

The background features a light gray surface with four silver laptops arranged in a circle. In the center, there is a large, semi-transparent red padlock. Behind the padlock, a faint, stylized graphic of a database cylinder is visible. The text is overlaid on this background.

# **MÓDULO II**

## **PROGRAMACIÓN DE BASES DE DATOS RELACIONALES**

### **Unidad Formativa 1**

#### **Diseño de Bases de Datos Relacionales**

##### **El Modelo Relacional**

# MODELO RELACIONAL

MÓDULOS	UNIDADES FORMATIVAS
PROGRAMACIÓN DE BASES DE DATOS RELACIONALES	<b>DISEÑO DE BASES DE DATOS RELACIONALES</b> Introducción a las bases de datos <b>Modelos conceptuales de bases de datos</b> El modelo relacional El ciclo de vida de un proyecto Creación y diseño de bases de datos
	<b>DEFINICIÓN Y MANIPULACIÓN DE DATOS</b> Lenguajes relacionales El lenguaje de manipulación de la base de datos
	<b>DESARROLLO DE PROGRAMAS EN EL ENTORNO DE LA BASE DE DATOS</b> Lenguajes de programación de bases de datos

# MODELO RELACIONAL

MÓDULOS	UNIDADES FORMATIVAS
PROGRAMACIÓN DE BASES DE DATOS RELACIONALES	<b>DISEÑO DE BASES DE DATOS RELACIONALES</b> Introducción a las bases de datos Modelos conceptuales de bases de datos <b>El modelo relacional</b> El ciclo de vida de un proyecto Creación y diseño de bases de datos
	<b>DEFINICIÓN Y MANIPULACIÓN DE DATOS</b> Lenguajes relacionales El lenguaje de manipulación de la base de datos
	<b>DESARROLLO DE PROGRAMAS EN EL ENTORNO DE LA BASE DE DATOS</b> Lenguajes de programación de bases de datos

# Origen del Modelo Relacional

- Edgar Frank Codd definió las bases del Modelo relacional.
- Publicó “A Relational Model of data for Large Shared Data Banks”. (1970) - Un modelo relacional de datos para grandes bancos de datos compartidos
- Codd era matemático y se basó en la teoría de conjuntos. La base del modelo relacional es la teoría de conjuntos.
  - Los datos se agrupan en “relaciones” denominadas “tablas” que aglutinan datos referidos a una misma entidad de forma organizada.
  - Las “relaciones” estructuran los datos de forma independiente a su almacenamiento real en el ordenador, es decir

ES UN ELEMENTO CONCEPTUAL, NO FÍSICO

# Conjuntos

- Es una colección de elementos. Están caracterizados por compartir alguna propiedad. Para que un conjunto esté bien definido debe ser posible discernir si un elemento arbitrario está o no en él.

## Explícita

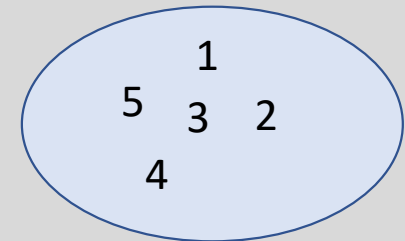
$$A = \{1, 2, 3, 4, 5\}$$

## Implícita

$$A = \{\text{números naturales del 1 al 5}\}$$

$$A = \{x / x \in \mathbb{N} \wedge x \leq 5\}$$

## Diagrama de Venn



## Conjunto vacío

$$\emptyset$$

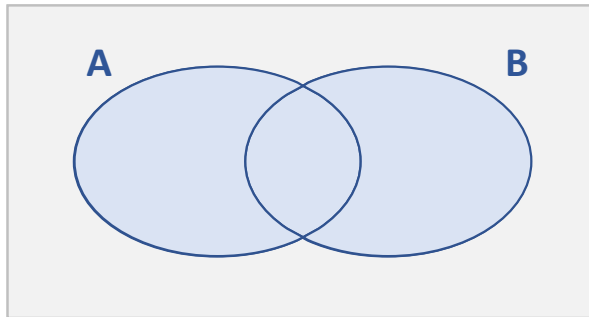
## Inclusión

$$\left. \begin{array}{l} A = \{1, 2, 3, 4, 5\} \\ B = \{5, 2\} \end{array} \right\} \begin{array}{l} B \subset A \text{ o } B \subseteq A \\ A \supset B \text{ o } A \supseteq B \end{array}$$

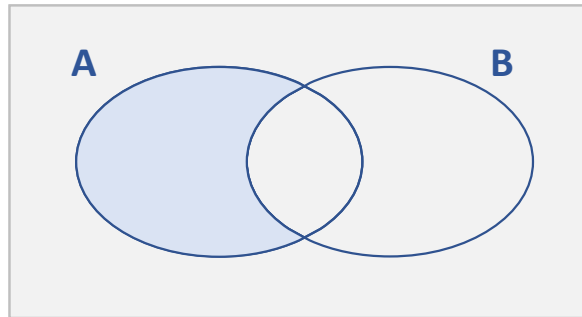
B es un subconjunto de A

# Operaciones de Conjuntos

**Unión:**  $A \cup B$



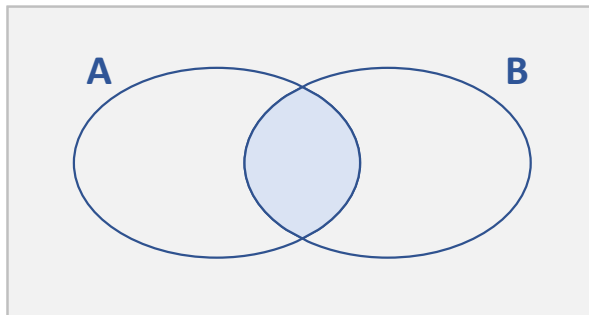
**Diferencia:**  $A - B$  o  $A \setminus B$



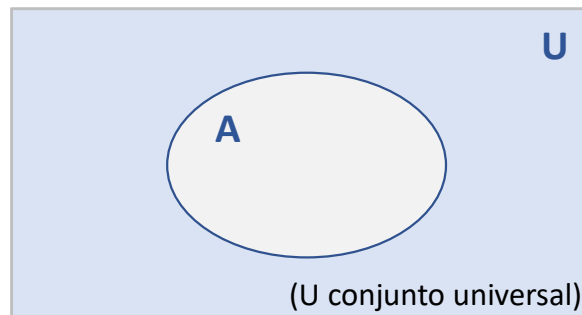
**Producto cartesiano:**  $A \times B$

El producto cartesiano de dos conjuntos A y B es el conjunto  $A \times B$  de todos los pares ordenados (a, b) formados con un primer elemento a perteneciente a A, y un segundo elemento b perteneciente a B.

**Intersección:**  $A \cap B$



**Complemento:**  $A^c$



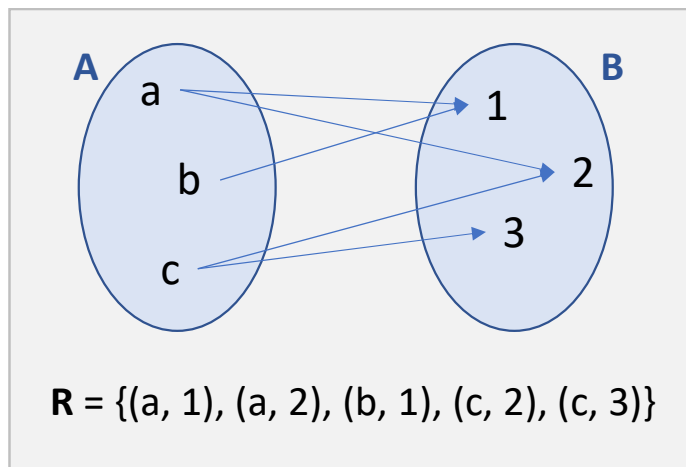
$$A = \{1, 2\}$$

$$B = \{a, b, c\}$$

$$A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$$

# Relaciones matemáticas

- Una “relación”  $R$  es una regla de correspondencia entre los elementos de  $n$  conjuntos.



- Una relación  $R$ , de  $n$  conjuntos, es un subconjunto del producto cartesiano de los  $n$  conjuntos

➤ Unaria:  $R \subseteq A, R(a)$

➤ Binaria:  $R \subseteq A_1 \times A_2, R(a_1, a_2)$

➤ Ternaria:  $R \subseteq A_1 \times A_2 \times A_3, R(a_1, a_2, a_3)$

➤  $n$ -aria:  $R \subseteq A_1 \times A_2 \times A_3 \times \dots \times A_n, R(a_1, a_2, a_3, \dots, a_n)$

$A \times B = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3), (c, 1), (c, 2), (c, 3)\}$

$R = \{(a, 1), (a, 2), (b, 1), (c, 2), (c, 3)\}$

$$R \subseteq (A \times B)$$

# Relaciones, atributos, tuplas, dominios

## Atributos

ID_JUGADOR	NOMBRE	APELLIDOS	DNI	FECHA_NAC
1	Javier	Ortega Desio	4473754P	02/10/1984
2	Marcos	Ayerza	7940816K	08/05/1984
3	Jeronimo	De La Fuente	5169607A	07/06/1986
4	Juan Martin	Fernandez Lobbe	5145360O	17/01/1983
5	Santiago	Garcia Botta	3763193R	26/01/1986

## Relación

### Grado

Cantidad de atributos de una relación.

### Cardinalidad

Cantidad de tuplas de una relación.

## Dominio

Se refiere a un atributo y es el conjunto de todos los posibles valores de ese atributo

**Dominio de ID\_JUGADOR**  
{naturales  $> 0$  y  $< 1.000.000$ }

**Dominio de NOMBRE**  
{todos los posibles nombres}

## Tuplas



# Sinónimos

Relacional	Visual	Ficheros	Representa
relación	tabla	fichero	entidad
tupla	fila	registro	una instancia
atributo	columna / campo /propiedad	campo	una propiedad
grado	nº de columnas	nº de campos	nº propiedades
cardinalidad	nº de filas	nº de registros	nº de instancias
clave primara	clave primaria	--	identificador instancia

# Propiedades de las tablas

- Unicidad de nombre: cada tabla debe tener un nombre distinto
- Atributos atómicos: cada atributo de la tabla solo puede tener un valor en cada tupla
- Cada atributo tiene un nombre distinto en cada tabla (aunque puede coincidir en tablas distintas)
- Cada tupla es única (no hay tuplas duplicadas), pk diferente.
- El orden de los atributos no importa
- El orden de las tuplas no importa

# Claves

- **superclave**

Conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma unívoca una tupla en una tabla.

Las superclaves que son mínimas, es decir, que no tienen ningún subconjunto de atributos que sea superclave, se denominan “**claves candidatas**”.

- **clave primaria** (*primary key*)

Es la “clave candidata” elegida por el diseñador de la BBDD como elemento principal para identificar a las tuplas dentro de la tabla. (identificador).

- **claves alternativas**

Son las claves candidatas de una tabla que no se han elegido como primaria.

- **clave foránea**, externa o ajena (*foreign key*)

*Conjunto de uno o más atributos cuyos valores deben coincidir con los valores de la clave primaria de otra tabla.*

# Claves

## Jugadores

ID_JUGADOR	NOMBRE	APELLIDOS	DNI	FECHA_NAC	NRO_LICENCIA
1	Javier	Ortega Desio	4473754P	02/10/1984	2934
2	Marcos	Ayerza	7940816K	08/05/1984	1640
3	Jeronimo	De La Fuente	5169607A	07/06/1986	3919
4	Juan Martin	Fernandez Lobbe	5145360O	17/01/1983	8742
5	Santiago	Garcia Botta	3763193R	26/01/1986	7578
6	Lucas	Gonzales Amorosino	2807609L	15/04/1993	2910
7	Marinao	Galarza	1485392G	19/02/1988	2310
8	Pablo	Matera	6089091L	23/05/1974	9584
9	Juan	Pablo Socino	6055307E	17/08/1975	9882
10	Guido	Petti Pagadizabal	2773999I	16/02/1997	9355
11	Juan	Figallo	8739151Y	09/12/1971	9731
12	Santiago	Gonzalez Iglesias	3846194N	27/08/1987	6555

## Equipos

ID_EQUIPO	NOMBRE_E	CATEGORIA	LIGA
1	SENIOR A	DHB	NACIONAL
2	SENIOR B	1A	REGIONAL
3	FEMENINO A	DH	NACIONAL
4	M23	1B	NACIONAL
5	M21	2A	REGIONAL

[superclave](#)

[claves candidatas](#)

[claves primarias](#)

