Tema 3. Creación de tablas

Índice

En esta unidad vamos a estudiar cómo convertimos un diagrama de entidad-relación en un modelo relacional (tablas). También vamos a estudiar SQL.

1.- Mapeo de diagramas Entidad-Relación

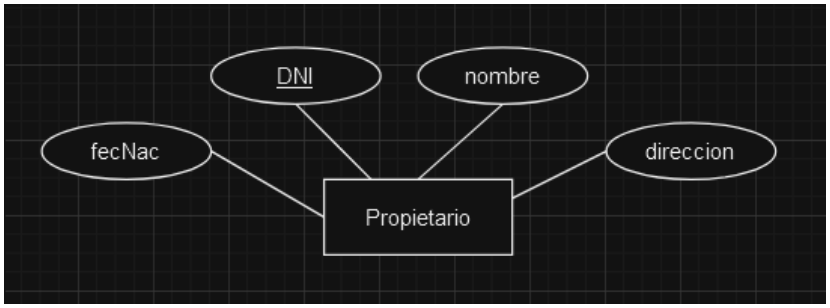
Veamos cómo mapear cada componente del modelo ER:

1.1.- Mapeo de entidades y atributos

Entidades regulares (fuertes) y entidades débiles.

- Cree una tabla para cada entidad.
- Los atributos de la entidad deben convertirse en campos de tablas con sus respectivos tipos de datos.
- La clave principal de la tabla es el atributo clave.

Ejemplo:

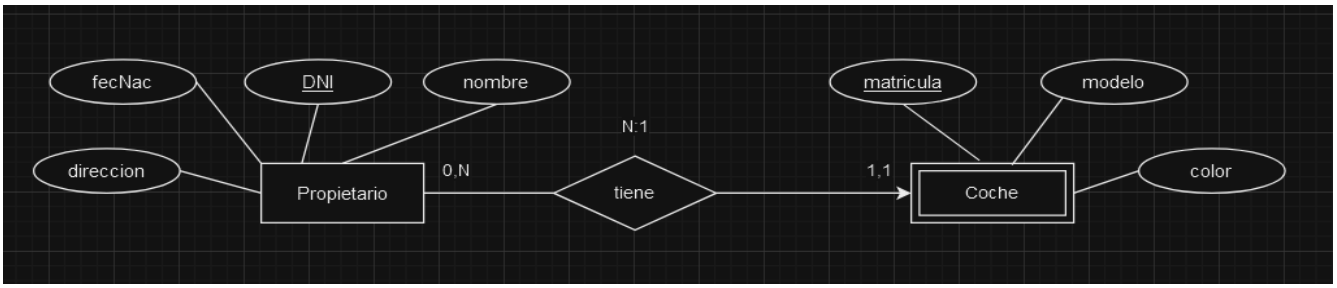



```
Propietario (DNI, nombre, dirección, fecNac)
    PK (DNI)
```

Entidad débil identificada

- Crear tabla para cada entidad
- Los atributos de la entidad deben convertirse en campos de tablas con sus respectivos tipos de datos.
- La clave principal de la tabla está compuesta por el atributo clave de la entidad débil y el atributo clave de la entidad fuerte es una clave externa.

Ejemplo:

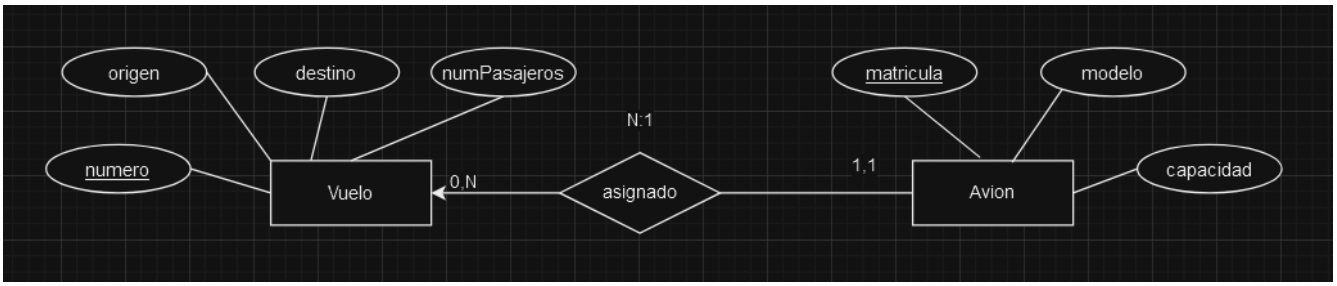



```
Coche (matricula, modelo, color, DNI)
    PK (matricula)
    FK (DNI / Propietario)
```

1.2.- Mapeo de relaciones 1:N

La clave principal de la entidad en el extremo “uno” se convierte en una clave externa en la entidad en el extremo “varios”. Observe que la flecha indica dónde debe colocar la clave externa.

