CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNIDAD 5

MODELO FÍSICO DML

BASES DE DATOS (BD) 22/23
CFGS DAW

PARTE 1. MANIPULACIÓN DE DATOS

Revisado por:
Sergio Badal, Abelardo Martínez y Pau Miñana
Autores:
Paco Aldarias
Fecha: 02/12/22

ÍNDICE DE CONTENIDO

UD05.1. MODELO FÍSICO DML. MANIPULACIÓN DE DATOS

> Importante
>
> Veremos en este tema las particularidades de Oracle, pero recordad que en esta primera parte del curso solo se os pide conocer y usar la sintaxis de MySQL.

1. INSERCIÓN DE DATOS

Hasta ahora hemos conseguido crear tablas, el próximo paso será incluir registros en las tablas que hemos creado.

> Importante
>
> El DML (Data Manipulaton Language) lo forman las instrucciones capaces de modificar los datos de las tablas.

Para insertar un registro utilizaremos la instrucción INSERT cuya sintaxis (una de las varias formas que tiene) es la siguiente:

```
INSERT INTO nombre_tabla [(nombre_col, ...)]
VALUES((expresión | DEFAULT),...)
```

En ella podemos distinguir las palabras reservadas INSERT INTO seguidas del nombre de la tabla en la que vamos a guardar los nuevos datos. Opcionalmente podemos poner entre paréntesis los nombres de los campos, si no los incluimos se deberán colocar los valores en el mismo orden en que fueron creados (es el orden de columnas según las devuelve el comando describe), pues de lo contrario se producirá un error si los tipos no coinciden o se almacenará la información de forma errónea en caso de que los tipos de datos de los campos sean compatibles.

```
NSERT INTO departamentos (CodDpto, Nombre, Ubicacion)
VALUES ('INF'. 'Informática', 'Planta sótano');
```

ATENCIÓN A LA SINTAXIS DE LA CLÁUSULA INSERT

- La lista de campos a rellenar (CodDpto, Nombre, Ubicacion) se indica solo si no queremos rellenar todos los campos.

```
– Si NO conocemos el orden de los campos al hacer CREATE TABLE

INSERT INTO departamentos (CodDpto, Nombre, Ubicacion)

VALUES ('INF', 'Informática', 'Planta sótano');
```

- Si no incluimos esa lista de campos, el SGBD entenderá que el orden es el especificado cuando creamos la tabla con CREATE TABLE.

```
– Si conocemos el orden de los campos al hacer CREATE TABLE

INSERT INTO departamentos

VALUES ('INF', 'Informática', 'Planta sótano');
```

- En cualquier caso, siempre podremos cambiar el orden de los mismos con la precaución de cambiar también el orden de los VALUES

```
– Lo conozcamos o no, siempre podemos cambiar el orden de los mismos

INSERT INTO departamentos (Nombre, Ubicacion, CodDpto)

VALUES ('Informática', 'Planta sótano', 'INF');
```

- Los campos no rellenados explícitamente con la orden INSERT, se rellenan con su valor por defecto (DEFAULT) o bien con NULL si no se indicó valor alguno.

```
– Asumimos que Ubicación tiene un valor por defecto

INSERT INTO departamentos (CodDpto, Nombre, Ubicacion)

VALUES ('INF', 'Informática');
```

- Si algún campo tiene restricción de obligatoriedad (NOT NULL), ocurrirá un error si no rellenamos el campo con algún valor.

Por ello, aunque sea un poco más pesado, nuestro consejo es que siempre que realices un INSERT incluyas los nombres de los campos, de tal forma que cuando vayas escribiendo los valores de los campos veas que el orden se corresponde con los campos indicados.

Veamos cómo funciona con un ejemplo. Recordemos las tablas Departamentos y Empleados que creamos en el tema pasado:

Tabla departamentos.

CodDpto	Nombre	Ubicación
INF	Informática	Planta sótano U3
ADM	Administración	Planta quinta U2
COM	Comercial	Planta tercera U3
CONT	Contabilidad	Planta quinta U1
ALM	Almacén	Planta baja U1

Tabla empleados.