20/05/2025

Pablo López

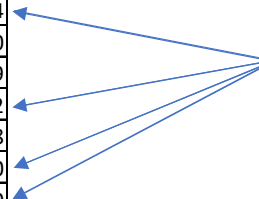
# Claves foráneas

## Jugadores

ID_JUGADOR	NOMBRE	APELLIDOS	DNI	FECHA_NAC	NRO_LICENCIA
1	Javier	Ortega Desio	4473754P	02/10/1984	2934
2	Marcos	Ayerza	7940816K	08/05/1984	1640
3	Jeronimo	De La Fuente	5169607A	07/06/1986	3919
4	Juan Martin	Fernandez Lobbe	5145360O	17/01/1983	8742
5	Santiago	Garcia Botta	3763193R	26/01/1986	7578
6	Lucas	Gonzales Amorosino	2807609L	15/04/1993	2910
7	Marinao	Galarza	1485392G	19/02/1988	2310
8	Pablo	Matera	6089091L	23/05/1974	9584
9	Juan	Pablo Socino	6055307E	17/08/1975	9882
10	Guido	Petti Pagadizabal	2773999I	16/02/1997	9355
11	Juan	Figallo	8739151Y	09/12/1971	9731
12	Santiago	Gonzalez Iglesias	3846194N	27/08/1987	6555

## Equipos

ID_EQUIPO	NOMBRE_E	CATEGORIA	LIGA
1	SENIOR A	DHB	NACIONAL
2	SENIOR B	1A	REGIONAL
3	FEMENINO A	DH	NACIONAL
4	M23	1B	NACIONAL
5	M21	2A	REGIONAL



claves primarias

20/05/2025

Pablo López

# Claves foráneas

## Jugadores

ID_JUGADOR	NOMBRE	APELLIDOS	DNI	FECHA_NAC	NRO_LICENCIA	ID_EQUIPO
1	Javier	Ortega Desio	4473754P	02/10/1984	2934	1
2	Marcos	Ayerza	7940816K	08/05/1984	1640	2
3	Jeronimo	De La Fuente	5169607A	07/06/1986	3919	2
4	Juan Martin	Fernandez Lobbe	5145360O	17/01/1983	8742	1
5	Santiago	Garcia Botta	3763193R	26/01/1986	7578	4
6	Lucas	Gonzales Amorosino	2807609L	15/04/1993	2910	1
7	Marinao	Galarza	1485392G	19/02/1988	2310	1
8	Pablo	Matera	6089091L	23/05/1974	9584	4
9	Juan	Pablo Socino	6055307E	17/08/1975	9882	4
10	Guido	Petti Pagadizabal	2773999I	16/02/1997	9355	2
11	Juan	Figallo	8739151Y	09/12/1971	9731	4
12	Santiago	Gonzalez Iglesias	3846194N	27/08/1987	6555	2

## Equipos

ID_EQUIPO	NOMBRE_E	CATEGORIA	LIGA
1	SENIOR A	DHB	NACIONAL
2	SENIOR B	1A	REGIONAL
3	FEMENINO A	DH	NACIONAL
4	M23	1B	NACIONAL
5	M21	2A	REGIONAL

claves primarias

clave foránea

20/05/2025

Pablo López

# Claves

## Jugadores

ID_JUGADOR	NOMBRE	APELLIDOS	DNI	FECHA_NAC	NRO_LICENCIA	ID_EQUIPO
1	Javier	Ortega Desio	4473754P	02/10/1984	2934	1
2	Marcos	Ayerza	7940816K	08/05/1984	1640	2
3	Jeronimo	De La Fuente	5169607A	07/06/1986	3919	2
4	Juan Martin	Fernandez Lobbe	5145360O	17/01/1983	8742	1
5	Santiago	Garcia Botta	3763193R	26/01/1986	7578	4
6	Lucas	Gonzales Amorosino	2807609L	15/04/1993	2910	1
7	Marinao	Galarza	1485392G	19/02/1988	2310	1
8	Pablo	Matera	6089091L	23/05/1974	9584	4
9	Juan	Pablo Socino	6055307E	17/08/1975	9882	4
10	Guido	Petti Pagadizabal	2773999I	16/02/1997	9355	2
11	Juan	Figallo	8739151Y	09/12/1971	9731	4
12	Santiago	Gonzalez Iglesias	3846194N	27/08/1987	6555	2

## Equipos

ID_EQUIPO	NOMBRE_E	CATEGORIA	LIGA
1	SENIOR A	DHB	NACIONAL
2	SENIOR B	1A	REGIONAL
3	FEMENINO A	DH	NACIONAL
4	M23	1B	NACIONAL
5	M21	2A	REGIONAL

superclave

~~claves candidatas~~

claves alternativas

claves primarias

clave foránea

20/05/2025

Pablo López

# Valor nulo (null)

- El valor null indica la ausencia de valor en un atributo.
  - Si por ejemplo, en el atributo “email” aparece un null, indica que esa persona no tiene email.
- El carácter espacio ' ' es diferente de null y 0 es diferente de null



# Proposiciones y tablas de verdad

Una proposición  $p$  es cualquier enunciado lógico del que se puede formular su veracidad o falsedad, es decir,  $p$  puede ser verdadera o falsa.

$p$ : Te he pagado lo que te debía.

$q$ : Te he pagado en efectivo.

Dadas las proposiciones “ $p$ ” y “ $q$ ” se definen los siguientes operadores:

## conjunción (and o $\wedge$ )

p and q		q	
		true	false
p	true	true	false
	false	false	false

## disyunción (or o $\vee$ )

p or q		q	
		true	false
p	true	true	true
	false	true	false

## negación (not o $\neg$ )

p	not p
true	false
false	true

# Lógica ternaria

null no se considera propiamente como un “valor” sino que indica la ausencia de un valor, por eso las comparaciones con null no resultan ciertas o falsas, sino desconocidas (unknown).

Sin embargo, algunas operaciones con null pueden devolver valores si el null no es relevante en la operación.

Ej:  $V \text{ or } \text{null} \rightarrow \text{será verdadero}$

## conjunción (and o $\wedge$ )

p and q		q		
		true	false	null
p	true	true	false	null
	false	false	false	false
	null	null	false	null

## disyunción (or o $\vee$ )

p or q		q		
		true	false	null
p	true	true	true	true
	false	true	false	null
	null	true	null	null

## negación (not o $\neg$ )

p	not p
true	false
false	true
null	null

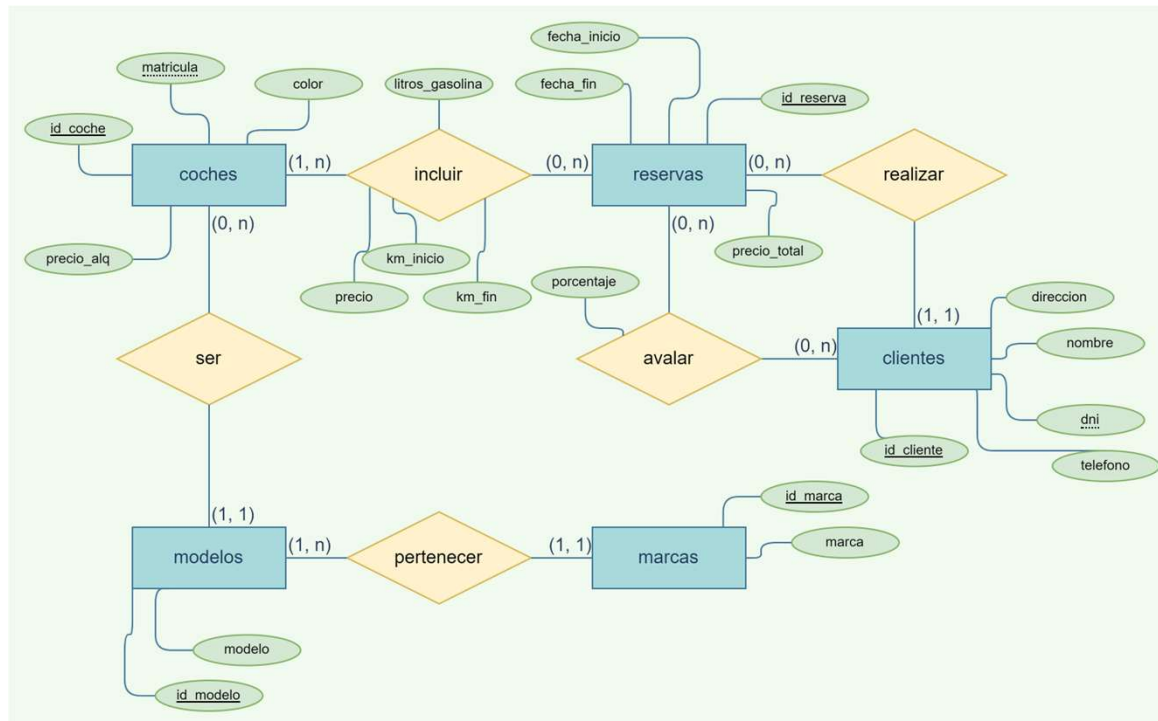
# Restricciones de integridad

- Son condiciones de obligado cumplimiento orientadas a mantener la integridad de los datos.
- Inherentes
  - Están definidas por el propio hecho que la BBDD sea relacional.
    - No puede haber dos filas iguales
    - El orden de las filas no es significativo
    - El orden de las columnas no es significativo
    - Cada atributo sólo puede tomar un valor en una fila.
- Semánticas...

# Restricciones de integridad semánticas

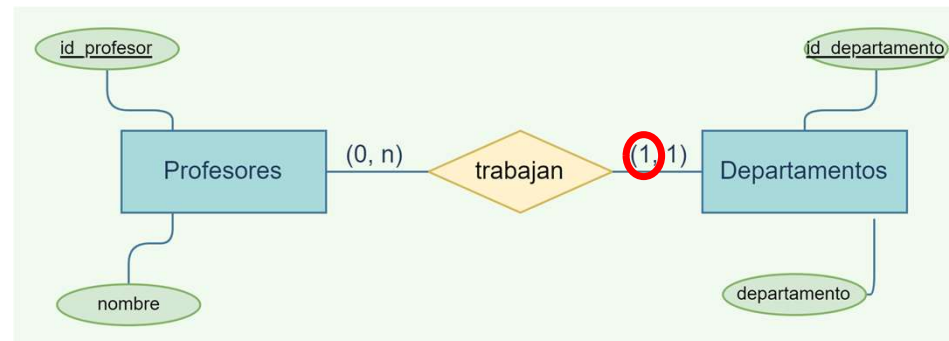
- Restricción de clave primaria (**primary key**)
  - Toda tabla requiere de una clave primaria, sus valores en una tabla deben ser todos diferentes y ningún atributo de ella puede contener null.
- Restricción de unicidad (**unique**)
  - Impide que los atributos marcados como “unique” puedan repetirse en distintas filas. Deben ser todos distintos o vacíos. Así deben indicarse en las columnas que son claves alternativas.
- Obligatoriedad (**not null**)
  - Prohíbe que el atributo marcado como “not null” contenga un null. Así deben indicarse en las columnas que son claves alternativas.
- Integridad referencial (**foreign key**)
  - Indica que las columnas así marcadas deben contener valores de la clave primaria de la tabla que relaciona (tabla principal) o un null.
- Regla de validación (**check**)
  - Permiten restringir los valores de un atributo. Mayor a 500. Fecha inicio < que Fecha fin.

# Paso de ER a Relacional



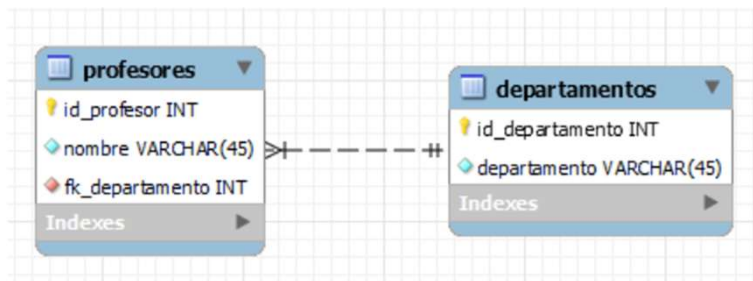
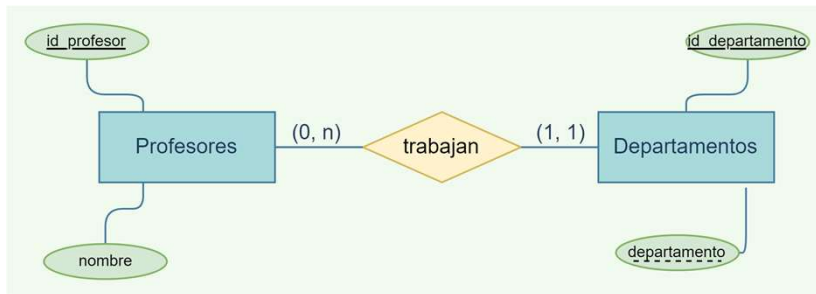
- Por cada entidad se crea una nueva tabla.
- Todos los atributos simples de la entidad pasan a ser atributos de la tabla.
- Los atributos compuestos, se descomponen y cada componente pasa a ser un atributo de la tabla.
- El atributo ID de la entidad será la Primary Key de la tabla.
- Los atributos que son claves alternativas se definirán como UNIQUE en la tabla.

