DESARROLLO WEB

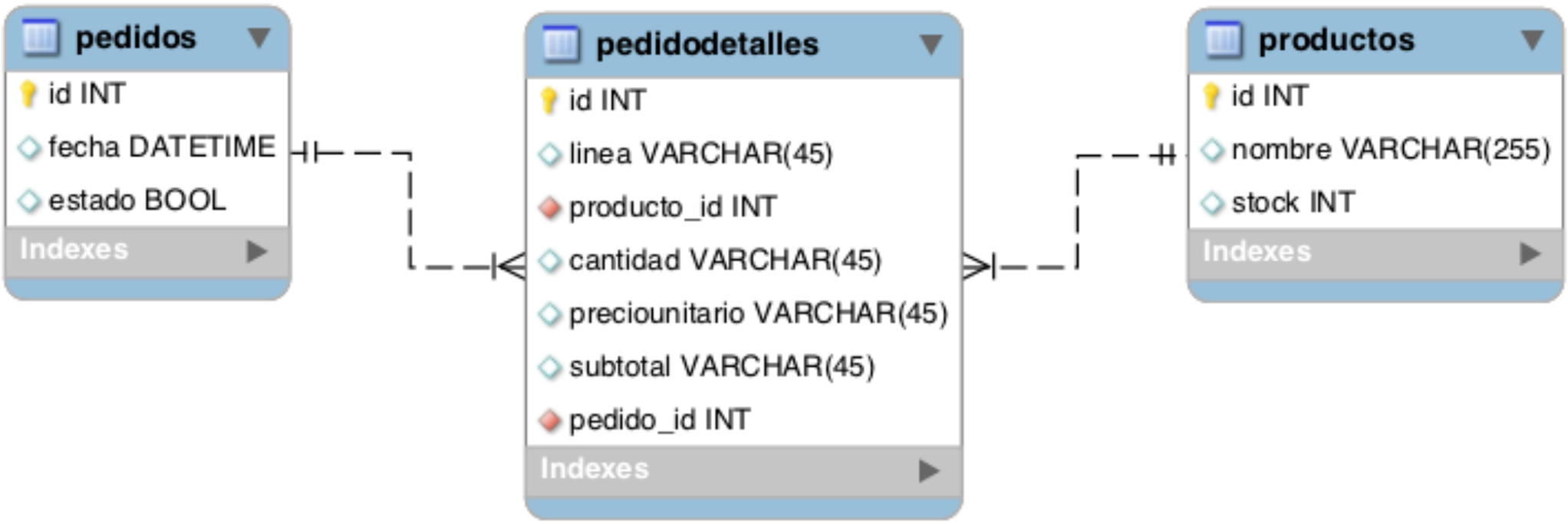
BACKEND

PERCY CRISOLOGO BARDALES

TABLAS RELACIONADAS

Una base de datos relacional consta de varias tablas relacionadas que se unen mediante columnas comunes que se conocen como columnas de clave externa.

