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

MODELO RELACIONAL

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

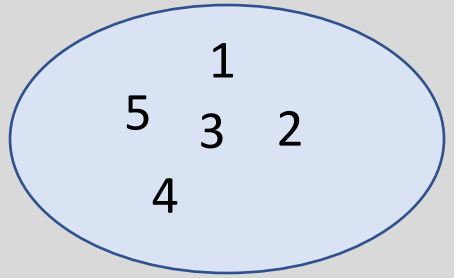
$A = \{1, 2, 3, 4, 5\}$

Implícita

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

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

Diagrama de Venn



Conjunto vacío

$\emptyset$

Inclusión

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

$B = \{5, 2\}$

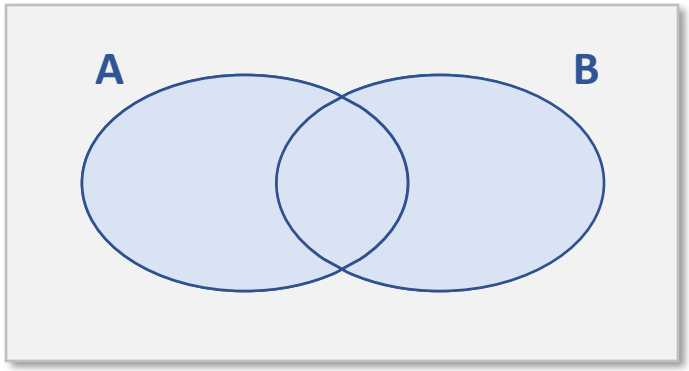
$B \subset A$ o $B \subseteq A$

$A \supset B$ o $A \supseteq B$

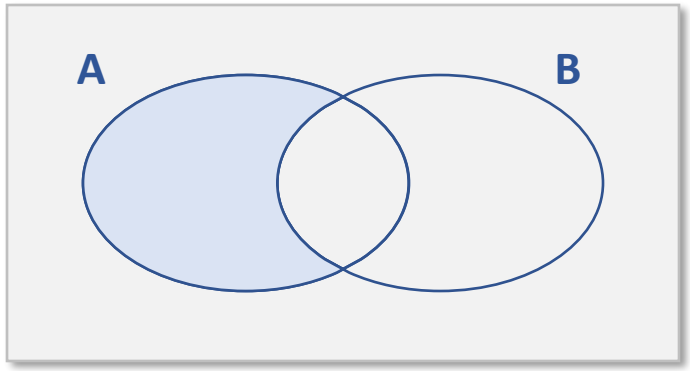
B es un subconjunto de A

Operaciones de Conjuntos

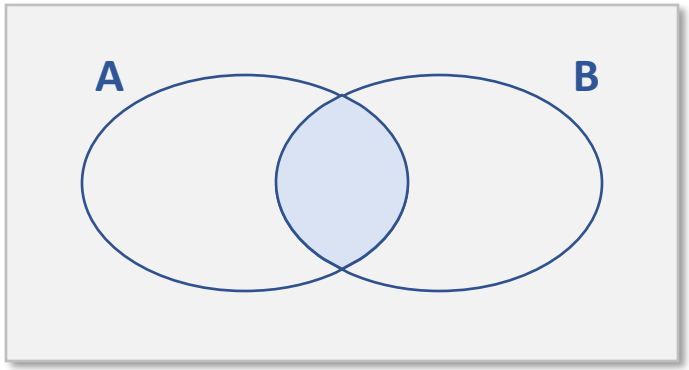
Unión: $A \cup B$



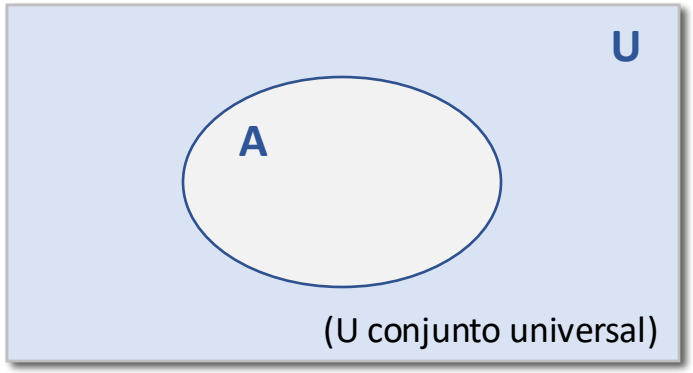
Diferencia: $A - B$ o $A \setminus B$



Intersección: $A \cap B$



Complemento: A^C



Producto cartesiano: $A \times B$

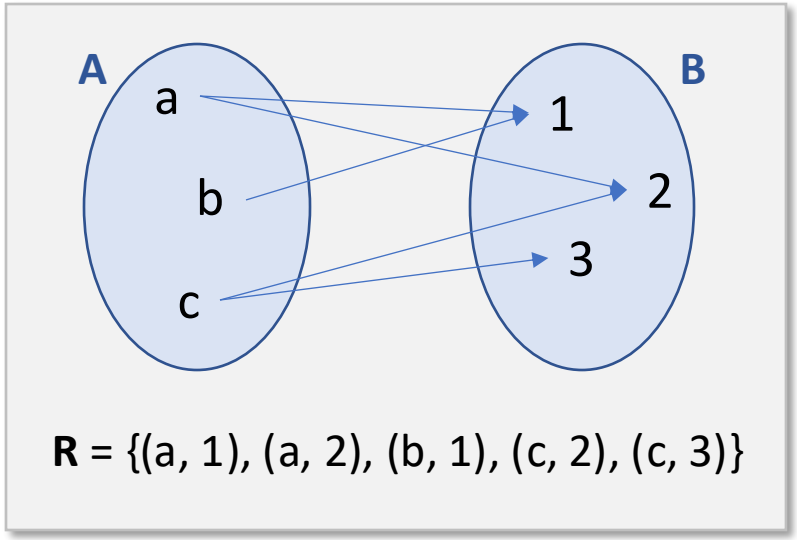
El producto cartesiano de dos conjuntos A y B es el conjunto A × B de todos los pares ordenados (a, b) formados con un primer elemento a perteneciente a A, y un segundo elemento b perteneciente a B.

A = {1, 2}
B = {a, b, c}

A x 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.



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

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

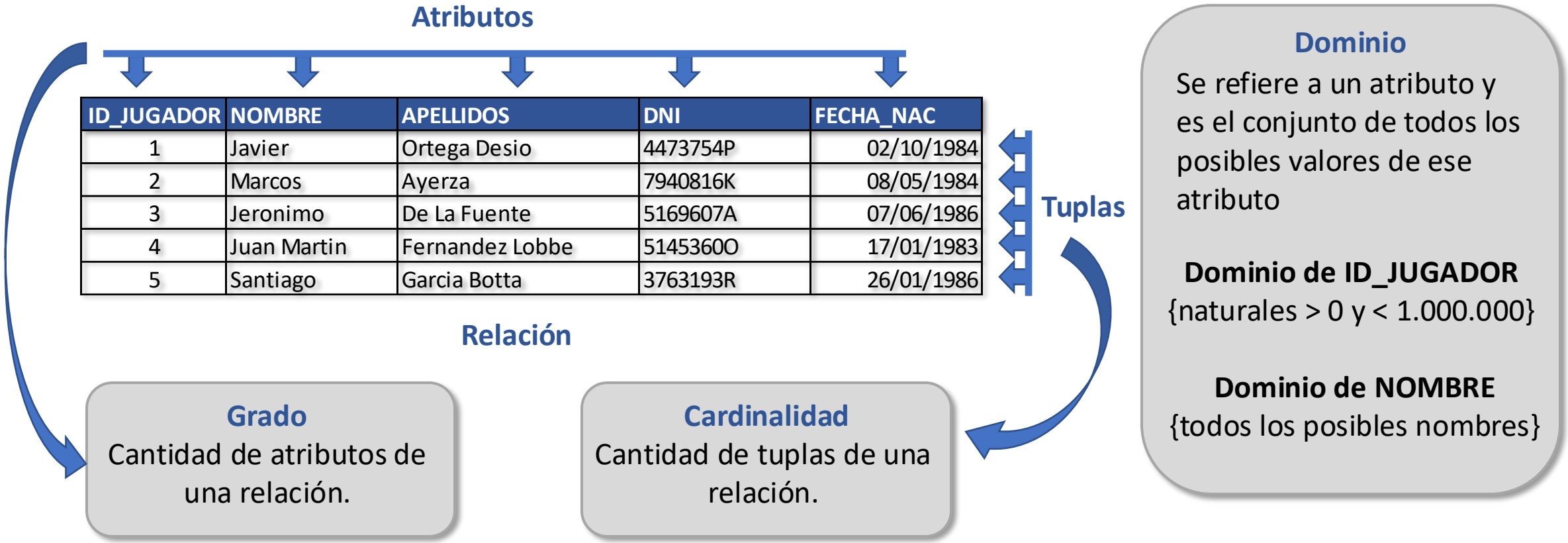
➢n-aria: $R \subseteq A_1 \times A_2 \times A_3 \times \ldots \times A_n$, $R(a_1, a_2, a_3, \ldots, a_n)$

A x 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



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

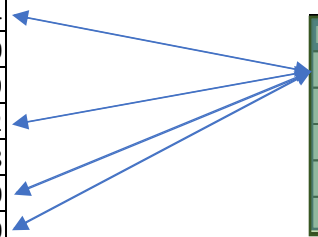
claves candidatas

claves primarias

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

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

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

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 or null -> será verdadero

conjunción (and o ∧)

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

disyunción (or o ∨)

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

negación (not o ¬)

p	not p
true	false
false	true
null	null

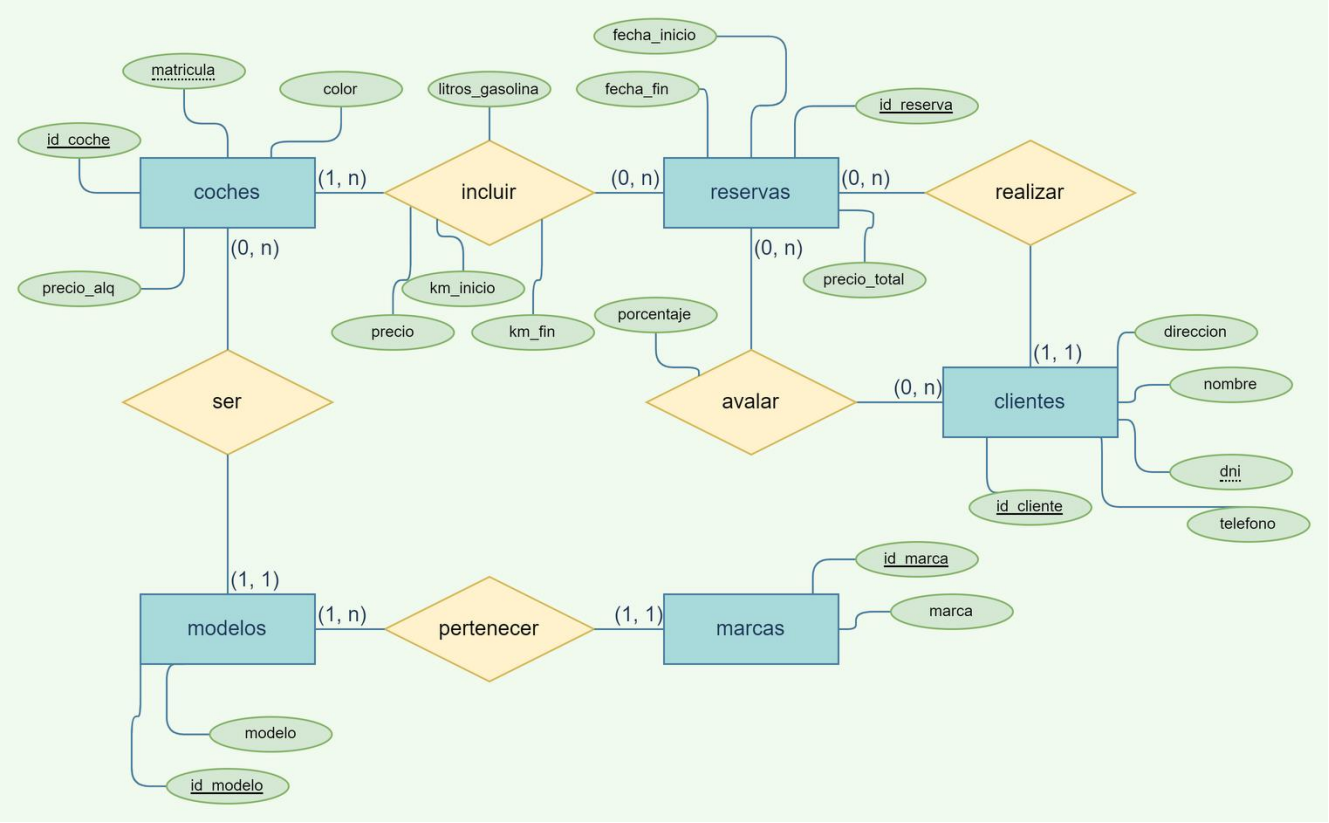
Restricciones de integridad

- 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

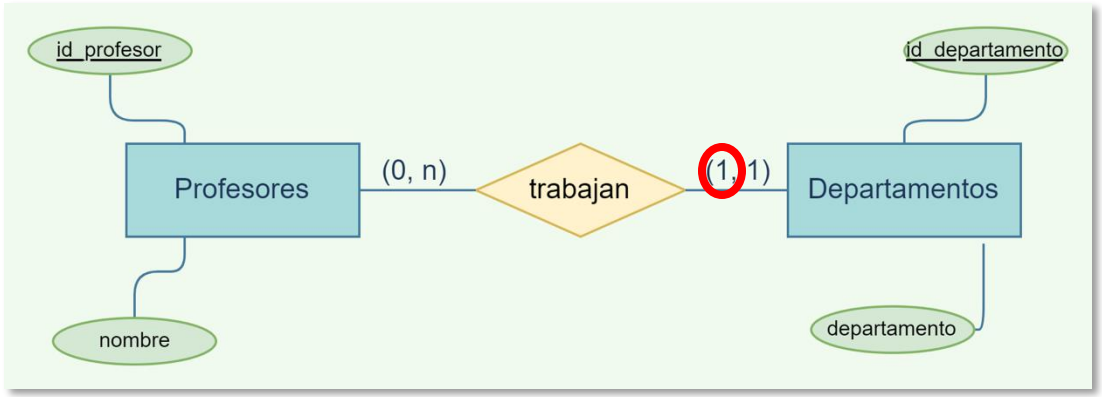
- Restricción de clave primaria (primary key)
  - Toda tabla requiere de una clave primaria, sus valores en una tabla deben ser todos diferentes y ningún atributo de ella puede contener null.
- Restricción de unicidad (unique)
  - Impide que los atributos marcados como “unique” puedan repetirse en distintas 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



