Guía visual y práctica de JOINS en SQL

Dataset de alumnos–matrículas para INNER, LEFT, RIGHT y FULL (emulado)

Departamento de Informática

October 17, 2025

Contents

1 Dataset

Objetivo

Vamos a descubrir, primero de forma visual y luego con SQL, las diferencias entre **INNER**, **LEFT**, **RIGHT** y **FULL** (emulado en MySQL).

Recuerda que esta base de datos la tienes disponible para cargarla en dbs/matriculaciones_tiposJoin.

Tablas

- `alumnos`(*id_alumno*, nombre)
- `cursos`(*id_curso*, nombre_curso)
- `matriculas`(*id_matricula*, id_alumno, id_curso)

Notación: Esta forma en la que se escriben las tablas se denomina modelo relacional. Se indica el nombre de la tabla y, entre paréntesis, los atributos que tiene. Se indica en cursiva la clave primaria.

Datos (mirada previa sin SQL)

`alumnos`

id_alumno	nombre
1	Ana
2	Bruno
3	Cris
4	Dani
5	Eva

`cursos`

id_curso	nombre_curso
101	BBDD
102	Programación
103	Sistemas
104	Redes

`matriculas`

id_matricula	id_alumno	id_curso
1001	1	101
1002	1	102
1003	3	101
1004	6	103
1005	4	105

Consultas a mano

Ahora, contesta estas preguntas a partir de la información presentada y fíjate en el proceso que sigues para obtener la información (tablas consultadas, como sabes si hay o no matrícula, etc).

1. ¿Qué `id_alumno` coinciden entre `alumnos` y `matriculas`?
2. ¿Qué alumnos *no* tienen matrícula? (pista: Bruno, Eva)
3. ¿Qué matrículas *no* tienen alumno? (`id_alumno`=6)
4. ¿Qué cursos del catálogo no tienen matrículas? (Redes=104)
5. ¿Qué matrículas apuntan a cursos inexistentes? (`id_curso`=105)

No todas estas preguntas se contestan con un `join` de los que hemos estado haciendo hasta ahora. ¿Qué ocurre al hacer un `join` de `alumnos` con `matriculas` si hay un alumno que no tiene matrícula asociada? ¿Aparece o no? Realiza la consulta y compruébalo.

```sql
SELECT a.id_alumno, a.nombre, m.id_matricula, m.id_curso
FROM alumnos a
JOIN matriculas m
  ON m.id_alumno = a.id_alumno
ORDER BY a.id_alumno, m.id_matricula;
```

id_alumno	nombre	id_matricula	id_curso
1	Ana	1001	101
1	Ana	1002	102
3	Cris	1003	101
4	Dani	1005	105

¿Y si quiero que me aparezcan los alumnos que no tienen matrículas asociadas? ¿Cómo puedo conseguirlo? ¿Cómo puedo sacar algo así?

id_alumno	nombre	id_matricula	id_curso
1	Ana	1001	101
1	Ana	1002	102
2	Bruno	NULL	
3	Cris	1003	101
4	Dani	1005	105
5	Eva	NULL	

Para preguntas como esas, podemos utilizar distintos tipos de JOINs.

2 Tipos de Joins

2.1 INNER JOIN: sólo emparejados

Idea

Intersección: devuelve únicamente las filas que cumplen la condición en ambas tablas.

```
SELECT a.id_alumno, a.nombre, m.id_matricula, m.id_curso
FROM alumnos a
INNER JOIN matriculas m
  ON m.id_alumno = a.id_alumno
ORDER BY a.id_alumno, m.id_matricula;
```

Resultado esperado (4 filas)

id_alumno	nombre	id_matricula	id_curso
1	Ana	1001	101
1	Ana	1002	102
3	Cris	1003	101
4	Dani	1005	105

2.2 LEFT JOIN: todos los de la izquierda

Idea

Todo A (alumnos) y lo que case en B (matrículas). Si no casa, columnas de la derecha como `NULL`.

```
SELECT a.id_alumno, a.nombre, m.id_matricula, m.id_curso
FROM alumnos a
LEFT JOIN matriculas m
  ON m.id_alumno = a.id_alumno
ORDER BY a.id_alumno, m.id_matricula;
```

Resultado esperado (6 filas)

id_alumno	nombre	id_matricula	id_curso
1	Ana	1001	101
1	Ana	1002	102
2	Bruno	NULL	
3	Cris	1003	101
4	Dani	1005	105
5	Eva	NULL	