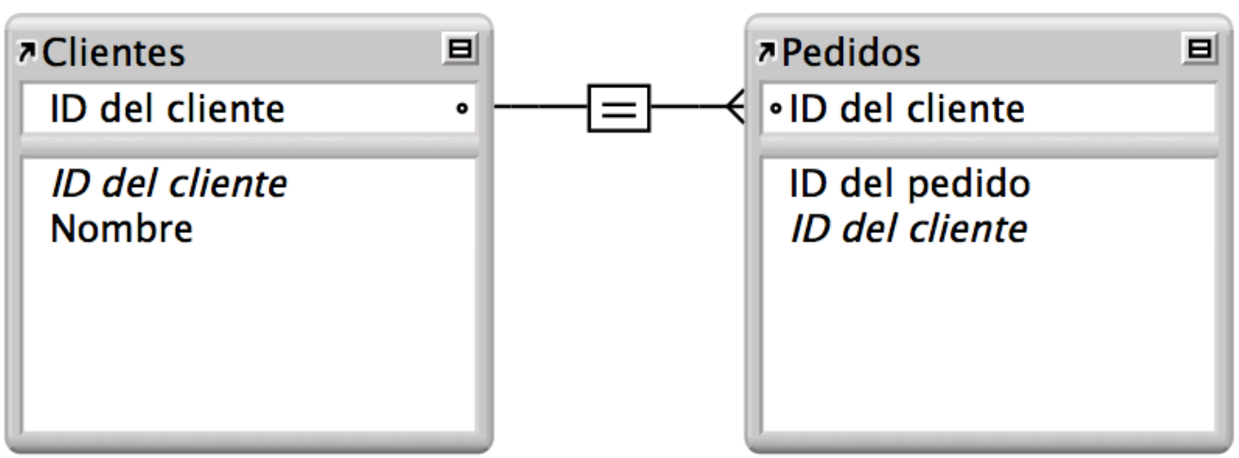
Por ejemplo, tenemos la tabla pedidos la cual se relaciona con la tabla pedidodetalles y la tabla productos que se relaciona con la tabla pedidodetalles.



RELACION 1 A MUCHOS

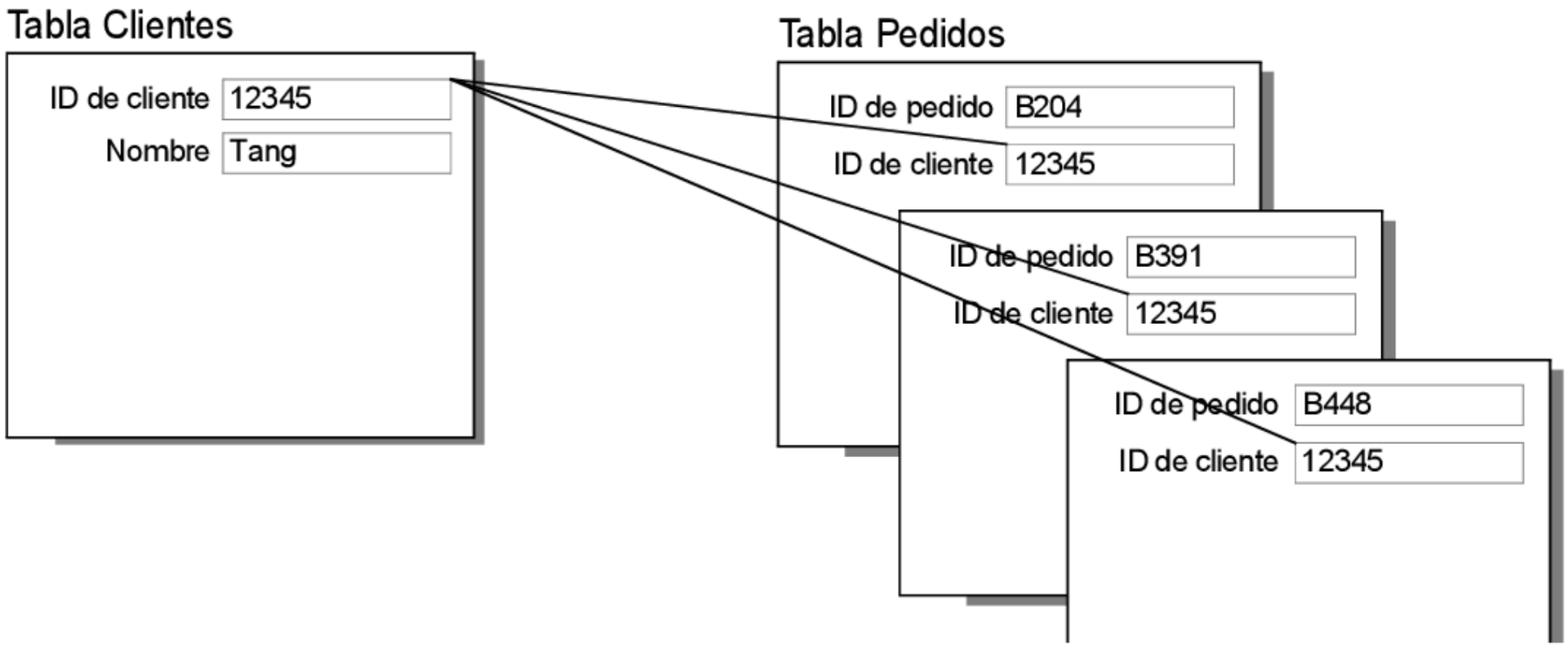
En una relación de uno a muchos, un registro de una tabla uno se puede asociar a uno o varios registros de la tabla muchos.