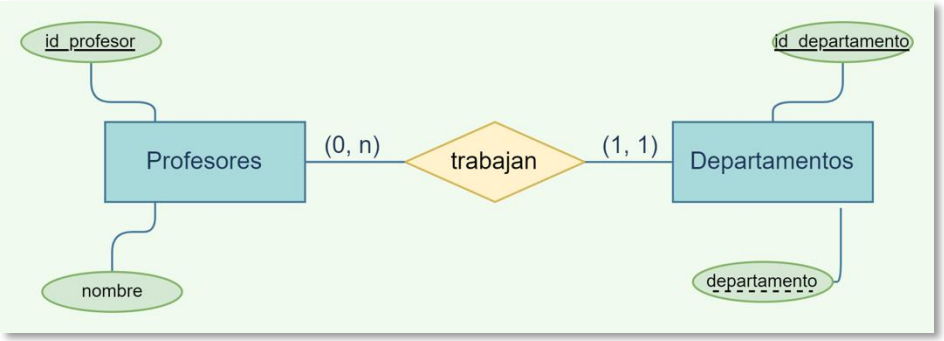
- Por cada entidad se crea una nueva tabla.
- Todos los atributos simples de la entidad pasan a ser atributos de la tabla.
- Los atributos compuestos, se descomponen y cada componente pasa a ser un atributo de la tabla.
- El atributo ID de la entidad será la Primary Key de la tabla.
- Los atributos que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

Relaciones 1-n

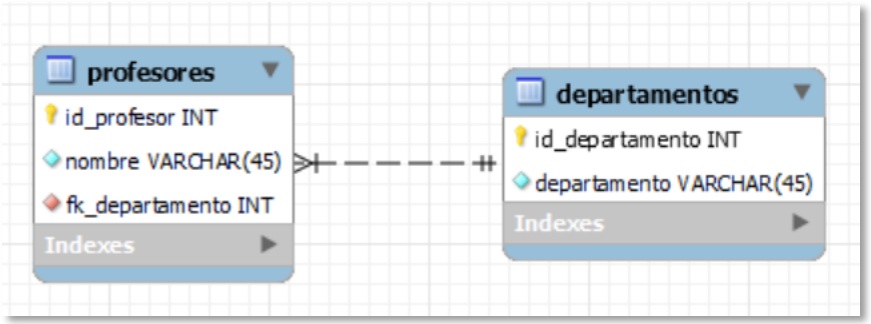


- En la tabla del lado de “n” se añade un atributo que funcionará como Foreign Key a la otra tabla.
- Este atributo será simple o compuesto, tal cual sea la primary key de la tabla que referencia y del mismo tipo de dato.
- Deberá crearse la restricción referencial “constraint” foreign key.
- Finalmente, 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

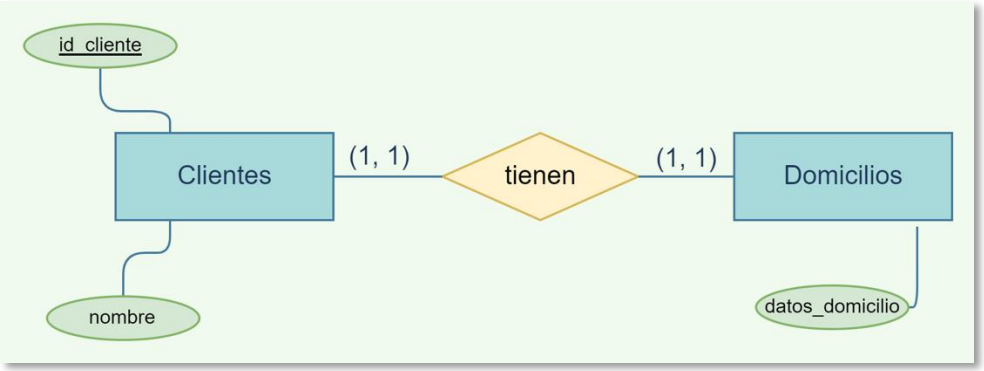



```
CREATE TABLE `departamentos` (
  `id_departamento` int NOT NULL AUTO_INCREMENT,
  `departamento` varchar(45) NOT NULL,
  PRIMARY KEY (`id_departamento`),
  UNIQUE KEY `departamento_UNIQUE` (`departamento`)
);
```



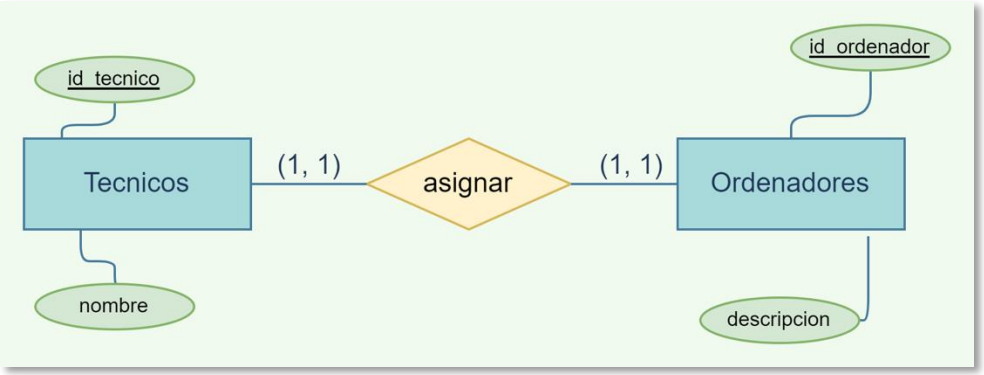

```
CREATE TABLE `profesores` (
  `id_profesor` int NOT NULL AUTO_INCREMENT,
  `nombre` varchar(45) NOT NULL,
  `fk_departamento` int NOT NULL,
  PRIMARY KEY (`id_profesor`),
  CONSTRAINT `profesores_departamentos` FOREIGN KEY (`fk_departamento`)
        references `departamentos` (`id_departamento`)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

Relaciones 1-1 “caso (1,1) – (1,1)”



- Opción 1:

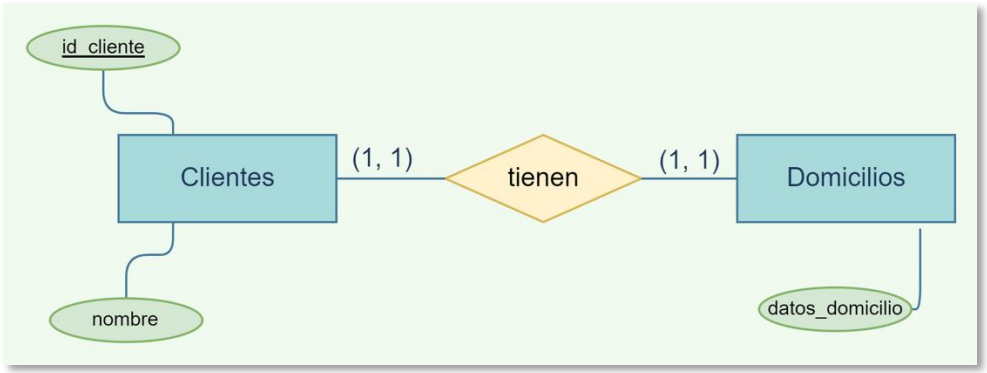
  Crear sólo una tabla con todos los atributos de las dos entidades.

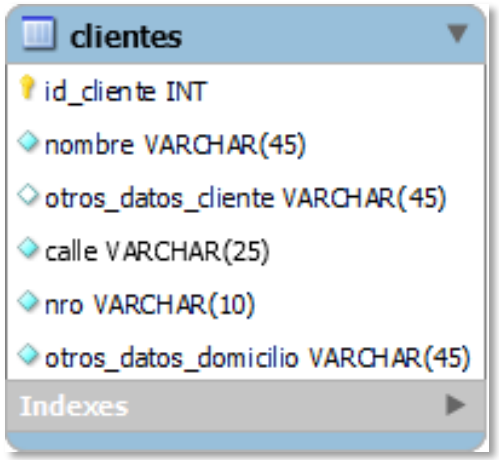


- Opción 2

  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)”



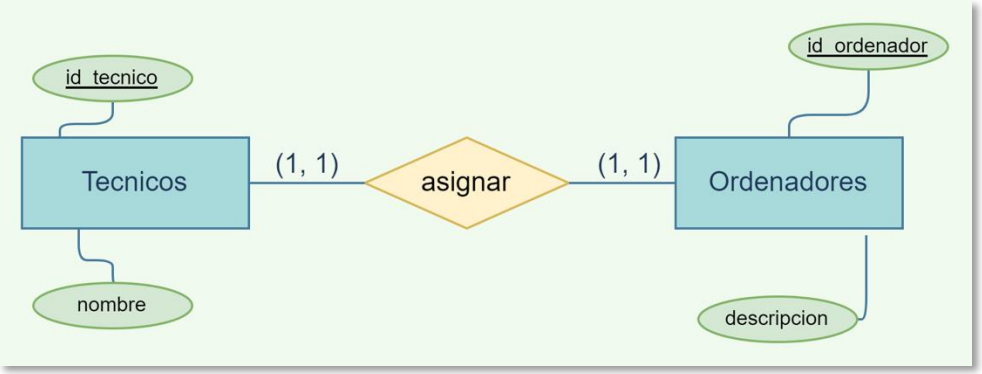


- Opción 1:

  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`)
);
```

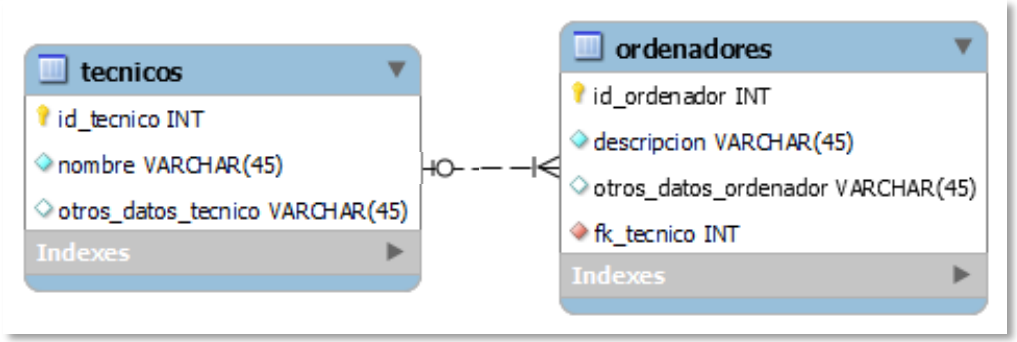
Relaciones 1-1 “caso (1,1) – (1,1)”



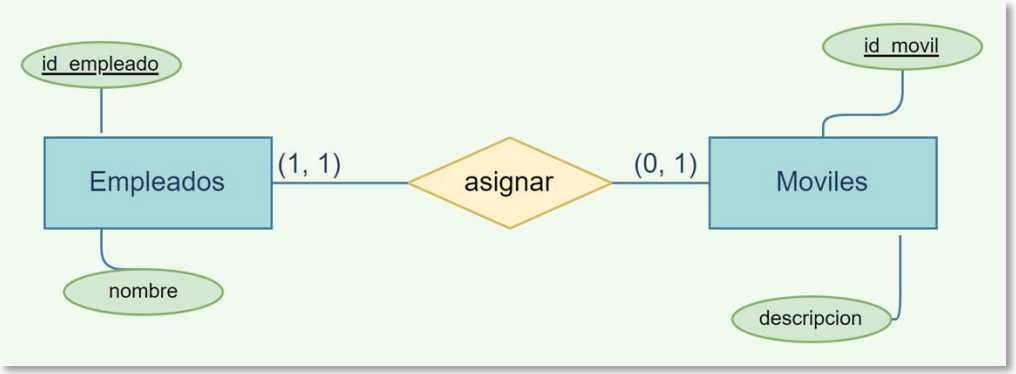

```
CREATE TABLE `ordenadores` (
  `id_ordenador` int NOT NULL,
  `descripcion` varchar(45) NOT NULL,
  `otros_datos_ordenador` varchar(45) DEFAULT NULL,
  `fk_tecnico` int NOT NULL,
  PRIMARY KEY (`id_ordenador`),
  UNIQUE KEY `fk_tecnico_UNIQUE` (`fk_tecnico`),
  CONSTRAINT `ordenadores_tecnicos` FOREIGN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



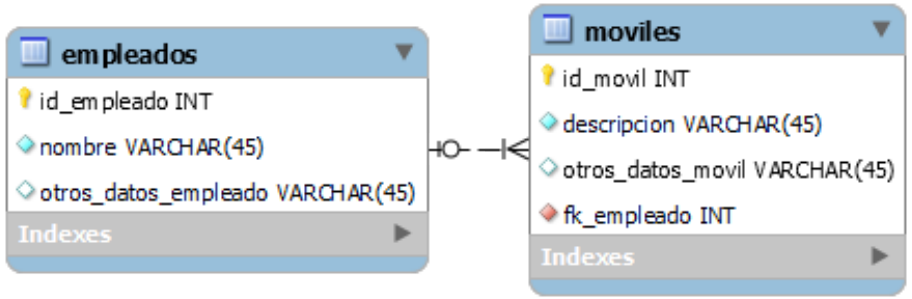
Relaciones 1-1 “caso (1,1) – (0,1)”



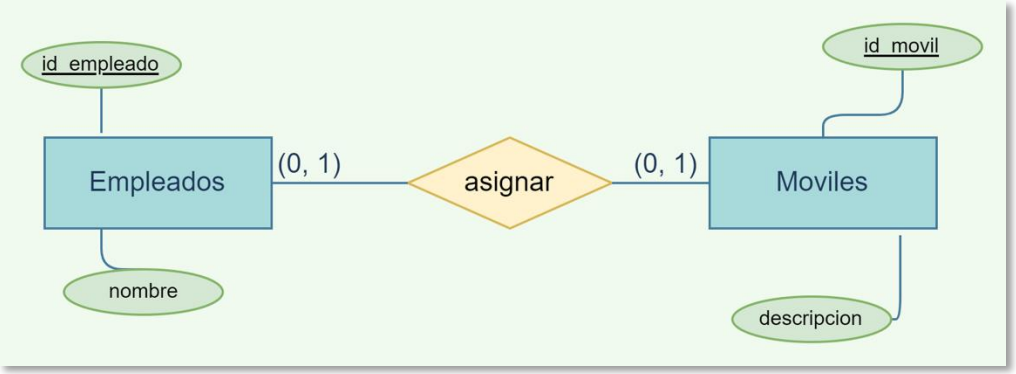
- Se trata como un caso especial de 1-n. Del lado (0,1) se agrega una foreign key y debe ser UNIQUE NOT NULL

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