2.3 RIGHT JOIN: simétrico del LEFT

Idea

Todo B (matrículas) y lo que case en A (alumnos). Útil para detectar huérfanas en la derecha.

```
SELECT a.id_alumno, a.nombre, m.id_matricula, m.id_curso
FROM alumnos a
RIGHT JOIN matriculas m
```

```
  ON m.id_alumno = a.id_alumno
ORDER BY m.id_matricula;
```

Resultado esperado (5 filas)

id_alumno	nombre	id_matricula	id_curso
1	Ana	1001	101
1	Ana	1002	102
3	Cris	1003	101
NULL		1004	103
4	Dani	1005	105

Left or right: dependen del orden en el que escribas las tablas en la consulta. Para convertir un `left` en un `right`, cambia de orden las tablas. Te dejo como reto que construyas las consultas de `left join` utilizando un `right join` y viceversa.

2.4 FULL OUTER JOIN (emulado en MySQL)

Idea

Todo de ambos lados: LEFT + sólo-lo-de-la-derecha (RIGHT con `WHERE a.id_alumno IS NULL`) y `UNION`.

Este tipo de `join` se utiliza poco y no viene por defecto en mysql. Por eso, se hace primero una consulta `left` y luego la otra `right` realizando la unión de ambas. La **unión** simplemente selecciona todas las filas que hayan salido en alguna de las dos consultas.

```
SELECT a.id_alumno, a.nombre, m.id_matricula, m.id_curso
FROM alumnos a
LEFT JOIN matriculas m
  ON m.id_alumno = a.id_alumno
UNION
SELECT a.id_alumno, a.nombre, m.id_matricula, m.id_curso
FROM alumnos a
RIGHT JOIN matriculas m
  ON m.id_alumno = a.id_alumno
WHERE a.id_alumno IS NULL
ORDER BY 1 IS NULL, a.id_alumno, m.id_matricula;
```

Resultado esperado (7 filas)

id_alumno	nombre	id_matricula	id_curso
1	Ana	1001	101
1	Ana	1002	102
2	Bruno	NULL	
3	Cris	1003	101
4	Dani	1005	105
5	Eva	NULL	
NULL		1004	103

Bonus: sumar cursos para ver matrículas “válidas”

```
SELECT m.id_matricula,
       a.nombre AS alumno,
       c.nombre_curso AS curso
```

```
FROM matriculas m
LEFT JOIN alumnos a ON a.id_alumno = m.id_alumno
LEFT JOIN cursos c ON c.id_curso = m.id_curso
ORDER BY m.id_matricula;
```

¿Cuál es el `join` por defecto?

Viendo lo visto, ¿cuál es el `join` por defecto?

3 Ejercicios: enunciados y resultados esperados

Nota: En este apartado *no* se incluyen las soluciones SQL. Están al final, en el Anexo.

Ejercicio 1

Enunciado: Listado de alumnos con sus `id_curso` (sólo emparejados).
Resultado esperado (4 filas):

id_alumno	nombre	id_curso
1	Ana	101
1	Ana	102
3	Cris	101
4	Dani	105

Ejercicio 2

Enunciado: Alumnos sin ninguna matrícula (anti-join).
Resultado esperado (2 filas):

id_alumno	nombre
2	Bruno
5	Eva

Ejercicio 3

Enunciado: Matrículas sin alumno (huérfanas).
Resultado esperado (1 fila):

id_matricula	id_alumno	id_curso
1004	6	103

Ejercicio 4

Enunciado: Cursos del catálogo sin ninguna matrícula.
Resultado esperado (1 fila):

id_curso	nombre_curso
104	Redes

Ejercicio 5

Enunciado: Número de matrículas por alumno (incluye 0).
Resultado esperado (5 filas):

id_alumno	nombre	n_matriculas
1	Ana	2
2	Bruno	0
3	Cris	1
4	Dani	1
5	Eva	0

Ejercicio 6

Enunciado: Alumnos con más de un curso.
Resultado esperado (1 fila):

id_alumno	nombre	n
1	Ana	2

Ejercicio 7

Enunciado: FULL OUTER JOIN emulado (alumnos y sus matrículas, incluyendo huérfanas).
Resultado esperado (7 filas): las 6 del LEFT + la fila `1004`.

Ejercicio 8

Enunciado: Para cada curso del catálogo, número de alumnos con matrícula válida (alumno y curso existen).
Resultado esperado (4 filas):

id_curso	nombre_curso	n_alumnos
101	BBDD	2
102	Programación	1
103	Sistemas	0
104	Redes	0

4 Tipo test

Marca una opción por pregunta.