Cada cliente puede tener varios pedidos de ventas.



Ejemplos:

- Categorías y Productos
- Personas y Telefonos.
- Empresa empleado



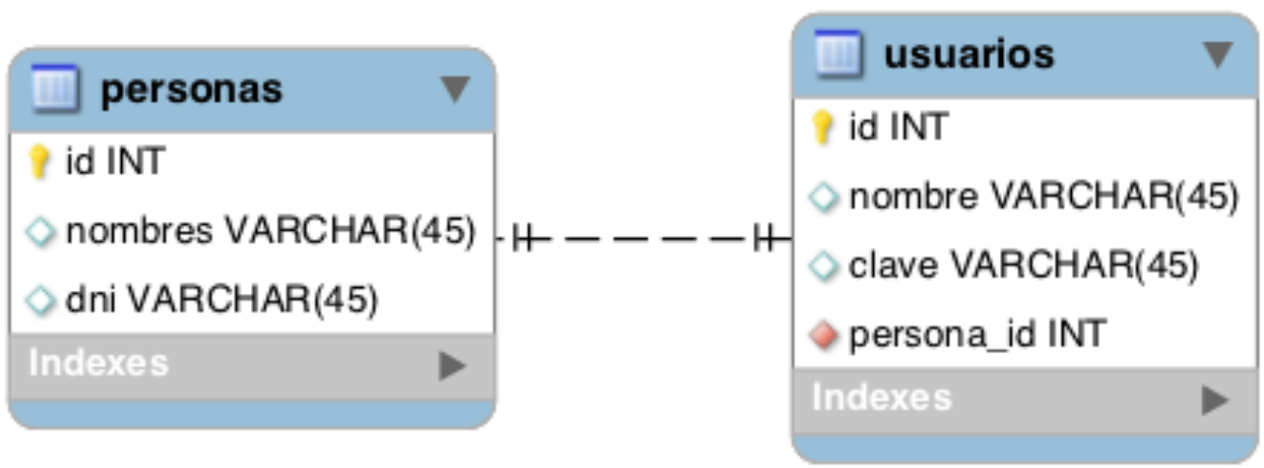
RELACION 1 A 1

Las relaciones Uno a Uno se utilizan sólo en casos especiales puesto que las tablas relacionadas por este tipo pueden combinarse en una sola tabla.

Algunas de las razones para utilizar relaciones Uno a Uno son:

- En la tabla existen datos de tipo Texto, Imagen o BLOB muy grandes. Estos campos podrían ralentizar las consultas si se cargan en la memoria cuando un registro se convierte en el registro actual. Al colocar los textos, imágenes o BLOBs en otra tabla, puede cargar en memoria sólo los datos que necesita y de esta forma optimizar el funcionamiento de la base de datos.
- La tabla contiene un número muy grande de campos y necesita dividirlos en grupos lógicos. Las tablas separadas pueden hacer que la base de datos sea más rápida y fácil de utilizar.
- Utilizando tablas separadas, podemos asignar diferentes privilegios de acceso a cada tabla, con lo cual podríamos limitar el acceso a ciertos campos.