# Relaciones 1-n



- En la tabla del lado de “n” se añade un atributo que funcionará como Foreign Key a la otra tabla.
- Este atributo será simple o compuesto, tal cual sea la primary key de la tabla que referencia y del mismo tipo de dato.
- Deberá crearse la restricción referencial “constraint” foreign key.
- ◉ Finalmente, dependiendo de la cardinalidad mínima del lado “1” la foreign key deberá definirse como NOT NULL si es 1, o que acepte null si es 0.

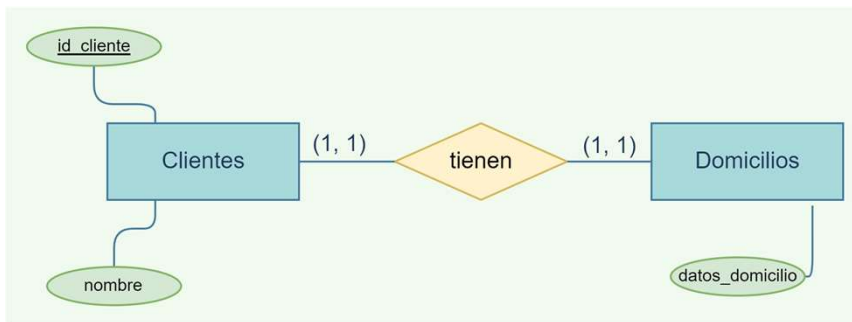
# Relaciones 1-n



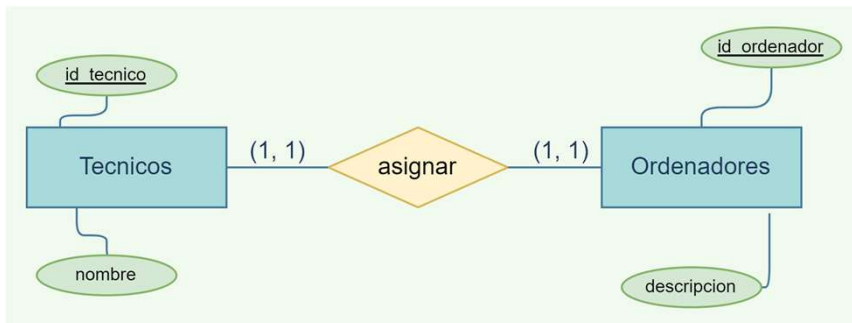
```
CREATE TABLE `departamentos` (  
  `id_departamento` int NOT NULL AUTO_INCREMENT,  
  `departamento` varchar(45) NOT NULL,  
  PRIMARY KEY (`id_departamento`),  
  UNIQUE KEY `departamento_UNIQUE` (`departamento`)  
);
```

```
CREATE TABLE `profesores` (  
  `id_profesor` int NOT NULL AUTO_INCREMENT,  
  `nombre` varchar(45) NOT NULL,  
  `fk_departamento` int NOT NULL,  
  PRIMARY KEY (`id_profesor`),  
  CONSTRAINT `profesores_departamentos` FOREIGN KEY (`fk_departamento`)  
    references `departamentos` (`id_departamento`)  
);
```

# Relaciones 1-1 “caso (1,1) – (1,1)”



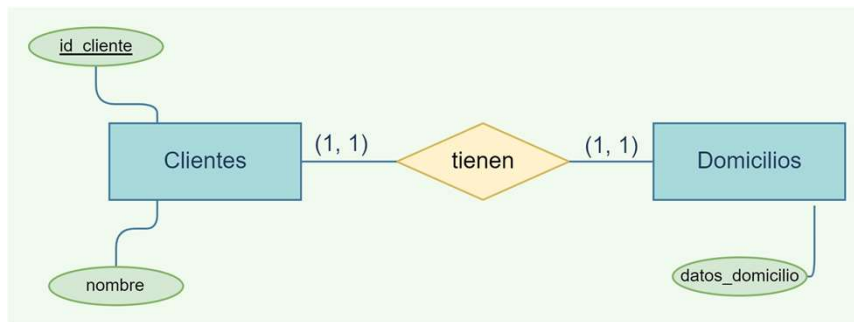
- Opción 1:  
Crear sólo una tabla con todos los atributos de las dos entidades.



- Opción 2  
Tratarlo como un caso especial de 1-n. Se agrega una foreign key que debe ser UNIQUE NOT NULL



# Relaciones 1-1 “caso (1,1) – (1,1)”



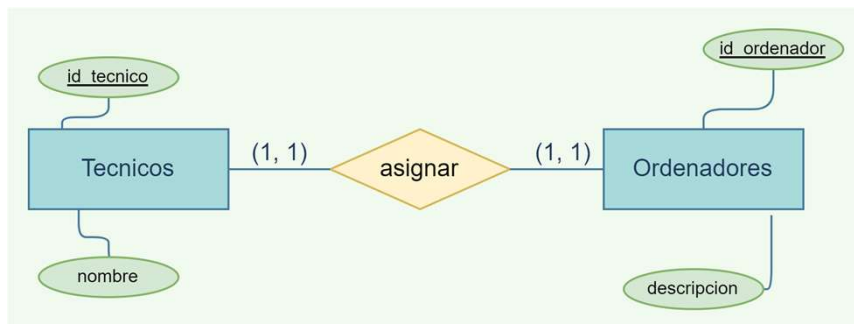
clientes	
id_cliente	INT
nombre	VARCHAR(45)
otros_datos_cliente	VARCHAR(45)
calle	VARCHAR(25)
nro	VARCHAR(10)
otros_datos_domicilio	VARCHAR(45)
Indexes	

- Opción 1:

Crear sólo una tabla con todos los atributos de las dos entidades.

```
CREATE TABLE `clientes` (  
  `id_cliente` int NOT NULL,  
  `nombre` varchar(45) NOT NULL,  
  `otros_datos_cliente` varchar(45) DEFAULT NULL,  
  `calle` varchar(25) NOT NULL,  
  `nro` varchar(10) NOT NULL,  
  `otros_datos_domicilio` varchar(45) NOT NULL,  
  PRIMARY KEY (`id_cliente`)  
);
```

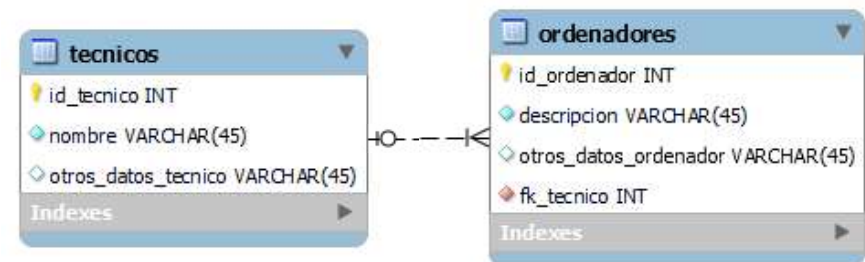
# Relaciones 1-1 “caso (1,1) – (1,1)”



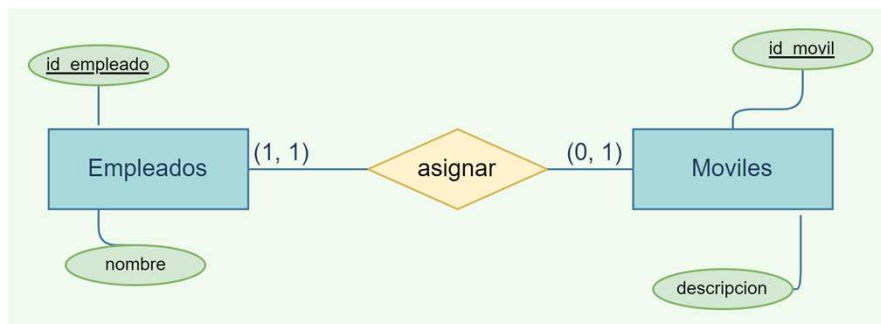
```
CREATE TABLE `ordenadores` (  
  `id_ordenador` int NOT NULL,  
  `descripcion` varchar(45) NOT NULL,  
  `otros_datos_ordenador` varchar(45) DEFAULT NULL,  
  `fk_tecnico` int NOT NULL,  
  PRIMARY KEY (`id_ordenador`),  
  UNIQUE KEY `fk_tecnico_UNIQUE` (`fk_tecnico`),  
  CONSTRAINT `ordenadores_tecnicos` FOREIGN KEY (`fk_tecnico`)  
    REFERENCES `tecnicos` (`id_tecnico`)  
);
```

- Opción 2: tratarlo como un caso especial de 1-n. Se agrega una foreign key “EN CUALQUIERA DE LAS DOS TABLAS” y debe ser UNIQUE NOT NULL

```
CREATE TABLE `tecnicos` (  
  `id_tecnico` int NOT NULL,  
  `nombre` varchar(45) NOT NULL,  
  `otros_datos_tecnico` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id_tecnico`)  
);
```



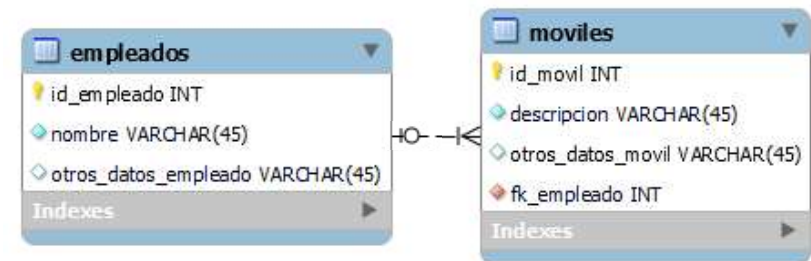
# Relaciones 1-1 “caso (1,1) – (0,1)”



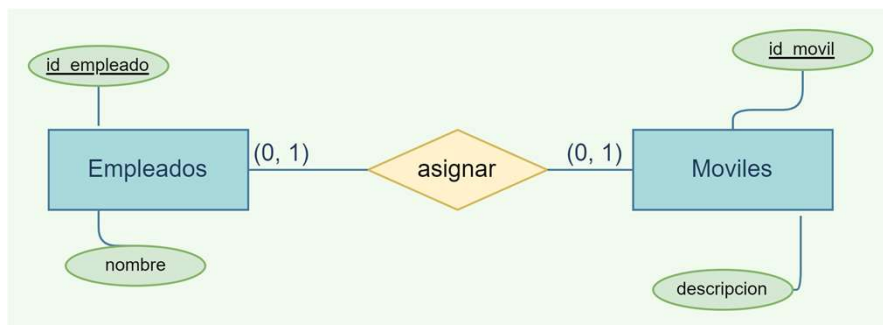
```
CREATE TABLE `moviles` (  
  `id_movil` int NOT NULL,  
  `descripcion` varchar(45) NOT NULL,  
  `otros_datos_movil` varchar(45) DEFAULT NULL,  
  `fk_empleado` int NOT NULL,  
  PRIMARY KEY (`id_movil`),  
  UNIQUE KEY `fk_empleado_UNIQUE` (`fk_empleado`),  
  CONSTRAINT `moviles_empleados` FOREIGN KEY (`fk_empleado`)  
    REFERENCES `empleados` (`id_empleado`)  
);
```

- Se trata como un caso especial de 1-n. Del lado (0,1) se agrega una foreign key y debe ser UNIQUE NOT NULL

```
CREATE TABLE `empleados` (  
  `id_empleado` int NOT NULL,  
  `nombre` varchar(45) NOT NULL,  
  `otros_datos_empleado` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id_empleado`)  
);
```



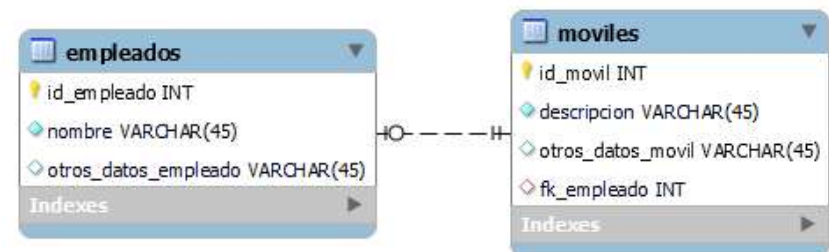
# Relaciones 1-1 “caso (0,1) – (0,1)”



```
CREATE TABLE `moviles` (  
  `id_movil` int NOT NULL,  
  `descripcion` varchar(45) NOT NULL,  
  `otros_datos_movil` varchar(45) DEFAULT NULL,  
  `fk_empleado` int DEFAULT NULL,  
  PRIMARY KEY (`id_movil`),  
  UNIQUE KEY `fk_empleado_UNIQUE` (`fk_empleado`),  
  CONSTRAINT `moviles_empleados` FOREIGN KEY (`fk_empleado`)  
    REFERENCES `empleados` (`id_empleado`)  
);
```

