Tema 9

Consultas multitabla.

Los datos de diferentes tablas se pueden relacionar.

Hay varias formas:

- Producto cartesiano.
- Composición:
  - Interna: producto cartesiano restringido.
    - Normal: `INNER JOIN`.
    - Cruzada: `CROSS JOIN`.
    - Natural: `NATURAL JOIN`.
  - Externa: no proceden de un producto cartesiano: pueden aparecer tuplas que no aparecen en el producto cartesiano. Si no existe tupla con la que combinar, se combina con una de valores nulos.
    - `LEFT [OUTER] JOIN`.
    - `RIGHT [OUTER] JOIN`.
    - `FULL [OUTER] JOIN`.
    - `NATURAL LEFT [OUTER] JOIN`.
    - `NATURAL RIGHT [OUTER] JOIN`.
    - `NATURAL FULL [OUTER] JOIN`.
- Otras:
  - `UNION`.
  - `INTERSECT`.
  - `MINUS`.

9.1. Producto cartesiano.

Todas las posibles combinaciones entre los datos de dos (o más) tablas. (Figura 9.1).

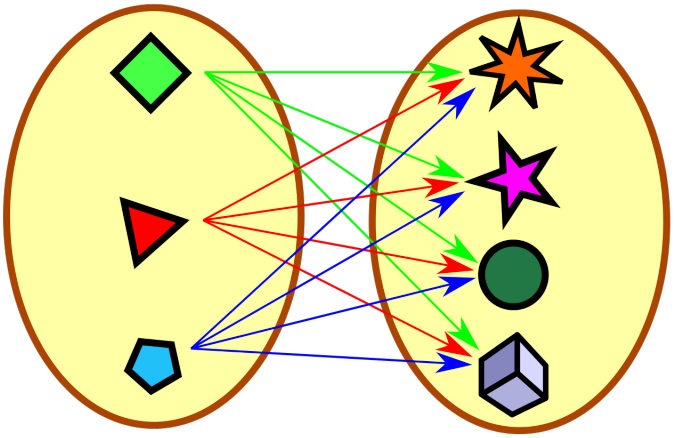

Figura 9.1: Tipos de Joins. Imagen encontrada aquí. De GermanX - Trabajo propio, CC BY-SA 4.0

No suele ser muy útil ya que no relaciona los datos, solo los combina.

Muchas de las alternativas que vamos viendo se pueden aplicar a los casos que iremos viendo más adelante.

```
SELECT * FROM empleados, departamentos;

SELECT
   empleados.apellido, departamento.nombre_departamento
FROM
   empleados,
   departamentos;
```

```
/* Usando Alias */
SELECT
   E.apellido, D.nombre_departamento
FROM
   empleados E,
   departamentos D;
```

```sql
/* Si no hay confusión en nombres de columnas
   no es necesario especificar */
SELECT
   apellido, nombre_departamento
FROM
   empleados,
   departamentos;
```

9.2. Composiciones internas.

Para evitar todas las combinaciones relacionamos haciendo que se cumpla una determinada condición.

Hay varios tipos: (Figura 9.2).