```
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`)
);
```



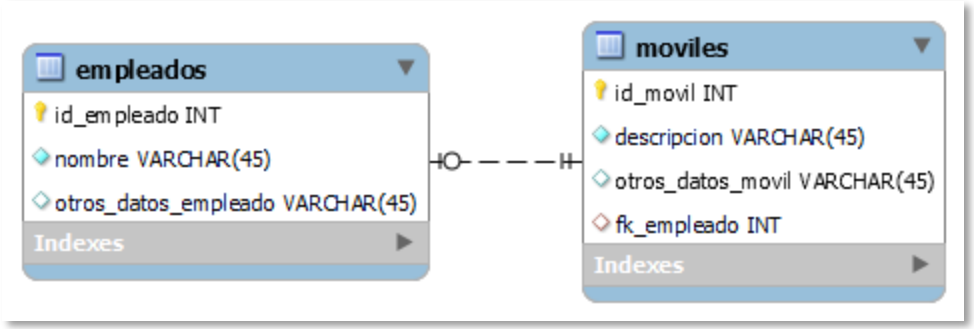
Relaciones 1-1 “caso (0,1) – (0,1)”



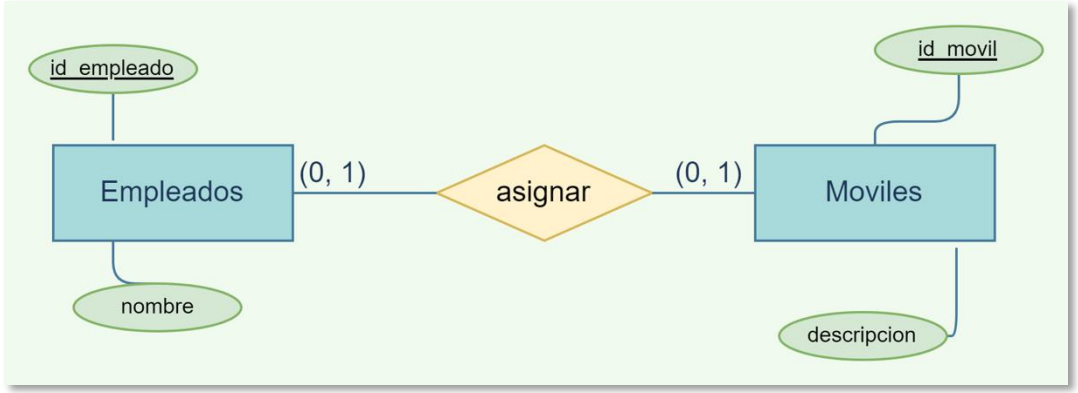
- Se trata como un caso especial de 1-n. Se agrega una foreign key y debe ser sólo UNIQUE

```
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`)
);
```

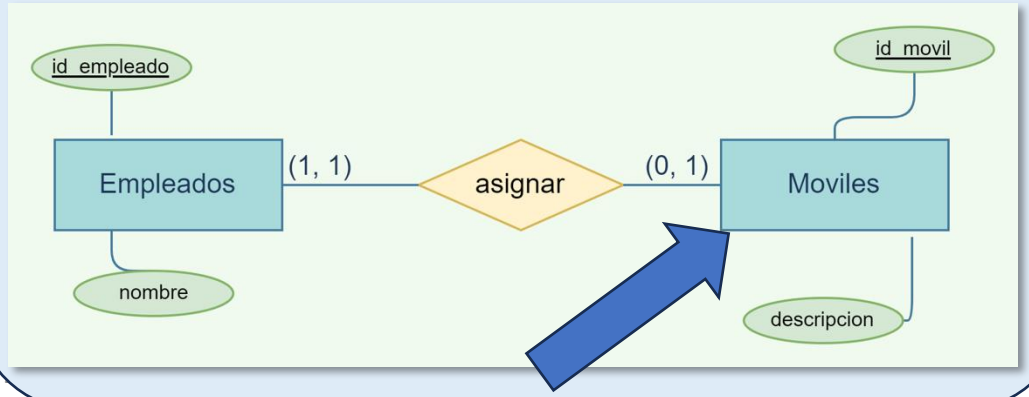
```
CREATE TABLE `empleados` (
  `id_empleado` int NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `otros_datos_empleado` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id_empleado`)
);
```



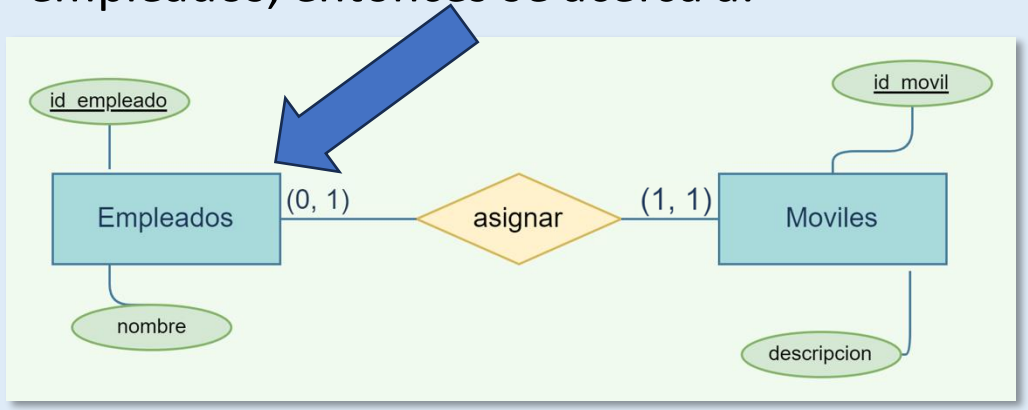
Relaciones 1-1 “caso (0,1) – (0,1)” ¿De qué lado poner la FK?



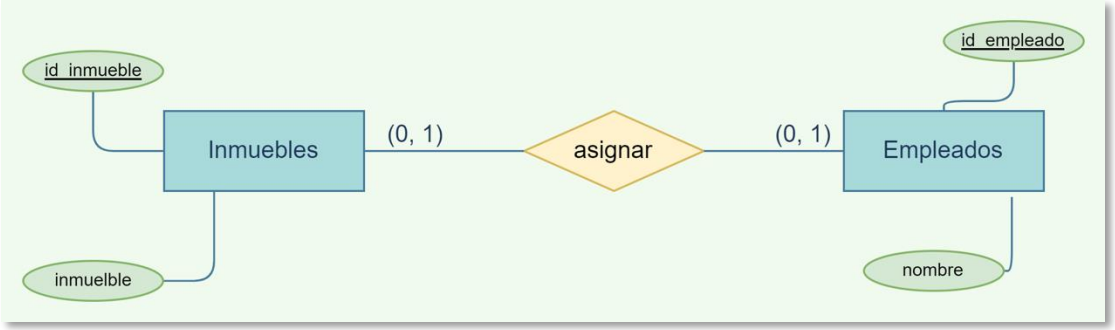
Si casi todos los móviles son entregados a empleados y muchos empleados no tienen móviles, entonces se acerca a:



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

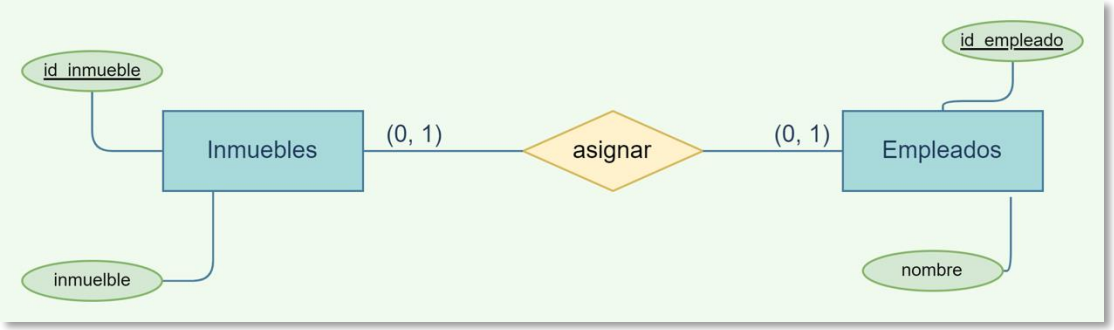


Relaciones 1-1 “caso (0,1) – (0,1)” ¿Y si hay muy pocas interrelaciones?



- 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?

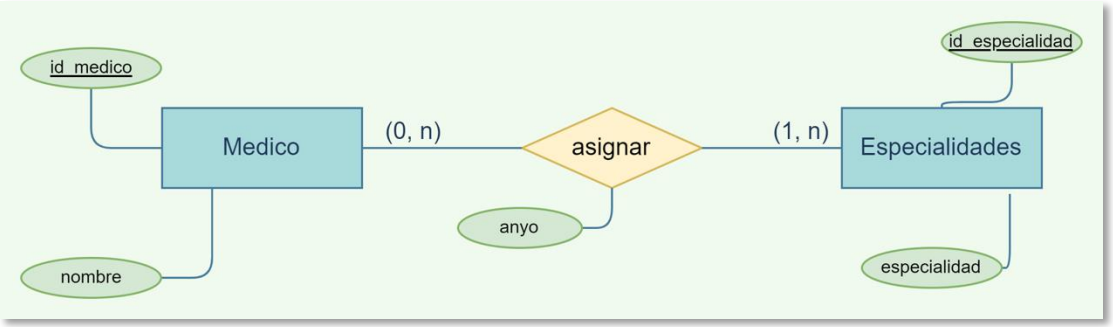



```
CREATE TABLE `empleados` (
  `id_empleado` int NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `otros_datos_empleado` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id_empleado`)
);
```

```
CREATE TABLE `inmuebles_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`)
);
```

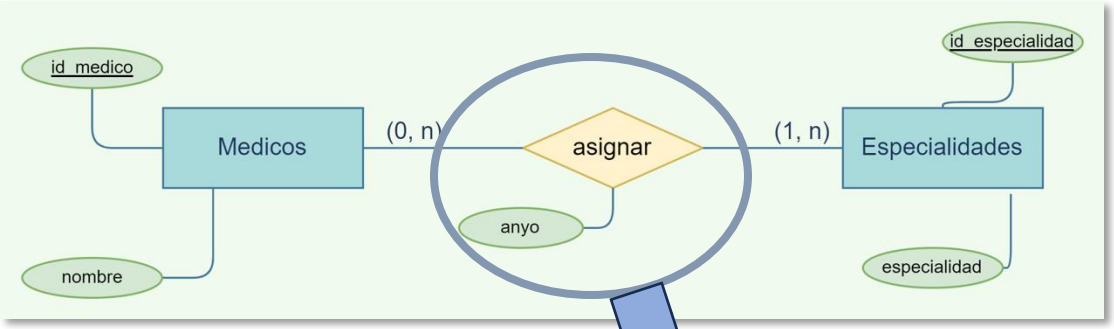
```
CREATE TABLE `inmuebles` (
  `id_inmueble` int NOT NULL,
  `inmueble` varchar(45) NOT NULL,
  `domicilio` varchar(45) NOT NULL,
  PRIMARY KEY (`id_inmueble`)
);
```

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

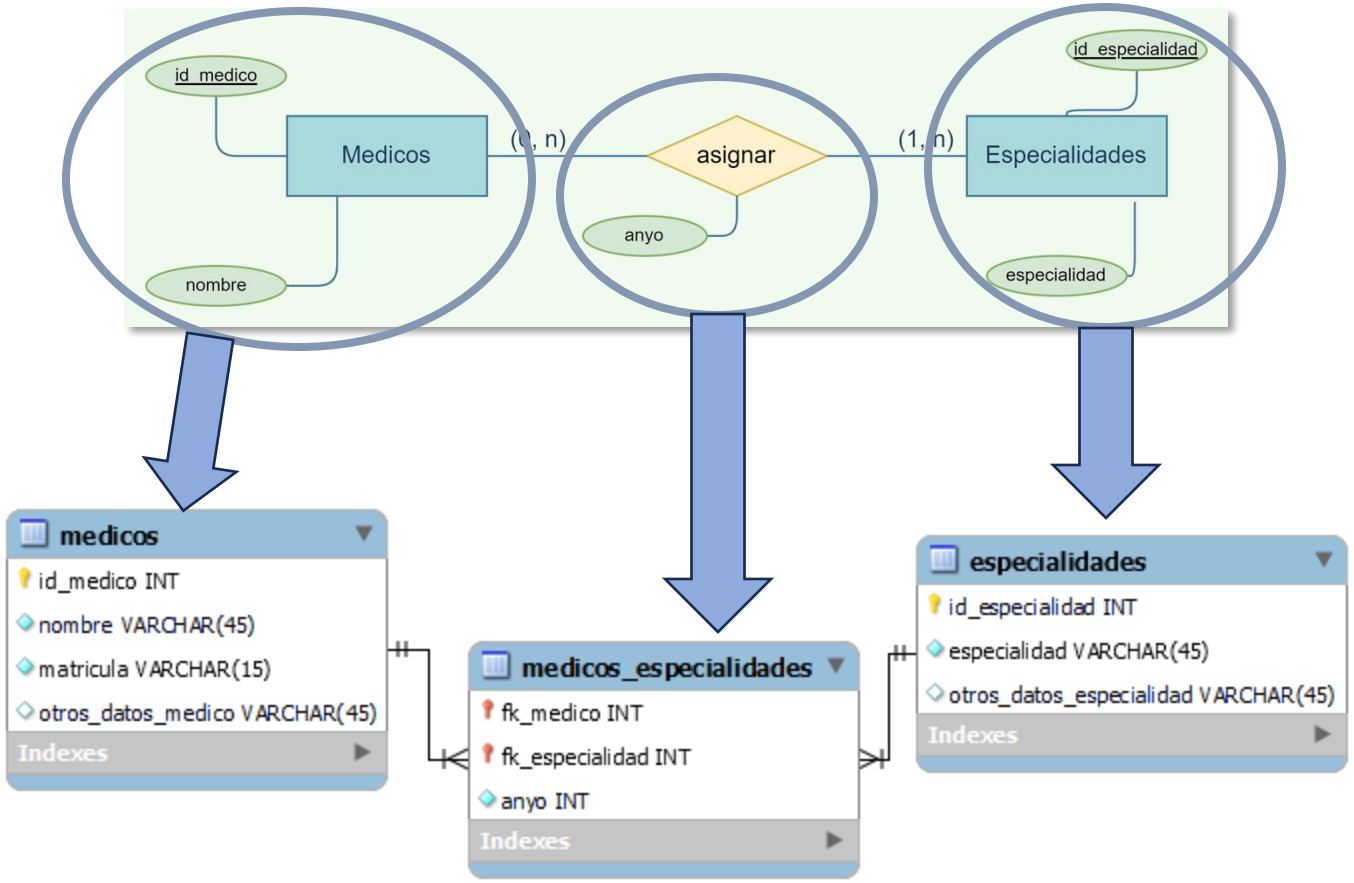