Binary Large Objects: Permite guardar archivos (pdfs, imágenes, etc) directamente en la base de datos.



RELACION MUCHOS A MUCHOS

Imaginemos que deseamos relacionar la tabla Alumnos con la tabla Cursos.
Un Alumno puede llevar varios cursos y un Curso puede ser llevado por varios alumnos.

Ejemplos:

- Proveedores y Productos: cada proveedor suministra diferentes productos y cada producto puede ser suministrado por diferentes proveedores.

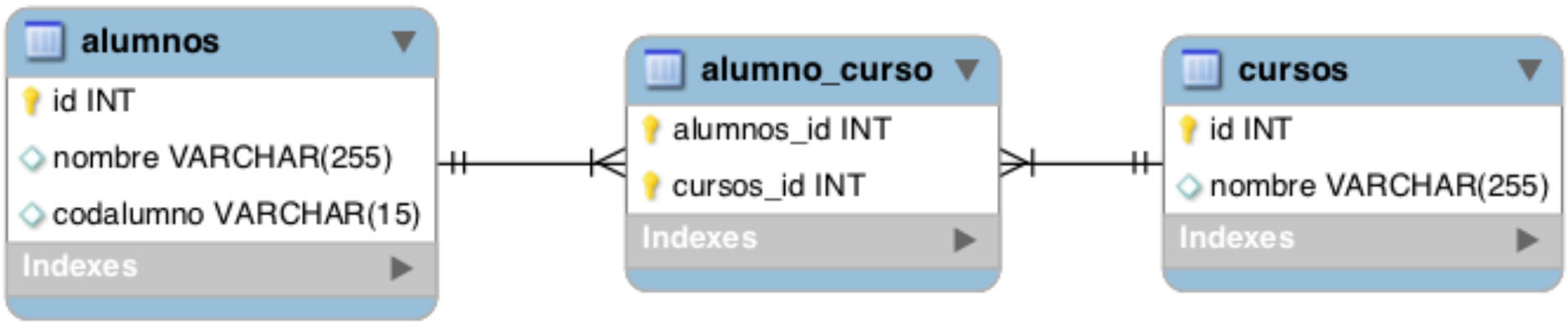
- Películas y Actores: en cada película participan varios actores y cada actor puede participar en diferentes películas.

Tablas intermedias:

En estos casos, se debe crear una tabla intermedia relacionada a las otras tablas utilizando relaciones Uno a Muchos.

Para los ejemplos anteriores, las tablas intermedias serían:

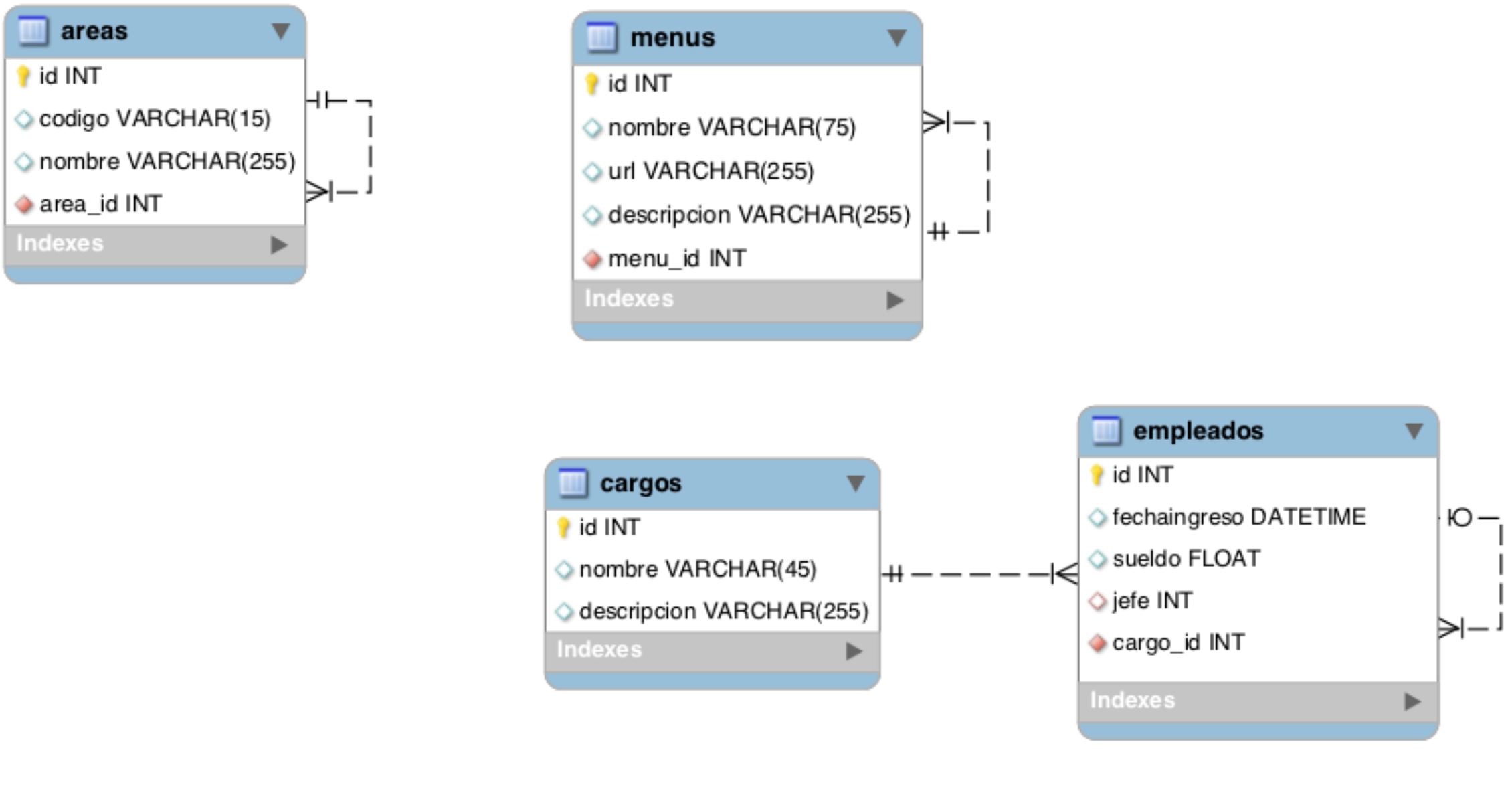
- Alumnos y Cursos: Alumno_curso.
- Proveedores y Productos: Proveedor_producto
- Películas y Actores: Actor_pelicula