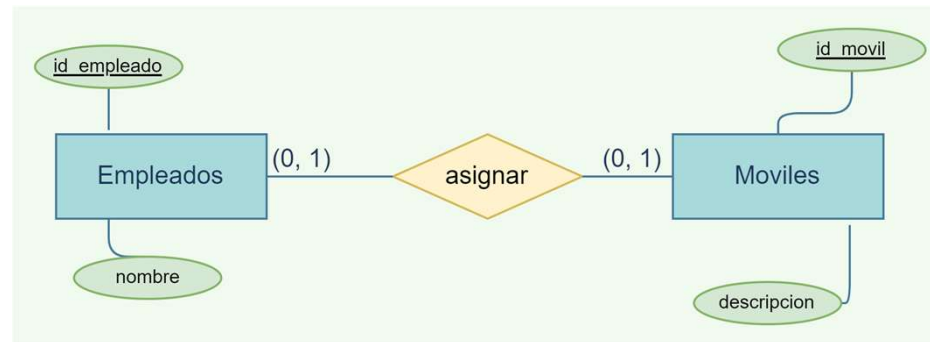
- Se trata como un caso especial de 1-n. Se agrega una foreign key y debe ser sólo UNIQUE

```
CREATE TABLE `empleados` (  
  `id_empleado` int NOT NULL,  
  `nombre` varchar(45) NOT NULL,  
  `otros_datos_empleado` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id_empleado`)  
);
```

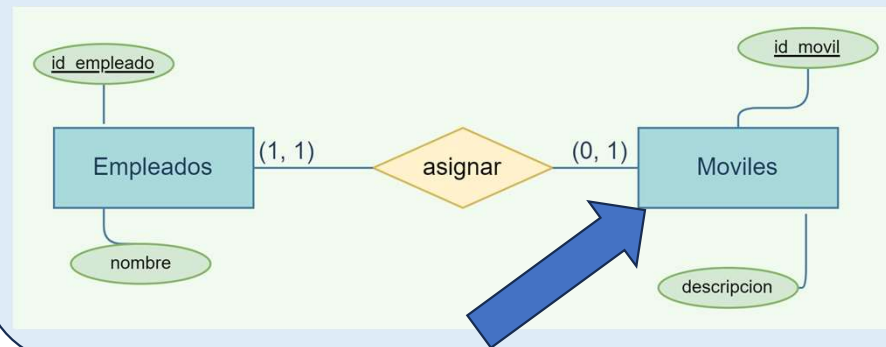


# Relaciones 1-1 “caso (0,1) – (0,1)”

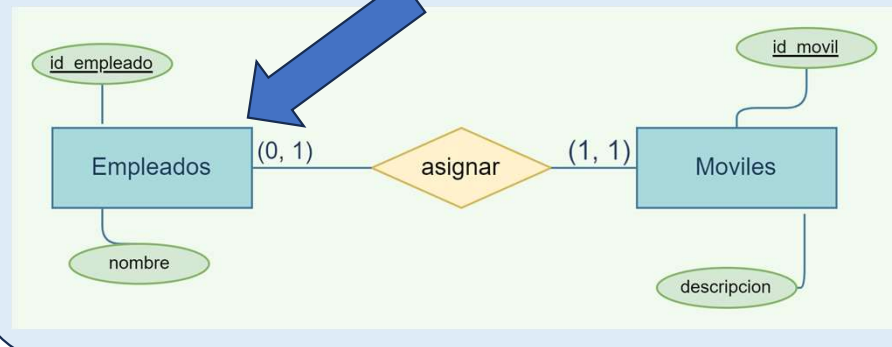
## ¿De qué lado poner la FK?



Si casi todos los móviles son entregados a empleados y muchos empleados no tienen móviles, entonces se acerca a:

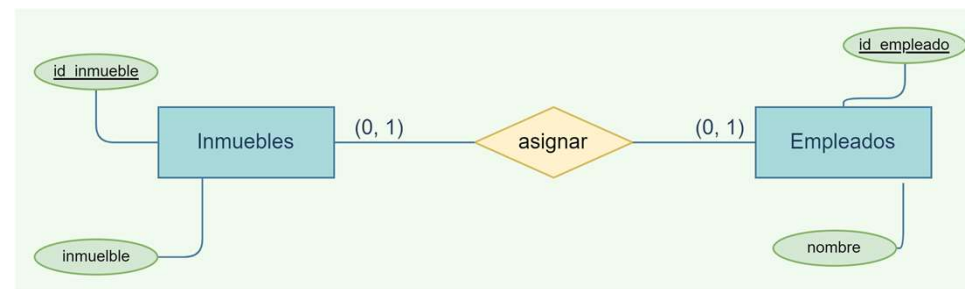


Si casi todos los empleados tienen un móvil y hay muchos móviles no asignados a empleados, entonces se acerca a:



# Relaciones 1-1 “caso (0,1) – (0,1)”

## ¿Y si hay muy pocas interrelaciones?

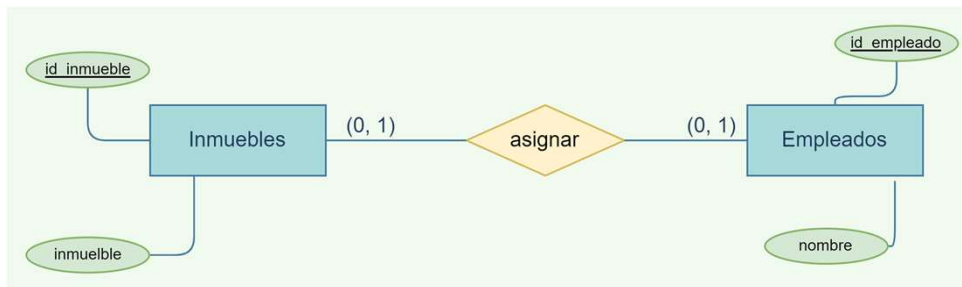


- Si la tratamos como caso especial de 1-n, deberíamos poner, por ejemplo, la FK a “Inmuebles” en “Empleados”, pero es FK, sería NULL para casi todos los empleados.
- Si decidimos poner la FK en “Inmuebles” que referencie a “Empleados”, pasaría lo mismo, prácticamente todos los inmuebles tendrían un NULL en la FK.
- Debemos crear una nueva tabla donde registrar las pocas interrelaciones que existen.
  - Tendrá las FK a las dos tablas y elegiremos una como PK (no se repetirá) y la otra como UNIQUE, tampoco se repetirá.



# Relaciones 1-1 “caso (0,1) – (0,1)”

## ¿Y si hay muy pocas interrelaciones?

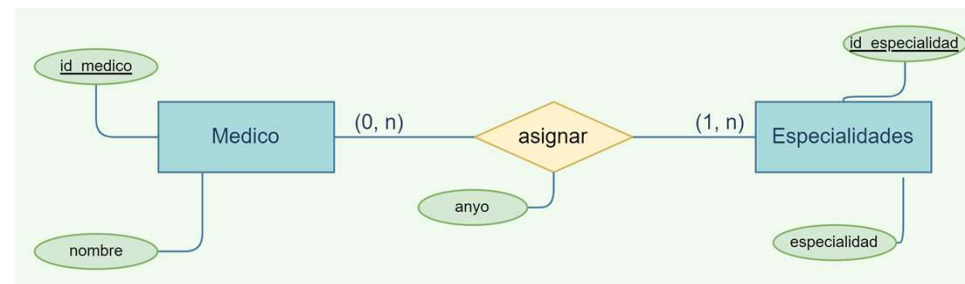


```
CREATE TABLE `inmuebles_empleados` (  
  `fk_inmueble` int NOT NULL,  
  `fk_empleado` int NOT NULL,  
  PRIMARY KEY (`fk_inmueble`),  
  UNIQUE KEY `id_empleado_UNIQUE` (`fk_empleado`),  
  CONSTRAINT `empleados_inmuebles` FOREIGN KEY (`fk_empleado`)  
    REFERENCES `empleados` (`id_empleado`),  
  CONSTRAINT `inmuebles_empleados` FOREIGN KEY (`fk_inmueble`)  
    REFERENCES `inmuebles` (`id_inmueble`)  
);
```

```
CREATE TABLE `empleados` (  
  `id_empleado` int NOT NULL,  
  `nombre` varchar(45) NOT NULL,  
  `otros_datos_empleado` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id_empleado`)  
);
```

```
CREATE TABLE `inmuebles` (  
  `id_inmueble` int NOT NULL,  
  `inmueble` varchar(45) NOT NULL,  
  `domicilio` varchar(45) NOT NULL,  
  PRIMARY KEY (`id_inmueble`)  
);
```

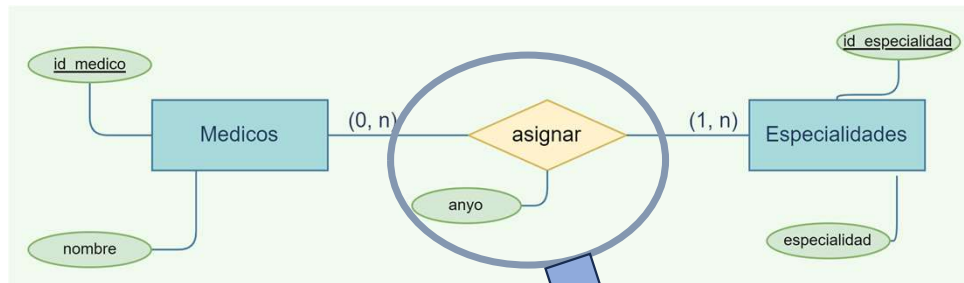
# Relaciones n-n



- Cada entidad pasa a ser una tabla con sus atributos.
- Toda relación n-n se convierte en una nueva tabla.
  - Todos los atributos de la relación pasan a ser atributos de la nueva tabla
  - Esta tabla tendrá dos FK, cada una referencia a una tabla de la relación.
  - Las FK en conjunto, conformar la PK compuesta de la tabla.



# Relaciones n-n



```

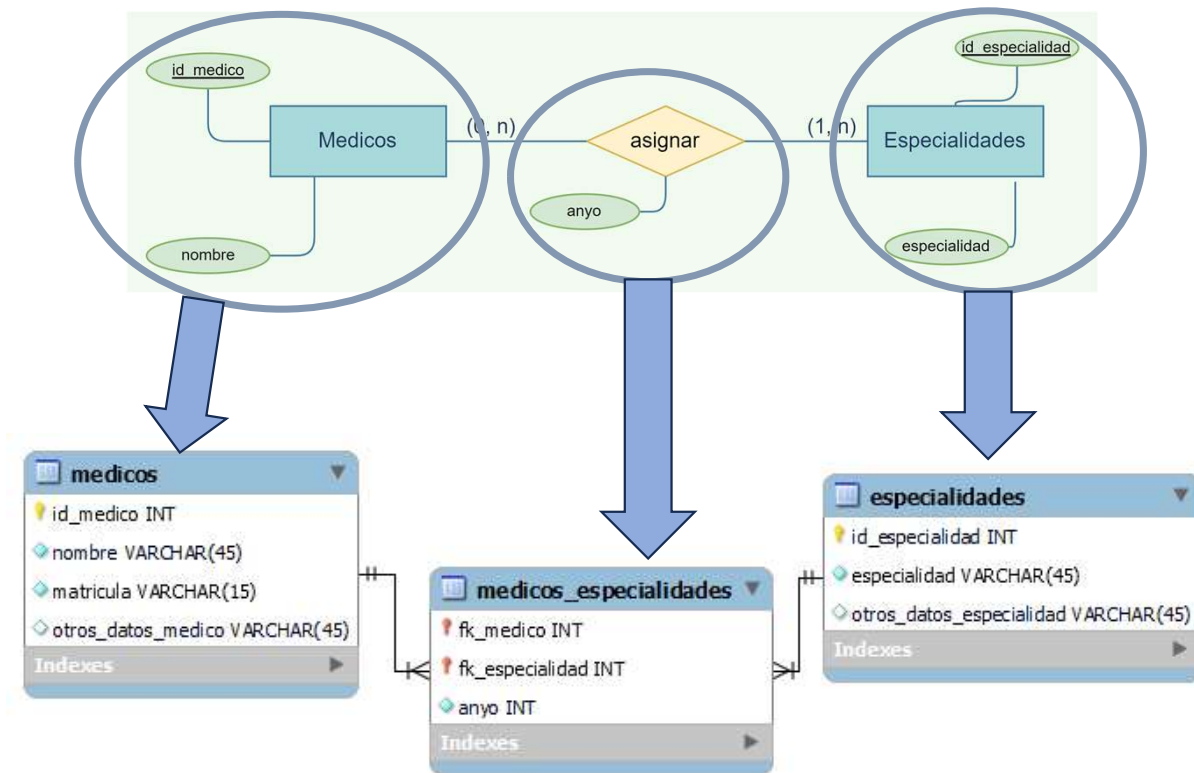
CREATE TABLE `medicos_especialidades` (
  `fk_medico` INT NOT NULL,
  `fk_especialidad` INT NOT NULL,
  `anyo` INT NOT NULL,
  PRIMARY KEY (`fk_medico`, `fk_especialidad`),
  CONSTRAINT `medicos_especialidades` FOREIGN KEY (`fk_medico`)
    REFERENCES `000_teoría`.`medicos` (`id_medico`),
  CONSTRAINT `especialidades_medicos` FOREIGN KEY (`fk_especialidad`)
    REFERENCES `000_teoría`.`especialidades` (`id_especialidad`)
);
    
```

```

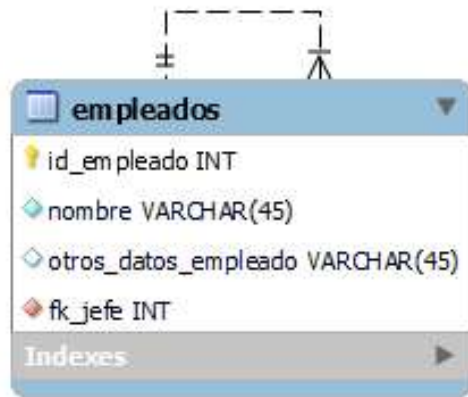
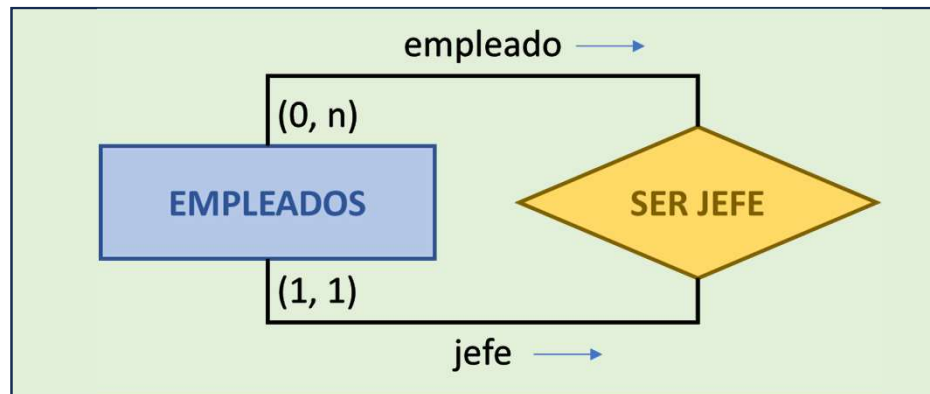
CREATE TABLE `medicos` (
  `id_medico` int NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `matricula` varchar(15) NOT NULL,
  `otros_datos_medico` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id_medico`),
  UNIQUE KEY `matricula_UNIQUE` (`matricula`)
);

CREATE TABLE `especialidades` (
  `id_especialidad` int NOT NULL,
  `especialidad` varchar(45) NOT NULL,
  `otros_datos_especialidad` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id_especialidad`),
  UNIQUE KEY `especialidad_UNIQUE` (`especialidad`)
);
    
```