DNI	Nombre	Especialidad	FechaAlta	Dpto
12345678A	Alberto Gil	Contable	10/12/2010	CONT
23456789B	Mariano Sanz	Informática	04/10/2011	INF
34567890C	Iván Gómez	Ventas	20/07/2012	COM
45678901D	Ana Silván	Informática	25/11/2012	INF
56789012E	María Cuadrado	Ventas	02/04/2013	COM
67890123A	Roberto Milán	Logística	05/02/2010	ALM

La primera forma será indicando todos los campos y todos los valores (recuerda que si queremos colocar una cadena el valor tendrá que ir entre comillas (da igual simples o dobles).

Para insertar el primer registro de la tabla departamentos haremos:

1.1 MYSQL

```
mysql> INSERT INTO departamentos (CodDpto, Nombre, Ubicacion)
    -> VALUES ('INF','Informática','Planta sótano U3');
Query OK, 1 row affected (0.05 sec)
```

Para ver si ha funcionado y el registro se ha insertado en la tabla utilizaremos la instrucción SELECT.

Esta instrucción, que veremos muy a fondo en el próximo tema, sirve para hacer consultas a la base de datos y mostrar la información que contiene. En este ejemplo vamos a utilizar la consulta más simple, por ejemplo:

SELECT * FROM departamentos;

Donde:

- SELECT es palabra reservada que indica seleccionar
- El asterisco indica todos los campos de la tabla

- La palabra reservada FROM indica de dónde se van a sacar los datos e irá seguida del nombre de la tabla de donde obtendremos la información.

Es decir, la instrucción selecciona todos los campos de la tabla departamentos y muestra su contenido. El resultado será:

```
mysql> select * from departamentos;
+---------+-------------+-------------------+
| CodDpto | Nombre      | Ubicacion         |
+---------+-------------+-------------------+
| INF     | Informática | Planta sótano U3  |
+---------+-------------+-------------------+
1 row in set (0.00 sec)
```

Podemos observar el resultado obtenido donde aparece el registro que hemos incluido con la instrucción INSERT.

Vamos a probar insertando el segundo registro de la tabla departamentos sin incluir los nombres de los campos en la instrucción.

```
mysql> INSERT INTO departamentos
    -> VALUES ('ADM','Administración','Planta quinta U2');
Query OK, 1 row affected (0.27 sec)
```

Como puedes apreciar la instrucción ha funcionado correctamente y ahora utilizaremos de nuevo el SELECT para ver el contenido de la tabla.

```
mysql> select * from departamentos;
+---------+----------------+-------------------+
| CodDpto | Nombre         | Ubicacion         |
+---------+----------------+-------------------+
| ADM     | Administración | Planta quinta U2  |
| INF     | Informática    | Planta sótano U3  |
+---------+----------------+-------------------+
2 rows in set (0.00 sec)
```

Correcto, ya tenemos dos registros en nuestra tabla departamentos, pero ¿siempre hay que ir uno a uno?

El SQL de MySQL nos proporciona un INSERT extendido que nos permite realizar la inserción de varios registros en una sola instrucción. La sintaxis es la siguiente:

```
INSERT [INTO] nombre_tabla [(nombre_col, ...)]
VALUES ({expre | DEFAULT}, ... ), (...), ...
```

```
INSERT INTO departamentos (CodDpto, Nombre, Ubicacion) VALUES
('INF'. 'Informática', 'Planta sótano'),
```

```
('VAL'. 'Valencià', 'Planta 4'),

('FOL'. 'Form y Orien Laboral', 'Planta 3');
```

Como puedes observar la sintaxis es prácticamente igual a la anterior con la salvedad que ahora podemos colocar los datos de varios registros entre paréntesis separados por comas. También ahora los nombres de las columnas son opcionales.

Para probarlo vamos a insertar dos registros más en nuestra tabla de departamentos:

```
mysql> INSERT INTO departamentos (CodDpto, Nombre, Ubicacion)
    -> VALUES ('COM','Comercial','Planta tercera U3'),
    -> ('CONT','Contabilidad','Planta quinta U1');
Query OK, 2 rows affected (0.05 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Podemos ver en el mensaje como indica que se han añadido dos registros.