```
CREATE TABLE `medicos_especialidades` (
  `fk_medico` INT NOT NULL,
  `fk_especialidad` INT NOT NULL,
  `anyo` INT NOT NULL,
  PRIMARY KEY (`fk_medico`, `fk_especialidad`),
  CONSTRAINT `medicos_especialidades` FOREIGN KEY (`fk_medico`)
    REFERENCES `000_teoria`.`medicos` (`id_medico`),
  CONSTRAINT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

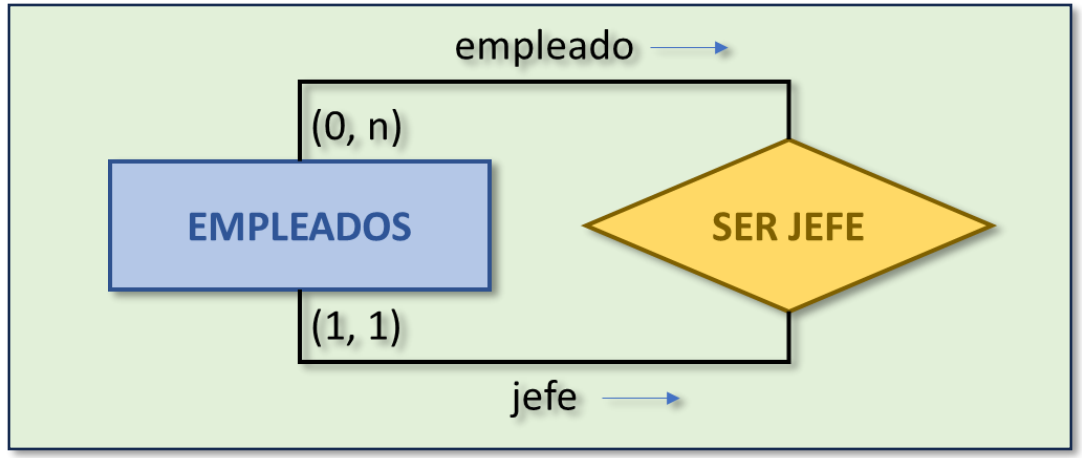
```
CREATE TABLE `medicos` (
  `id_medico` int NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `matricula` varchar(15) NOT NULL,
  `otros_datos_medico` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id_medico`),
  UNIQUE KEY `matricula_UNIQUE` (`matricula`)
);
```

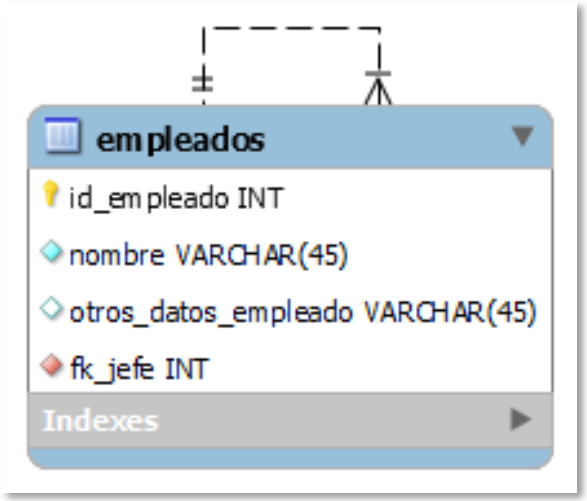
```
CREATE TABLE `especialidades` (
  `id_especialidad` int NOT NULL,
  `especialidad` varchar(45) NOT NULL,
  `otros_datos_especialidad` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id_especialidad`),
  UNIQUE KEY `especialidad_UNIQUE` (`especialidad`)
);
```

Relaciones n-n



Relaciones reflexivas (1-n)

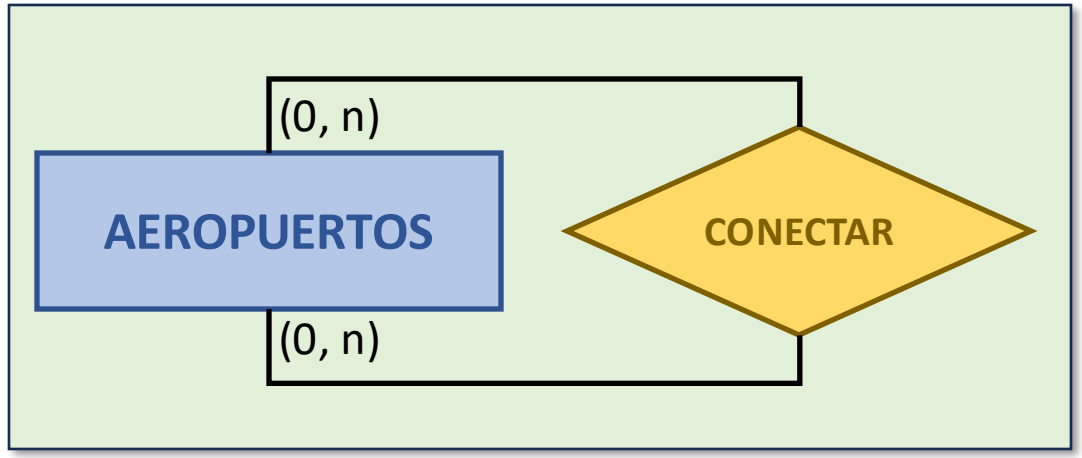


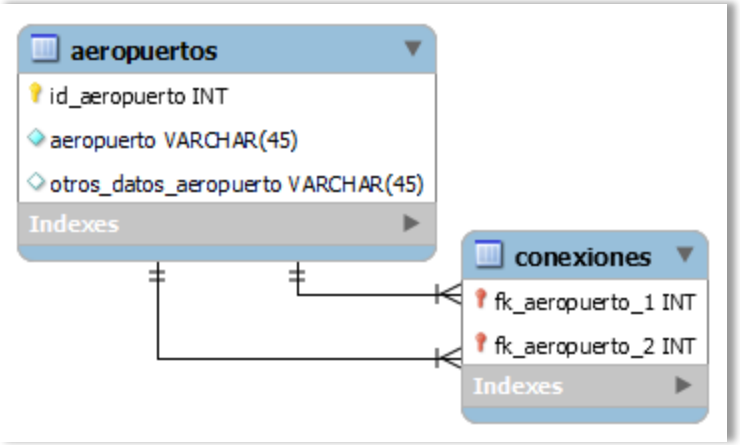


- Se siguen las mismas reglas que una relación 1-n, pero la referencia es a la misma tabla.

```
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`)
);
```

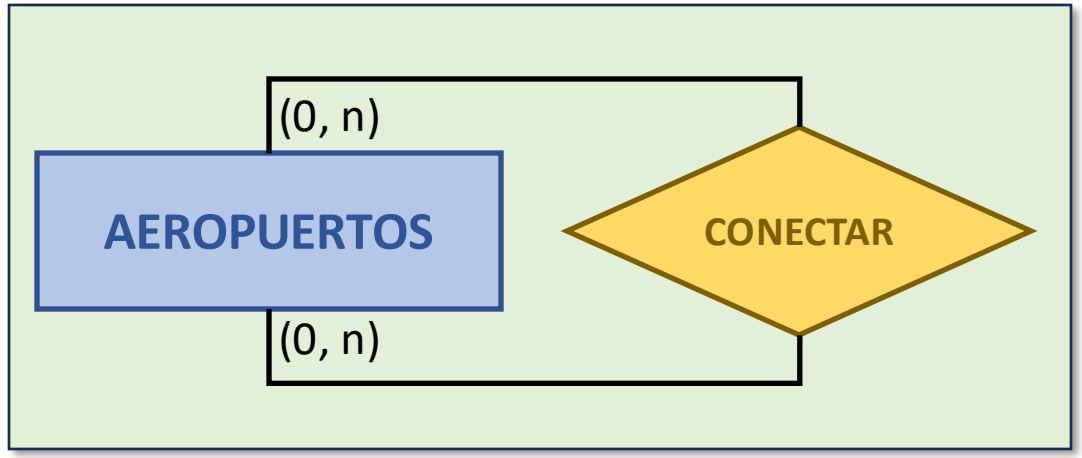
Relaciones reflexivas (n-n)

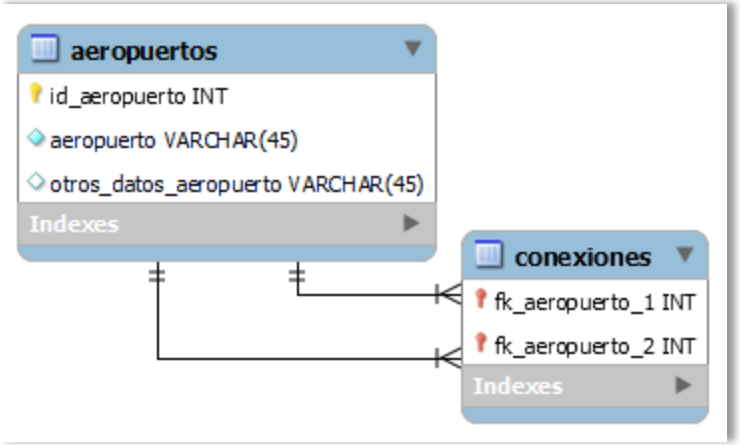




- Se siguen las mismas reglas que una relación n-n.
- Una nueva tabla que tendrá dos referencias a la misma tabla.
- Las dos referencias (fk) serán la primary key.

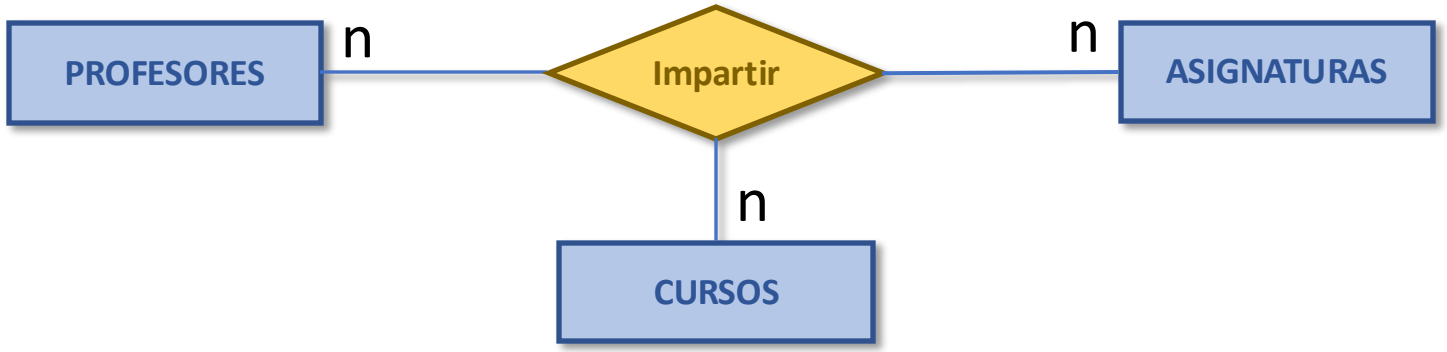
Relaciones reflexivas (n-n)





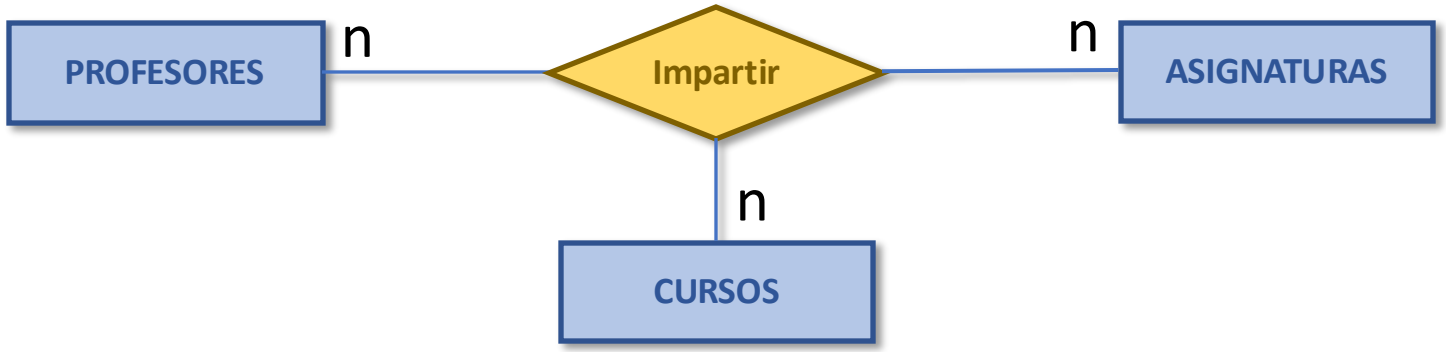

```sql
CREATE TABLE `aeropuertos` (
  `id_aeropuerto` int NOT NULL,
  `aeropuerto` varchar(45) NOT NULL,
  `otros_datos_aeropuerto` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id_aeropuerto`),
  UNIQUE KEY `aeropuerto_UNIQUE` (`aeropuerto`)
);

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`)
);
```