Ejemplo:

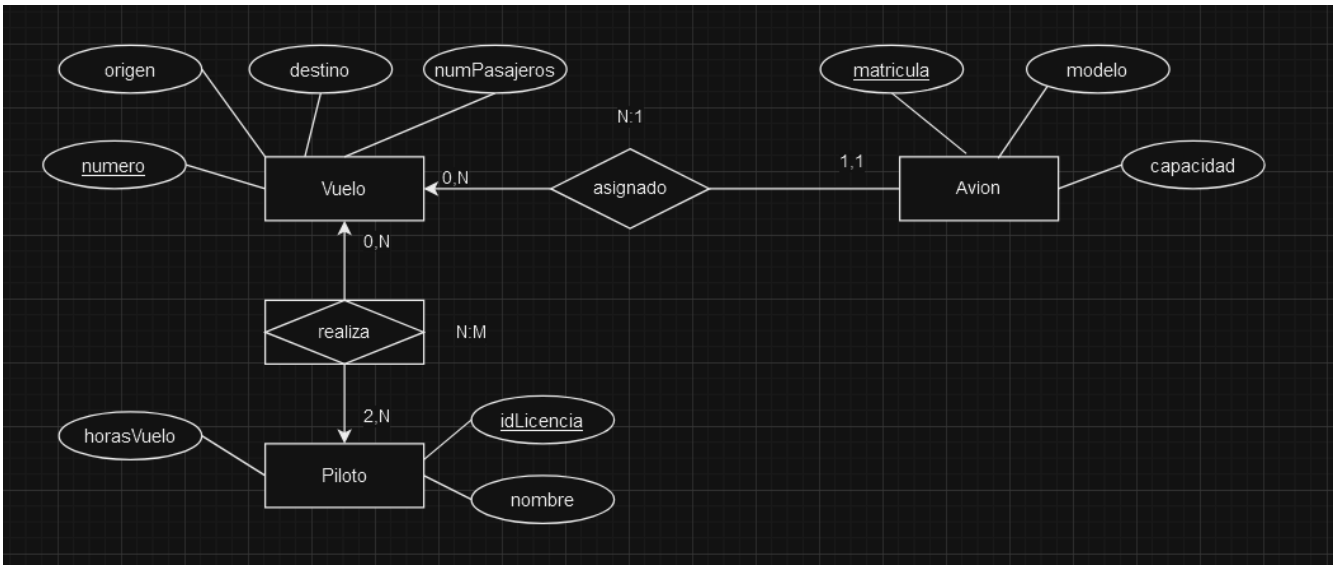



```
Avion (matricula, modelo, capacidad)
    PK (matricula)
Vuelo (numero, origen, destino, numPasajeros, matricula)
    PK (numero)
    FK (matricula / Avion)
```

1.3.- Mapeo de relaciones N:M

Para ello, tenemos que crear una nueva tabla. La clave principal de esta tabla es la combinación de los atributos clave de ambas entidades que participan en la relación. Si hay algunos atributos en la relación, también aparecerán en la nueva tabla. Esta nueva tabla tiene dos claves externas a las tablas relacionadas.

Ejemplo:




```
Avion (matricula, modelo, capacidad)
    PK (matricula)
Vuelo (numero, origen, destino, numPasajeros, matricula)
    PK (numero)
    FK (matricula / Avion)
Piloto (idLicencia, nombre, horasVuelo)
    PK (idLicencia)
Realiza (numeroVuelo, idPiloto)
    PK (numeroVuelo, idPiloto)
    FK1 (numeroVuelo / Vuelo)
    FK2 (idPiloto / Piloto)
```

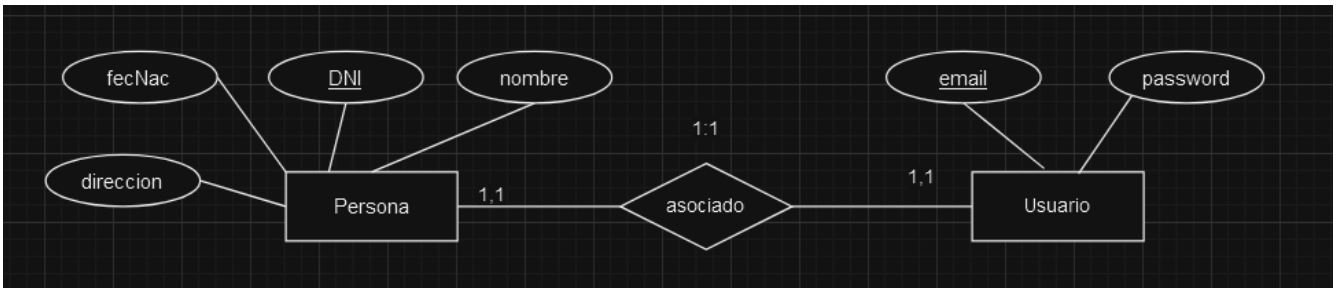
1.4.- Mapeo de relaciones 1:1

Existen tres opciones posibles, dependiendo de la cardinalidad mínima:

- Agregar la clave primaria de A como clave externa en B
- Agregar la clave primaria de B como clave externa en A
- Ambas

Debes poner la clave externa en la tabla (o entidad) que participa con 1 en la cardinalidad mínima, para evitar valores nulos en la tabla.