# Relaciones n-n



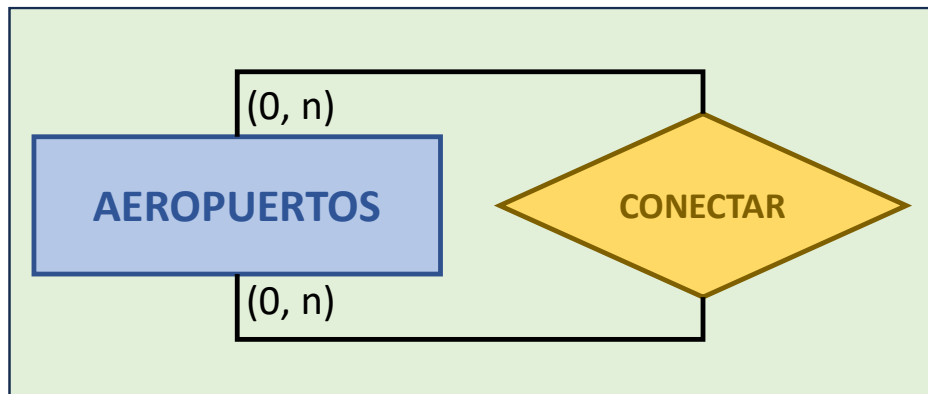
# Relaciones reflexivas (1-n)



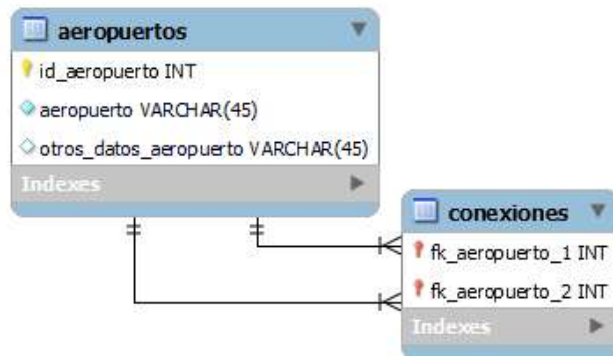
- Se siguen las mismas reglas que una relación 1-n, pero la referencia es a la misma tabla.

```
CREATE TABLE `empleados` (  
  `id_empleado` int NOT NULL,  
  `nombre` varchar(45) NOT NULL,  
  `otros_datos_empleado` varchar(45) DEFAULT NULL,  
  `fk_jefe` int NOT NULL,  
  PRIMARY KEY (`id_empleado`),  
  KEY `empleados_jefe_idx` (`fk_jefe`),  
  CONSTRAINT `empleados_jefe` FOREIGN KEY (`fk_jefe`)  
    REFERENCES `empleados` (`id_empleado`)  
);
```

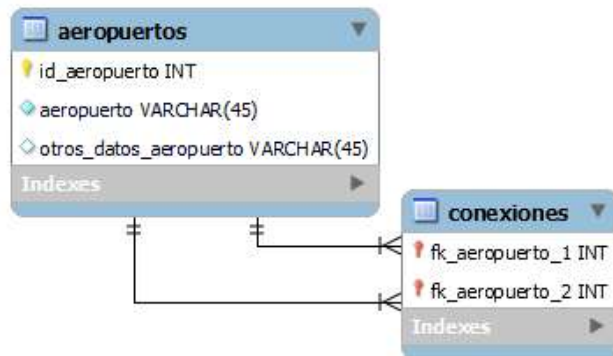
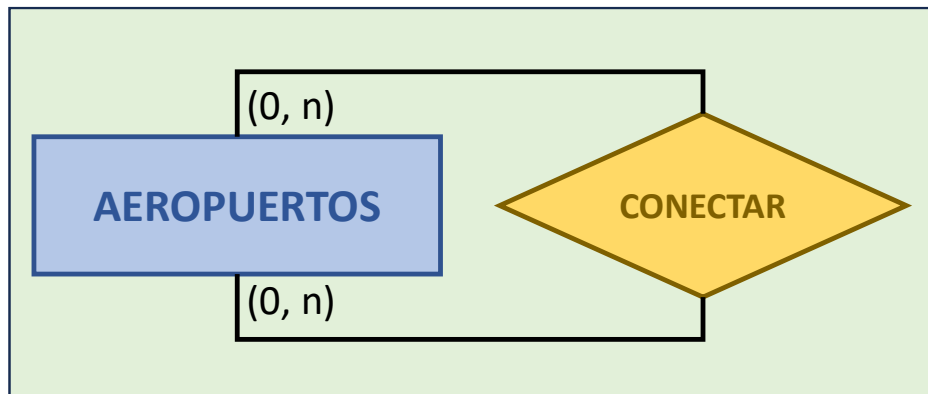
# Relaciones reflexivas (n-n)



- Se siguen las mismas reglas que una relación n-n.
- Una nueva tabla que tendrá dos referencias a la misma tabla.
- Las dos referencias (fk) serán la primary key.



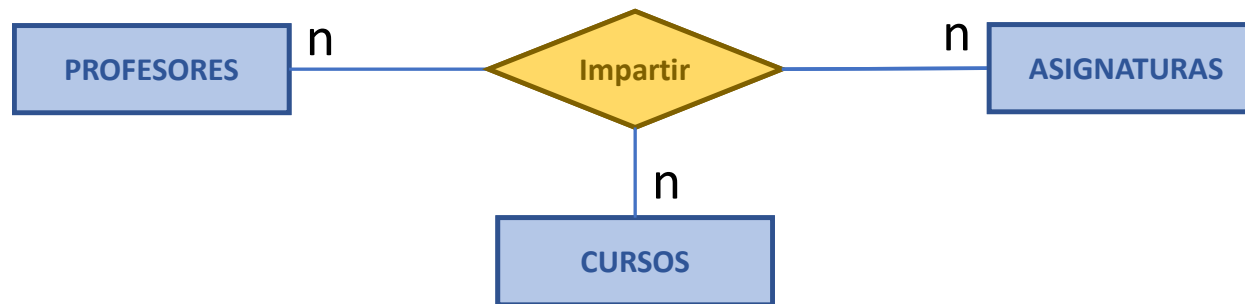
# Relaciones reflexivas (n-n)



```
CREATE TABLE `aeropuertos` (  
  `id_aeropuerto` int NOT NULL,  
  `aeropuerto` varchar(45) NOT NULL,  
  `otros_datos_aeropuerto` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id_aeropuerto`),  
  UNIQUE KEY `aeropuerto_UNIQUE` (`aeropuerto`)  
);
```

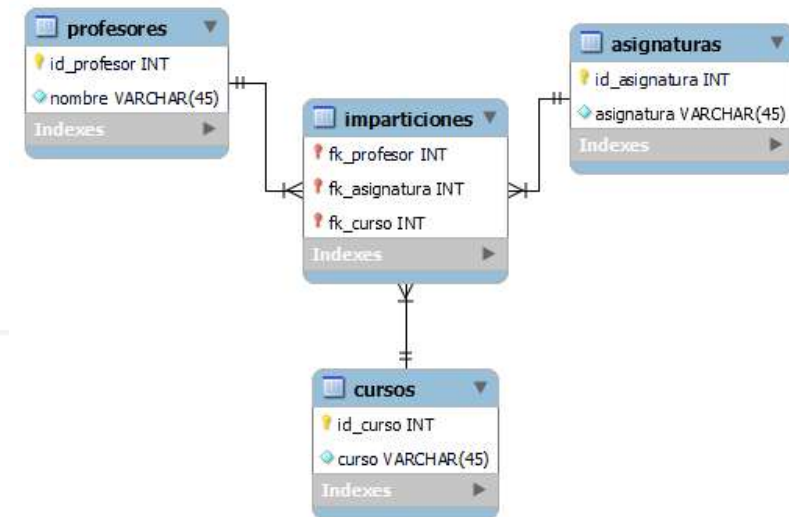
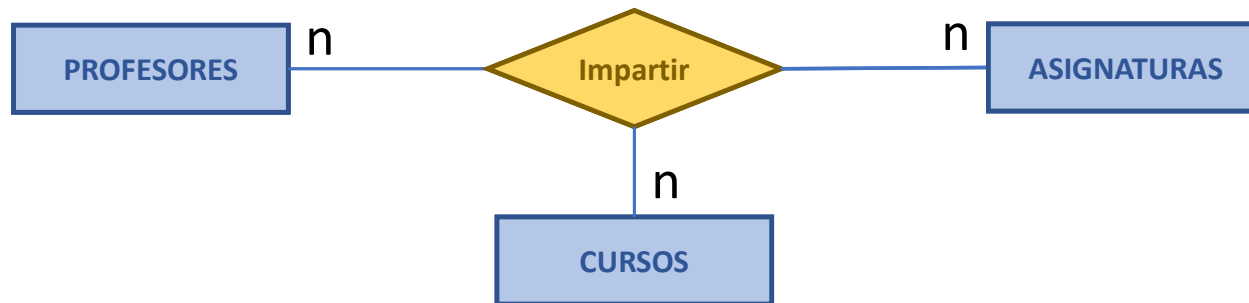
```
CREATE TABLE `conexiones` (  
  `fk_aeropuerto_1` int NOT NULL,  
  `fk_aeropuerto_2` int NOT NULL,  
  PRIMARY KEY (`fk_aeropuerto_1`, `fk_aeropuerto_2`),  
  KEY `conexion_aeropuerto_2_idx` (`fk_aeropuerto_2`),  
  CONSTRAINT `conexion_aeropuerto_1` FOREIGN KEY (`fk_aeropuerto_1`)  
    REFERENCES `aeropuertos` (`id_aeropuerto`),  
  CONSTRAINT `conexion_aeropuerto_2` FOREIGN KEY (`fk_aeropuerto_2`)  
    REFERENCES `aeropuertos` (`id_aeropuerto`)  
);
```

