INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

CREACCIÓN DE TABLAS

Una tabla es una colección de datos estructurados en campos.
Cada fila de la tabla, toma un único valor en cada campo.

Para crear una tabla con comandos SQL, debemos tener en cuenta el tipo de datos que van a contener los campos, y el tamaño máximo que van a ocupar.

Un ejemplo de creación de una tabla llamada empleados sería:

```
CREATE TABLE  Empleados(
DNI               VARCHAR(10),
Nombre  VARCHAR(20),
Edad              NUMBER(2,0)
);
```

RESTRICCIONES DE LOS CAMPOS

A los campos de una tabla se le pueden imponer ciertas restricciones de tal forma que los valores válidos en ese campo deban de cumplir ciertas propiedades además del tipo de dato.

- NOT NULL constraint
  - Se asegura de que la columna no acepte valores nulos.
- UNIQUE constraint
  - Se asegura de que el valor en la columna sea único con respecto a los demás valores en la misma columna.
- DEFAULT constraint
  - Asigna una valor por defecto cuando se va a insertar una nueva fila.
- CHECK constraint
  - Valida la data cuando el valor del atributo se entra.

CLAVE PRIMARIA

Toda tabla tiene una clave primaria, que es un campo o conjunto de campos, cuyos valores no se pueden repetir y no pueden tomar valor nulo.

La clave primaria puede estar formada por varios campos, cuya unión cumple la condición anterior.

La clave primaria en una tabla es única, es decir solo puede haber una clave primaria aunque como ya dijimos puede estar formada por más de un campo. En la definición de una tabla la restricción de PRIMARY KEY solo puede aparecer una vez.

Las restricciones en una tabla pueden ser impuestas de dos formas:

A nivel de campo:

```
CREATE TABLE  Empleados(
DNI               VARCHAR(10)     PRIMARY KEY,
Nombre  VARCHAR(20),
Edad              NUMBER(2,0)
);
```

O a nivel de tabla:

```
CREATE TABLE  Empleados(
DNI               VARCHAR(10),
Nombre  VARCHAR(20),
Edad              NUMBER(2,0),
PRIMARY KEY (DNI)
);
```

Cuando la clave primaria está formada por dos o más campos, la restricción solo puede imponerse a nivel de tabla.

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

```
CREATE TABLE  Empleados_Puesto(
DNI                VARCHAR(10),
Cod_puesto         VARCHAR(5),
Hora               TIMESTAMP,
PRIMARY KEY (DNI,CodPuesto)
);
```

Toda restricción tiene un nombre, puesto con el fin de borrar la restricción sin borrar la tabla ni el campo ó campos que afecta. En los ejemplo anteriores como el usuario en el momento de imponer la restricción no especificó un nombre para ella, el propio sistema le impuso dicho nombre, por ejemplo si quisiéramos que en la tabla empleados DNI dejara de ser clave primaria, tendríamos que consultar como ha nombrado el sistema dicha restricción y luego borrarla. Para especificar en el momento de la creación de la tabla un nombre para la restricción de clave primaria sería:

```
CREATE TABLE  Empleados(
DNI                VARCHAR(10)     CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre   VARCHAR(20),
Edad               NUMBER(2,0)
);
```

```
CREATE TABLE  Empleados(
DNI                VARCHAR(10),
Nombre   VARCHAR(20),
Edad               NUMBER(2,0),
CONSTRAINT ClaveEmpl PRIMARY KEY (DNI)
);
```

Índices

Tenemos tres tipos de índices. El primero corresponde a las claves primarias, que como vimos, también se pueden crear en la parte de definición de columnas.