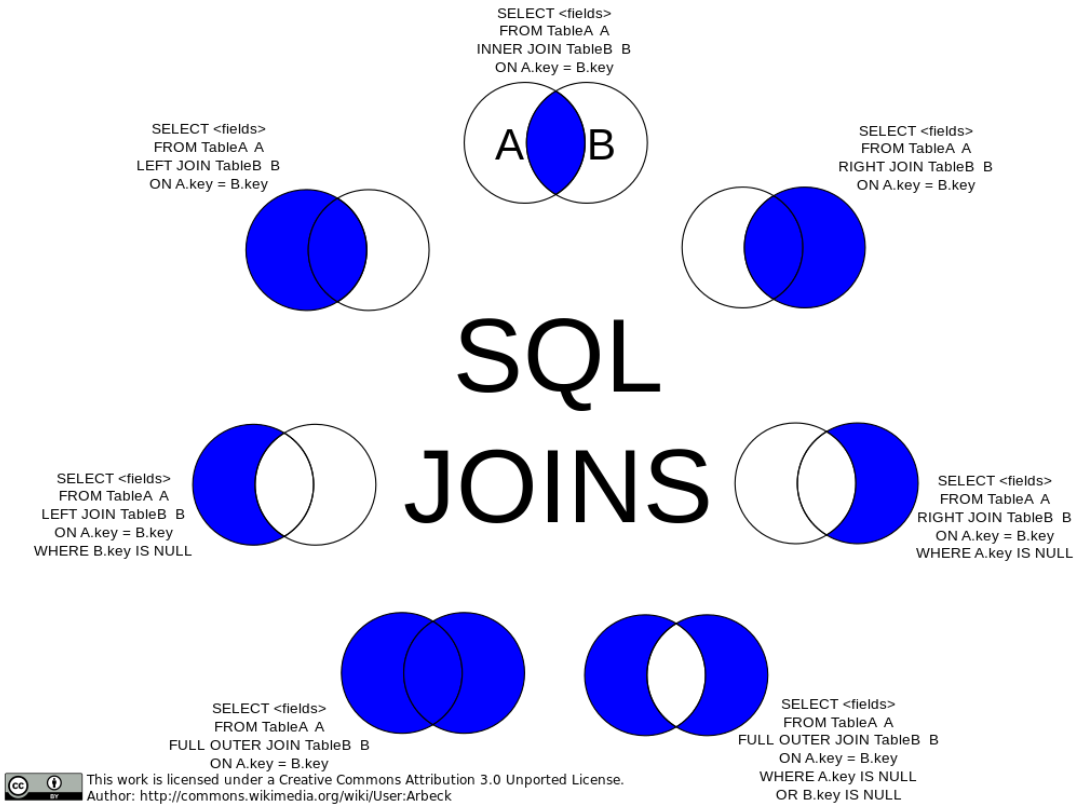

Figura 9.2: Tipos de Joins. Imagen encontrada aquí

9.2.1. Composición interna: `INNER JOIN`.

Combina las filas si se da una coincidencia entre las columnas usadas en la condición. (Figura 9.3).