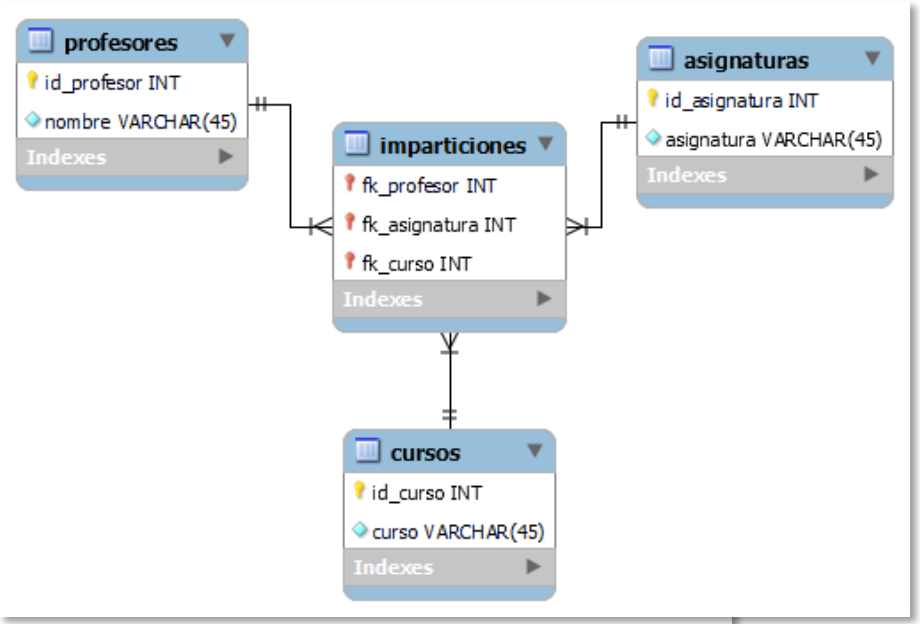
Relaciones ternarias – n:m:p



- 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

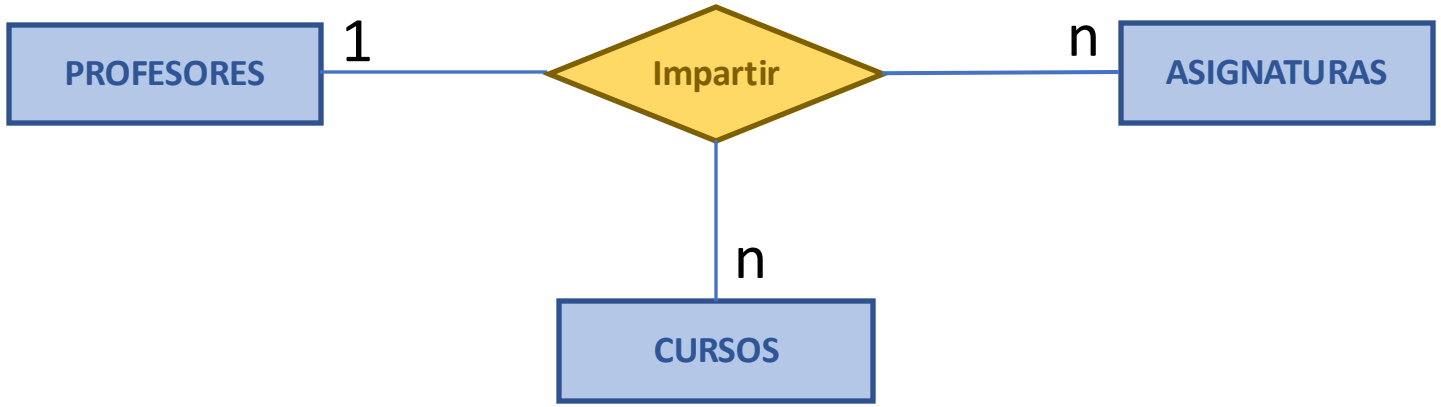
Relaciones ternarias – n:m:p





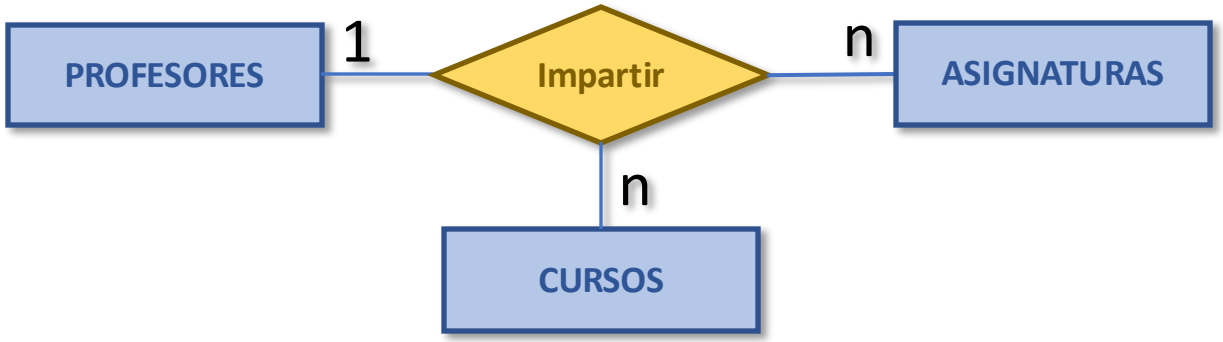

```
CREATE TABLE `imparticiones` (
  `fk_profesor` int NOT NULL,
  `fk_asignatura` int NOT NULL,
  `fk_curso` int NOT NULL,
  PRIMARY KEY (`fk_profesor`,`fk_asignatura`,`fk_curso`),
  CONSTRAINT `imp_asignaturas` FOREIGN KEY (`fk_asignatura`) REFERENCES `asignaturas` (`id_asignatura`),
  CONSTRAINT `imp_cursos`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

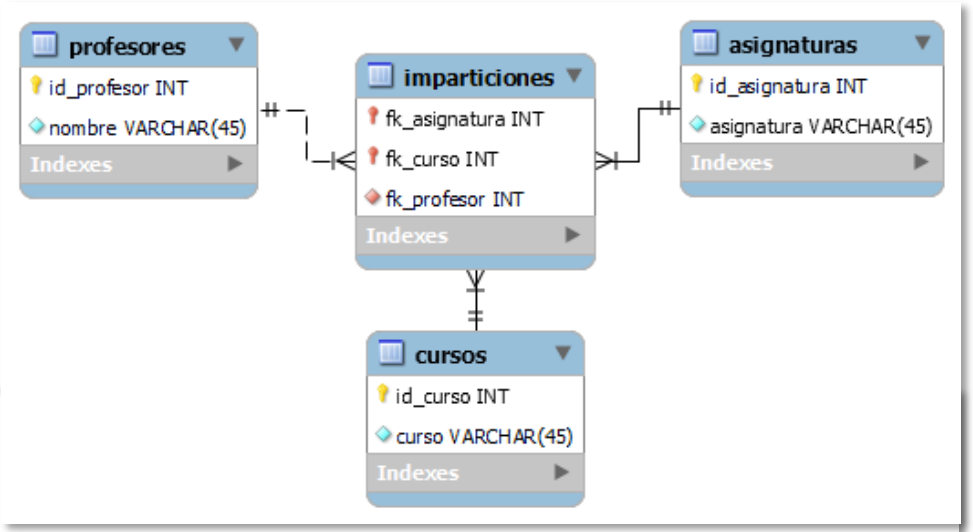
Relaciones ternarias – n:m:1



- 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

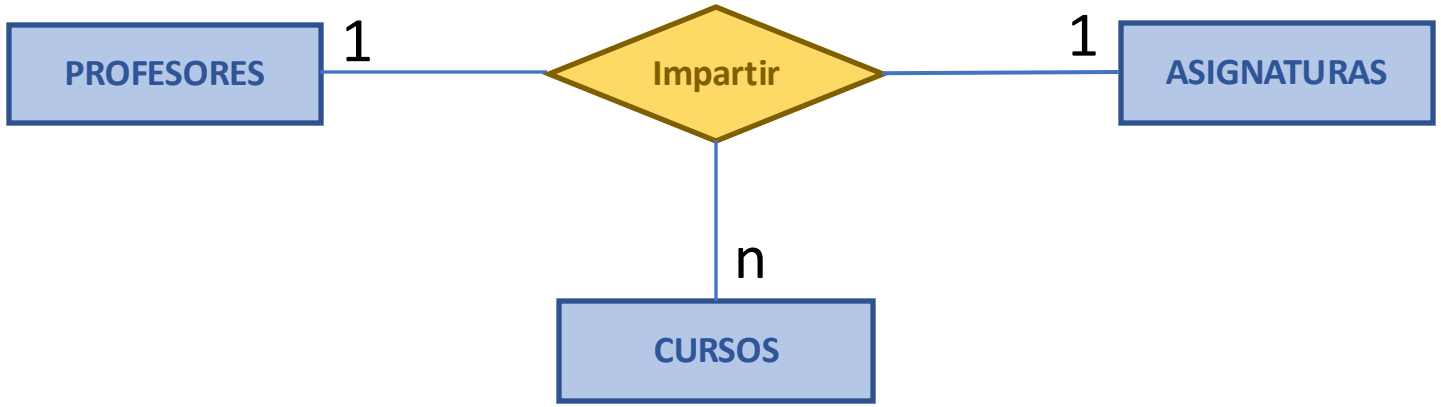
Relaciones ternarias – n:m:1





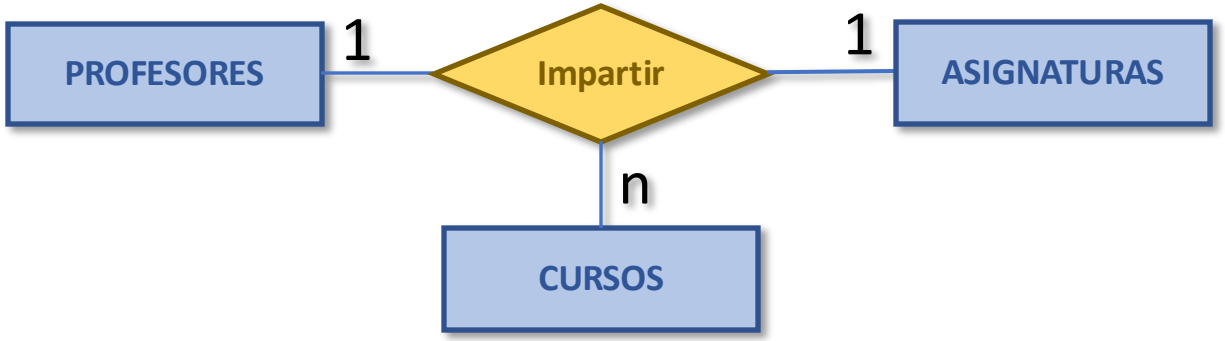

```
CREATE TABLE `imparticiones` (
  `fk_asignatura` int NOT NULL,
  `fk_curso` int NOT NULL,
  `fk_profesor` int NOT NULL,
  PRIMARY KEY (`fk_asignatura`,`fk_curso`),
  CONSTRAINT `imp_asignaturas` FOREIGN KEY (`fk_asignatura`) REFERENCES `asignaturas` (`id_asignatura`),
  CONSTRAINT `imp_cursos` FOREIGN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

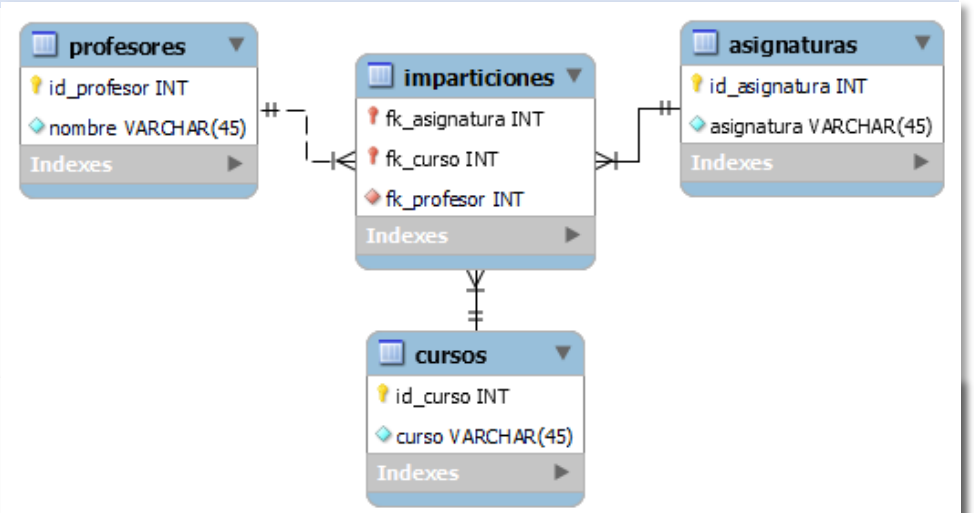
Relaciones ternarias – n:1:1



- 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.