Claves primarias

La sintaxis para definir claves primarias es:

```
   definición_columnas
 | PRIMARY KEY (index_nombre_col,...)
```

El ejemplo anterior que vimos para crear claves primarias, usando esta sintaxis, quedaría así:

```
mysql> CREATE TABLE ciudad4 (nombre CHAR(20) NOT NULL,
   -> poblacion INT NULL DEFAULT 5000,
   -> PRIMARY KEY (nombre));
Query OK, 0 rows affected (0.17 sec)
```

Pero esta forma tiene más opciones, por ejemplo, entre los paréntesis podemos especificar varios nombres de columnas, para construir claves primarias compuestas por varias columnas:

```
mysql> CREATE TABLE mitabla1 (
   -> id1 CHAR(2) NOT NULL,
   -> id2 CHAR(2) NOT NULL,
   -> texto CHAR(30),
   -> PRIMARY KEY (id1, id2));
Query OK, 0 rows affected (0.09 sec)

mysql>
```

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

Índices

El segundo tipo de índice permite definir índices sobre una columna, sobre varias, o sobre partes de columnas. Para definir estos índices se usan indistintamente las opciones KEY o INDEX.

```
mysql> CREATE TABLE mitabla2 (
    -> id INT,
    -> nombre CHAR(19),
    -> INDEX (nombre));
Query OK, 0 rows affected (0.09 sec)
```

O su equivalente:

```
mysql> CREATE TABLE mitabla3 (
    -> id INT,
    -> nombre CHAR(19),
    -> KEY (nombre));
Query OK, 0 rows affected (0.09 sec)
```

También podemos crear un índice sobre parte de una columna:

```
mysql> CREATE TABLE mitabla4 (
    -> id INT,
    -> nombre CHAR(19),
    -> INDEX (nombre(4)));
Query OK, 0 rows affected (0.09 sec)
```

Este ejemplo usará sólo los cuatro primeros caracteres de la columna 'nombre' para crear el índice.

Claves únicas

El tercero permite definir índices con claves únicas, también sobre una columna, sobre varias o sobre partes de columnas. Para definir índices con claves únicas se usa la opción UNIQUE.

La diferencia entre un índice único y uno normal es que en los únicos no se permite la inserción de filas con claves repetidas. La excepción es el valor NULL, que sí se puede repetir.

```
mysql> CREATE TABLE mitabla5 (
    -> id INT,
    -> nombre CHAR(19),
    -> UNIQUE (nombre));
Query OK, 0 rows affected (0.09 sec)
```

Una clave primaria equivale a un índice de clave única, en la que el valor de la clave no puede tomar valores NULL. Tanto los índices normales como los de claves únicas sí pueden tomar valores NULL.

Por lo tanto, las definiciones siguientes son equivalentes:

```
mysql> CREATE TABLE mitabla6 (
    -> id INT,
    -> nombre CHAR(19) NOT NULL,
    -> UNIQUE (nombre));
Query OK, 0 rows affected (0.09 sec)
```

Y:

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

```
mysql> CREATE TABLE mitabla7 (
   -> id INT,
   -> nombre CHAR(19),
   -> PRIMARY KEY (nombre));
Query OK, 0 rows affected (0.09 sec)
```

Los índices sirven para optimizar las consultas y las búsquedas de datos. Mediante su uso es mucho más rápido localizar filas con determinados valores de columnas, o seguir un determinado orden. La alternativa es hacer búsquedas secuenciales, que en tablas grandes requieren mucho tiempo.

Diferencia entre clave primaria y unique

Una clave primaria es el campo (o los campos) usado para identificar una fila. Por ejemplo, el número de identificación fiscal de una persona.

Una simple combinación única de campos (UNIQUE) no tiene nada que ver con la identificación de la columna. Es simplemente una restricción de integridad. Por ejemplo, yo tengo una colección de enlaces. Cada colección se identifica por medio de un número único, que es la clave primaria. Esta clave se usa en relaciones.