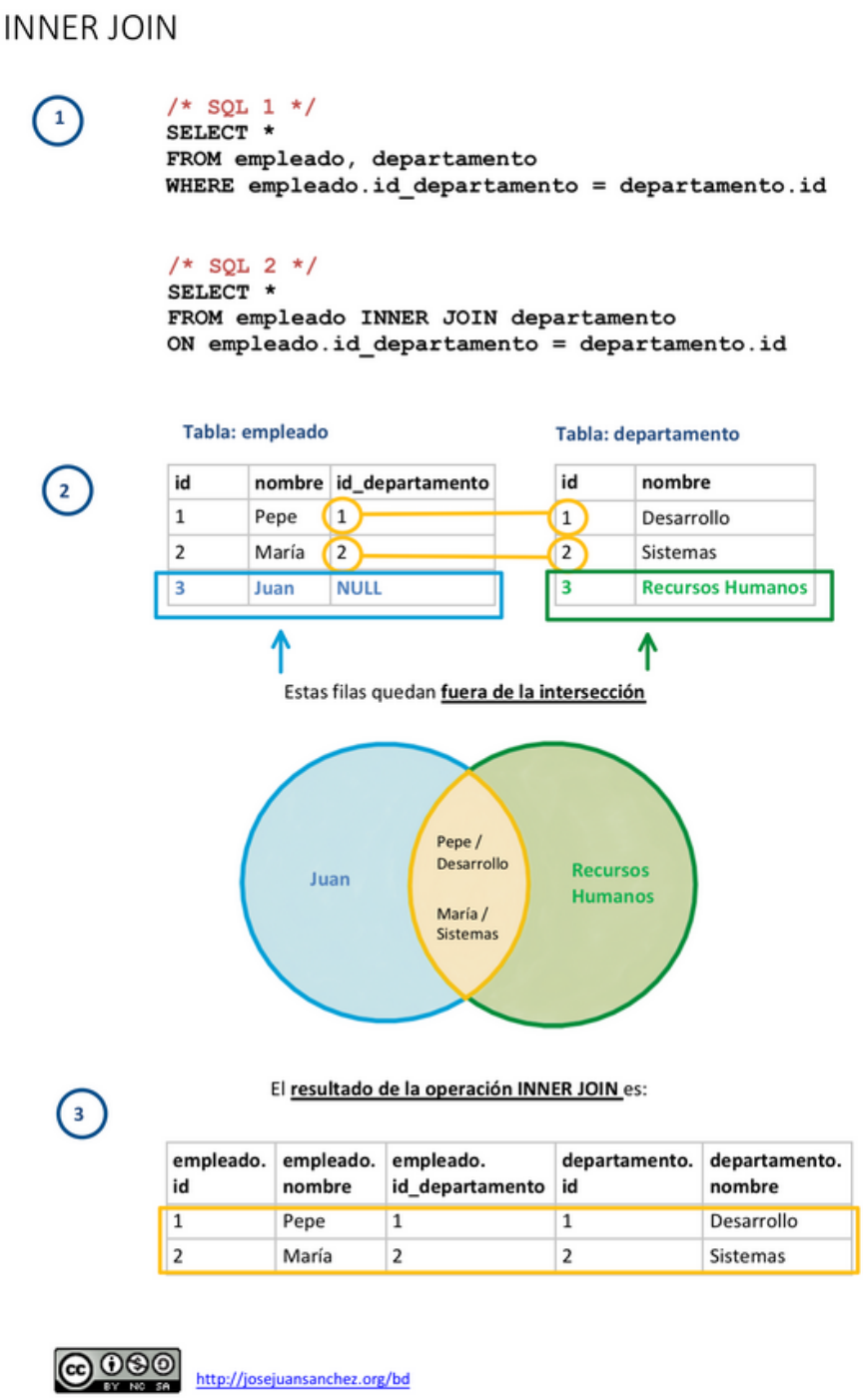

Figura 9.3: Inner Join. Por Juan José Sánchez

```
SELECT
   apellido, nombre_departamento
FROM
   empleados
      INNER JOIN
   departamentos ON
      empleados.id_departamento = departamentos.id_departamento;
```


Nota

El uso de `INNER` es opcional.

```
SELECT
   apellido, nombre_departamento
FROM
   empleados E
      JOIN
   departamentos D ON E.id_departamento = D.id_departamento;
```

Nota

Si no ponemos la cláusula `ON` obtendremos el producto cartesiano.

Si el campo de la condición se llama igual en ambas tablas:

```
SELECT
   apellido, nombre_departamento
FROM
   empleados
      JOIN
   departamentos USING (id_departamento);
```

Hay otra sintáxis equivalente (muy habitual) que no usa la palabra `JOIN`:

```
SELECT
   apellido, nombre_departamento
FROM
   empleados E,
   departamentos D
WHERE
   E.id_departamento = D.id_departamento;
```


¡CUIDADO!

Un descuido muy habitual es olvidarse de combinar las tablas con `WHERE`, `ON`, `USING`, etc.

9.2.2. Natural: `NATURAL JOIN`.

Otra sintáxis para cuando usamos un campo que se llama igual en ambas tablas en una composición interna.

```
SELECT
   direccion, ciudad, nombre_pais
FROM
   localizaciones
      NATURAL JOIN
   paises;
```

¡CUIDADO!

Unirá ambas tablas por TODOS los campos que se llamen igual en las dos, esto puede dar lugar a efectos no deseados, por ejemplo en empleados y departamentos existe `id_director` con significados diferentes (director del departamentoo jefe de una persona).

9.2.3. `CROSS JOIN`.

Dos situaciones:

- Si NO hay claúsula `WHERE` que se aplique al `CROSS JOIN` es como un producto cartesiano.
- Si SÍ hay claúsula `WHERE` que se aplique al `CROSS JOIN` es como una composición interna.


```sql
/* Equivalente a producto cartesiano */
SELECT
   apellido, nombre_departamento
FROM
   empleados
      CROSS JOIN
   departamentos;
```

```sql
/* Equivalente a INNER JOIN */
SELECT
   apellido, nombre_departamento
FROM
   empleados E
      CROSS JOIN
   departamentos D
WHERE
   E.id_departamento = D.id_departamento;
```

Nota

El uso de `CROSS` es opcional.

Ejercicios 9.1

- Devuelve el nombre, apellido y nombre de departamento de todos los empleados con todas las sintaxis vistas para el `INNER JOIN` incluyendo el `NATURAL JOIN`.
- Devuelve el nombre de todos los departamentos que se encuentren en Estados Unidos.
- Selecciona todos los empleados que pertenezcan al departamento ejecutivo o al de administración.
- Selecciona los empleados con un salario que no esté entre los límites establecidos para su trabajo.

9.3. Composiciones externas.

9.3.1. `LEFT (OUTER) JOIN`

Si queremos que aparezcan todas las filas de la tabla de la izquierda aunque no tengan correspondencia en la de la derecha. (Figura 9.4).