Ejemplo:



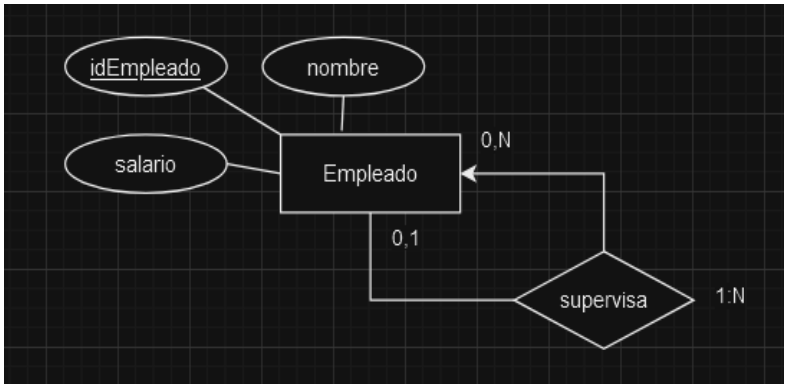

```
Persona (DNI, nombre, direccion, fecNac)
    PK (DNI)
Usuario (email, password, DNI)
    PK (email)
    FK (DNI / Persona)
```

1.5.- Mapeo de relaciones reflexivas

El proceso es el mismo:

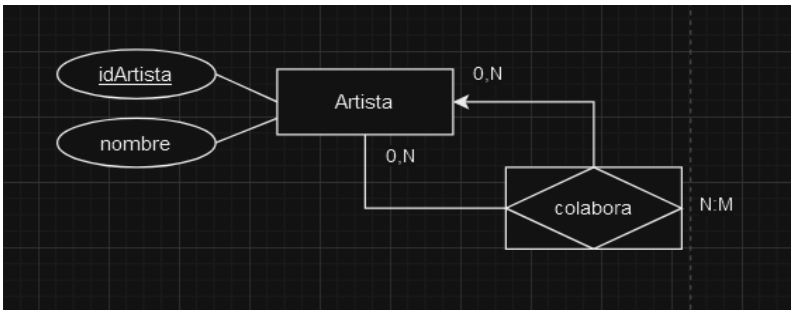
- Si la relación reflexiva es de muchos a muchos, crea una nueva tabla. En esta tabla, la clave principal debe estar compuesta por dos atributos, que es el atributo ID dos veces.
- Si la relación reflexiva es de uno a muchos o de uno a uno, agrega una clave externa en la tabla que haga referencia a sí misma.

Ejemplo de relación reflexiva 1:N




```
Empleado (idEmpleado, nombre, salario, supervisor)
PK (idEmpleado)
FK (supervisor / Empleado)
```

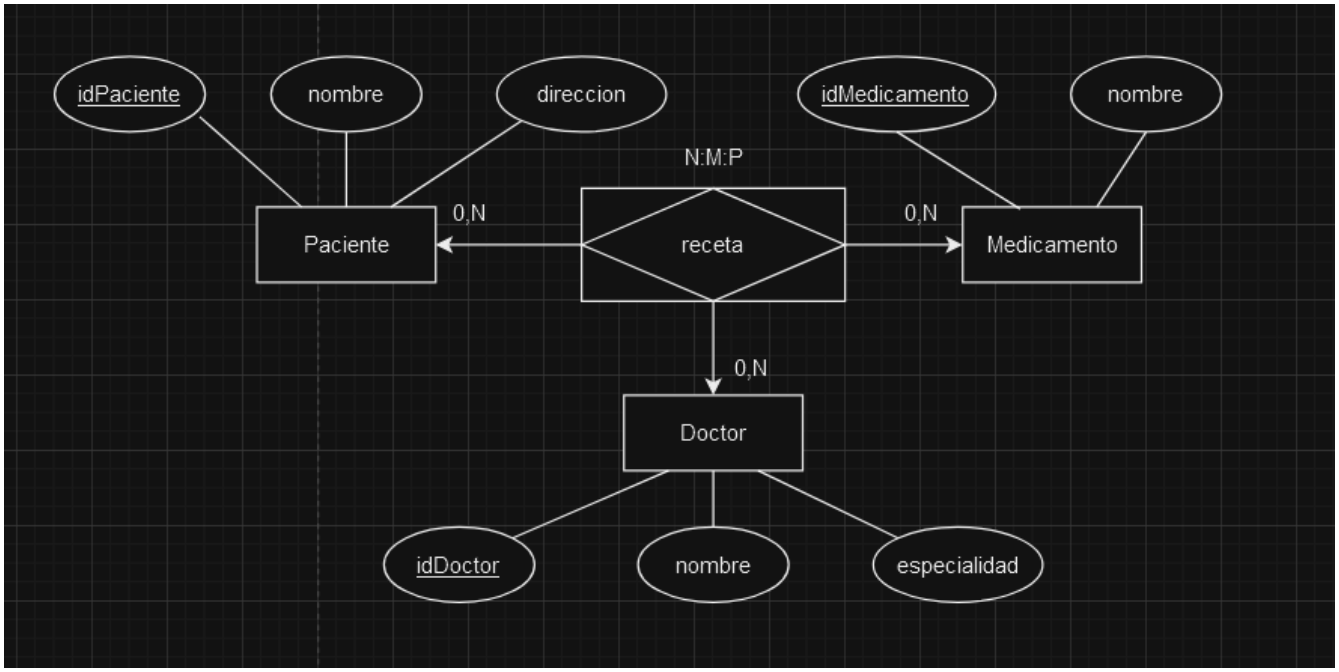
Ejemplo de relación reflexiva N:M




```
Artista (idArtista, nombre)
    PK (idArtista)
Colabora (idArtista1, idArtista2)
    PK (idArtista1, idArtista2)
    FK1 (idArtista1 / Artista)
    FK2 (idArtista2 / Artista)
```