Sin embargo, mi aplicación exige que cada colección tenga también un nombre único. ¿Por qué? Para que un ser humano que quiera modificar una colección también sea capaz de identificarla. Es mucho más difícil saber, si se tienen dos colecciones llamadas "Ciencia de la Vida", que la que tiene el número 24433 es la que usted necesita y no la que tiene el número 29882.
De esta forma, el usuario selecciona las colecciones por sus nombres. Por lo tanto nos aseguramos que los nombres sean únicos dentro de la base de datos. Sin embargo ninguna otra tabla en la base de datos se refiere a la tabla de colecciones por su nombre. Eso sería bastante ineficiente.
¡Aún más, a pesar de ser único, el nombre de la colección no define realmente la colección! Por ejemplo, si alguien decidiera cambiar el nombre de la colección de "Ciencia de la Vida" por "Biología", aún seguiría siendo la misma colección, solo que con un nombre diferente. Mientras el nombre sea único no hay problema.

Resumiendo:

- Clave primaria:
  - Usada para identificar la fila y para referirse a ella.
  - Es imposible (o muy difícil) de actualizar.
  - No debe aceptar valores NULL.

  Campos "unique":
  - Se usan como alternativa para acceder una fila.
  - Pueden ser actualizados siempre y cuando mantengan su valor único.
  - Aceptan valores NULL.

Diferencia entre clave primaria e índice

La principal diferencia entre un índice y una clave primaria es la repetición de valores. En una clave primaria no pueden repetirse y en un índice si podrían.

Una clave primaria es, un campo o varios que identifican cada registro de la base de manera unívoca, es decir, sin posibilidad de confusión.

Un índice es un campo o varios por los que se pueden "ordenar" los registros de manera que acceder a uno concreto es más rápido.

Suponiendo una tabla de clientes, se podría hacer una clave primaria con el NIF del cliente ya que es único para cada uno (salvando errores administrativos). De esta manera, sabiendo el NIF sabemos a qué cliente nos referimos sin lugar a dudas. No podríamos hacer clave primaria el nombre, porque se podría repetir, incluso con los apellidos detrás.
En esa misma tabla, sería útil hacer índices en los apellidos o la población ya que es previsible que acabemos buscando clientes según esos campos.

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

Especificar un campo como clave primaria lo único que modifica en la base de datos es que ya no vamos a poder repetir valores en ese campo (o grupo de campos) y que vamos a poder emplearlos como clave externa en otras tablas...
Especificar un índice suele crear un fichero auxiliar de indexado.

Una clave primaria puede estar compuesta por más de un campo. Por ejemplo, si tomamos una tabla de empleos, podemos distribuirla ordenada por departamentos, y dentro de cada departamento por niveles. Hacemos clave primaria ambos campos (departamento-nivel) de manera que se podrá repetir departamento, o nivel, pero no se podrá repetir el conjunto de un departamento y un nivel.

Cuando dicen que la Clave Primaria no se repite y el indice si es una verdad a media; ya que podes crear el indice permitiendo que se repita o no.

Crear indice y que el mismo no se repita es decir que sea único

CREATE UNIQUE INDEX [nombre_indice] ON [nombre_tabla].[nombre_campo]

Crear indice y que el mismo se pueda repitir

CREATE INDEX [nombre_indice] ON [nombre_tabla].[nombre_campo]

Y para borrar un indice

DROP INDEX [nombre_tabla].[nombre_indice]

VALORES REQUERIDOS

Algunos campos, aunque no sean claves primarias, por el propio diseño de la base de datos, se requiere que no acepte valores nulos, es decir que cualquier registro o fila no puede quedar en blanco en dicho campo., aunque si se puede repetir.
La restricción es NOT NULL y se establece únicamente a nivel de campo:
Esta restricción no puede aplicarse al conjunto de dos campos, pero si a varios campos de una tabla.
Esta restricción acepta un nombre como en el caso de la clave primaria, pero no resulta muy útil porque para su modificación no se precisa un nombre de restricción.

```
CREATE TABLE  Empleados(
DNI           VARCHAR (10)   CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR (20)   NOT NULL,
Edad          NUMBER (2,0)
);
```

```
CREATE TABLE  Empleados(
DNI           VARCHAR (10)   CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR (20)   NOT NULL,
Apellidos     VARCHAR (30)   NOT NULL,
Edad          NUMBER (2,0)
);
```

