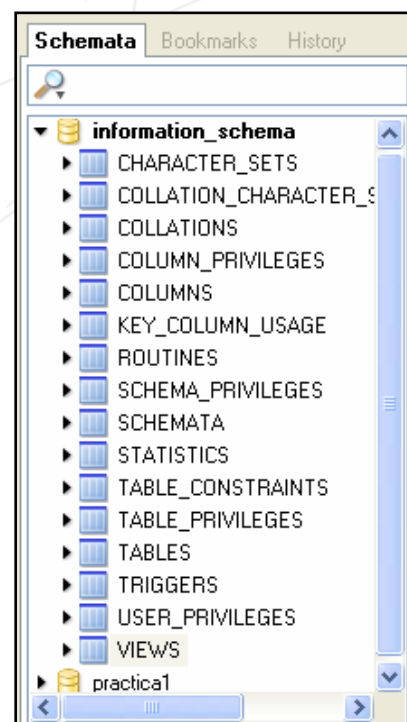


Ilustración 10. Schemata

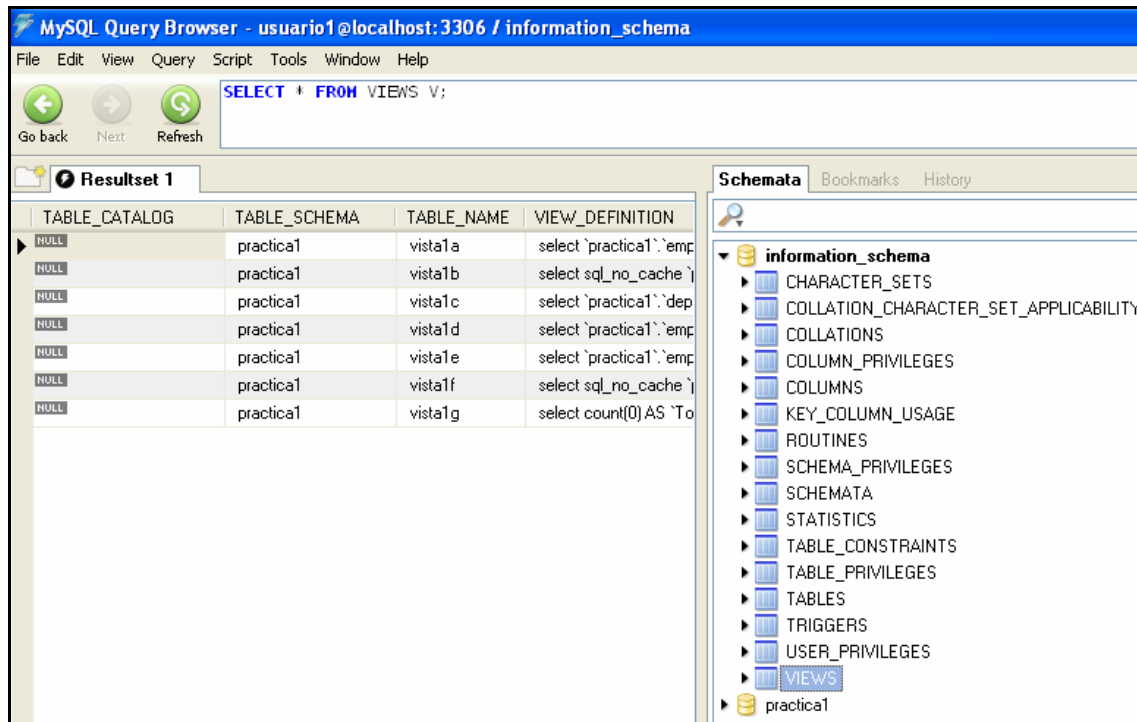


Ilustración 11. Vista VIEWS

Si tuvieras acceso a varias bases de datos y sólo quisieras saber las vistas de que dispones en la base de datos *practical*, la misma información anterior la podrías obtener:

```
SELECT * FROM information_schema.VIEWS
WHERE table_schema = 'practical';
```