1.5.- Mapeo de relaciones ternarias

El proceso es similar a una relación de muchos a muchos. Debe estudiar especialmente cada caso para determinar qué atributos son componentes de la clave principal. En este caso, es necesario que todos los atributos de la clave principal de la tabla Receta.



```
Paciente (idPaciente, nombre, direccion)
    PK (idPaciente)
Medicamento (idMedicamento, nombre)
    PK (idMedicamento)
Doctor (idDoctor, nombre, especialidad)
    PK (idDoctor)
Receta (idPaciente, idMedicamento, idDoctor)
    PK (idPaciente, idMedicamento, idDoctor)
    FK1 (idPaciente / Paciente)
    FK2 (idMedicamento / Medicamento)
    FK3 (idDoctor / Doctor)
```

1.6.- Mapeo de jerarquías (superclases y subclases)

Existen tres posibilidades:

1. Crear una tabla solo para la superclase. Incluirá también los atributos de las subclases. Es necesario tener un atributo con el tipo.
2. Crear una tabla para cada subclase. Cada tabla incluirá los atributos de la superclase.
3. Crear una tabla para la superclase y una tabla para cada subclase. En este caso, la clave principal de las subclases también será una clave externa a la superclase.

Opción 1

Es una opción posible cuando:

- Las subclases tienen pocos atributos
- No existen relaciones en las que participen las subclases.

Opción 2

Es una opción posible cuando:

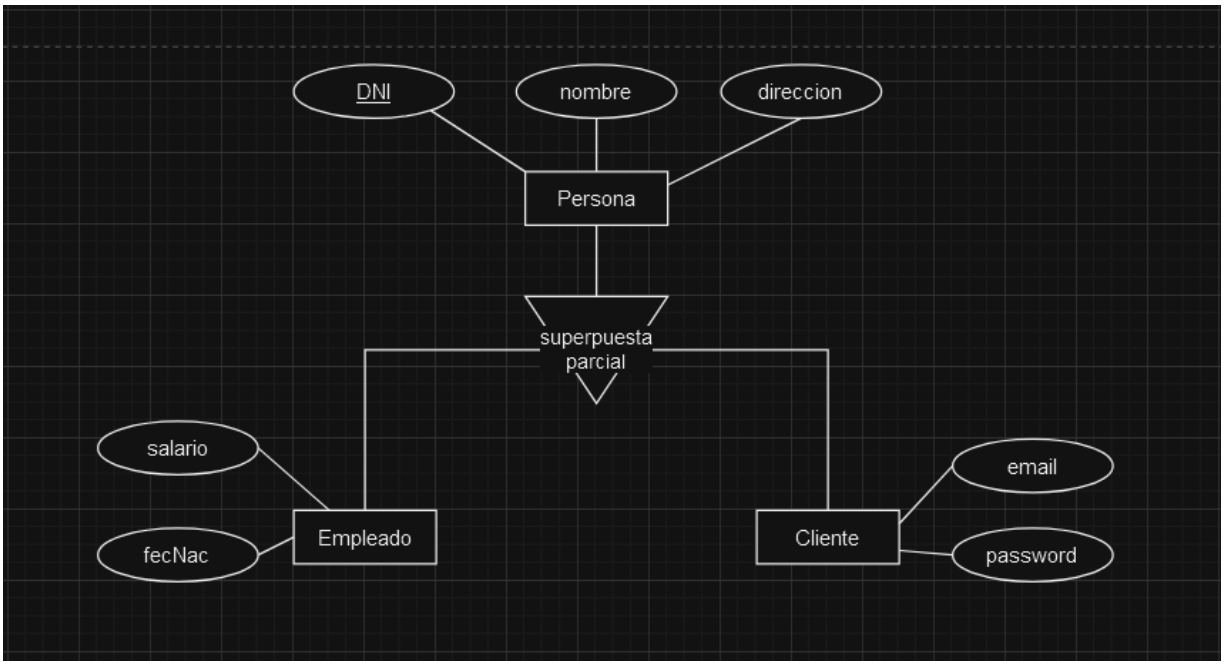
- La relación no es parcial.
- No existen relaciones en las que participe la superclase.
- Los atributos de las subclases son muy diferentes.