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)    CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR(20)    CONSTRAINT nn_nombre NOT NULL,
Apellidos     VARCHAR(30)    NOT NULL,
Edad          NUMBER(2,0)
);
```

Valores de la tabla válidos

DNI	NOMBRE	APELLIDOS	EDAD

1111111111U	Carlos	González	23
2222222222J	Carlos	González	
3333333333L	Juan	Pérez	

De esta forma el campo Nombre no podría quedar en blanco, al insertar una fila si no ponemos el nombre del empleado no nos deja guardar el resto de campos, pero puede haber dos empleados con el mismo nombre.

VALORES ÚNICOS

Si queremos que un campo que no sea clave primaria, no acepte 2 filas con el mismo valor, pero que sí pueden quedar en blanco, se aplicaría la restricción UNIQUE, puede aplicarse a nivel de tabla, o a nivel de campo.
La restricción puede aplicarse a un conjunto de campos como la clave primaria.

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)   CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR(20)   NOT NULL,
Apellidos     VARCHAR(30)   UNIQUE,
Edad          NUMBER(2,0),
);
```

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)   CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR(20)   NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0),
UNIQUE (Apellidos)
);
```

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)   CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR(20)   NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0),
UNIQUE (Nombre,Apellidos)
);
```

También puede darse un nombre a la restricción, para luego modificarla sin borrar la tabla.

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)   CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR(20)   NOT NULL,
Apellidos     VARCHAR(30)   CONSTRAINT ApellidoUnico UNIQUE,
Edad          NUMBER(2,0),
);
```

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)   CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR(20)   NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0),
CONSTRAINT ClaveApel   UNIQUE (Apellidos)
);
```

Valores de la tabla válidos:

DNI	**NOMBRE**	**APELLIDOS**	**EDAD**
1111111111U	Carlos	González	23
2222222222J	Carlos	García	45

DOCENTE: ELIZABETH CORTEZ RAZO

3333333333L	Juan	Pérez	
4444444444G	Juan		
555555555M	María		

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)    CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR(20)    NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0),
CONSTRAINT RestrUnicaNomApel  UNIQUE (Nombre,Apellidos)
);
```

Valores de la tabla válidos:

DNI	**NOMBRE**	**APELLIDOS**	**EDAD**
1111111111U	Carlos	González	23
2222222222J	Carlos	García	45
3333333333L	Juan	Pérez	
4444444444G	Juan		
555555555M	María		
666666666R	María	González	

Hay que tener en cuenta que no es lo mismo aplicar la restricción a dos campos de forma separada que aplicar la restricción a dos campos de forma conjunta, por ejemplo la sentencia:

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)    CONSTRAINT ClaveEmpl PRIMARY KEY,
Nombre        VARCHAR(20)    NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0),
UNIQUE (Nombre),
UNIQUE (Apellidos)
);
```

No adptaría las filas de nombres, ni de apellidos repetidos, es decir la tabla anterior , solo tendría de valores válidos:

DNI	**NOMBRE**	**APELLIDOS**	**EDAD**
1111111111U	Carlos	González	23
3333333333L	Juan	Pérez	
555555555M	María		

VALORES POR DEFECTO

Cuando una fila en un campo es normal que tenga siempre el mismo valor, conviene darle dicho valor por defecto. Esto quiere decir que al insertar una nueva fila, si no indicamos lo contrario el valor de la fila en ese campo será el predeterminado.

Por ejemplo:

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)    PRIMARY KEY,
Nombre        VARCHAR(20)    NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0),
Cargo         VARCHAR(20)  DEFAULT  'Administrativo'
);
```

Esto quiere decir que en todas las filas que insertemos se nos propondrá de forma automática el cargo de administrativo.

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)    PRIMARY KEY,
Nombre        VARCHAR(20)    NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0),
FehaAlta      DATE DEFAULT SYSDATE
);
```

La función SYSDATE asigna la fecha actual del sistema operativo.