Instrucción 15

Viendo las filas resultantes puedes comprobar que la columna `TABLE_NAME` contiene el nombre de la vista y que la columna `VIEW_DEFINITION` la sentencia `SELECT` que la construye. Por tanto si intentarás ver esta última columna solamente y para una vista en concreto:

```
SELECT VIEW_DEFINITION FROM information_schema.VIEWS
WHERE table_schema = 'practical'
AND table_name = 'vista1g';
```

Instrucción 16

Lo que ocurre con la anterior consulta es que puedes encontrar una expresión un poco confusa a la hora de mostrar la sentencia `SELECT` y además en caso de que ésta sea larga no verse por completo y aparecer partida. Por eso quizás sea mejor seleccionar la vista deseada del diccionario de datos, en concreto su columna `VIEW_DEFINITION` (p.ej. de cualquiera de las vistas que aparece en la ilustración 11 anterior) y con el botón derecho elegir la opción *View Field in Popup Editor*.

También puedes conseguir ver completa la instrucción que construye la vista mediante una ventana cliente (*Tools / MySQL Command Line Client* de la barra de menús) y el comando:

```
SHOW CREATE VIEW practica1.vista1g;
```

A través de los conceptos aquí expuestos podrás comprender la funcionalidad del resto de columnas de la tabla (vista) anterior.

## 1.4 Modificación

La sintaxis básica es muy similar a la de creación:

```
ALTER
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
[DEFINER = {nombreusuario} | CURRENT_USER]
[SQL SECURITY {DEFINER | INVOKER}]
VIEW nombre_vista [(lista_columnas)]
  AS sentencia_select
  [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Lo que hace es modificar la especificación de la vista. Realmente es como si se borrara y se volviera a crear con la nueva especificación de la sentencia SELECT por lo que no se suele utilizar esta sentencia puesto que se puede conseguir los mismos resultados que utilizando la cláusula REPLACE de la sentencia CREATE. Como equivale a borrarla y volverla a crear, el usuario que lo haga debe tener los privilegios de creación y borrado para vistas y los de las columnas referenciadas en la sentencia SELECT de creación de la vista.

Ejemplo, si queremos que la última columna de la *vistaIf* no sea la antigüedad en la empresa sino la edad de los empleados, podemos borrar la vista (siguiente apartado) y volverla a crear o:

```
ALTER VIEW vistalf
AS
SELECT nomem, numde, YEAR(CURRENT_DATE) - YEAR(fecna) AS AÑOS
FROM empleados;
```

**Instrucción 17**

## 1.5 Borrado

A partir de la versión de MySQL 5.0.1 como las anteriores sentencias. Sintaxis:

```
DROP VIEW [IF EXISTS]
vista1 [, vista2] ...
```

Borra una o más vistas. Para poder realizarlo se debe poseer los privilegios de borrado sobre vistas. La cláusula IF EXISTS evita que aparezca mensaje de error en caso de que la vista a borrar no exista y en su lugar aparece un mensaje informativo.



Ejemplo de borrado de la vista vista1b

```
DROP VIEW vista1b
```

### Instrucción 18

## 1.6 Operaciones sobre vistas. Restricciones

Antes de indicar las operaciones que se pueden realizar sobre las vistas y sus restricciones, apuntar las normas más importantes a tener en cuenta a la hora de la crear vistas:

- La sentencia SELECT de creación de la vista no puede hacer referencia en la cláusula FROM ni a una tabla temporal (TEMPORARY TABLE) ni contener una subconsulta.
  - La sentencia SELECT no puede referirse a variables de usuario o de sistema.
  - Dentro de un procedimiento almacenado (se estudian en el siguiente tema), la definición de la vista no puede hacer referencia ni a los argumentos del procedimiento ni a las variables locales del mismo.
  - Cualquier tabla o vista a la que referencia la nueva vista debe existir previamente.
  - No se puede asociar un disparador con una vista (se tratan los disparadores también en el siguiente tema de este curso).
- a) **Consulta** Ya visto en los apartados anteriores.
- b) **Actualización** En general, si una vista está basada en una sola tabla, al modificar la vista se está modificando directamente la tabla.

Ejemplo: Si tenemos la siguiente vista:

```
CREATE VIEW vista1h
AS SELECT numem, nomem, extel
FROM empleados
```

### Instrucción 19

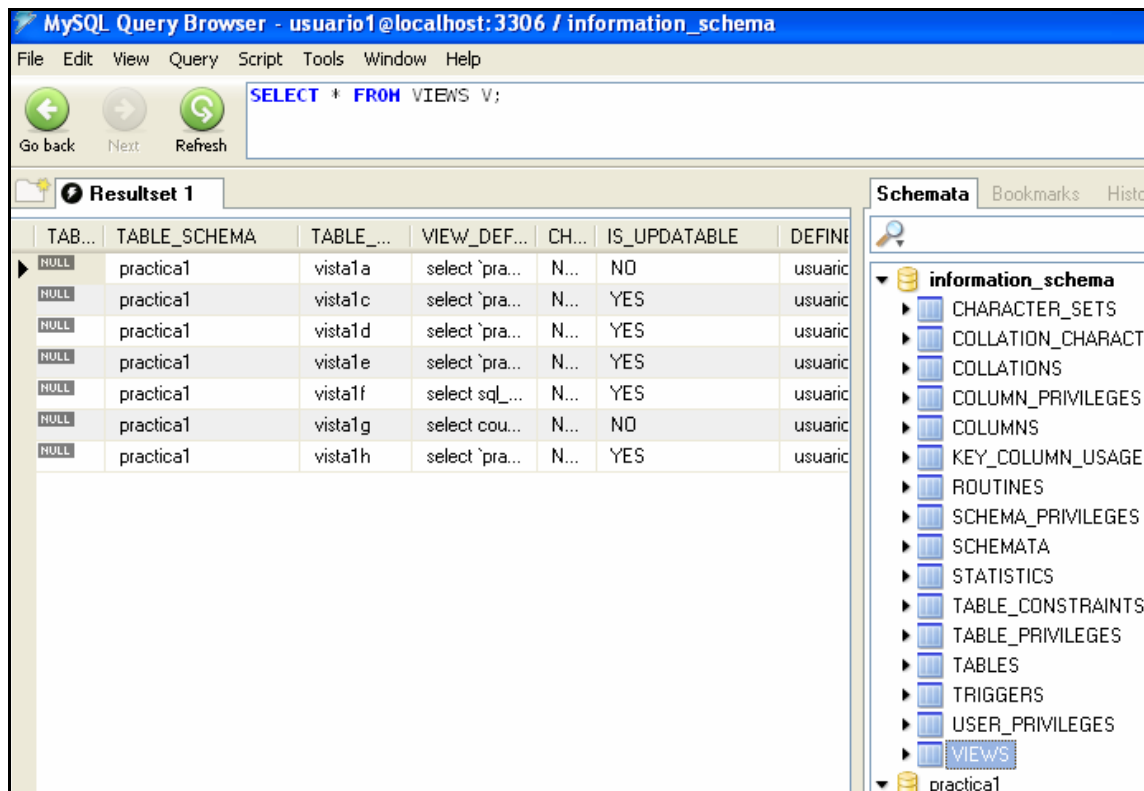
Cualquiera de las dos actualizaciones siguientes nos lleva al mismo resultado, actualizando 3 filas, cambiando la extensión telefónica 760 por la 990 a 3 empleados de la empresa. La diferencia está en que sentencia de la derecha se está modificando la tabla empleados a través de la vista y en la de la izquierda se hace directamente a través de la tabla.

<pre>UPDATE empleados SET extel=990 WHERE extel=760</pre> <p><b>Instrucción 20</b></p>	<pre>UPDATE vista1h SET extel=990 WHERE extel=760</pre> <p><b>Instrucción 21</b></p>
--	--

¿Es posible poder realizar actualizaciones de las tablas a través de las vistas?

No siempre. Depende de la sentencia SELECT que construye la vista a partir de la tabla. Antes de ver algunas restricciones ¿Cómo se puede averiguar si se pueden

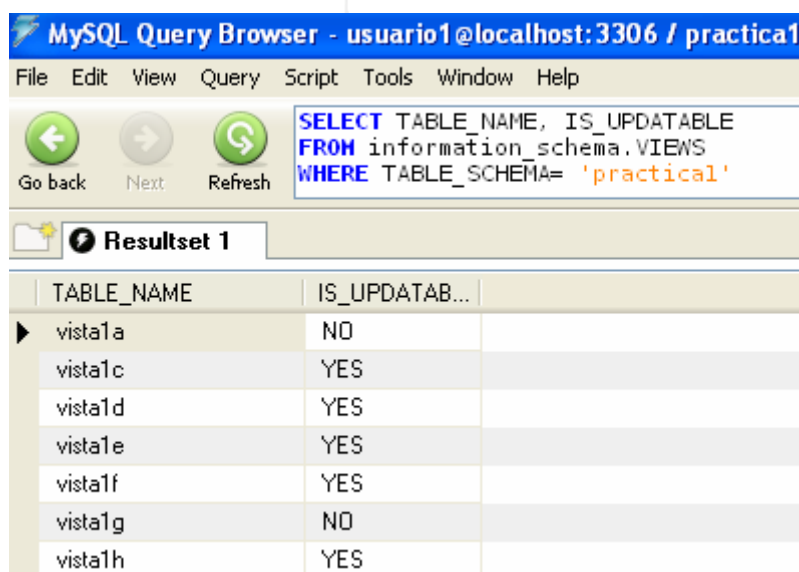
modificar datos de una tabla a través de una vista? Muy sencillo, consultando el diccionario de datos o lo que es lo mismo, la base de datos *information\_schema*



**Ilustración 12. Vista VIEWS**

La columna `IS_UPDATABLE` de la vista `VIEWS` nos indica si a través de una vista puedo modificar (YES) o no una tabla.

Recuerda que puedes obtener la misma información “sin moverte” de tu esquema (base de datos) *practical*:



**Instrucción 22**

¿Por qué *vista1a* y *vista1g* no son actualizables y el resto sí?

Hay una serie de restricciones a tener en cuenta. Si se da alguna de las características siguientes en la sentencia SELECT que construye la vista a partir de la tabla, la vista no es actualizable:

- Contiene funciones de agregado como SUM(), MIN(), MAX(), COUNT() (por eso la *vista1g* no es actualizable).
  - Contiene la cláusula DISTINCT o GROUP BY o HAVING o UNION o JOIN u ORDER BY (*vista1a*).
  - Contiene una subconsulta.
- c) **Inserción** ¿Es posible realizar inserciones en la tabla a través de una vista definida sobre ella? Si como podrás comprobar con este ejemplo, que inserta una fila en la tabla empleados a través de una vista construida sobre ella:

```
INSERT INTO vista1h values (560, 'CARRERA, ALBERTO', 990)
```

#### Instrucción 23

Pero la tabla base empleados tiene muchos más campos como número de departamento, salario, comisión, número de hijos... ¿Con qué valores se rellenan estos valores? Mira y verás:

numem	numde	extel	fecna	fecin	sal...	comisi...	numhi	nomem
560	NULL	990	NULL	NULL	NULL	NULL	NULL	CARRERA, ALBERTO

Ilustración 13. Campos de la tabla empleados

¿Por qué se ha podido hacer esto?

Si miras la descripción de la tabla mediante el script que lanzaste para crearla o directamente desde el browser (botón derecho sobre la tabla *empleados* de la base de datos *practical* – opción *Edit Table*) como aparece en la siguiente figura, verás que el único campo que no debe quedar nulo (por ser clave primaria en este caso) es el de número de empleado (*numem*):

Column Name	Datatype	NOT NULL
numem	INTEGER	✓
numde	INTEGER	
extel	INTEGER	
fecna	DATE	
fecin	DATE	
salario	INTEGER	
comision	INTEGER	
numhi	INTEGER	
nomem	VARCHAR(18)	

Ilustración 14. Descripción de la tabla empleados

Siempre que no nos dejemos de rellenar los campos obligatorios de la tabla base podremos hacer inserciones a través de la vista.

En cambio, si utilizamos la vista *vistald* (contiene las columnas nombre, salario y número de departamento) para insertar en la tabla base de *empleados* no podremos hacerlo pues nos estamos dejando de introducir el campo *numem* (que es obligatorio y no autonumérico).

Realmente la información de esas tres últimas líneas no es del todo cierta, pues si pruebas a realizar la siguiente inserción:

```
INSERT INTO vistald VALUES ('CARRERA, RAQUEL', 300, 120)
```

#### Instrucción 24

te va a dejar hacerlo siendo que falta el campo obligatorio (y no autonumérico) del número de empleado (*numem*). La razón de ello, como podrás comprobar tal y como hemos hecho en la descripción de esta tabla, es que esta columna tiene como valor por defecto 0 y por tanto le asigna a Raquel el número de empleado 0. A partir de ese momento ya no puedes hacer más inserciones como la que viene a continuación pues ya no puede volver a asignarle el 0 como número de empleado al estar la clave repetida:

```
INSERT INTO vistald VALUES ('CARRERA, MARIO', 250, 100)
```

#### Instrucción 25

Lo que debes recordar es que siempre podremos realizar inserciones en las tablas a través de sus vistas siempre y cuando no nos dejemos los campos obligatorios de la tabla. Deberemos tener en cuenta además de la anterior todas las restricciones que aparecen al final del apartado b anterior.

Antes de terminar el tema comentar una opción que puede añadir cierta seguridad a las vistas y que se había dejado planteada en el apartado 1.3 de este tema en relación a la creación de vistas y su sintaxis básica. La cláusula es:

[WITH CHECK OPTION]
---------------------

¿Para qué sirve ?

Observa la siguiente vista:

```
CREATE OR REPLACE VIEW vistali
AS
SELECT * FROM empleados
WHERE numde=100;
```

#### Instrucción 26

La sentencia anterior crea una vista con todos los datos de los empleados del Departamento 100. A través de esta vista, como se ha trabajado en los puntos anteriores se puede realizar cualquier operación de manipulación de la tabla base empleados. Por eso, estas dos instrucciones son totalmente correctas:

```
INSERT INTO vistali
(numem, numde, nomem)
VALUES (600, 100, 'CARRERA, MARIO')
```

**Instrucción 27**

```
INSERT INTO vistali
(numem, numde, nomem)
VALUES (610, 120, 'BAILIN, CARMEN')
```

**Instrucción 28**

Pero en la segunda de ellas, hemos utilizado una vista que representa los empleados del Departamento 100 para introducir un empleado que no cumple realmente la cláusula WHERE del SELECT de creación de la vista ya que su departamento es el 120.

Si hubiéramos creado la vista de esta otra forma (se ha añadido la comprobación al final):