Opción 3

Es una opción posible cuando:

- Existen relaciones en las que intervienen subclases y superclases.
- Los atributos de las subclases son muy diferentes.

Ten en cuenta que en todas estas opciones, podría ser necesario incluir restricciones en las tablas.




```
Persona (DNI, nombre, direccion)
    PK (DNI)
Empleado (salario, fecNac, DNI)
    PK (DNI)
    FK (DNI / Persona)
Cliente (email, password, DNI)
    PK (DNI)
    FK (DNI / Persona)
```

2.- SQL. Lenguaje de definición de datos (DDL)

SQL significa (Structured Query Language) lenguaje de consulta estructurado. Según ANSI (Instituto Nacional de Estándares de Estados Unidos), es el lenguaje estándar para los Data Base Manage System (Sistemas gestores de bases de datos SGBD) relacionales. Todos los DBMS relacionales utilizan SQL, pero la mayoría de ellos también tienen sus propias extensiones propietarias adicionales que normalmente solo se utilizan en su sistema.

Las sentencias SQL se dividen en dos categorías: Lenguaje de definición de datos (Data Definition Language, DDL) y Lenguaje de manipulación de datos (Data Manipulation Language, DML).

Las sentencias DDL se invocan cuando se necesita manipular la estructura de la base de datos, mientras que las sentencias DML se utilizan para manipular los datos reales contenidos en una base de datos.

Puede utilizar sentencias DDL para definir, modificar y eliminar todo tipo de objeto que pueda existir en una base de datos. Por este motivo, es de vital importancia que comprenda las sentencias DDL y sepa cómo utilizarlas correctamente. En esta unidad, vamos a estudiar las sentencias DDL más utilizadas:

- `CREATE y DROP DATABASE`
- `CREATE TABLE`
- `DROP TABLE`
- `ALTER TABLE`

2.- Crear y eliminar bases de datos

La creación de la base de datos la realiza normalmente el Administrador de la base de datos (Data Base Administrator, DBA). Cada base de datos tiene un identificador (un sustantivo). En un DBMS en particular, existen varias opciones de configuración de las que es responsable el DBA.

La sentencia SQL para crear una base de datos es:

```
CREATE DATABASE databaseName;
```

La sentencia SQL para eliminar (eliminar) una base de datos es:

```
DROP DATABASE databaseName;
```

2.2.- Crear y eliminar tablas

Para crear una tabla en SQL, debe utilizar la instrucción **CREATE TABLE** para definir las columnas de la tabla y configurar las restricciones adecuadas en la tabla. Contiene numerosos componentes y proporciona muchas opciones para definir la naturaleza exacta de una tabla en particular.

La siguiente sintaxis representa los elementos que componen una declaración CREATE TABLE:

```
CREATE TABLE [IF NOT EXISTS] <nombre tabla>
( <elemento tabla> [{, <elemento tabla>}...] )
[<opcion tabla> [<opcion tabla>...]]

<elemento tabla>::=
<definicion columna>
| {[CONSTRAINT <nombre restriccion>] PRIMARY KEY
(<nombre columna> [{, <nombre columna>}...])}
| {[CONSTRAINT <nombre restriccion>] FOREIGN KEY [<nombre indice>]
(<nombre columna> [{, <nombre columna>}...]) <definicion referencia>}
| {[CONSTRAINT <nombre restriccion>] UNIQUE [INDEX] [<nombre indice>]
(<nombre columna> [{, <nombre columna>}...])}
| [CONSTRAINT [simbolo]] CHECK (expresion)
| {{INDEX | KEY} [<index name>] (<nombre columna> [{, <nombre columna>}...])}
```

```
<definicion columna>::=
<nombre columna> <tipo> [NOT NULL | NULL] [DEFAULT <valor>] [AUTO_INCREMENT]
[PRIMARY KEY] [COMMENT '<cadena>'] [<definicion referencia>]

<definicion referencia>::=
REFERENCES <nombre tabla> [(<nombre columna> [{, <nombre columna>}...])]
[ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT }]
[ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT }]
```

Puedes ver la sintaxis completa de esta frase en este enlace CREATE TABLE

Vamos a estudiar esta oración con este ejemplo:

```sql
CREATE TABLE
    IF NOT EXISTS Departamento (
        depID INT,
        nombre VARCHAR(20) UNIQUE NOT NULL,
        PRIMARY KEY (depID)
    );

CREATE TABLE
    IF NOT EXISTS Tareas (
        tareaID INT AUTO_INCREMENT,
        titulo VARCHAR(20) NOT NULL,
        fechaIni DATE,
        fechaFin DATE,
        prioridad TINYINT NOT NULL DEFAULT 1,
        fechaCreacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        depID INT NOT NULL,
        PRIMARY KEY (tareaID),
        FOREIGN KEY (depID) REFERENCES Departamento (depID)
    );
```

Podemos eliminar una tabla con la sentencia **DROP TABLE**:

```
DROP TABLE nombreTabla;
```

Puedes ver la sintaxis completa de esta oración en este enlace Drop table.