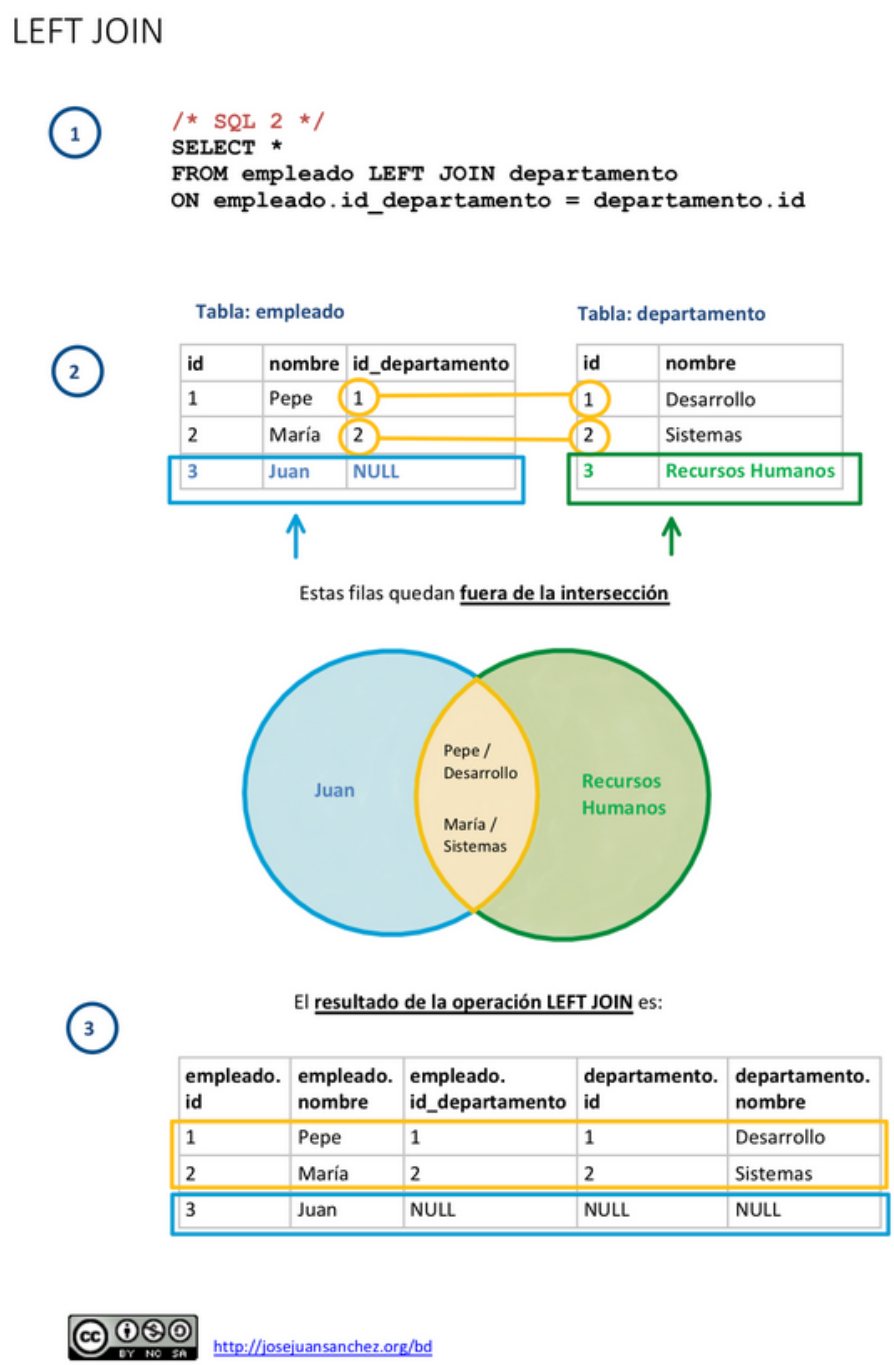

Figura 9.4: Left Join. Por Juan José Sánchez

```sql
SELECT
   apellido, nombre_departamento
FROM
   empleados
      LEFT JOIN  /* equivalente a poner LEFT OUTER JOIN */
   departamentos
      ON empleados.id_departamento = departamentos.id_departamento;
```

Vemos como aparecen también los empleados que no pertenecen a ningún departamento. (Figura 9.5).