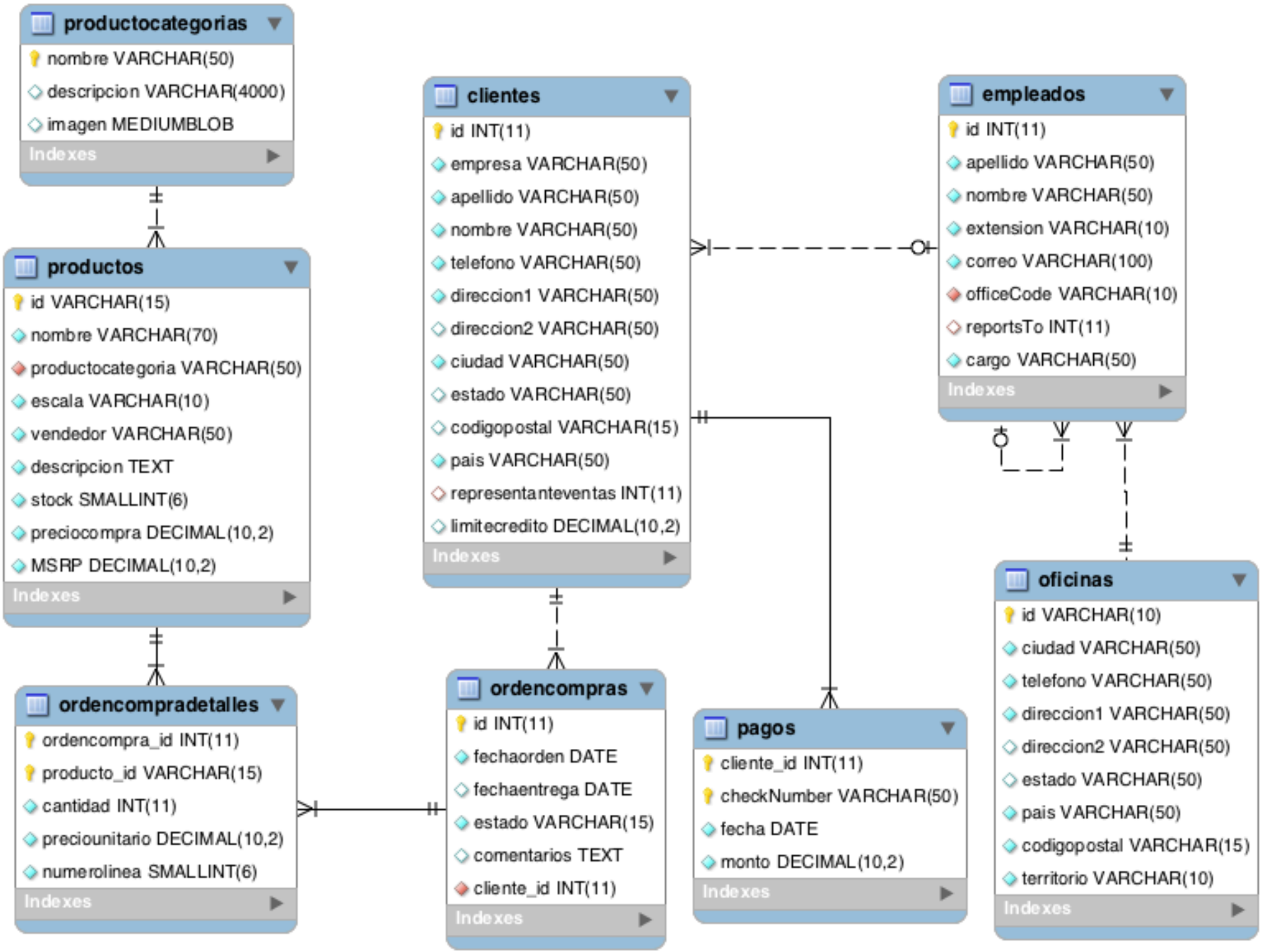
RELACIONES AUTOUNION

Una relación de autounion implica relacionar a la tabla consigo misma, por ejemplo tenemos la tabla empleados.

Cada empleado tiene un jefe que es otro empleado, esta relación la podemos representar como una relación a sis misma en la tabla empleados.



EJEMPLO BASE DE DATOS



INNERS

Vamos a crear dos tablas, una llamada miembros y la otra llamada comites, luego insertar datos en cada una de ellas.

```sql
CREATE TABLE miembros (
    id INT AUTO_INCREMENT,
    nombre VARCHAR(100),
    PRIMARY KEY (id)
);
```

```sql
CREATE TABLE comites (
    id INT AUTO_INCREMENT,
    nombre VARCHAR(100),
    PRIMARY KEY (id)
);
```

```sql
INSERT INTO
    miembros ( nombre )
VALUES('John'),
    ('Jane'),
    ('Mary'),
    ('David'),
    ('Amelia');
```