```
mysql> select * from departamentos;
+---------+----------------+--------------------+
| CodDpto | Nombre         | Ubicacion          |
+---------+----------------+--------------------+
| ADM     | Administración | Planta quinta U2   |
| COM     | Comercial      | Planta tercera U3  |
| CONT    | Contabilidad   | Planta quinta U1   |
| INF     | Informática    | Planta sótano U3   |
+---------+----------------+--------------------+
4 rows in set (0.00 sec)
```

Ahora nuestra tabla ya tiene cuatro registros insertados.

1.2 ORACLE

```
SQL> INSERT INTO departamentos (CodDpto, Nombre, Ubicacion)
  2  VALUES ('INF', 'Informática', 'Planta sótano U3');

1 row created.
```

Hemos ejecutado la misma instrucción en Oracle y el resultado ha sido satisfactorio, indicando una fila creada.

Para ver si ha funcionado utilizaremos la misma instrucción SELECT comentada en la parte de MySQL.

```
SQL> select * from departamentos;

CODDPTO    NOMBRE                         UBICACION
---------- ------------------------------ ------------------------------
INF        Informática                    Planta sótano U3
```

Como podemos ver ha funcionado perfectamente y el departamento de Informática ya está en la tabla. Incluyamos ahora el segundo registro sin especificar los campos:

```
SQL> insert into departamentos
  2  values('ADM','Administración','Planta quinta U2');

1 row created.
```

El registro se creó satisfactoriamente y comprobamos el contenido de la tabla con SELECT.

```
SQL> select * from departamentos;

CODDPTO    NOMBRE                         UBICACION
---------- ------------------------------ ------------------------------
INF        Informática                    Planta sótano U3
ADM        Administración                 Planta quinta U2
```

La inserción extendida que hemos visto en MySQL no se puede aplicar en Oracle. Pero Oracle también tiene su forma de poder insertar varios registros seguidos. Sin embargo, el ahorro de código a teclear es prácticamente nulo, ya que es casi lo mismo que hacer varios INSERT individuales en un script normal.

Para insertar los dos registros en Oracle podemos utilizar:

```
INSERT ALL
    INTO departamentos values('COM','Comercial','Planta tercera U3')
    INTO departamentos values('CONT','Contabilidad','Planta quinta U1')
SELECT * FROM DUAL;
```

La tabla DUAL es una tabla especial de una sola columna presente de manera predeterminada en todas las instalaciones de Oracle.

Lo guardamos en un archivo y ejecutamos el script correspondiente.

```
SQL> @ c:\src\insertvarios.sql

2 rows created.
```

Después podemos observar el resultado.

```
SQL> SELECT * FROM DEPARTAMENTOS;

CODDPTO    NOMBRE                         UBICACION
---------- ------------------------------ ------------------------------
COM        Comercial                      Planta tercera U3
CONT       Contabilidad                   Planta quinta U1
INF        Informática                    Planta sótano U3
ADM        Administración                 Planta quinta U2
```

1.3 Errores en la inserción

A la hora de insertar datos se pueden producir errores por diversas causas, algunos errores que irán en contra de las reglas establecidas mostrarán el error y ya está, pero otros, los más difíciles de encontrar son los que producimos los usuarios y que para la base de datos pueden pasar como correctos.

ATENCIÓN A LOS ERRORES MÁS COMUNES DE LA CLÁUSULA INSERT

1. **Repetir la inserción de un registro que ya existe. Como la clave primaria debe ser única debe mostrar un error al intentar insertar un registro con la misma clave. (Ejemplo: insertar dos veces el mismo empleado)**
2. **Insertar registro(s) con un campo clave ajena a otro registro que todavía no se ha insertado. (Ejemplo: insertar un empleado asociado a un departamento que no se ha insertado aún)**
3. **Insertar registro(s) con campos NOT NULL y no indicar el valor de esos campos.**
4. **Insertar registro(s) con valores que no corresponden con el tipo de datos de ese atributo.**

1.3.1 MySQL

```
mysql> select * from departamentos;
+---------+----------------+--------------------+
| CodDpto | Nombre         | Ubicacion          |
+---------+----------------+--------------------+
| ADM     | Administración | Planta quinta U2   |
| COM     | Comercial      | Planta tercera U3  |
| CONT    | Contabilidad   | Planta quinta U1   |
| INF     | Informática    | Planta sótano U3   |
+---------+----------------+--------------------+
4 rows in set (0.00 sec)

mysql> insert into departamentos values('INF','Informática','Planta sótano U3');