CHEQUEO DE VALORES

Los valores en algunos campos, además del tipos de dato, deben estar limitados por alguna otra norma, por ejemplo, la edad es un número, pero como debe estar en edad laboral, deberíamos comprobar que la edad está entre los 18 y los 70 y no admitir ningún otro valor. Esto se conseguiría con:

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)    PRIMARY KEY,
Nombre        VARCHAR(20)    NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0) CHECK (Edad BETWEEN 18 AND 70),
FechaAlta     DATE DEFAULT SYSDATE
);
```

También puede establecerse a nivel de tabla:

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)    PRIMARY KEY,
Nombre        VARCHAR(20)    NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0) ,
FechaAlta     DATE DEFAULT SYSDATE,
CHECK (Edad BETWEEN 18 AND 70)
);
```

También puede darse un nombre a la restricción:

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)    PRIMARY KEY,
Nombre        VARCHAR(20)    NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0) CONSTRAINT  ck_edad CHECK (Edad BETWEEN 18 AND 70),
FechaAlta     DATE DEFAULT SYSDATE
);
```

```
CREATE TABLE  Empleados(
DNI           VARCHAR(10)    PRIMARY KEY,
Nombre        VARCHAR(20)    NOT NULL,
Apellidos     VARCHAR(30),
Edad          NUMBER(2,0) ,
FechaAlta     DATE DEFAULT SYSDATE,
CONSTRAINT ck_edad CHECK (Edad BETWEEN 18 AND 70)
);
```

CLAVES AJENAS

La restricción de clave ajena se establece sobre algún campo que necesariamente es clave primaria en otra tabla, ambos campos deben contener los mismos valores, es decir deben ser del mismo tipo aunque pueden llamarse de forma distinta.
La clave ajena señala una relación con la tabla donde dicho campo es clave primaria.

DOCENTE: ELIZABETH CORTEZ RAZO

Además los valores de la tabla donde es clave ajena, deben existir previamente en la tabla donde es clave primaria. Esta condición se denomina **integridad referencial.**

Por ejemplo creamos un tabla DEPARTAMENTOS:

```
CREATE TABLE Departamentos(
Codigo_dep              VARCHAR(3) PRIMARY KEY,
Nombre_dep              VARCHAR(10)
);
```

En ella introducimos los siguientes valores:

Codigo_dep	**Nombre_dep**
ADM	Administración
FOR	Formación
DIS	Diseño

Ahora la tabla empleados, con un campo que representa el código del departamento en el que trabaja y exigimos integridad referencial códigos que se recogen como valores en la tabla anterior.

```
CREATE TABLE  Empleados(
DNI             VARCHAR(10)     PRIMARY KEY,
Nombre          VARCHAR(20)     NOT NULL,
Apellidos       VARCHAR(30),
Departamento    VARCHAR(3) ,
CONSTRAINT fk_dep_emp FOREIGN KEY (Departamento)  REFERENCES Departamentos
);
```

Esto quiere decir que los valores de las filas en el campo departamento, deben ser exclusivamente los valores contenidos en el codigo_dep (clave primaria) de la tabla Departamentos.

Es decir son válidos:

DNI	**NOMBRE**	**APELLIDOS**	**DEPARTAMENTO**
1111111111U	Carlos	González	ADM
3333333333L	Juan	Pérez	FOR
555555555M 444444444Y	María Pilar	Lópes	FOR

Es decir la clave ajena puede tomar valor nulo, pero no puede asignarse ningún empleado a un departamento que no exista.

La clave ajena siempre se establece a nivel de tabla, y conviene darle un nombre a la restricción aunque no es obligatorio, es decir sería váida:

```
CREATE TABLE  Empleados(
DNI             VARCHAR(10)     PRIMARY KEY,
Nombre          VARCHAR(20)     NOT NULL,
Apellidos       VARCHAR(30),
Departamento    VARCHAR(3) ,
FOREIGN KEY (Departamento)  REFERENCES Departamentos
);
```

El campo clave ajena debe ser del mismo tipo que su referencia en la clave primaria, aunque como observamos en el ejemplo anterior, no tienen porque llamarse igual, aunque sí que es posible.