Gonzaga	Ventas
Schroeder	NULL
Kemp	Ventas
Johnson	Ventas
Hess	Ventas
Martín	Ventas
Gonzalez	Ventas
Ferreras	NULL
Fenix	Contabilidad

Figura 9.5: Left Join

9.3.2. RIGHT (OUTER) JOIN

Lo contrario: si queremos que aparezcan todas las filas de la tabla de la derecha aunque no tengan correspondencia en la de la izquierda.

```sql
SELECT
   apellido, nombre_departamento
FROM
   empleados
      RIGHT JOIN  /* equivalente a poner RIGHT OUTER JOIN */
   departamentos
      ON empleados.id_empleado = departamentos.id_director;
```


Nota

Es la primera vez que combinamos tablas por un campo que no se llama igual en ambas tablas. Es muy habitual.
Vamos a volver a mirar el modelo E/R y físico de la BdD de empleados para entender un poco mejor esto.

Vemos como aparece también el departamento que no tiene director. (Figura 9.6).

López	Ejecutivo
Ernst	Administración
Gonzales	Marketing
Van Pobel	Envíos
García	Investigación y Desarrollo
Chopenhauer	Ventas
Fenix	Contabilidad
NULL	Recursos Humanos

Figura 9.6: Right Join

9.3.3. `FULL (OUTER) JOIN`

Combinación de las dos anteriores.

```
SELECT
   apellido, nombre_departamento
FROM
   empleados
      FULL JOIN  /* equivalente a poner FULL OUTER JOIN */
   departamentos
      ON empleados.id_empleado = departamentos.id_director;
```

Nota

MySQL no implementa `FULL JOIN` por lo que se puede simular con una unión, que veremos a continuación.

```
SELECT
   apellido, nombre_departamento
FROM
   empleados
      LEFT JOIN
   departamentos
      ON empleados.id_departamento = departamentos.id_departamento
UNION
SELECT
   apellido, nombre_departamento
FROM
   empleados
      RIGHT JOIN
   departamentos
      ON empleados.id_departamento = departamentos.id_departamento;
```

Aparecen filas sin correspondencia en ambos lados. (Figura 9.7).

Figura 9.7: Full Join

9.3.4. `NATURAL LEFT JOIN` y `NATURAL RIGHT JOIN`

Si se combinan por el campo que se llame igual.


```sql
SELECT
   direccion, ciudad, nombre_pais
FROM
   localizaciones
      NATURAL RIGHT JOIN /* equivale a NATURAL RIGHT OUTER JOIN */
   paises;
```

¡CUIDADO!

Tal y como vimos en `NATURAL JOIN` si hay varios campos que se llaman igual, el efecto puede no ser el deseado, como se puede ver si ejecutas la siguiente consulta, ya que existe `id_director` con significados diferentes (director del departamentoo jefe de una persona).

```sql
SELECT
   apellido, nombre_departamento
FROM
   empleados
      NATURAL LEFT JOIN
   departamentos;
```

9.4. Otras consultas multitabla.

Para poder realizar las siguientes operaciones se deben cumplir unas condiciones en las columnas seleccionadas:

- Mismo número.
- Mismo tipo de datos.
- Mismo orden.

9.4.1. `UNION`.

Combinación de los elementos resultantes de ambas consultas. (Figura 9.8).

Es como un *OR* lógico.

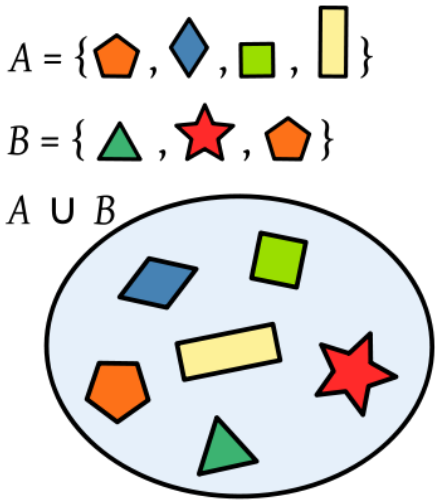

Figura 9.8: Unión. Imagen encontrada aquí.

```
SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario BETWEEN 2000 AND 3000
UNION SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario > 8000;
```

Evita repetidos. (Añado `ORDER BY` para facilitar comprobarlo).


```
SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario BETWEEN 5000 AND 9000
UNION SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario > 8000
ORDER BY salario, apellido, nombre;
```

9.4.1.1. `UNION ALL`.