ERROR 1062 (23000): Duplicate entry 'INF' for key 1
mysql>
```

Como podéis ver en la imagen aparece el error de entrada duplicada para la clave 1.

Puede ocurrir que al realizar un alta o inserción de un registro no rellenemos todos los campos de la tabla. Por ejemplo supongamos que vamos a tener un nuevo departamento de Marketing con el código MKT pero que no sabemos aún su ubicación. Podemos hacer su alta de la siguiente forma:

```
mysql> insert into departamentos (CodDpto, Nombre)
    -> values ('MKT','Marketing');
Query OK, 1 row affected (0.06 sec)

mysql> select * from departamentos;
+---------+----------------+--------------------+
| CodDpto | Nombre         | Ubicacion          |
+---------+----------------+--------------------+
| ADM     | Administración | Planta quinta U2   |
| COM     | Comercial      | Planta tercera U3  |
| CONT    | Contabilidad   | Planta quinta U1   |
| INF     | Informática    | Planta sótano U3   |
| MKT     | Marketing      | NULL               |
+---------+----------------+--------------------+
5 rows in set (0.03 sec)
```

Como puedes observar en el INSERT solamente hemos colocado el nombre de los dos campos que conocemos y el tercero, la ubicación se ha puesto a NULL.

Supongamos ahora que conocemos el código de otro nuevo departamento, AUD, pero no tenemos claro el nombre que va a tener, aunque su ubicación será en la Planta cuarta U1. Veamos cómo hacer el INSERT.

Como habrás imaginado, la nueva instrucción será igual que la anterior, pero cambiando el campo Nombre por el campo Ubicación:

```
mysql> insert into departamentos (CodDpto, Ubicacion)
    -> values ('AUD','Planta cuarta U1');
ERROR 1364 (HY000): Field 'Nombre' doesn't have a default value
mysql>
```

Vaya, esto sí que es un error. Dice que Nombre no tiene un valor por defecto. **¿Qué ocurre?**

Pues bien, si miramos la estructura de nuestra tabla:

```
mysql> desc departamentos;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| CodDpto   | varchar(10) | NO   | PRI | NULL    |       |
| Nombre    | varchar(30) | NO   |     | NULL    |       |
| Ubicacion | varchar(30) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.03 sec)

mysql>
```

Podemos ver que el campo nombre no puede ser NULO ya que se creó con NOT NULL. Por otro lado, no tiene establecido ningún valor por defecto (aparece NULL), luego no podemos insertar un registro que no tenga un valor para el campo Nombre; es decir, el valor de ese campo es obligatorio incluirlo en el INSERT.

Pero, tal y como he comentado antes, el error más difícil de encontrar es el error que cometemos los usuarios sin que para la base de datos sea considerado un error. Por ejemplo, si hacemos el siguiente INSERT:

```
mysql> insert into departamentos
    -> values('ALM','Planta baja U1','Almacén');
Query OK, 1 row affected (0.02 sec)
```

Vemos que no hay ningún mensaje de error, la instrucción INSERT es correcta y se añade el nuevo registro indicado. Vamos a comprobar los datos para verificarlo.

```
mysql> select * from departamentos;
+---------+----------------+--------------------+
| CodDpto | Nombre         | Ubicacion          |
+---------+----------------+--------------------+
| ADM     | Administración | Planta quinta U2   |
| ALM     | Planta baja U1 | Almacén            |
| COM     | Comercial      | Planta tercera U3  |
| CONT    | Contabilidad   | Planta quinta U1   |
| INF     | Informática    | Planta sótano U3   |
| MKT     | Marketing      | NULL               |
+---------+----------------+--------------------+
6 rows in set (0.00 sec)
```

Aparentemente todo parece correcto, pero ¿qué ha ocurrido con el segundo registro, el de código ALM? Fíjate, donde está el nombre aparece la ubicación y en ubicación aparece el nombre. Esto ha ocurrido porque espera los datos de los campos en el mismo orden en el que están en la tabla, ya que no hemos incluido el nombre de los campos en la instrucción.

1.3.2 Oracle

```
SQL> select * from departamentos;

CODDPTO    NOMBRE                         UBICACION
---------- ------------------------------ ------------------------------
INF        Informática                    Planta sótano U3
ADM        Administración                 Planta quinta U2
COM        Comercial                      Planta tercera U3
CONT       Contabilidad                   Planta quinta U1

SQL> insert into departamentos
  2  values('INF','Informática','Planta sótano U3');
insert into departamentos
*
ERROR at line 1:
ORA-00001: unique constraint (USUARIO_PRUEBA.SYS_C006997) violated


SQL> _
```

El error que muestra Oracle es diferente pero con el mismo significado. Indica que la restricción de unicidad está siendo incumplida.