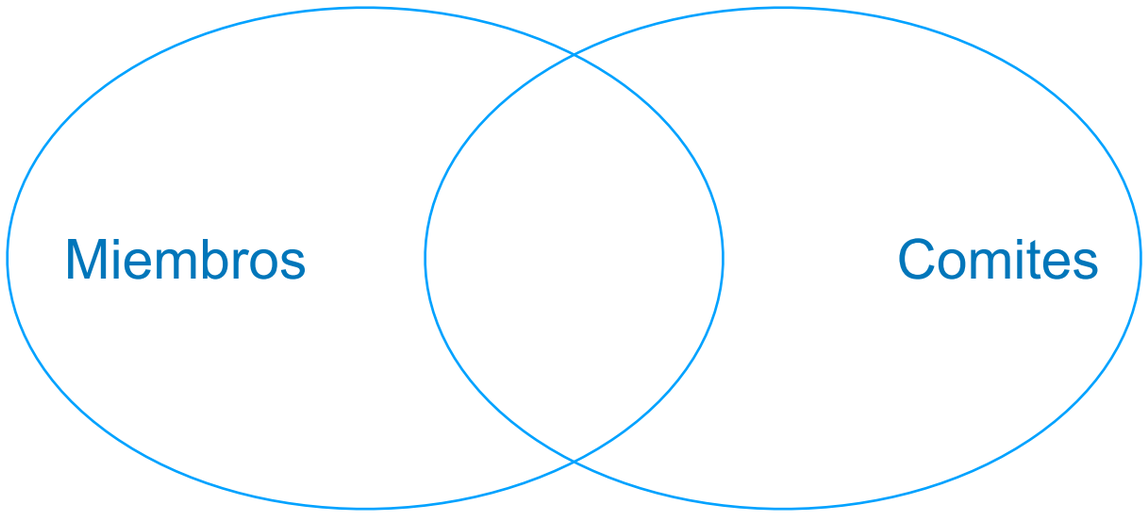
```sql
INSERT INTO
    comites(nombre)
VALUES
    ('John'),
    ('Mary'),
    ('Amelia'),
    ('Joe');
```

INNER JOIN

Ejemplo 1: Inner Join entre la tabla miembros y comités a través del campo nombre

```
SELECT
      m.id,
      m.nombre miembro,
      c.id,
      c.nombre comite
FROM
      miembros m
INNER JOIN
      comites c
ON c.nombre = m.nombre;
```

id	miembro	id	comite
1	John	1	John
3	Mary	2	Mary
5	Amelia	3	Amelia



INNER JOIN

Ejemplo 1: Inner Join entre la tabla miembros y comités.

Consulta Equivalente

```
SELECT
        m.id,
        m.nombre miembro,
        c.id,
        c.nombre comite
FROM
        miembros m
INNER JOIN
        comites c
USING(nombre)
```

id	miembro	id	comite
1	John	1	John
3	Mary	2	Mary
5	Amelia	3	Amelia

Como el campo (columna) nombre esta en ambas tablas, podemos utilizar USING

LEFT JOIN

LEFT JOIN selecciona datos a partir de la tabla izquierda.
Para cada fila de la tabla izquierda, se compara con cada fila de la tabla derecha.
Si los valores coinciden con los de la tabla derecha se crea una fila con los datos de ambas tablas; de lo contrario se crea una fila con los datos de la tabla izquierda y se rellena con NULL los datos de la tabla derecha.

Ejemplo 1: Crear la consulta Left Join de las tablas miembros y comites

```
SELECT
        m.id,
        m.nombre
        miembro,
        c.id,
        c.nombre comite
FROM miembros m
LEFT JOIN
comites c
USING(nombre)
```

id ▲ 1	miembro	id	comite
1	John	1	John
2	Jane	*NULL*	*NULL*
3	Mary	2	Mary
4	David	*NULL*	*NULL*
5	Amelia	3	Amelia