```
CREATE TABLE  Empleados(
DNI             VARCHAR(10)     PRIMARY KEY,
Nombre          VARCHAR(20)     NOT NULL,
Apellidos       VARCHAR(30),
Codigo_dep      VARCHAR(3) ,
FOREIGN KEY (Codigo_dep)  REFERENCES Departamentos
);
```

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

Un campo puede hacer referencia a una clave primaria formada por más de un campo, por ejemplo:

```
CREATE TABLA Viviendas (
Calle            VARCHAR(29),
Numero           NUMBER(4,0),
Piso_letra       VARCHAR(8),
PRIMARY KEY (Calle, Numero, Piso_letra)
);
CREATE TABLE Ciudadanos(
DNI              VARCHAR(10)    PRIMARY KEY,
Nombre           VARCHAR(20)    NOT NULL,
Apellidos        VARCHAR(30),
Calle            VARCHAR(29),
Numero           NUMBER(4,0),
Piso_letra       VARCHAR(8),
CONSTRAINT ciu_viv FOREIGN KEY (Calle, Numero, Piso_letra) REFERENCES Viviendas
);
```

También puede darse el caso de un campo que sea clave primaria o parte de una clave primaria y clave ajena al mismo tiempo:

Supongamos que en una empresa un empleado puede ser de dos departamentos a la vez:

```
CREATE TABLE Departamentos(
Codigo_dep              VARCHAR(3) PRIMARY KEY,
Nombre_dep              VARCHAR(10)
);

CREATE TABLE  Empleados(
DNI              VARCHAR(10)    PRIMARY KEY,
Nombre           VARCHAR(20)    NOT NULL,
Apellidos        VARCHAR(30),
);
```

No podemos insertar el campo codigo_dep en empleados pues cada empleado puede trabajar en más de un departamento, creamos una nueva tabla que recoja esa información:

```
CREATE TABLE Departamentos_Empleados (
Codigo_dep              VARCHAR(3) ,
DNI                     VARCHAR(10),
Función                 VARCHAR(20),
PRIMARY KEY(Codigo_dep, DNI),
CONSTRAINT ca1 FOREIGN KEY (Codigo_dep) REFERENCES Departamentos,
CONSTRAINT ca2 FOREIGN KEY (DNI) REFERENCES Empleados
);
```

Otro ejemplo, supongamos que añadimos una tabla con los familiares del empleado, cuya clave primaria sea el DNI del familiar y un código interno que es:
C→Conyuge
H1→1ºHijo
H2→2ºHijo
Etc..

La tabla sería:

```
CREATE TABLE  Familiares(
DNI_empleado            VARCHAR(10),
Codigo_parentesco       VARCHAR(4),
Nombre                  VARCHAR(20)    NOT NULL,
Apellidos               VARCHAR(30),
PRIMARY KEY (DNI_empleado, Codigo_parentesco),
CONSTRAINT fam_empl FOREIGN KEY (DNI_empleado) REFERENCES Empleados
```

);

Claves foráneas e integridad referencial

- Referencia de integridad – Son *constraints* que aseguran que los valores del FK de una tabla deben parear los valores del PK de una tabla en una relación 1:M.
- Restringe:
  - Eliminación de records primarios
  - Actualización de records primarios
  - Insertar records dependientes

Es imprescindible que la columna que contiene una definición de clave foránea esté indexada (2). Pero esto no debe preocuparnos demasiado, ya que si no lo hacemos de forma explícita, MySQL lo hará por nosotros de forma implícita.