```
SQL> insert into departamentos (CodDpto, Nombre)
  2  values ('MKT', 'Marketing');

1 row created.

SQL> select * from departamentos;

CODDPTO    NOMBRE                         UBICACION
---------- ------------------------------ ------------------------------
INF        Informática                    Planta sótano U3
ADM        Administración                 Planta quinta U2
COM        Comercial                      Planta tercera U3
CONT       Contabilidad                   Planta quinta U1
MKT        Marketing
```

En Oracle, no aparece el valor nulo al hacer el SELECT pero deja el campo en blanco. Veamos ahora cómo reacciona Oracle al introducir solo los valores para CODDPTO y UBICACION:

```
SQL> insert into departamentos (CodDpto,Ubicacion)
  2  values ('AUD','Planta cuarta U1');
insert into departamentos (CodDpto,Ubicacion)
*
ERROR at line 1:
ORA-01400: cannot insert NULL into ("USUARIO_PRUEBA"."DEPARTAMENTOS"."NOMBRE")
```

Pero fíjate en la información que aporta el mensaje de error, dice que no puede insertar NULO en el esquema USUARIO_PRUEBA, dentro de la tabla DEPARTAMENTOS, en el campo NOMBRE.

Si ahora ejecutamos el siguiente INSERT:

```
SQL> insert into departamentos
  2  values('ALM','Planta baja U1','Almacén');

1 row created.
```

2. MODIFICACIÓN DE DATOS

Si en el apartado anterior nos hemos equivocado al insertar los datos, ahora tendremos que modificarlos para dejarlos correctos. Esto implica actualizar con nuevos datos el registro erróneo y para ello SQL nos suministra la instrucción UPDATE. La sintaxis que utilizaremos será:

```
UPDATE Nombre_tabla
SET columna1=valor1 [, columna2=valor2]...
[WHERE condición]
```

La instrucción indica que queremos actualizar (UPDATE) una tabla (Nombre_tabla) estableciendo (SET) los campos que queremos modificar a una expresión o valor determinado. Podemos colocar tantos campos con sus nuevos valores separados por comas como necesitemos.

Por último, aunque es opcional, debemos incluir la cláusula WHERE condición que nos permitirá seleccionar sobre qué registro se debe realizar la actualización.

> ⚠ CUIDADO: Si no ponemos el WHERE la actualización se realizará en todos los registros de la tabla

Veamos cómo podemos cambiar en nuestra tabla los datos que aún están pendientes de introducir. Para realizar la actualización del registro de Marketing haremos:

```
UPDATE departamentos
SET ubicacion = 'Planta cuarta U5'
WHERE CodDpto = 'MKT';
```

Aunque las condiciones las veremos con muchísima más profundidad en el próximo tema, tenemos que saber que WHERE CodDpto = 'MKT' indica que la actualización se aplicará a los registros cuyo código de departamento sea igual a MKT.

En la condición se puede utilizar cualquiera de los siguientes operadores de comparación:

Operador	Significado
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
=	Igual
!=	Distinto
<>	Distinto

Además se puede utilizar:

Operador	Significado
AND	Devuelve verdadero si las expresiones a su izquierda y derecha son ambas verdaderas.
OR	Devuelve verdadero si cualquiera de las dos expresiones a izquierda y derecha del OR, son verdaderas.
NOT	Invierte la lógica de la expresión que está a su derecha. Si era verdadera, mediante NOT pasa a ser falso.

2.1 MySQL

Probamos con MySQL.

```
mysql> update departamentos
    -> set ubicacion = 'Planta cuarta U5'
    -> where CodDpto = 'MKT';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

Vemos el resultado de la actualización.

```
mysql> select * from departamentos;
+---------+----------------+--------------------+
| CodDpto | Nombre         | Ubicacion          |
+---------+----------------+--------------------+
| ADM     | Administración | Planta quinta U2   |
| ALM     | Planta baja U1 | Almacén            |
| COM     | Comercial      | Planta tercera U3  |
| CONT    | Contabilidad   | Planta quinta U1   |
| INF     | Informática    | Planta sótano U3   |
| MKT     | Marketing      | Planta cuarta U5   |
+---------+----------------+--------------------+
6 rows in set (0.00 sec)
```

Podemos apreciar cómo se ha modificado la ubicación del departamento de Marketing.