LEFT JOIN

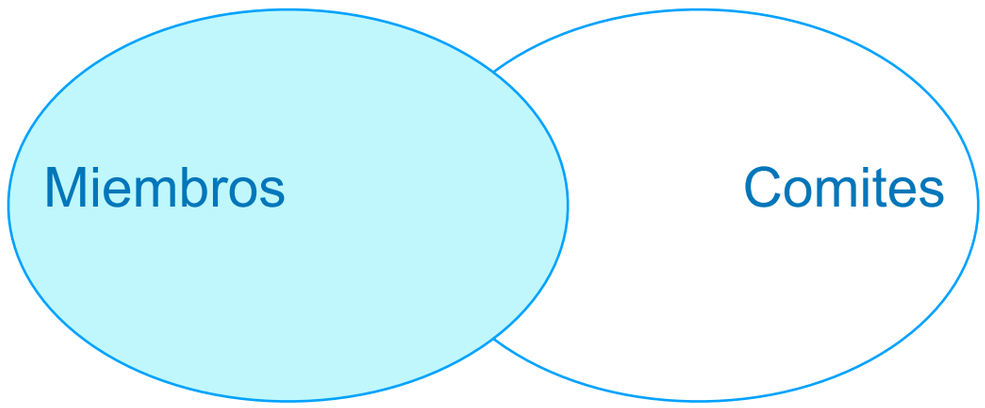
Ejemplo 2:

Buscar miembros que no sean miembros de algún comité.

```
SELECT
      m.id,
      m.nombre miembro,
      c.id,
      c.nombre comite
FROM miembros m
LEFT JOIN comites c
USING(nombre)
WHERE c.id IS NULL
```

id	miembro	id	comite
2	Jane	*NULL*	*NULL*
4	David	*NULL*	*NULL*

La cláusula WHERE se ejecuta a partir de los resultados de la consulta LEFT JOIN



RIGHT JOIN

Selecciona datos a partir de la tabla derecha.
Para cada fila de la tabla derecha, se compara con cada fila de la tabla izquierda.
Si los valores coinciden con los de la tabla izquierda se crea una fila con los datos de ambas tablas; de lo contrario se crea una fila con los datos de la tabla derecha y se rellena con NULL los datos de la tabla izquierda.

Ejemplo 1: Crear la consulta Right Join de las tablas miembros y comites

```
SELECT
      miembros.id,
      miembros.nombre miembro,
      comites.id,
      comites.nombre comite
FROM miembros
RIGHT JOIN comites
ON comites.nombre = miembros.nombre
```

id	miembro	id	comite
1	John	1	John
3	Mary	2	Mary
5	Amelia	3	Amelia
NULL	*NULL*	4	Joe

RIGHT JOIN

Ejemplo 2:
Encontrar los miembros de algún comité que no están en la tabla miembros.

```
SELECT
      m.id,
      m.nombre miembro,
      c.id,
      c.nombre comite
FROM
      miembros m
RIGHT JOIN
      comites c
USING(nombre)
WHERE m.id IS NULL
```

id	miembro	id	comite
NULL	*NULL*	4	Joe

CROSS JOIN

Realiza el producto cartesiano de las filas de la tabla izquierda con cada una de las filas de la tabla derecha.
Si la tabla A tiene N filas y la tabla B tiene M filas, el resultado de Cross Join será NxM filas

Ejemplo:

```
SELECT
    m.id,
    m.nombre miembro,
    c.id,
    c.nombre comite
FROM
    miembros m
CROSS JOIN comites c
```

id	miembro	id	comite
1	John	1	John
1	John	2	Mary
1	John	3	Amelia
1	John	4	Joe
2	Jane	1	John
2	Jane	2	Mary
2	Jane	3	Amelia
2	Jane	4	Joe
3	Mary	1	John
3	Mary	2	Mary
3	Mary	3	Amelia
3	Mary	4	Joe
4	David	1	John
4	David	2	Mary
4	David	3	Amelia
4	David	4	Joe
5	Amelia	1	John
5	Amelia	2	Mary
5	Amelia	3	Amelia
5	Amelia	4	Joe

EJERCICIOS