2.3.- Tipos de datos

Principales tipos de datos	
Números enteros (sin decimales)	TINYINT, SMALLINT, MEDIUMINT, INTEGER, BIGINT
Números decimales	FLOAT, DOUBLE, DECIMAL (length, decimals)
Fechas	DATE (YYYY-MM-DD), TIME (HH:MM:SS), DATETIME, YEAR, TIMESTAMP
Cadenas de texto	CHAR(length), VARCHAR(length), ENUM

Tipos de datos numéricos

Los tipos de datos numéricos difieren en el almacenamiento requerido y el rango de valores que tienen. Puede ver estos rangos para tipos enteros en este enlace Rangos para tipos enteros.

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

Autoincrementos

Una columna de números enteros puede tener el atributo adicional AUTO_INCREMENT. Normalmente, es valor+1, donde valor es el valor más grande de la columna que se encuentra actualmente en la tabla. Las secuencias AUTO_INCREMENT comienzan con 1.

Timestamp

El tipo TIMESTAMP es ligeramente diferente de los demás tipos de datos. Cuando se configura una columna con este tipo de datos, o cuando se inserta o actualiza una fila, se proporciona automáticamente un valor para la columna TIMESTAMP que se basa en la fecha y hora actuales. Esto proporciona una forma práctica de registrar cada transacción que ocurre en una tabla en particular.

Char y Varchar

La diferencia entre CHAR y VARCHAR es que CHAR almacena solo tipos de datos de cadenas individuales de longitud fija, mientras que VARCHAR almacena caracteres variables de diferentes cadenas y la longitud depende de la cadena.

Enumeraciones

Una ENUM es una cadena con un valor elegido de una lista de valores permitidos que se enumeran explícitamente en la especificación de la columna en el momento de la creación de la tabla.

```
CREATE TABLE
    Camiseta (
        camisetaID INT AUTO_INCREMENT,
        nombre VARCHAR(40),
        talla ENUM ('x-small', 'small', 'medium', 'large', 'x-large'),
        PRIMARY KEY (camisetaID)
    );
```

Puede encontrar más información sobre los tipos de datos de MySQL en Tipos de datos.

2.4.- Opciones de claves ajenas

En la cláusula de clave ajena, existen estas opciones:

- **CASCADE**: elimina o actualiza la fila de la tabla principal y elimina o actualiza automáticamente las filas coincidentes en la tabla secundaria. Por lo tanto, si especifica esta opción, cuando elimina una fila en la tabla principal, el DBMS también elimina cualquier fila asociada con esa fila (claves externas) en una tabla secundaria. Observe que esta es la opción que debe aparecer en la clave ajena de una entidad débil que la relaciona con la entidad fuerte.
- **SET NULL**: elimina o actualiza la fila de la tabla principal y establece la columna o columnas de clave ajena en la tabla secundaria en NULL. Si especifica una acción SET NULL, asegúrese de no haber declarado las columnas en la tabla secundaria como NOT NULL.
- **RESTRICT**: rechaza la operación de eliminación o actualización para la tabla principal. Especificar RESTRICT (o NO ACTION) es lo mismo que omitir la cláusula ON DELETE o ON UPDATE.
- **SET DEFAULT**: MySQL no reconoce esta acción. Tiene estas opciones para operaciones de eliminación (ON DELETE CASCADE) y para operaciones de actualización (ON UPDATE CASCADE).

Tiene estas opciones para operaciones de eliminación (ON DELETE CASCADE) y para operaciones de actualización (ON UPDATE CASCADE).

Ejemplo con opción CASCADE:

En este caso, cuando eliminas un país, las provincias de ese país se eliminan automáticamente. Si actualizas el provinciaID en la tabla Pais, se actualizará automáticamente en la tabla Provincia.

```sql
CREATE TABLE
    Pais (
        paisID INT,
        nombre VARCHAR(20) UNIQUE NOT NULL,
        PRIMARY KEY (paisID)
    );

CREATE TABLE
    Provincia (
        provinciaID INT,
        nombre VARCHAR(20) UNIQUE NOT NULL,
        paisID INT NOT NULL,
        PRIMARY KEY (provinciaID),
        FOREIGN KEY (paisID) REFERENCES Pais (paisID) ON DELETE CASCADE ON UPDATE CASCADE
    );
```

Ejemplo con opción SET NULL:

En este caso, al eliminar un departamento, todos los empleados de ese departamento tendrán un valor “nulo” (el DBMS lo generará automáticamente). Por lo tanto, el campo depID debe permitir valores nulos.

```sql
CREATE TABLE
    Departamento (
        depID INT,
        nombre VARCHAR(20) UNIQUE NOT NULL,
        PRIMARY KEY (depID)
    );

CREATE TABLE
    Empleado (
        empleadoID INT,
        nombre VARCHAR(20) UNIQUE NOT NULL,
        depID INT,
        FOREIGN KEY (depID) REFERENCES Departamento (depID) ON DELETE SET NULL ON UPDATE SET NULL
    );
```