Bien, ahora vamos a realizar la modificación del registro del departamento de Almacén cambiando los dos campos en la misma instrucción.

Fíjate en la condición, el registro que queremos modificar es el del Almacén, por ello, la condición será CodDpto = 'ALM'.

```
mysql> UPDATE departamentos
    -> SET nombre = 'Almacén',
    -> ubicacion = 'Planta baja U1'
    -> WHERE CodDpto = 'ALM';
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Comprobamos el resultado.

```
mysql> select * from departamentos;
+---------+----------------+--------------------+
| CodDpto | Nombre         | Ubicacion          |
+---------+----------------+--------------------+
| ADM     | Administración | Planta quinta U2   |
| ALM     | Almacén        | Planta baja U1     |
| COM     | Comercial      | Planta tercera U3  |
| CONT    | Contabilidad   | Planta quinta U1   |
| INF     | Informática    | Planta sótano U3   |
| MKT     | Marketing      | Planta cuarta U5   |
+---------+----------------+--------------------+
6 rows in set (0.00 sec)
```

Podemos observar que el cambio se ha realizado y ahora tenemos todos los datos correctos.

Para usar condiciones compuestas podríamos hacer esto:

```
UPDATE departamentos
SET ubicacion = 'Planta cuarta U5'
WHERE CodDpto = 'MKT' OR CodDpto = 'ALM';
```

2.2 Oracle

Probemos ahora con Oracle.

```
SQL> update departamentos
  2  set ubicacion = 'Planta cuarta U5'
  3  where CodDpto = 'MKT';

1 row updated.
```

Oracle nos muestra un mensaje indicando que una de las filas ha sido actualizada.

Comprobamos el resultado.

```
SQL> select * from departamentos;

CODDPTO    NOMBRE                         UBICACION
---------- ------------------------------ -------------------------
INF        Informática                    Planta sótano U3
ADM        Administración                 Planta quinta U2
COM        Comercial                      Planta tercera U3
CONT       Contabilidad                   Planta quinta U1
MKT        Marketing                      Planta cuarta U5
ALM        Planta baja U1                 Almacén

6 rows selected.
```

Si realizamos el siguiente cambio:

```
SQL> update departamentos
  2  set nombre = 'Almacén',
  3  ubicacion = 'Planta baja U1'
  4  where CodDpto = 'ALM';

1 row updated.
```

Comprobamos el resultado:

```
SQL> select * from departamentos;

CODDPTO    NOMBRE                         UBICACION
---------- ------------------------------ -------------------------
INF        Informática                    Planta sótano U3
ADM        Administración                 Planta quinta U2
COM        Comercial                      Planta tercera U3
CONT       Contabilidad                   Planta quinta U1
MKT        Marketing                      Planta cuarta U5
ALM        Almacén                        Planta baja U1

6 rows selected.
```

3. BORRADO DE DATOS

Para eliminar un registro utilizaremos la instrucción DELETE de la siguiente forma:

```
DELETE FROM nombre_tabla
[WHERE condición]
```

¡Recordad que si no ponemos la condición se eliminarán todos los registros de la tabla!

Imaginemos que queremos eliminar el registro correspondiente al departamento de Marketing, ejecutaríamos la siguiente instrucción:

```
DELETE FROM departamentos
WHERE CodDpto = 'MKT';
```

3.1 MySQL

```
mysql> delete from departamentos
    -> where CodDpto = 'MKT';
Query OK, 1 row affected (0.03 sec)
```

Comprobamos el resultado, el departamento de Marketing ha sido eliminado.

```
mysql> select * from departamentos;
+---------+----------------+--------------------+
| CodDpto | Nombre         | Ubicacion          |
+---------+----------------+--------------------+
| ADM     | Administración | Planta quinta U2   |
| ALM     | Almacén        | Planta baja U1     |
| COM     | Comercial      | Planta tercera U3  |
| CONT    | Contabilidad   | Planta quinta U1   |
| INF     | Informática    | Planta sótano U3   |
+---------+----------------+--------------------+
5 rows in set (0.00 sec)
```

3.2 Oracle

No hay diferencias significativas con MySQL en este punto.