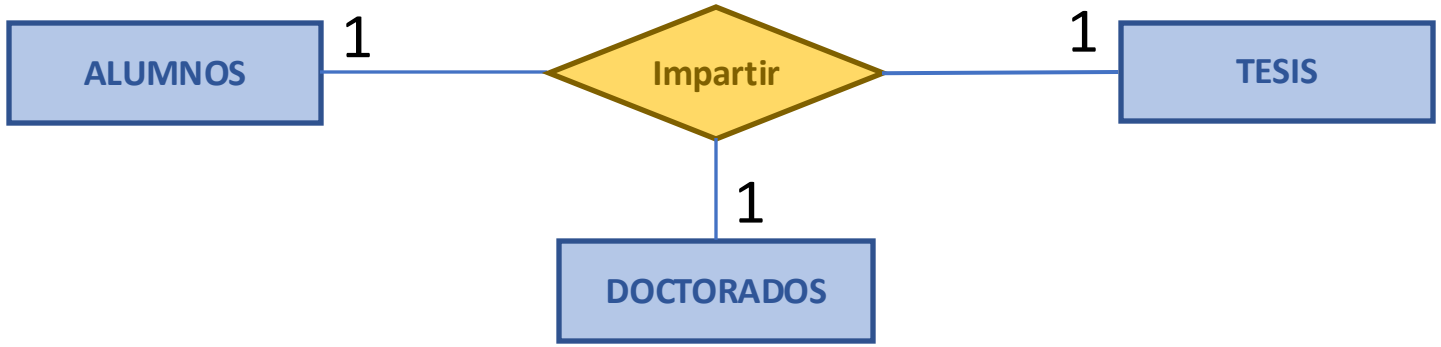
Relaciones ternarias – n:1:1





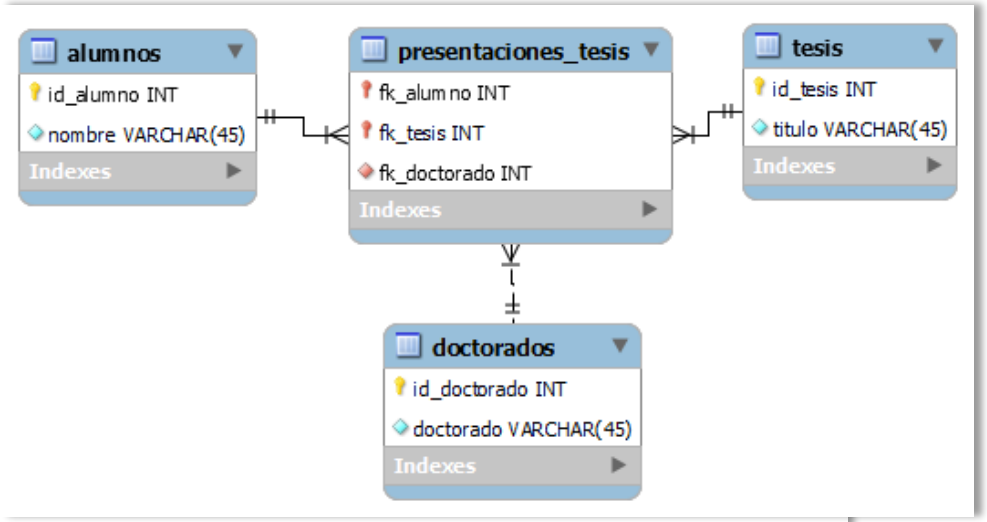

```
CREATE TABLE `imparticiones` (
  `fk_asignatura` int NOT NULL,
  `fk_curso` int NOT NULL,
  `fk_profesor` int NOT NULL,
  PRIMARY KEY (`fk_asignatura`,`fk_curso`),
  CONSTRAINT `imp_asignaturas` FOREIGN KEY (`fk_asignatura`) REFERENCES `asignaturas` (`id_asignatura`),
  CONSTRAINT `imp_cursos` FOREIGN 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



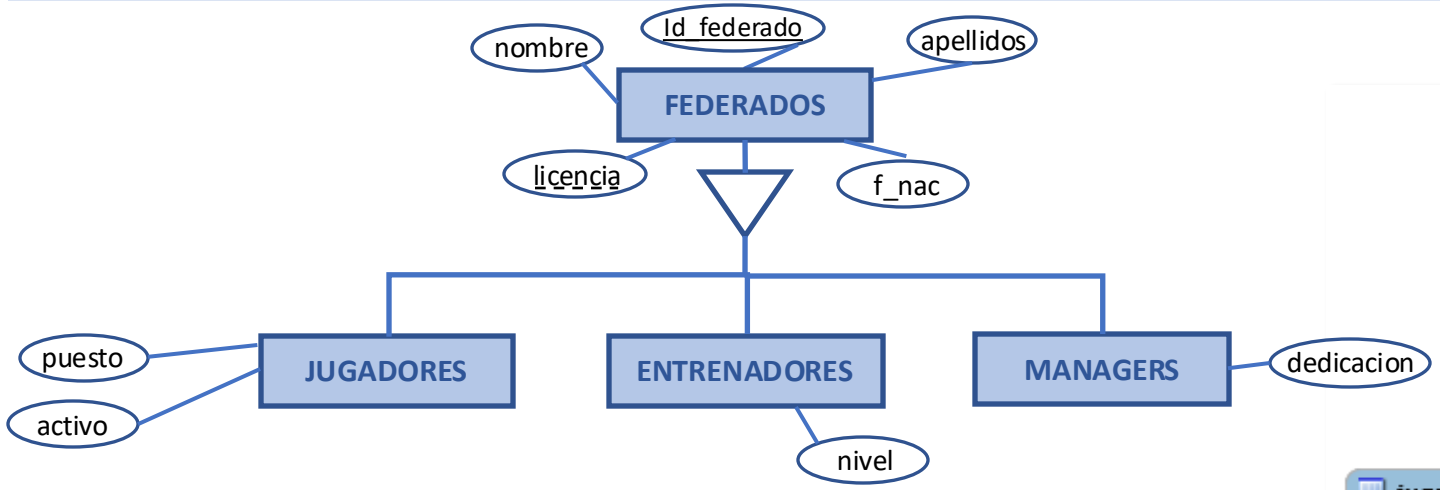
- Cada entidad genera una tabla.
- La relación genera una tabla
  - Contiene una FK a cada entidad de la relación
  - Existirán tres claves candidatas compuestas por las combinaciones de 2 FK.
  - Elegida una como PK, las otras dos claves candidata debe ser NOT NULL y UNIQUE.

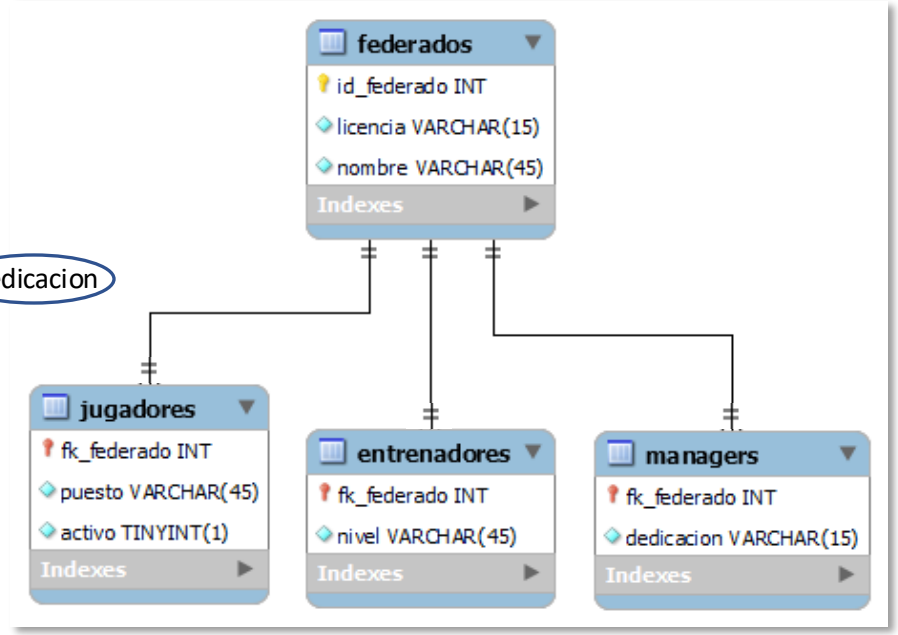
Relaciones ternarias – 1:1:1




```
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`)
);
```

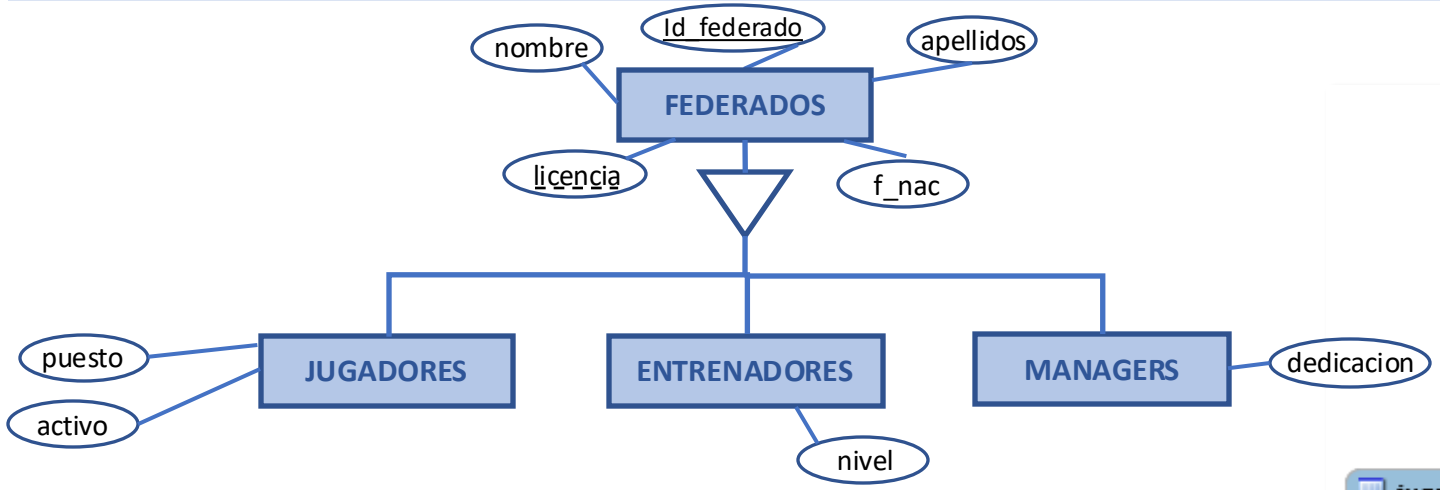
Relaciones de Herencia (IsA)
Estrategia: Tablas relacionadas

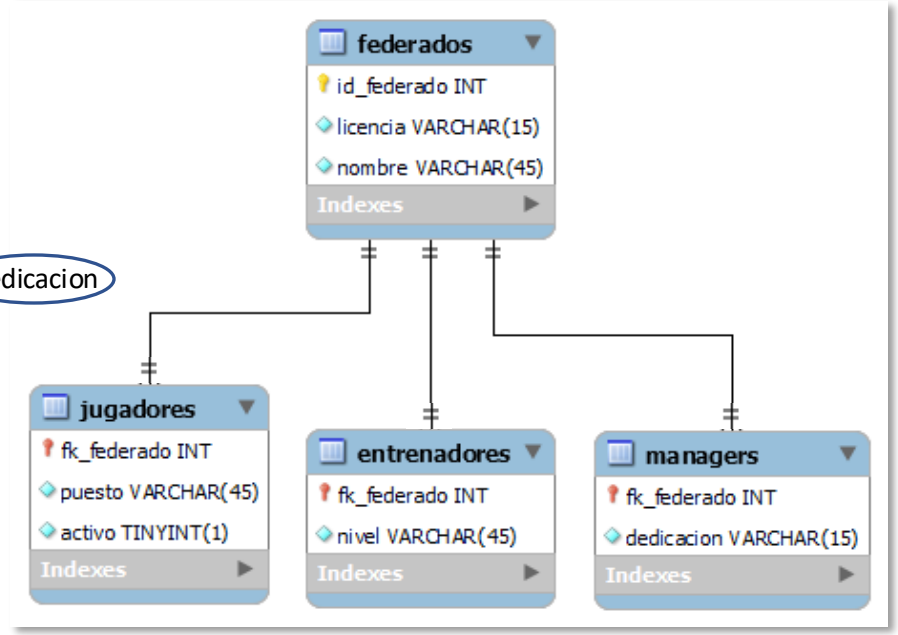




- Cada entidad genera una tabla.
  - Una tabla para la generalización y una por cada especialización.
  - La tabla generalización tendrá los atributos en común y las otras sólo los propios; y una FK a la de generalización y será su PK.
- ¿Es total o parcial? - ¿Es con solapamiento o sin solapamiento?

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

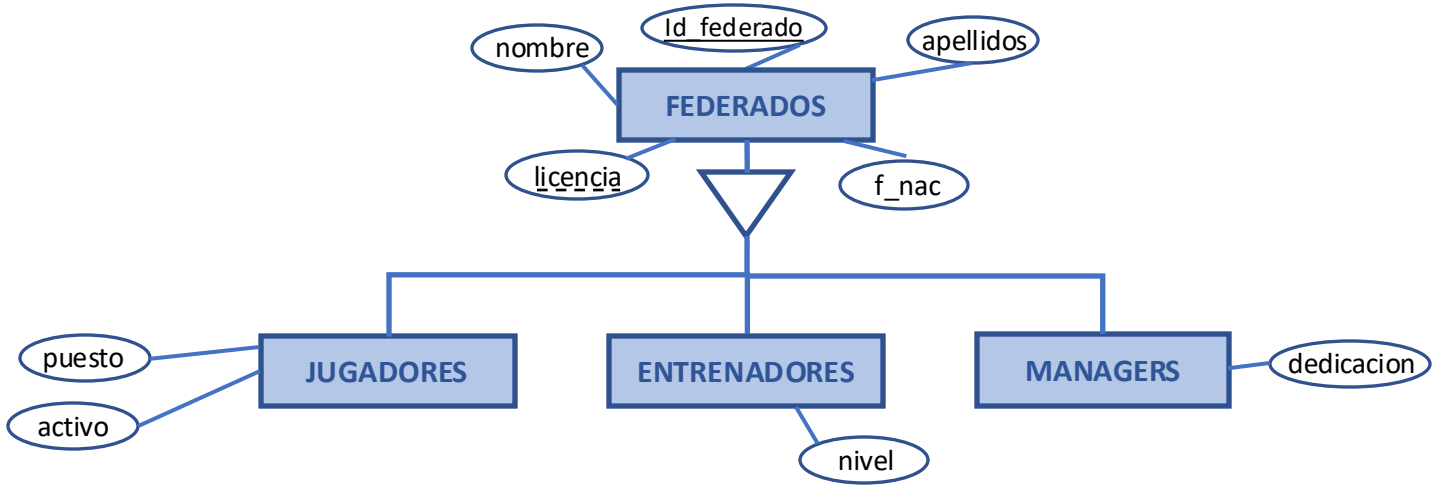
```
CREATE TABLE `federados` (
  `id_federado` int NOT NULL,
  `licencia` varchar(15) NOT NULL,
  `nombre` varchar(45) NOT NULL,
  PRIMARY KEY (`id_federado`),
  UNIQUE KEY `licencia_UNIQUE` (`licencia`)
);
```

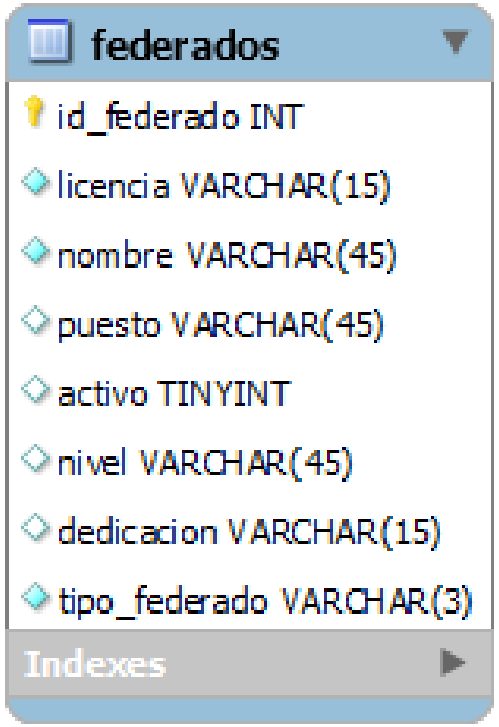
```
CREATE TABLE `entrenadores` (
  `fk_federado` int NOT NULL,
  `nivel` varchar(45) NOT NULL,
  PRIMARY KEY (`fk_federado`),
  CONSTRAINT `ent_fed` FOREIGN KEY (`fk_federado`)
      REFERENCES `federados` (`id_federado`)
);
```

```
CREATE TABLE `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 `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

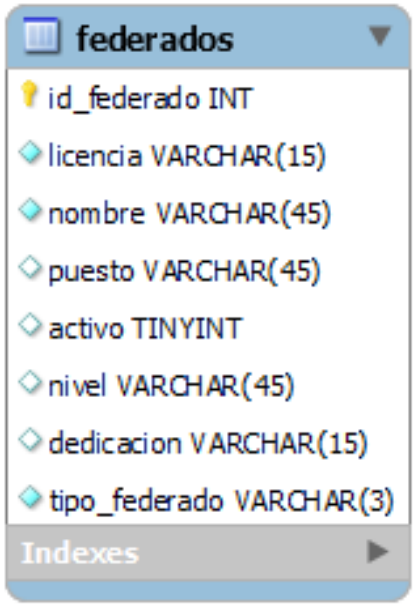




- Una sola tabla para toda la familia
  - Debe contener todos los atributos en común, los de cada especialización y un discriminante.
  - Se deben generar checks con controles indicando qué atributos deben ser null y cuáles no; en base al valor del discriminante.