Ejemplo con opción RESTRICT:

En este caso, no es posible eliminar un departamento si existen empleados de ese departamento. Cuando intentas actualizar el depID en la tabla Departamento, el sistema no te lo permite, mostrando un error.

```sql
CREATE TABLE
    Departamento (
        depID INT,
        nombre VARCHAR(20) UNIQUE NOT NULL,
        PRIMARY KEY (depID)
    );

CREATE TABLE
    Empleado (
        empleadoID INT,
        nombre VARCHAR(20) UNIQUE NOT NULL,
        depId INT NOT NULL,
        FOREIGN KEY (depID) REFERENCES Departamento (depID) ON DELETE RESTRICT ON UPDATE RESTRICT
    );
```

2.5.- Restricciones (CONSTRAINTS)

Las restricciones de SQL se utilizan para especificar reglas para los datos de una tabla. Es posible crear tablas que describan las restricciones, como puede ver en estos ejemplos:

```sql
CREATE TABLE
    IF NOT EXISTS Departamento (
        depID INT,
        nombre VARCHAR(20) NOT NULL,
        CONSTRAINT DepartmentoPK PRIMARY KEY (depID),
        CONSTRAINT NombreUnico UNIQUE (nombre)
    );

CREATE TABLE
    IF NOT EXISTS Tarea (
        tareaID INT AUTO_INCREMENT,
        titulo VARCHAR(20) NOT NULL,
        fechaIni DATE,
        fechaFin DATE,
        prioridad TINYINT NOT NULL DEFAULT 1,
        fechaCreacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        depID INT NOT NULL,
        CONSTRAINT TareaPK PRIMARY KEY (tareaID),
        CONSTRAINT TareaDepIDFK FOREIGN KEY (depID) REFERENCES Departmento (depID),
        CONSTRAINT FechaIniFin CHECK (fechaFin > fechaIni)
    );
```

Puede incluir otras restricciones con la cláusula CHECK utilizando operadores aritméticos (+, -, *, /), operadores de comparación (>, <, =, >=,<=, <>) y operadores lógicos (AND, OR, NOT). Consulte todos los operadores posibles Operadores MySQL.

2.6.- Modificar una tabla

La sintaxis simplificada de esta operación es:

```
ALTER TABLE nombreTabla
    [opcion_modificar [, opcion_modificar] ...]
opcion_modificar: {
    | ADD [COLUMN] (nombreColumna definicionColumna,...)
    | ADD [CONSTRAINT [simbolo]] PRIMARY KEY
          [tipoIndice] (clave,...)
          [opcionIndice] ...
    | ADD [CONSTRAINT [simbolo]] UNIQUE [INDEX | KEY]
          [nombreIndice] [tipoIndice] (clave,...)
          [opcionIndice] ...
    | ADD [CONSTRAINT [simbolo]] FOREIGN KEY
          [nombreIndice] (nombreColumna,...)
```

```
        definicionReferencia
    | ADD [CONSTRAINT [simbolo]] CHECK (expresion)
    | DROP {CHECK | CONSTRAINT} simbolo
    | ALTER [COLUMN] nombreColumna {
        SET DEFAULT {literal | (expresion)}
        | SET {VISIBLE | INVISIBLE}
        | DROP DEFAULT
        }
    | DROP [COLUMN] nombreColumna
    | DROP PRIMARY KEY
    | DROP FOREIGN KEY fkSimbolo
    | MODIFY [COLUMN] nombreColumna definicionColumna
    | RENAME COLUMN antiguoNombre TO nuevoNombre
}
```

Puedes ver la sintaxis completa de esta operación en este enlace Alter table.

Veamos algunos ejemplos:

- Añadir una nueva columna.

```
1   ALTER TABLE Departmento ADD COLUMN numeroDeEmpleados INT DEFAULT 0;
```

- Eliminar una columna.

```
1   ALTER TABLE Departmento DROP COLUMN nombre;
```

- Modificar una columna.

```
1   ALTER TABLE Departmento MODIFY nombre VARCHAR(40);
```

- Renombrar una columna.

```
1   ALTER TABLE Departmento RENAME COLUMN name TO nombreDepartamento;
```

- *Renombrar columna en MySQL

```
ALTER TABLE Departamento CHANGE nombre nombreDepartamento varchar;
```

- Añadir restricciones clave. Se pueden crear tablas sin ninguna restricción y añadirlas después.

```
1   / / Creamos la tabla sin primary key
2   CREATE TABLE IF NOT EXISTS Departmento (depID INT, nombre VARCHAR(20) UNIQUE NOT NULL,);
3
4   / / Añadimos ahora el constraint (primary key)
5   ALTER TABLE Departmento ADD CONSTRAINT DepartmentoPK PRIMARY KEY (depID);
```

- Añadir restricciones de comprobación. Es posible añadir restricciones a los valores de los atributos (columnas).

```
ALTER TABLE Tarea ADD CONSTRAINT PrioridadPositiva CHECK(prioridad > 0);

ALTER TABLE Tarea ADD CONSTRAINT Prioridad1a5 CHECK(prioridad >= 1 AND prioridad<=5);
```

- Eliminar una restricción.

```
ALTER TABLE Tarea DROP CONSTRAINT PrioridadPositiva;
```

3.- SQL. Lenguaje de manipulación de datos (LDM)

En esta sección vamos a estudiar las sentencias SQL para manipular datos.