No elimina los repetidos.

```
SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario BETWEEN 5000 AND 9000
UNION ALL SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario > 8000
ORDER BY salario, apellido, nombre;
```

9.4.2. `INTERSECT`.

Los que coinciden en ambas consultas. (Figura 9.9).

Es como un *AND* lógico.

No existe en MySQL pero la sintaxis sería:

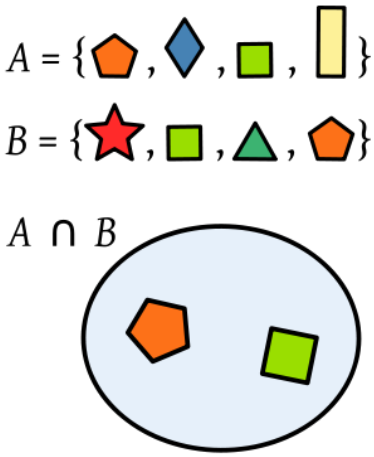

Figura 9.9: Intersección. Imagen encontrada aquí.

```sql
SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario BETWEEN 5000 AND 9000
INTERSECT SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario > 8000;
```

Se puede obtener el mismo resultado usando `IN`:


```
SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario BETWEEN 5000 AND 9000
      AND (nombre , apellido, salario) IN (SELECT
         nombre, apellido, salario
      FROM
         empleados
      WHERE
         salario > 8000);
```

Nota

Ten en cuenta que es necesario incluir todos los campos a seleccionar en el filtrado con `IN` para que el resultado sea el mismo que usando `INTERSECT`.

También se puede obtener el resultado usando `EXISTS` pero no lo veremos.

9.4.3. `MINUS`.

En algunos SGBDs se llama `EXCEPT`.

P.Ej: una academia de idiomas imparte inglés y francés. Se quiere enviar una comunicación a todos los alumnos. Ya se envió a los que estaban matriculados en francés así que al coger la lista de inglés deberíamos evitar a todos los que cursan ambos idiomas. (Figura 9.10).

No existe en MySQL pero la sintaxis sería:

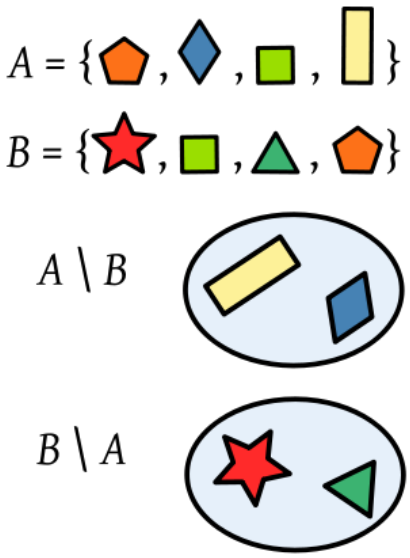

Figura 9.10: Diferencia. Imagen encontrada aquí.

```
SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario BETWEEN 3000 AND 6000
MINUS SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario BETWEEN 4000 AND 5000;
```

Se puede obtener el mismo resultado usando `NOT IN`:

```
SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario BETWEEN 3000 AND 6000
      AND (nombre , apellido, salario) NOT IN (SELECT
         nombre, apellido, salario
      FROM
         empleados
      WHERE
         salario BETWEEN 4000 AND 5000);
```

También se puede obtener el resultado usando `LEFT JOIN` e `IS NULL` pero no lo veremos.

Ejercicios 9.2

1. Selecciona el nombre de todos los países y si se tienen oficinas en alguna ciudad de dicho país, también el nombre de la ciudad. Usa al menos dos sintaxis diferentes.

   ESTE EJERCICIO QUEDÓ DEMASIADO COMPLEJO PARA LO VISTO HASTA AHORA, ASÍ QUE PONGO LA SOLUCIÓN Y ASÍ VEMOS CÓMO SE HACE.