# Relaciones ternarias – n:m:p



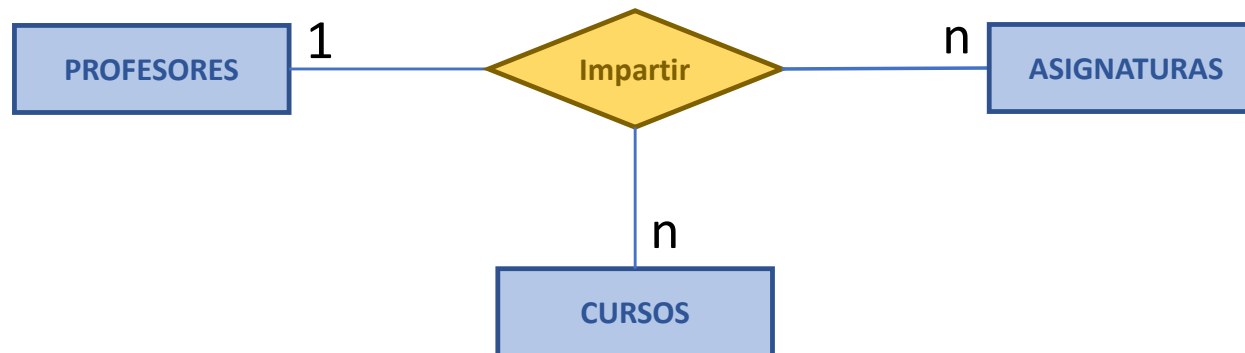
- Cada entidad genera una tabla.
- La relación genera una tabla
  - Contiene una FK a cada entidad de la relación
  - Las tres FKs conforman la PK compuesta de la nueva tabla

# Relaciones ternarias – n:m:p



```
CREATE TABLE `imparticiones` (  
  `fk_profesor` int NOT NULL,  
  `fk_asignatura` int NOT NULL,  
  `fk_curso` int NOT NULL,  
  PRIMARY KEY (`fk_profesor`,`fk_asignatura`,`fk_curso`),  
  CONSTRAINT `imp_asignaturas` FOREIGN KEY (`fk_asignatura`) REFERENCES `asignaturas` (`id_asignatura`),  
  CONSTRAINT `imp_cursos` FOREIGN KEY (`fk_curso`) REFERENCES `cursos` (`id_curso`),  
  CONSTRAINT `imp_profesores` FOREIGN KEY (`fk_profesor`) REFERENCES `profesores` (`id_profesor`)  
);
```

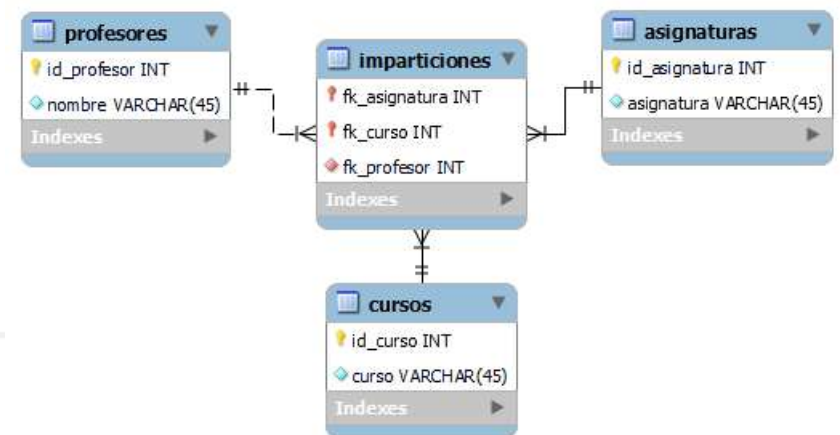
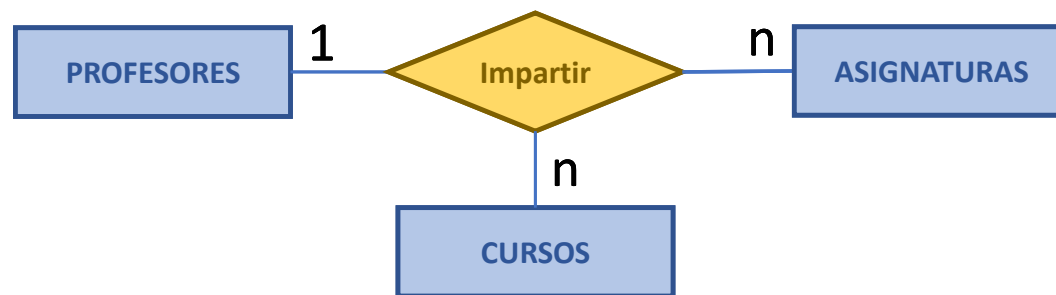
# Relaciones ternarias – n:m:1



- Cada entidad genera una tabla.
- La relación genera una tabla
  - Contiene una FK a cada entidad de la relación
  - Las dos FKs de los lados a muchos conforman la PK compuesta de la nueva tabla
  - La FK que referencia al lado 1 debe ser NOT NULL

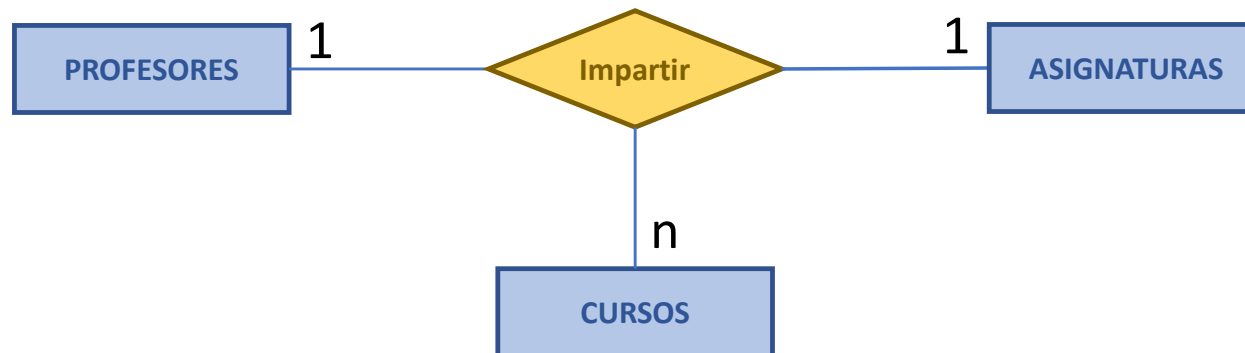


# Relaciones ternarias – n:m:1



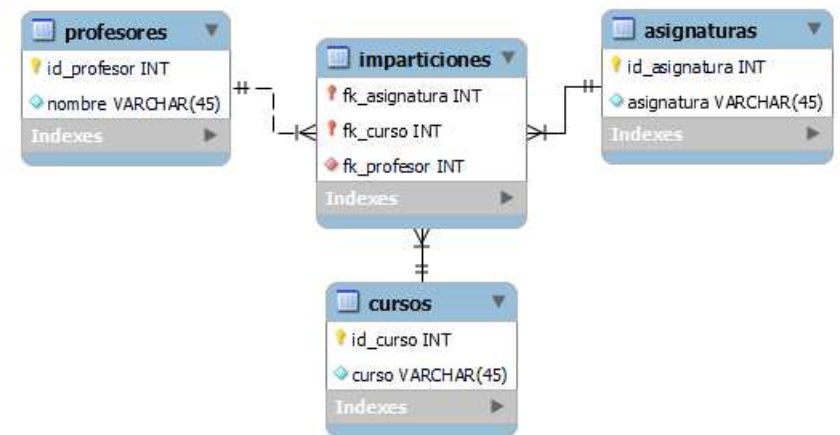
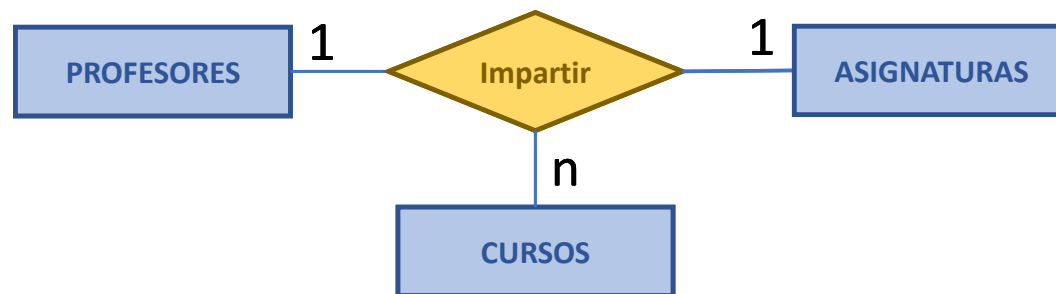
```
CREATE TABLE `imparticiones` (  
  `fk_asignatura` int NOT NULL,  
  `fk_curso` int NOT NULL,  
  `fk_profesor` int NOT NULL,  
  PRIMARY KEY (`fk_asignatura`,`fk_curso`),  
  CONSTRAINT `imp_asignaturas` FOREIGN KEY (`fk_asignatura`) REFERENCES `asignaturas` (`id_asignatura`),  
  CONSTRAINT `imp_cursos` FOREIGN KEY (`fk_curso`) REFERENCES `cursos` (`id_curso`),  
  CONSTRAINT `imp_profesores` FOREIGN KEY (`fk_profesor`) REFERENCES `profesores` (`id_profesor`)  
);
```

# Relaciones ternarias – n:1:1



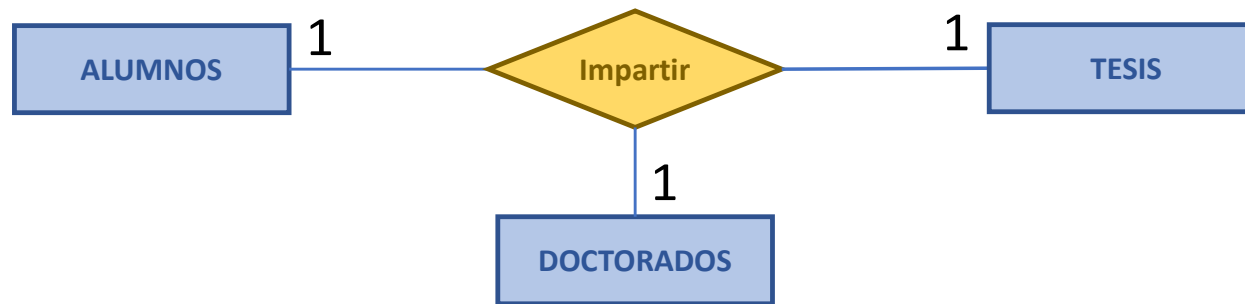
- Cada entidad genera una tabla.
- La relación genera una tabla
  - Contiene una FK a cada entidad de la relación
  - Existirán dos claves candidatas compuestas por 2 FK, las que combinan el lado a n con cada uno de los otros.
  - Elegida una como PK, la otra clave candidata debe ser NOT NULL y UNIQUE.

# Relaciones ternarias – n:1:1



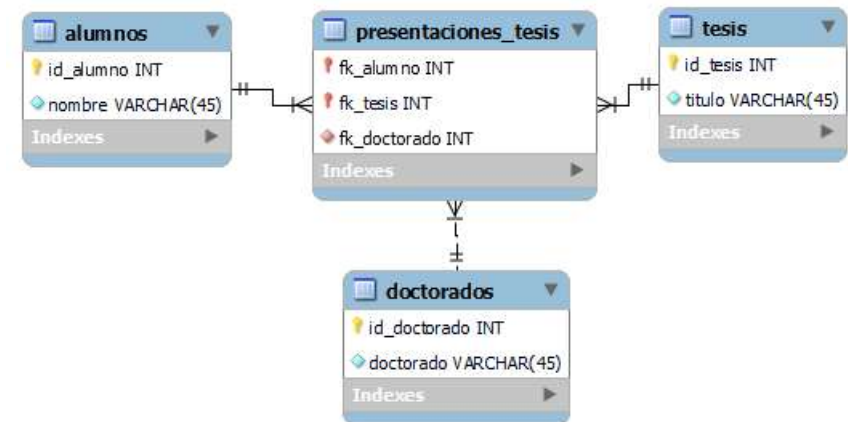
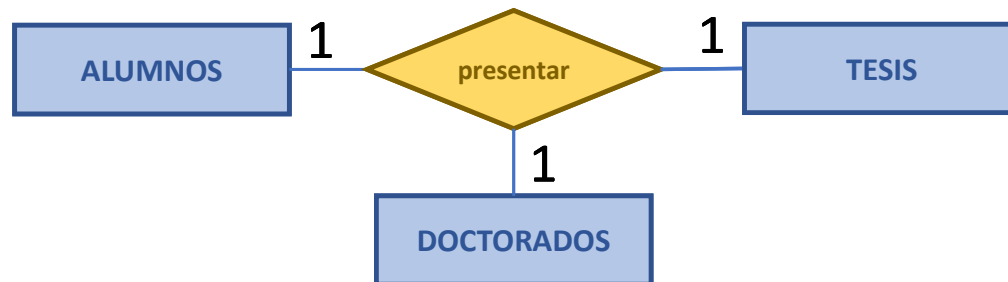
```
CREATE TABLE `imparticiones` (  
  `fk_asignatura` int NOT NULL,  
  `fk_curso` int NOT NULL,  
  `fk_profesor` int NOT NULL,  
  PRIMARY KEY (`fk_asignatura`,`fk_curso`),  
  CONSTRAINT `imp_asignaturas` FOREIGN KEY (`fk_asignatura`) REFERENCES `asignaturas` (`id_asignatura`),  
  CONSTRAINT `imp_cursos` FOREIGN KEY (`fk_curso`) REFERENCES `cursos` (`id_curso`),  
  CONSTRAINT `imp_profesores` FOREIGN KEY (`fk_profesor`) REFERENCES `profesores` (`id_profesor`),  
  UNIQUE KEY (`fk_profesor`,`fk_curso`)  
);
```

# Relaciones ternarias – 1:1:1



- Cada entidad genera una tabla.
- La relación genera una tabla
  - Contiene una FK a cada entidad de la relación
  - Existirán tres claves candidatas compuestas por las combinaciones de 2 FK.
  - Elegida una como PK, las otras dos claves candidata debe ser NOT NULL y UNIQUE.