- INSERT
- UPDATE
- DELETE

3.1.- Insertar (INSERT)

Inserta nuevas filas en una tabla existente:

`INSERT INTO NombreTabla(nombreCampo1, nombreCampo2..) VALUES (valor1, valor2..)`

Ejemplos:

```
INSERT INTO
    Departmento (depID, nombre)
VALUES
    (1, 'PROGRAMACIÓN');

INSERT INTO
    Tarea (titulo, fechaIni, fechaFin, depID)
VALUES
    ('Probar programa A', '2024-10-29', '2025-02-15', 1);
```

Tenga en cuenta que:

- Los valores de cadena y las fechas deben representarse entre comillas simples.
- Si un campo es AUTO_INCREMENT no aparece en la sentencia INSERT.
- Si un campo no es NULL o tiene un valor predeterminado, no es obligatorio que aparezca en la sentencia INSERT.
- Es posible otra sentencia INSERT (nombre INSERT INTO SELECT). Se estudiará en las siguientes unidades.

3.2.- Actualizar (UPDATE)

Actualiza las filas que cumplen la condición en la cláusula WHERE. En la siguiente sección puede ver operadores para construir condiciones.

```
UPDATE NombreTabla
SET columna1 = valor1, columna2 = valor2, ...
WHERE condicion;
```

NOTA: ¡Tenga cuidado al actualizar registros en una tabla! Observe la cláusula WHERE en la declaración UPDATE. La cláusula WHERE especifica qué registro(s) se deben actualizar. Si omite la cláusula WHERE, se actualizarán todos los registros de la tabla.

Ejemplos:

```
1   UPDATE Departmento SET nombre='PROGRAMACIÓN Y TESTING' WHERE depID=1;
2
3   UPDATE Tarea SET prioridad=4 WHERE depID=1;
```

3.3.- Eliminar (DELETE)

Elimina las filas que cumplen la condición

```
DELETE FROM NombreTabla WHERE condicion;
```

NOTA: ¡Tenga cuidado al eliminar registros de una tabla! Observe la cláusula WHERE en la declaración DELETE. La cláusula WHERE especifica qué registro(s) se deben eliminar. Si omite la cláusula WHERE, se eliminarán todos los registros de la tabla.

Ejemplos:

```
DELETE FROM Departamento WHERE depID = 1;
```

3.4.- Operadores SQL

SQL tiene muchos operadores. En las siguientes tablas puedes ver algunos de ellos utilizados para construir una cláusula condicional compleja. Algunos de estos operadores se estudiarán en profundidad en la siguiente unidad.

Operadores	Descripción
> >=	Mayor que, mayor igual que
< <=	Menor que, menor igual que
=	Igual
<> !=	Distinto
IS NULL, IS NOT NULL	Comprueba si el campo es nulo o NO nulo
BETWEEN...AND... NOT BETWEEN... AND...	Comprueba si el valor está en el rango o NO lo está
LIKE , NOT LIKE	Cumple con el patrón
condicion1 AND condicion2	Se deben cumplir las dos condiciones para que sea verdad
condicion1 OR condicion2	Se debe cumplir una de las condiciones para que sea verdad
NOT condicion	Da la vuelta al valor de la condición

Ejemplos:

Sentencia SQL	¿Qué hace?
DELETE FROM Departmento WHERE depID >= 10 AND depID <= 20;	Elimina los departamentos con id entre 10 y 20
DELETE FROM Departmento WHERE depID BETWEEN 10 AND 20;	Elimina los departamentos con id entre 10 y 20
DELETE FROM Departmento WHERE depID = 10 OR depID = 20;	Elimina los departamentos con id 10 y 20 sí existen
DELETE FROM Departmento WHERE NOT (depID > 10);	Elimina los departamentos con id 10 o menor
DELETE FROM Departmento WHERE NOT (depID >= 10 AND depID <= 20);	Eliminar los departamentos con id 10 o menor y 20 o mayor
UPDATE Tarea SET prioridad = 1 WHERE depID BETWEEN 1 AND 5;	Actualiza las tareas de los departamentos con identificador entre 1 y 5 y les asigna prioridad 1
UPDATE Tarea SET prioridad = prioridad + 1 WHERE depID=1;	Actualiza las tareas del departamento 1 y les añade 1 a la prioridad que tengan
UPDATE Tarea SET prioridad = 5 WHERE fechaFin is NULL;	Actualiza las tareas que no tienen fecha de finalización cambiando su prioridad a 5.