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))
) ;
```



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')),
```

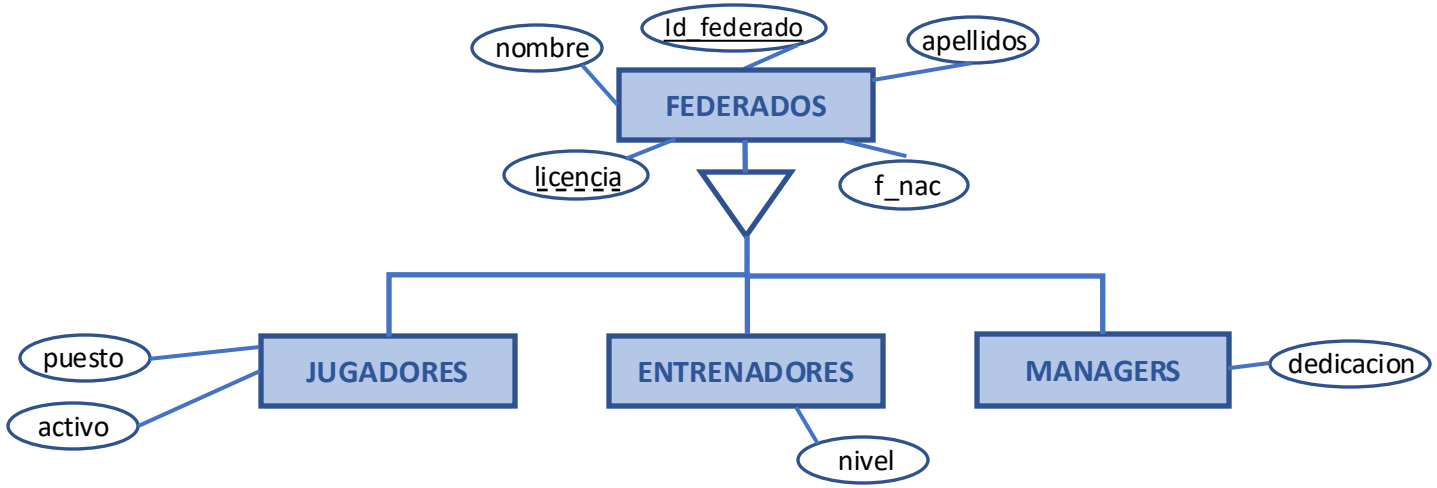
  - Y considerándolo en el check que controla qué atributos pueden o no ser null.

- ¿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.

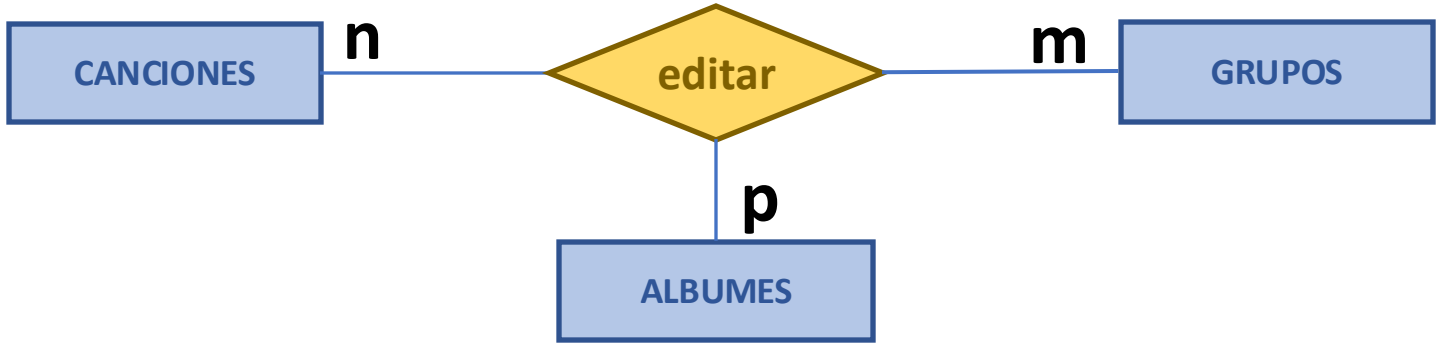
Relaciones de Herencia (IsA)
Estrategia: Tabla independiente por entidad

NO UTILIZAR

- 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)

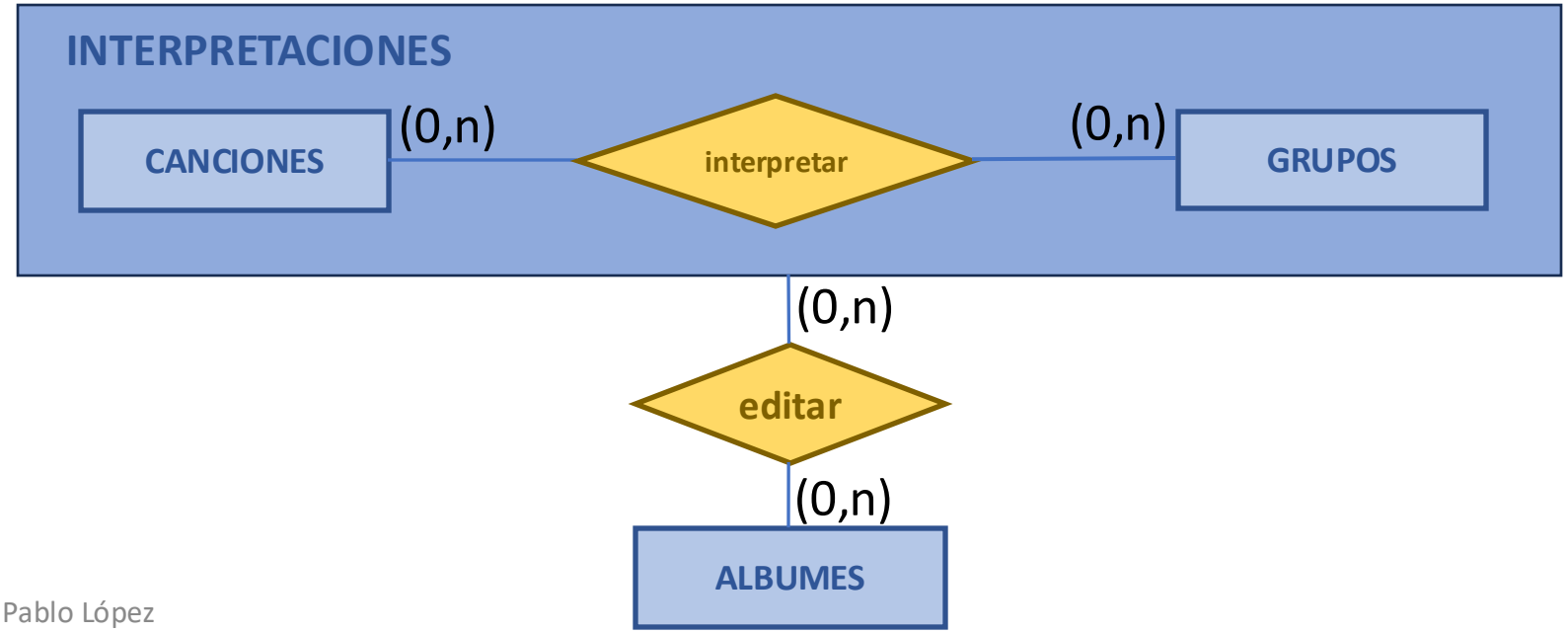
- Tenemos 3 entidades: Canciones, Grupos, Álbumes.
- Una canción contenida en un álbum podría estar interpretada por varios grupos.
- Un grupo en un álbum puede tener muchas canciones.
- Una canción interpretada por un grupo puede estar publicada en muchos álbumes.



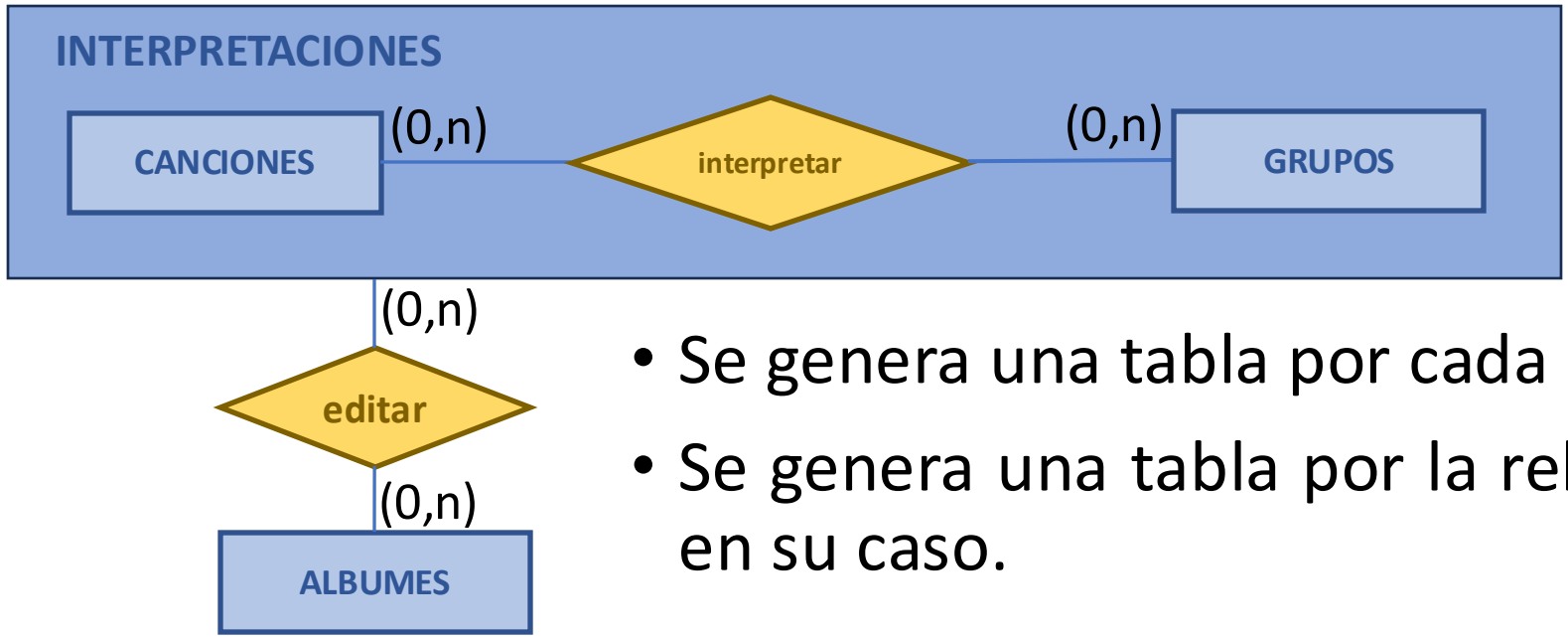
- ¿Y si una canción, interpretada por un grupo no se ha publicado en ningún álbum?, por ejemplo, solo la ha interpretado en conciertos.