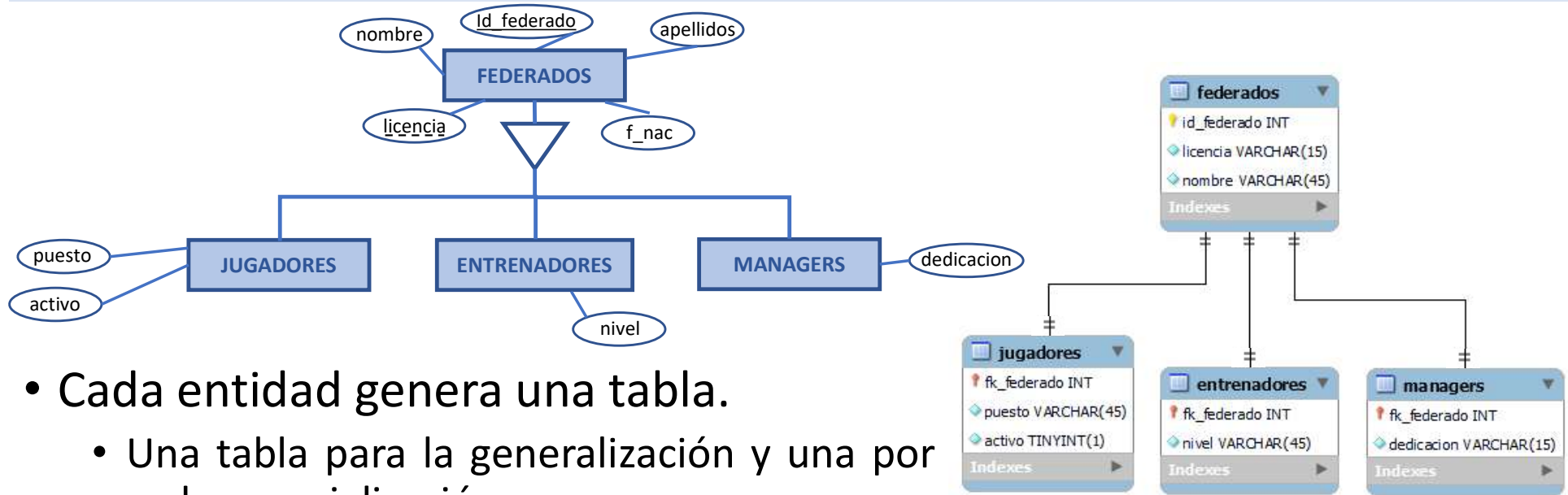
# Relaciones ternarias – 1:1:1



```
CREATE TABLE `presentaciones_tesis` (  
  `fk_alumno` int NOT NULL,  
  `fk_tesis` int NOT NULL,  
  `fk_doctorado` int NOT NULL,  
  PRIMARY KEY (`fk_alumno`,`fk_tesis`),  
  CONSTRAINT `p_alumnos` FOREIGN KEY (`fk_alumno`) REFERENCES `alumnos` (`id_alumno`),  
  CONSTRAINT `p_doctorados` FOREIGN KEY (`fk_doctorado`) REFERENCES `doctorados` (`id_doctorado`),  
  CONSTRAINT `p_tesis` FOREIGN KEY (`fk_tesis`) REFERENCES `tesis` (`id_tesis`),  
  UNIQUE KEY (`fk_alumno`,`fk_doctorado`),  
  UNIQUE KEY (`fk_tesis`,`fk_doctorado`)  
);
```

# Relaciones de Herencia (IsA)

## Estrategia: Tablas relacionadas



- Cada entidad genera una tabla.
  - Una tabla para la generalización y una por cada especialización.
  - La tabla generalización tendrá los atributos en común y las otras sólo los propios; y una FK a la de generalización y será su PK.
- ¿Es total o parcial? - ¿Es con solapamiento o sin solapamiento?

# Relaciones de Herencia (IsA)

## Estrategia: Tablas relacionadas



- Cada entidad genera una tabla.
  - Una tabla para la generalización y una por cada especialización.
  - La tabla generalización tendrá los atributos en común y las otras sólo los propios; y una FK a la de generalización y será su PK.
- NO se puede controlar ni la totalidad ni la exclusividad



# Relaciones de Herencia (IsA)

## Estrategia: Tablas relacionadas

```
CREATE TABLE `federados` (  
  `id_federado` int NOT NULL,  
  `licencia` varchar(15) NOT NULL,  
  `nombre` varchar(45) NOT NULL,  
  PRIMARY KEY (`id_federado`),  
  UNIQUE KEY `licencia_UNIQUE` (`licencia`)  
);
```

```
CREATE TABLE `jugadores` (  
  `fk_federado` int NOT NULL,  
  `puesto` varchar(45) NOT NULL,  
  `activo` tinyint(1) NOT NULL,  
  PRIMARY KEY (`fk_federado`),  
  CONSTRAINT `jug_fed` FOREIGN KEY (`fk_federado`)  
    REFERENCES `federados` (`id_federado`)  
);
```

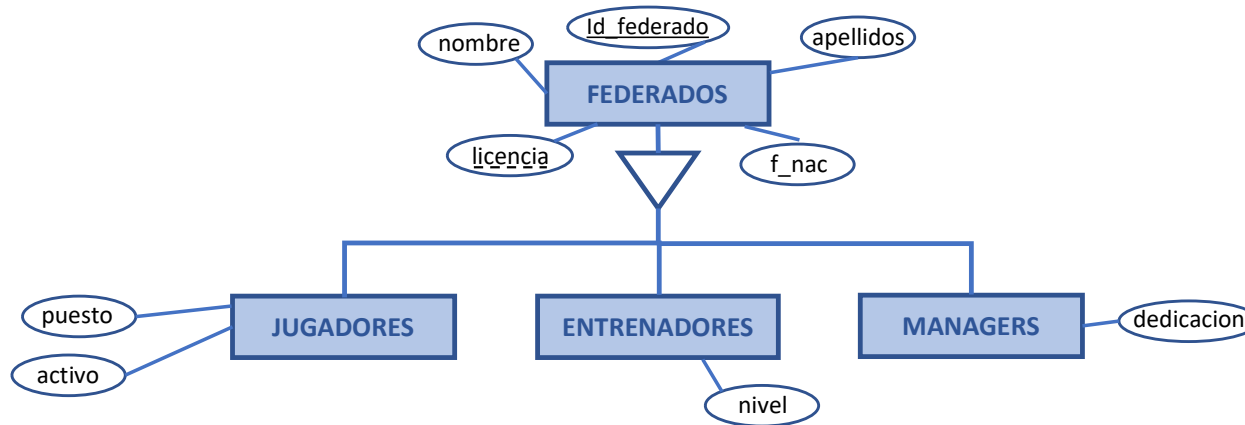
```
CREATE TABLE `entrenadores` (  
  `fk_federado` int NOT NULL,  
  `nivel` varchar(45) NOT NULL,  
  PRIMARY KEY (`fk_federado`),  
  CONSTRAINT `ent_fed` FOREIGN KEY (`fk_federado`)  
    REFERENCES `federados` (`id_federado`)  
);
```

```
CREATE TABLE `managers` (  
  `fk_federado` int NOT NULL,  
  `dedicacion` varchar(15) NOT NULL,  
  PRIMARY KEY (`fk_federado`),  
  CONSTRAINT `man_fed` FOREIGN KEY (`fk_federado`)  
    REFERENCES `federados` (`id_federado`)  
);
```



# Relaciones de Herencia (IsA)

## Estrategia: Una sola tabla



federados	
id_federado	INT
licencia	VARCHAR(15)
nombre	VARCHAR(45)
puesto	VARCHAR(45)
activo	TINYINT
nivel	VARCHAR(45)
dedicacion	VARCHAR(15)
tipo_federado	VARCHAR(3)
Indexes	

- Una sola tabla para toda la familia
  - Debe contener todos los atributos en común, los de cada especialización y un discriminante.
  - Se deben generar checks con controles indicando qué atributos deben ser null y cuáles no; en base al valor del discriminante.

# Relaciones de Herencia (IsA)

## Estrategia: Una sola tabla

```
CREATE TABLE `federados` (  
  `id_federado` int NOT NULL,  
  `licencia` varchar(15) NOT NULL,  
  `nombre` varchar(45) NOT NULL,  
  `puesto` varchar(45) DEFAULT NULL,  
  `activo` tinyint DEFAULT NULL,  
  `nivel` varchar(45) DEFAULT NULL,  
  `dedicacion` varchar(15) DEFAULT NULL,  
  `tipo_federado` varchar(3) NOT NULL CHECK(tipo_federado IN ('J', 'E', 'M')),  
  PRIMARY KEY (`id_federado`),  
  UNIQUE KEY `licencia_UNIQUE` (`licencia`),  
  CHECK ((tipo_federado = 'J' AND puesto IS NOT NULL AND activo IS NOT NULL AND nivel IS NULL AND dedicacion IS NULL) OR  
    (tipo_federado = 'E' AND puesto IS NULL AND activo IS NULL AND nivel IS NOT NULL AND dedicacion IS NULL) OR  
    (tipo_federado = 'M' AND puesto IS NULL AND activo IS NULL AND nivel IS NULL AND dedicacion IS NOT NULL))  
);
```



federados	
id_federado	INT
licencia	VARCHAR(15)
nombre	VARCHAR(45)
puesto	VARCHAR(45)
activo	TINYINT
nivel	VARCHAR(45)
dedicacion	VARCHAR(15)
tipo_federado	VARCHAR(3)
Indexes	

# Relaciones de Herencia (IsA)

## Estrategia: Una sola tabla

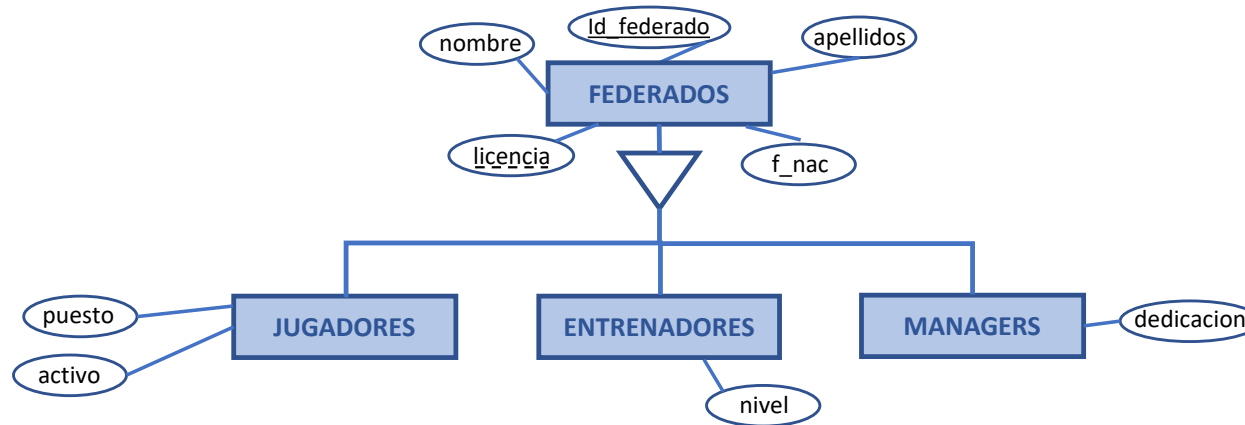
- ¿Cómo podemos controlar si es total o parcial?
  - Agregando un nuevo tipo de federado

```
`tipo_federado` varchar(3) NOT NULL CHECK(tipo_federado IN ('F', 'J', 'E', 'M'));
```
  - Y considerándolo en el check que controla qué atributos pueden o no ser null.
- ¿Cómo podemos controlar si es exclusiva o con solapamiento?
  - Agregando todas las combinaciones posibles como tipos de federado

```
`tipo_federado` varchar(3) NOT NULL CHECK(tipo_federado IN ('J', 'E', 'M', 'JE', 'JM', 'EM', 'JEM'));
```
  - Y considerarlos en el check que controla qué atributos pueden o no ser null.

# Relaciones de Herencia (IsA)

## Estrategia: Tabla independiente por entidad



- Se genera una tabla por cada especialización con los atributos propios y los heredados.
- Si es parcial, también se genera una tabla para la generalización.
- Las tablas no están relacionadas.
- No se aplica herencia y se pierden los controles de integridad referencial.