```
CREATE OR REPLACE VIEW vistali
AS
SELECT * FROM empleados
WHERE numde=100
WITH CHECK OPTION;
```

**Instrucción 29**

no hubiéramos tenido problemas en añadir al empleado Mario Carrera por ser del Departamento 100 pero no nos hubiera dejado realizar la inserción de la empleada Carmen Bailín dando mensaje de error ya que antes de hacer la inserción MySQL chequea que los datos a introducir cumplan con la selección de la cláusula WHERE de la sentencia SELECT de creación de la vista que seleccionaba las filas de empleados del Departamento 100 y estamos intentando introducir una empleada de otro Departamento. Recordad que tal y como se expuso en el apartado 1.3, la cláusula WITH CHECK OPTION equivale a WITH CASCADED CHECK OPTION en caso de que no se especifique la opción LOCAL, al ser CASCADED la opción por defecto, y por tanto todas las vistas que se creen a partir de la vista *vistali* deberán seguir respetando las restricciones de la cláusula WHERE de esta última.

## 1.7 Ejercicios resueltos.

Los siguientes ejemplos sirven para completar los vistos en este tema así como para recordar las sentencias SQL y poder hacer los ejercicios propuestos.

Centros y sus Departamentos (puedes utilizar si lo deseas la cláusula JOIN... ON en lugar de la aquí propuesta):

```
CREATE VIEW centros_departamentos
AS
SELECT nomce 'Centro', nomde 'Departamento'
FROM centros, departamentos
WHERE centros.numce=departamentos.numce
ORDER BY nomce
```

**Instrucción 30**

Centro	Departamento
RELACION CON CLIENTES	DIRECCION COMERCIAL
RELACION CON CLIENTES	SECTOR INDUSTRIAL
RELACION CON CLIENTES	SECTOR SERVICIOS
SEDE CENTRAL	PROCESO DE DATOS
SEDE CENTRAL	FINANZAS
SEDE CENTRAL	ORGANIZACION
SEDE CENTRAL	DIRECCION GENERAL
SEDE CENTRAL	PERSONAL

**Ilustración 15. Resultado de la Instrucción 30**

**Instrucción 31. Departamentos que no dependen jerárquicamente de otros**

```
CREATE OR REPLACE VIEW empleados_por_departamento AS
SELECT CONCAT('En el Departamento ', numde, ' trabajan ', count(*), ' empleados ')
AS 'Número de empleados por Departamento'
FROM empleados
GROUP BY numde
```

**Instrucción 32. Número de empleados por Departamento**

Número de empleados por Departamento	
En el Departamento 100	trabajan 3 empleados
En el Departamento 110	trabajan 3 empleados
En el Departamento 111	trabajan 8 empleados
En el Departamento 112	trabajan 7 empleados
En el Departamento 120	trabajan 1 empleados
En el Departamento 121	trabajan 4 empleados
En el Departamento 122	trabajan 5 empleados
En el Departamento 130	trabajan 3 empleados

**Ilustración 16. Resultado de la Instrucción 32**

La misma consulta que la anterior instrucción 32 pero especificando el nombre del Departamento en lugar de su número:

```
CREATE OR REPLACE VIEW empleados_por_departamento AS
SELECT CONCAT('En el Departamento de ', nomde, ' trabajan ',
count(*), ' empleados')
AS 'Número de empleados por departamento'
FROM empleados, departamentos
WHERE empleados.numde=departamentos.numde
GROUP BY nomde
```

**Instrucción 33**

Número de empleados por departamento	
En el Departamento de DIRECCION GENERAL	trabajan 3 empleados
En el Departamento de DIRECCION COMERCIAL	trabajan 3 empleados
En el Departamento de FINANZAS	trabajan 3 empleados
En el Departamento de ORGANIZACION	trabajan 1 empleados
En el Departamento de PERSONAL	trabajan 4 empleados
En el Departamento de PROCESO DE DATOS	trabajan 5 empleados
En el Departamento de SECTOR INDUSTRIAL	trabajan 8 empleados
En el Departamento de SECTOR SERVICIOS	trabajan 7 empleados

**Ilustración 17. Resultado de la instrucción 33**