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



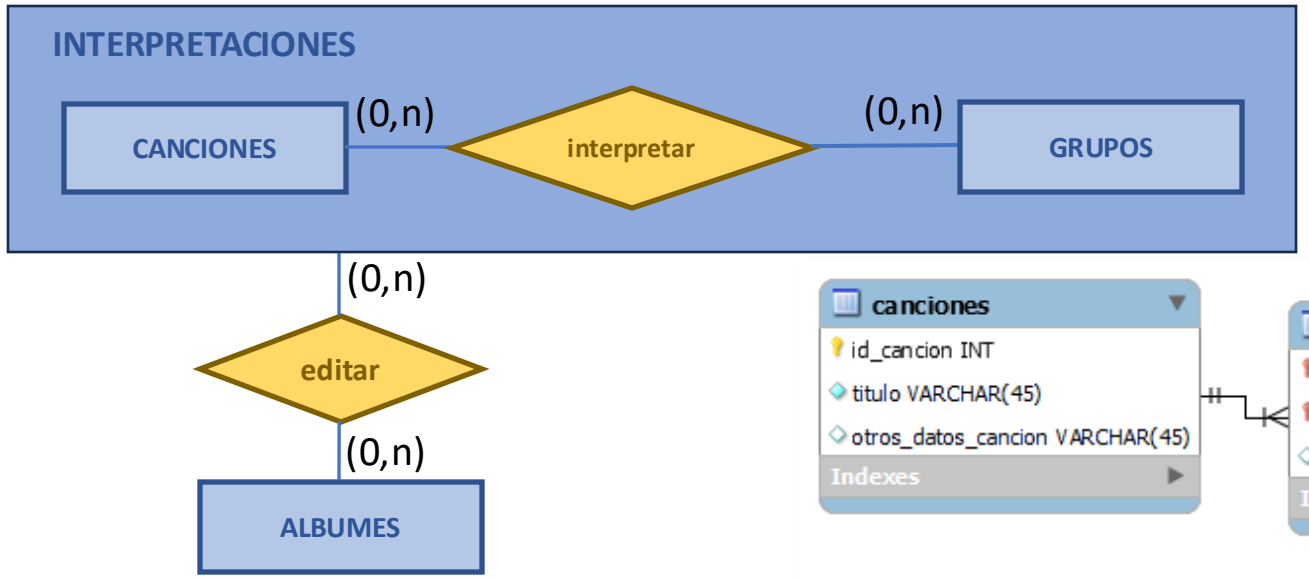
Agregaciones

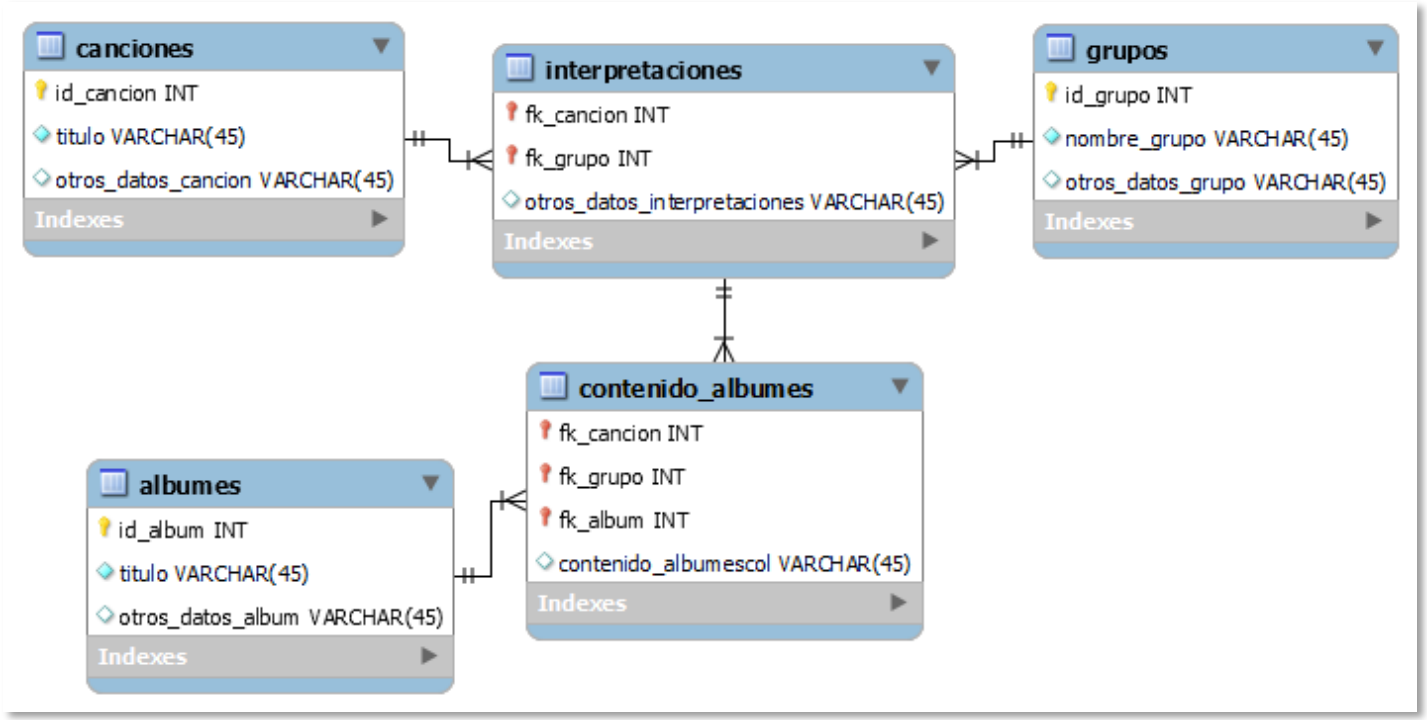


- Se genera una tabla por cada entidad.
- Se genera una tabla por la relación n-n o ternaria en su caso.

- 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`)
);
```

Normalización

- Es una técnica que consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso a tablas desde el DER.
  - Almacenamiento de la información sin redundancias innecesarias.
  - Recuperación eficiente de la información.
  - Minimizar el mantenimiento de la BBDD.
  - Minimizar los problemas de actualización de los datos.

- Definidas por Codd
  - **1FN** – Primera Forma Normal
  - **2FN** – Segunda Forma Normal
  - **3FN** – Tercera Forma Normal

- Luego, se definieron
  - **FNBC** – Forma Normal de Boyce-Codd
  - **4FN** – Cuarta Forma Normal
  - **5FN** – Quinta Forma Normal

Primera forma normal - 1FN

- Un dominio es atómico si se considera que los elementos del dominio son unidades indivisibles.

- Una relación R están en **primera forma normal (1FN)** si los dominios de todos los atributos de R son atómicos.

Primera forma normal - 1FN

Jugadores(id_jugador, nombre, apellidos, domicilio, telefonos)

ID_JUGADOR	NOMBRE	APELLIDOS	DOMICILIO	TELEFONOS
1	Javier	Ortega Desio	Francia 15, 3 A - Madrid - CCAA Madrid	612321456 912987159
2	Marcos	Ayerza Fernandez	Alcala 28 - Alcobendas - Madrid	654987147
3	Jeronimo	De La Fuente	Aragon 12 BAJO- Leganes - 28913 - Madrid	611122233 614785236

Jugadores(id_jugador, nombre, apellido1, apellido2, calle, nro, piso, ciudad, provincia, CCAA, telefonos)

ID_JUGADOR	NOMBRE	APELLIDO_1	APELLIDO_2	CALLE	NRO	PISO	CIUDAD	PROVINCIA	CCAA	TELEFONOS
1	Javier	Ortega	Desio	Francia	15	3 A	Madrid	Madrid	Madrid	612321456
1	Javier	Ortega	Desio	Francia	15	3 A	Madrid	Madrid	Madrid	912987159
2	Marcos	Ayerza	Fernandez	Alcala	28	null	Alcobendas	Madrid	Madrid	654987147
3	Jeronimo	De La Fuente	null	Aragon	12	BAJO	Leganes	Madrid	Madrid	611122233
3	Jeronimo	De La Fuente	null	Aragon	12	BAJO	Leganes	Madrid	Madrid	614785236

Jugadores(id_jugador, nombre, apellido1, apellido2, calle, nro, piso, ciudad, provincia, CCAA)

ID_JUGADOR	NOMBRE	APELLIDO_1	APELLIDO_2	CALLE	NRO	PISO	CIUDAD	PROVINCIA	CCAA
1	Javier	Ortega	Desio	Francia	15	3 A	Madrid	Madrid	Madrid
2	Marcos	Ayerza	Fernandez	Alcala	28	null	Alcobendas	Madrid	Madrid
3	Jeronimo	De La Fuente	null	Aragon	12	BAJO	Leganes	Madrid	Madrid

Telefonos(telefono, id_jugador)

TELEFONO	ID_JUGADOR
612321456	1
912987159	1
654987147	2
611122233	3
614785236	3

Ejemplo

- Se debe registrar la información de un club deportivo, de un deporte en equipo.
- Nos centraremos en cómo registrar los datos de “partidos”.
  - En un partido, que se identifica por un nro_partido, una descripción, que se juega en una fecha y hora determinada, en un campo.
  - Participa un equipo de nuestro club y un oponente. Nuestro equipo, se identifica con un nro_equipo, tiene un nombre, participa en una división en una temporada.
  - Y para este partido, este equipo se conforma de un grupo de jugadores que tienen su nro_jugador, nombre, apellidos, puesto en este partido, etc.

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada, nro_jugador, nombre, apellido1, apellido2, dni, puesto)

1FN

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada, nro_jugador, nombre, apellido1, apellido2, dni, puesto)

nro_partido	partido	fecha	hora	campo	oponente	nro_equipo	equipo	division	temporada	nro_jugador	nombre	apellido_1	apellido_2	dni	puesto
1	Jornada 5	12/04/2021	15:00	Central	Deportivo Sur	15	Senior A	2	20-21	1	Javier	Ortega	Desio	4473754P	pilar
1	Jornada 5	12/04/2021	15:00	Central	Deportivo Sur	15	Senior A	2	20-21	12	Marcos	Ayerza	Fernandez	1424527M	pilar
1	Jornada 5	12/04/2021	15:00	Central	Deportivo Sur	15	Senior A	2	20-21	15	Jeronimo	De La Fuente	null	4499786E	apertura
1	Jornada 5	12/04/2021	15:00	Central	Deportivo Sur	15	Senior A	2	20-21	7	Joaquin	Tuculet	null	6065642U	centro 12
1	Jornada 5	12/04/2021	15:00	Central	Deportivo Sur	15	Senior A	2	20-21	8	Lucas	Noguera	Lopez	5245191F	centro 13

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada, nro_jugador, nombre, apellido1, apellido2, dni, puesto)

1FN

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada)

JugadoresPorPartido(nro_partido, nro_jugador, nombre, apellido1, apellido2, dni, puesto)

nro_partido	partido	fecha	hora	campo	oponente	nro_equipo	equipo	division	temporada
1	Jornada 5	12/04/2021	15:00	Central	Deportivo Sur	15	Senior A	2	20-21

nro_partido	nro_jugador	nombre	apellido_1	apellido_2	dni	puesto
1	1	Javier	Ortega	Desio	4473754P	pilar
1	12	Marcos	Ayerza	Fernandez	1424527M	pilar
1	15	Jeronimo	De La Fuente	null	4499786E	apertura
1	7	Joaquin	Tuculet	null	6065642U	centro 12
1	8	Lucas	Noguera	Lopez	5245191F	centro 13

Dependencia Funcional

- Es una relación entre atributos de una misma tabla.
- Si “x” e “y” son atributos o conjuntos de atributos de R, se dice que “y” es funcionalmente dependiente de “x” ($x \rightarrow y$) si cada valor de “x” tiene asociado un solo valor de “y”.
- “x” se denomina “determinante”, ya que determina el valor de y.
- Se dice que “y” es completamente dependiente de “x” si “y” no depende de ningún subconjunto de “x”

nro_partido	nro_jugador	nombre	apellido_1	apellido_2	dni
1	1	Javier	Ortega	Desio	4473754P
1	12	Marcos	Ayerza	Fernandez	1424527M

?

(nro_partido → nombre)

(nro_partido, nro_jugador → nombre)

(nro_jugador → nombre)

Segunda Forma Normal – 2FN

- Una relación R está en **segunda forma normal (2FN)**, si y sólo si, está en 1FN y cada atributo que no forma parte de la clave primaria depende completamente de la clave primaria.

- Se aplica sólo a tablas que tienen claves primarias compuestas. Si una tabla está en 1FN y tiene una clave primaria simple, está en 2FN.

2FN

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada)

JugadoresPorPartido(nro_partido, nro_jugador, nombre, apellido1, apellido2, dni, puesto)

nro_partido	partido	fecha	hora	campo	oponente	nro_equipo	equipo	division	temporada
1	Jornada 5	12/04/2021	15:00	Central	Deportivo Sur	15	Senior A	2	20-21

Clave simple, ya está en 2FN

nro_partido	nro_jugador	nombre	apellido_1	apellido_2	dni	puesto
1	1	Javier	Ortega	Desio	4473754P	pilar
1	12	Marcos	Ayerza	Fernandez	1424527M	pilar
1	15	Jeronimo	De La Fuente	null	4499786E	apertura
1	7	Joaquin	Tuculet	null	6065642U	centro 12
1	8	Lucas	Noguera	Lopez	5245191F	centro 13

Dependencias funcionales de "parte" de la clave

(nro_jugador → nombre) (nro_jugador → apellido2)

(nro_jugador → apellido1) (nro_jugador → dni)

2FN

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada)

Jugadores (nro_jugador, nombre, apellido1, apellido2, dni)

JugadoresPorPartido(nro_partido, nro_jugador, puesto)

2FN

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada)

Jugadores (nro_jugador, nombre, apellido1, apellido2, dni)

JugadoresPorPartido(nro_partido, nro_jugador, puesto)

Partidos

nro_partido	partido	fecha	hora	campo	oponente	nro_equipo	equipo	division	temporada
1	Jornada 5	12/04/2021	15:00	Central	Deportivo Sur	15	Senior A	2	20-21

Jugadores

nro_jugador	nombre	apellido_1	apellido_2	dni
1	Javier	Ortega	Desio	4473754P
12	Marcos	Ayerza	Fernandez	1424527M
15	Jeronimo	De La Fuente	null	4499786E
7	Joaquin	Tuculet	null	6065642U
8	Lucas	Noguera	Lopez	5245191F

JugadoresPorPartido

nro_partido	nro_jugador	puesto
1	1	pilar
1	12	pilar
1	15	apertura
1	7	centro 12
1	8	centro 13

Tercera Forma Normal – 3FN

- Una relación R está en **tercera forma normal (3FN)**, si y sólo si, está en 2FN y cada atributo que no forma parte de la clave primaria no depende transitivamente de la clave primaria.

Dependencia Funcional Transitiva

Si “x”, “y”, “z” son atributos o conjuntos de atributos de R, se dice que existe una dependencia funcional transitiva si “y” es funcionalmente dependiente de “x” ($x \rightarrow y$) y además, “z” es funcionalmente dependiente de “y” ($y \rightarrow z$)

$$\boldsymbol{(x \rightarrow y)\ \text{y}\ (y \rightarrow z) \Rightarrow (x \rightarrow z)}$$

3FN

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada)

Jugadores (nro_jugador, nombre, apellido1, apellido2, dni) **No hay dependencias transitivas, ya está en 3FN**

JugadoresPorPartido(nro_partido, nro_jugador, puesto) **No hay dependencias transitivas, ya está en 3FN**

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, division, temporada)

Dependencias funcionales transitivas

(nro_partido → nro_equipo) y (nro_equipo → equipo)
(nro_partido → nro_equipo) y (nro_equipo → division)
(nro_partido → nro_equipo) y (nro_equipo → temporada)

3FN

Equipos(nro_equipo, equipo, division, temporada)

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo)

Jugadores (nro_jugador, nombre, apellido1, apellido2, dni)

JugadoresPorPartido(nro_partido, nro_jugador, puesto)

Normalización a 3FN

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada, nro_jugador, nombre, apellido1, apellido2, dni, puesto)

1FN

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada)

JugadoresPorPartido(nro_partido, nro_jugador, nombre, apellido1, apellido2, dni, puesto)

2FN

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo, equipo, división, temporada)

Jugadores (nro_jugador, nombre, apellido1, apellido2, dni)

JugadoresPorPartido(nro_partido, nro_jugador, puesto)

3FN

Equipos(nro_equipo, equipo, division, temporada)

Partidos(nro_partido, nombre_partido, fecha, hora, campo, oponente, nro_equipo)

Jugadores (nro_jugador, nombre, apellido1, apellido2, dni)

JugadoresPorPartido(nro_partido, nro_jugador, puesto)

Forma Normal de Boyce-Codd – BCFN

- Definición:

Una dependencia funcional A -> B es trivial **cuando B es parte de A**. Esto sucede cuando A es un conjunto de atributos, y B es a su vez un subconjunto de A.

- Una relación R está en **forma normal de Boyce-Codd (BCFN)**, si y solo si está en 3FN y cada dependencia funcional no trivial tiene una clave candidata como determinante.

Dicho de otra forma, una tabla está en FNBC si está en 3FN y los únicos determinantes son claves candidatas.

Otras formas normales

- 4FN

  Una relación está en 4FN si se encuentra en 3FN y además las únicas dependencias funcionales multivaluadas son aquellas que dependen de claves candidatas.

- 5FN
  - Una relación está en 5FN si se encuentra en 4FN y, además, toda dependencia de combinación es consecuencia de las claves candidatas.