```
CREATE TABLE cliente
(
   id_cliente INT NOT NULL,
   nombre VARCHAR(30),
   PRIMARY KEY (id_cliente)
) TYPE = INNODB;

CREATE TABLE venta
(
   id_factura INT NOT NULL,
   id_cliente INT NOT NULL,
   cantidad   INT,
   PRIMARY KEY(id_factura),
   INDEX (id_cliente),
   FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente)
) TYPE = INNODB;
```

La sintaxis completa de una restricción de clave foránea es la siguiente:

```
[CONSTRAINT símbolo] FOREIGN KEY (nombre_columna, ...)
            REFERENCES nombre_tabla (nombre_columna, ...)
            [ON DELETE {CASCADE | SET NULL | NO ACTION
                  | RESTRICT}]
            [ON UPDATE {CASCADE | SET NULL | NO ACTION
                  | RESTRICT}]
```

Las columnas correspondientes en la clave foránea y en la clave referenciada deben tener tipos de datos similares para que puedan ser comparadas sin la necesidad de hacer una conversión de tipos. El tamaño y el signo de los tipos enteros debe ser el mismo. En las columnas de tipo carácter, el tamaño no tiene que ser el mismo necesariamente.

Existen cinco opciones diferentes. Veamos lo que hace cada una de ellas:

- RESTRICT: esta opción impide eliminar o modificar filas en la tabla referenciada si existen filas con el mismo valor de clave foránea.
- CASCADE: borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado de clave, implica borrar las filas con el mismo valor de clave foránea o modificar los valores de esas claves foráneas.
- SET NULL: borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado de clave, implica asignar el valor NULL a las claves foráneas con el mismo valor.
- NO ACTION: las claves foráneas no se modifican, ni se eliminan filas en la tabla que las contiene.

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

Por ejemplo, veamos esta definición de la tabla 'telefonos':

```
mysql> CREATE TABLE personas3 (
   -> id INT AUTO_INCREMENT PRIMARY KEY,
   -> nombre VARCHAR(40),
   -> fecha DATE)
   -> ENGINE=InnoDB;

mysql> CREATE TABLE telefonos3 (
   -> numero CHAR(12),
   -> id INT NOT NULL,
   -> KEY (id),
   -> FOREIGN KEY (id) REFERENCES personas3 (id)
   -> ON DELETE RESTRICT ON UPDATE CASCADE)
   -> ENGINE=InnoDB;
```

- Si se intenta borrar una fila de 'personas3' con un determinado valor de 'id', se producirá un error si existen filas en la tabla 'telefonos3' con mismo valor en la columna 'id'. La fila de 'personas3' no se eliminará, a no ser que previamente eliminemos las filas con el mismo valor de clave foránea en 'teléfonos3'.
- Si se modifica el valor de la columna 'id' en la tabla 'personas3', se modificarán los valores de la columna 'id' para mantener la relación.
-

Veamos un ejemplo más práctico:

personas3

id	nombre	fecha
1	Fulanito	1998/04/14
2	Menganito	1975/06/18
3	Tulanito	1984/07/05

telefonos3

numero	id
12322132	1
12332221	1
55546545	3
55565445	3

Si intentamos borrar la fila correspondiente a "Fulanito" se producirá un error, ya que existen dos filas en 'telefonos' con el valor 1 en la columna 'id'.
Sí será posible borrar la fila correspondiente a "Menganito", ya que no existe ninguna fila en la tabla 'telefonos3' con el valor 2 en la columna 'id'.
Si modificamos el valor de 'id' en la fila correspondiente a "Tulanito", por el valor 4, por ejemplo, se asignará el valor 4 a la columna 'id' de las filas 3ª y 4ª de la tabla 'telefonos3':

personas3

id	nombre	fecha
1	Fulanito	1998/04/14
2	Menganito	1975/06/18
4	Tulanito	1984/07/05

telefonos3

numero	id

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

12322132 1
12332221 1
55546545 4
55565445 4

- Las operaciones que nos ofrece DDL para trabajar con la estructura de la base de datos son:
  - CREATE
  - USE
  - ALTER
  - DROP
  - SHOW
  - TRUNCATE
  - RENAME.

Podemos ver las bases de datos en nuestro servidor con la orden:
  - SHOW DATABASES

Borrado de una base de datos

- La orden para borrar una base de datos es:
  - DROP DATABASE nombre_bdd

- La orden DROP borra de forma irreversible una base de datos, por tanto debemos usarla cuidadosamente.

Al borrar la base de datos es obvio que perderemos todos los datos que se guardaban en ella.

- Ejemplo:

```
CREATE TABLE Alumno (
        Matricula INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
        dni         varchar(9) NOT NULL UNIQUE,
        nombre    varchar (50),
        apellido   varchar (50),
        edad        INT UNSIGNED,
        nota_media int DEFAULT 5,
        tutor       INT NOT NULL,
        CONSTRAINT FK_Alumno_Tutor FOREIGN KEY (tutor) REFERENCES Tutor(codigo));
```

Después de haber creado una tabla podemos querer verla, revisar su estructura o borrarla.
Para ver las tablas de una base de datos:
  - SHOW TABLES

Para ver la estructura de una tabla:
  - SHOW COLUMNS FROM nombre_tabla

Para borrar una tabla:
  - DROP TABLE nombre_tabla

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

Modificación de tablas.

Después de haber creado una tabla, podremos cambiarle el nombre, agregar, eliminar o renombrar una columna, o cambiar el tipo de datos y cualquier otro atributo de la columna.

- La estructura básica de la orden para modificar una tabla es:
  - ALTER TABLE nombre_tabla

- Añadir una columna a una tabla:
  - ALTER TABLE Alumno ADD campo INT

- Añadir una descripción a una columna:
  - ALTER TABLE Alumno ALTER campo SET DEFAULT 5

- Eliminar el valor predeterminado de una columna:
  - ALTER TABLE Alumno ALTER campo DROP DEFAULT

- Cambiar el nombre y la definición de una columna:
  - ALTER TABLE Alumno CHANGE campo campo2 float(8,2)

- Modificar la definición de una columna conservando su nombre:
  - ALTER TABLE Alumno MODIFY campo2 float(10,2)

- Eliminar una columna de una tabla:
  - ALTER TABLE Alumno DROP campo2

- Modificar el nombre de una tabla:
  - ALTER TABLE Alumno RENAME Estudiante

- Añadir una restricción a una tabla:
  - ALTER TABLE Alumno ADD CONSTRAINT FK_Alumno_Delegado FOREIGN KEY (cod_delegado) REFERENCES Alumno(codigo_alumno);

- Eliminar una restricción de una tabla:
  - ALTER TABLE Alumno DROP FOREIGN KEY FK_Alumno_Delegado;

- Eliminar llaves primarias de una tabla:
  - ALTER TABLE Alumno DROP PRIMARY KEY;

Sentencia RENAME

Hemos visto que con la sentencia ALTER podemos modificar el nombre de una tabla de nuestra base de datos.

MySQL proporciona además la sentencia RENAME para el mismo propósito. Su sintaxis es la siguiente:

- RENAME nombre_actual TO nombre_nuevo;

Podemos usar la sentencia RENAME para renombrar varias tablas:

RENAME table_1 TO table_1.0, table_2 TO table_2.0;

Sentencia TRUNCATE

La sentencia TRUNCATE elimina todos los datos de una tabla. La sintaxis es la siguiente:

- TRUNCATE TABLE nombre_tabla;

INSTITUTO TECNOLOGICO DE CIIUDAD MADERO
TALLER DE BASE DE DATOS
UNIDAD 1.
ESQUEMA DE BASE DE DATOS (DLL)

DOCENTE: ELIZABETH CORTEZ RAZO

La sentencia TRUNCATE puede parecer equivalente a realizar un DELETE de todos los datos de una tabla, pero hay diferencias:

- TRUNCATE es más rápido que DELETE. Se destruye la tabla y se crea de nuevo.
- El campo AUTO_INCREMENT se reinicia (la próxima fila que se añada a la tabla tomará el primer valor del campo AUTO_INCREMENT).
- TRUNCATE es una operación atómica (no transaccional).