# Relaciones de Herencia (IsA)

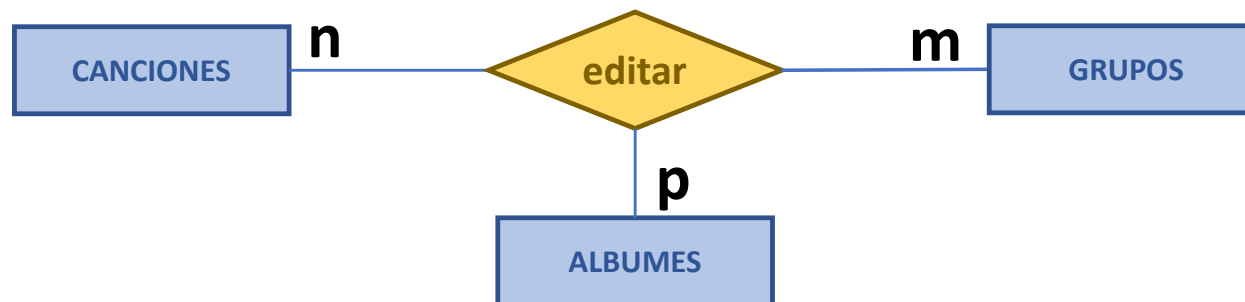
Estrategia: Tabla independiente por entidad



- Se genera una tabla por cada especialización con los atributos propios y los heredados.
- Si es parcial también se genera una tabla para la generalización.
- Las tablas no están relacionadas.
- No se aplica herencia y se pierden los controles de integridad referencial.

# Agregaciones (Modelo ER)

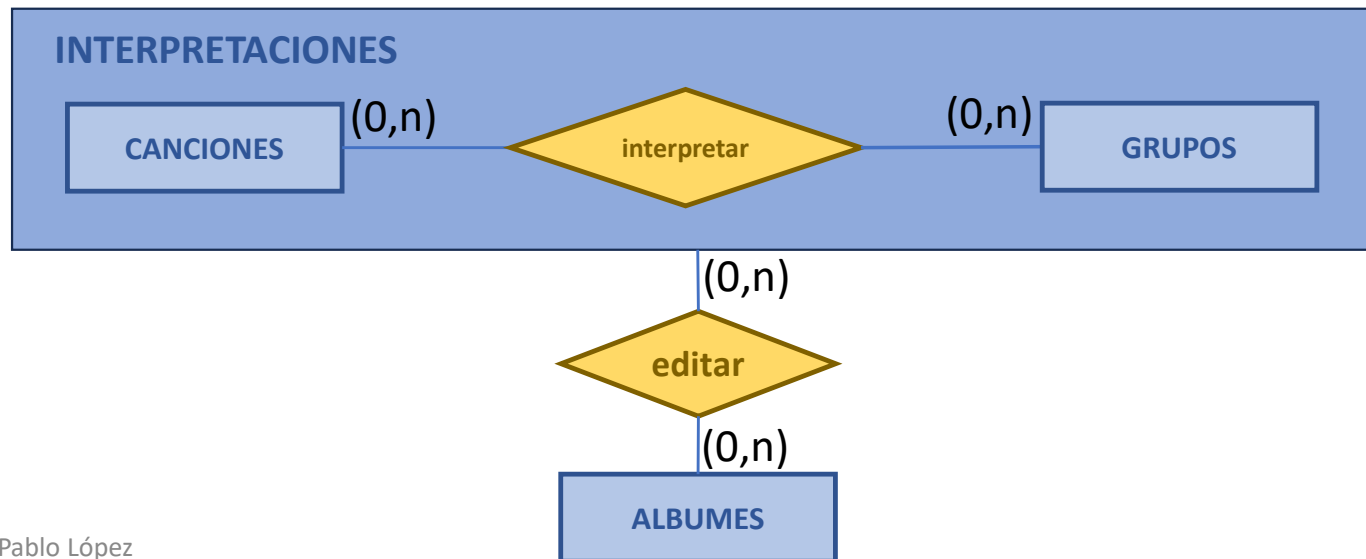
- Tenemos 3 entidades: Canciones, Grupos, Álbumes.
- Una canción contenida en un álbum podría estar interpretada por varios grupos.
- Un grupo en un álbum puede tener muchas canciones.
- Una canción interpretada por un grupo puede estar publicada en muchos álbumes.



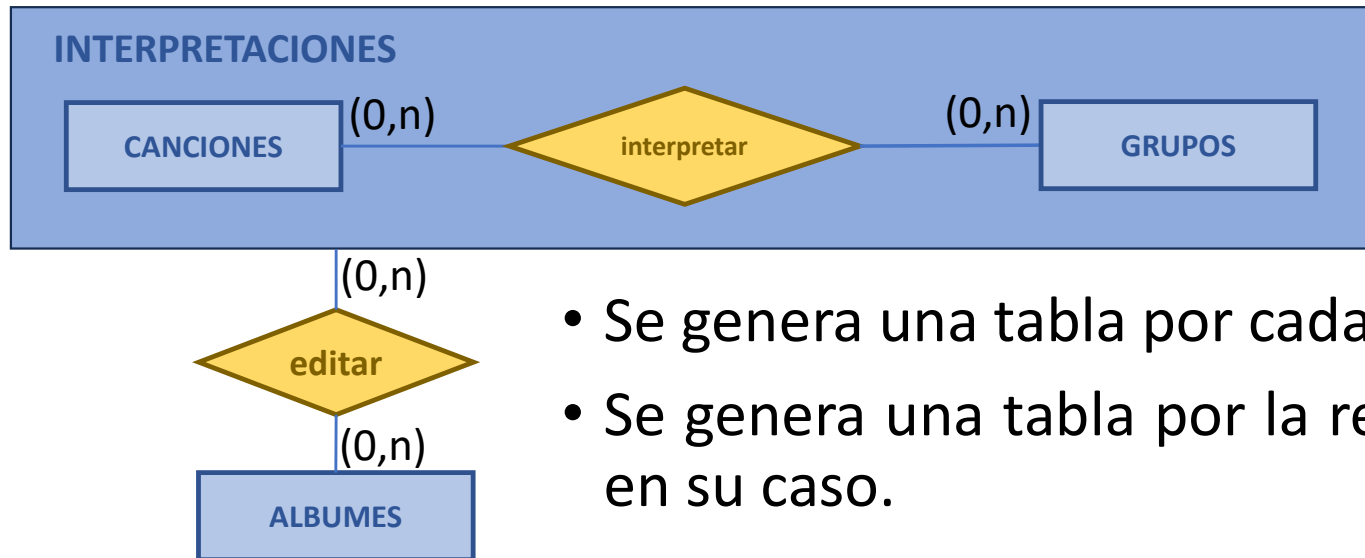
- ¿Y si una canción, interpretada por un grupo no se ha publicado en ningún álbum?, por ejemplo, solo la ha interpretado en conciertos.

# Agregaciones (Modelo ER)

- Necesitamos poder mantener relaciones entre Canciones y Grupos, independientemente que estén o no publicadas en álbumes.
- Ahora tenemos que relacionar esas interpretaciones con álbumes.
- Se crea una agrupación. Es como una entidad, formada por esta relación.
- Finalmente, se relaciona Álbumes con esta nueva entidad



# Agregaciones

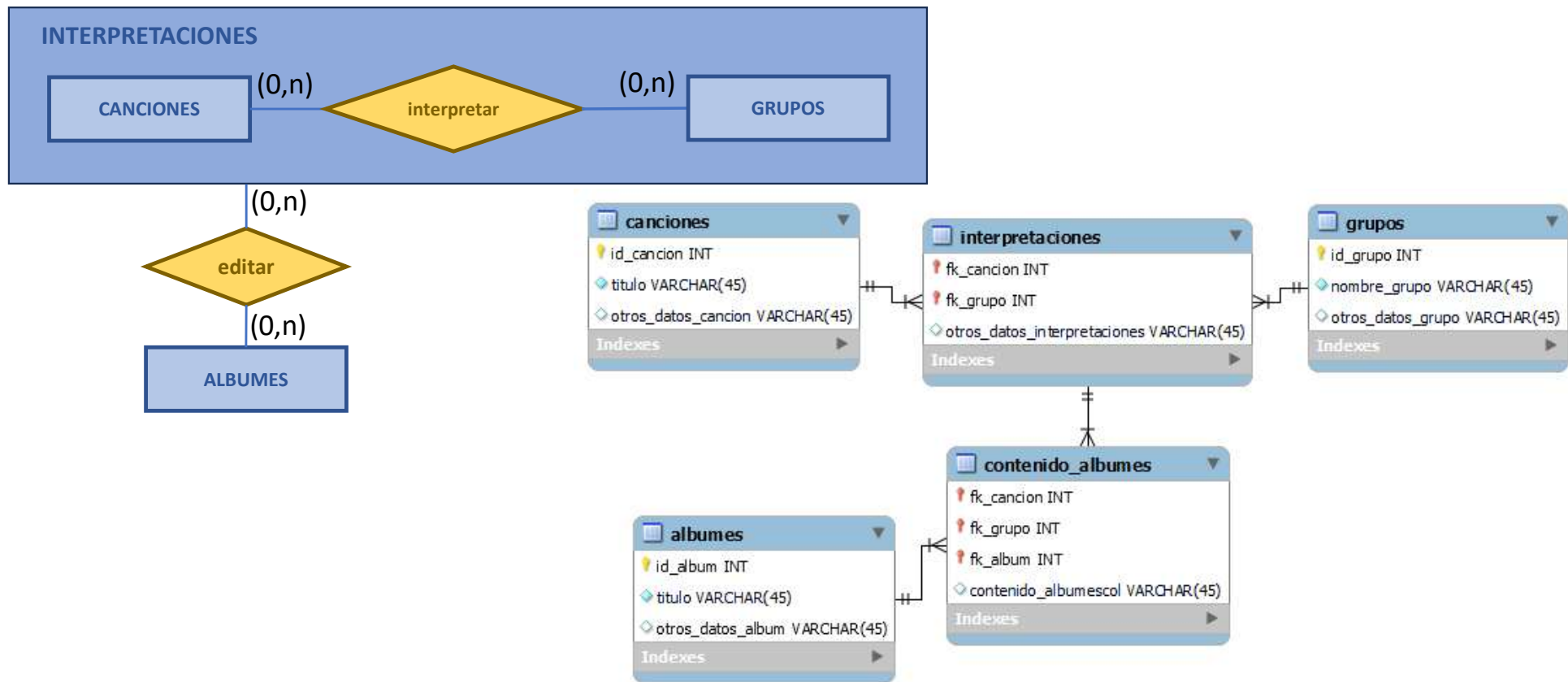


- Se genera una tabla por cada entidad.
- Se genera una tabla por la relación n-n o ternaria en su caso.

- La agregación no genera una tabla sino la representa la tabla de la relación anterior. En esta tabla se incorporan los atributos de la agregación.
- Las relaciones con la interrelación se harán con esta tabla.



# Agregaciones



# Agregaciones

```
CREATE TABLE `canciones` (  
  `id_cancion` int NOT NULL,  
  `titulo` varchar(45) NOT NULL,  
  `otros_datos_cancion` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id_cancion`)  
);
```

```
CREATE TABLE `grupos` (  
  `id_grupo` int NOT NULL,  
  `nombre_grupo` varchar(45) NOT NULL,  
  `otros_datos_grupo` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id_grupo`)  
);
```

```
CREATE TABLE `interpretaciones` (  
  `fk_cancion` int NOT NULL,  
  `fk_grupo` int NOT NULL,  
  `otros_datos_int` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`fk_cancion`, `fk_grupo`),  
  FOREIGN KEY (`fk_cancion`)  
    REFERENCES `canciones` (`id_cancion`),  
  FOREIGN KEY (`fk_grupo`)  
    REFERENCES `grupos` (`id_grupo`)  
);
```

# Agregaciones

```
CREATE TABLE `interpretaciones` (  
  `fk_cancion` int NOT NULL,  
  `fk_grupo` int NOT NULL,  
  `otros_datos_int` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`fk_cancion`,`fk_grupo`),  
  FOREIGN KEY (`fk_cancion`)  
    REFERENCES `canciones` (`id_cancion`),  
  FOREIGN KEY (`fk_grupo`)  
    REFERENCES `grupos` (`id_grupo`)  
);
```

```
CREATE TABLE `albumes` (  
  `id_album` int NOT NULL,  
  `titulo` varchar(45) NOT NULL,  
  `otros_datos_album` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id_album`)  
);
```

```
CREATE TABLE `contenido_albumes` (  
  `fk_cancion` int NOT NULL,  
  `fk_grupo` int NOT NULL,  
  `fk_album` int NOT NULL,  
  `contenido_albumescol` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`fk_cancion`,`fk_grupo`,`fk_album`),  
  FOREIGN KEY (`fk_album`) REFERENCES `albumes` (`id_album`),  
  FOREIGN KEY (`fk_cancion`,`fk_grupo`)  
    REFERENCES `interpretaciones` (`fk_cancion`,`fk_grupo`)  
);
```