   Lo que sucede es que hay que usar una sintaxis "diferente":

   ```
   SELECT DISTINCT
      nombre_pais, ciudad
   FROM
      paises P
         LEFT JOIN
      (localizaciones L
            JOIN departamentos D
            ON D.id_localizacion = L.id_localizacion)
         ON P.id_pais = L.id_pais;
   ```

   Lo que va dentro de los paréntesis es como si obtuviéramos el resultado de la siguiente consulta pero la sintaxis al combinarla tiene que ser como muestro arriba:


```
SELECT
   *
FROM
   localizaciones L
JOIN
   departamentos D ON D.id_localizacion = L.id_localizacion;
```

Realmente se puede usar la consulta completa pero hay que añadir un alias a la tabla temporal resultante:

```
SELECT DISTINCT
   nombre_pais, ciudad
FROM
   paises P
      LEFT JOIN
   (SELECT
      id_pais, ciudad
   FROM
      localizaciones L
         JOIN departamentos D
         ON D.id_localizacion = L.id_localizacion) AS DL
      ON P.id_pais = DL.id_pais;
```

2. Selecciona todos los países de la región 1 y los de la región 4.
3. Selecciona todos los países de América y los de Oriente Medio y África.
4. Selecciona el nombre de los departamentos que tengan más de tres empleados.
5. Selecciona el nombre de los departamentos que tengan más de tres empleados quitando el de aquellos que tengan más de 7;
6. Si has hecho la consulta anterior simulando un `MINUS` hazlo ahora sin usarlo, filtrando solo los grupos. En caso contrario, hazlo simulando un `MINUS`.

9.5. Apuntes sobre consultas multitabla.

9.5.1. Orden de las tablas.

El orden en el que combinemos las tablas no afecta a:

- `INNER JOIN`.
- `CROSS JOIN`.
- `NATURAL JOIN`.
- `FULL [OUTER] JOIN`.
- `NATURAL FULL [OUTER] JOIN`.
- `UNION`.
- `INTERSECT`.

9.5.2. Unir tres o más tablas.

Por ejemplo:

```
SELECT
   nombre_region, nombre_pais, ciudad
FROM
   regiones R
      JOIN
   paises P ON R.id_region = P.id_region
      JOIN
   localizaciones L ON P.id_pais = L.id_pais;
```

Si la estructura lo permite se pueden hacer de forma natural:

```
SELECT
   nombre_region, nombre_pais, ciudad
FROM
   regiones
      NATURAL JOIN
   paises
      NATURAL JOIN
   localizaciones;
```

O siempre con sintaxis clásica:

```
SELECT
   nombre_region, nombre_pais, ciudad
FROM
   regiones R,
   paises P,
   localizaciones L
WHERE
   R.id_region = P.id_region
      AND P.id_pais = L.id_pais;
```

Importante

Siempre hará falta una condición menos que el número de tablas.

9.5.3. Relaciones reflexivas.

De una tabla consigo misma.

```
SELECT
   E.nombre 'Emp Nom',
   E.apellido 'Emp Ape',
   J.nombre 'Jefe Nom',
   J.apellido 'Jefe Ape'
FROM
   empleados E
JOIN
   empleados J ON E.id_director = J.id_empleado;
```

Sintaxis clásica:

```sql
SELECT
   E.nombre 'Emp Nom',
   E.apellido 'Emp Ape',
   J.nombre 'Jefe Nom',
   J.apellido 'Jefe Ape'
FROM
   empleados E,
   empleados J
WHERE
   E.id_director = J.id_empleado;
```

Ejercicios 9.3

1. Obtén la ciudad y el código postal donde trabaja María González.
2. Obtén la ciudad y el código postal donde trabajan los empleados cuyo nombre empiece por M.
3. Selecciona el nombre del trabajo, nombre del departamento en el que lo ejerció, la fecha en la que empezó y la fecha en la que terminó para todos los trabajos anteriores que haya tenido Sara James en la empresa.
4. Selecciona los datos de todos los empleados cuya directora sea Maya Van Pobel.
5. Obtén el nombre y el apellido del director de departamento en el que trabaja Indira Virma.

9.6. Bibliografía.

- José Luis Comesaña (2011). Curso MEFP-IFCS03-BD. Ministerio de Educación y Formación Profesional.
- Structured Query Language. Wikibook. Wikipedia.
- Curso de MySQL. Con Clase.
- MySQL 8.0 Reference Manual.
- José Juan Sánchez Hernández (2021/2022). Bases de datos / Gestión de Bases de datos
- Iván López Montalbán, Manuel de Castro Vázquez, John Ospino Rivas (2022). Bases de Datos (2ª Edición) . Garceta.