1. INNER JOIN devuelve...
   A. Todas las filas de la izquierda.
   B. Solo las filas que cumplen la condición en ambas tablas.
   C. Todas las filas de ambas tablas.
   D. Solo las filas no emparejadas.
2. En un LEFT JOIN, las filas no emparejadas de la tabla derecha...
   A. Se descartan.
   B. Se rellenan como `NULL` en sus columnas.

C. Se duplican.

D. Se convierten a 0.

3. En un RIGHT JOIN, ¿qué se garantiza?

A. Ver todas las filas de la izquierda.

B. Ver todas las filas de la derecha.

C. Ver solo la intersección.

D. Ver ninguna fila `NULL`.

4. En MySQL, FULL OUTER JOIN...

A. Existe y se escribe `FULL JOIN`.

B. Existe, pero solo con `NATURAL`.

C. No existe; se emula con LEFT + RIGHT y UNION.

D. Solo en vistas.

5. En un LEFT JOIN, poner en `WHERE` una condición sobre la tabla derecha (`WHERE m.id_curso=101`)...

A. No cambia nada.

B. Puede hacer que el resultado se comporte como un INNER (filtra las `NULL`).

C. Duplica filas nulas.

D. Lanza error.

6. Para encontrar alumnos sin matrícula, lo más idiomático es...

A. INNER JOIN + `HAVING COUNT(*)=0`.

B. LEFT JOIN y `WHERE m.id_matricula IS NULL`.

C. RIGHT JOIN y `WHERE a.id_alumno IS NULL`.

D. FULL JOIN.

7. Detectar matrículas huérfanas de alumno:

A. LEFT JOIN alumnos→matriculas + `IS NULL` en `id_matricula`.

B. RIGHT JOIN alumnos←matriculas + `IS NULL` en `a.id_alumno`.

C. INNER JOIN con `WHERE a.id_alumno IS NULL`.

D. LEFT JOIN cursos→matriculas + `IS NULL` en `id_matricula`.

8. Si haces INNER JOIN `matriculas m` con `cursos c` ON `m.id_curso=c.id_curso`, ¿aparece la matrícula 1005?

A. Sí, con `NULL`.

B. No, porque 105 no existe en `cursos`.

C. Sí, siempre.

D. Solo con `USING`.

9. Para listar todos los cursos y cuántos alumnos tienen (incluye 0), lo más correcto es...

A. INNER JOIN cursos ↔ matriculas.

B. LEFT JOIN desde `cursos` hacia `matriculas` y `GROUP BY`.

C. RIGHT JOIN desde `matriculas`.

D. FULL JOIN.

10. En el FULL emulado, la parte `RIGHT ... WHERE a.id_alumno IS NULL` sirve para...

A. Quitar duplicados del `UNION`.

B. Añadir solo las filas que están solo en la derecha (huérfanas).

C. Ordenar el resultado.

D. Evitar `NULL` en la izquierda.

Anexo: Soluciones

Soluciones de los ejercicios

1.
```sql
SELECT a.id_alumno, a.nombre, m.id_curso
FROM alumnos a
INNER JOIN matriculas m ON m.id_alumno = a.id_alumno
ORDER BY a.id_alumno, m.id_curso;
```

2.
```sql
SELECT a.id_alumno, a.nombre
FROM alumnos a
LEFT JOIN matriculas m ON m.id_alumno = a.id_alumno
WHERE m.id_matricula IS NULL
ORDER BY a.id_alumno;
```

3.
```sql
SELECT m.id_matricula, m.id_alumno, m.id_curso
FROM alumnos a
RIGHT JOIN matriculas m ON m.id_alumno = a.id_alumno
WHERE a.id_alumno IS NULL
ORDER BY m.id_matricula;
```

4.
```sql
SELECT c.id_curso, c.nombre_curso
FROM cursos c
LEFT JOIN matriculas m ON m.id_curso = c.id_curso
WHERE m.id_matricula IS NULL
ORDER BY c.id_curso;
```

5.
```sql
SELECT a.id_alumno, a.nombre, COUNT(m.id_matricula) AS n_matriculas
FROM alumnos a
LEFT JOIN matriculas m ON m.id_alumno = a.id_alumno
GROUP BY a.id_alumno, a.nombre
ORDER BY a.id_alumno;
```

6.
```sql
SELECT a.id_alumno, a.nombre, COUNT(*) AS n
FROM alumnos a
INNER JOIN matriculas m ON m.id_alumno = a.id_alumno
GROUP BY a.id_alumno, a.nombre
HAVING COUNT(*) > 1
ORDER BY a.id_alumno;
```

7.
```sql
SELECT a.id_alumno, a.nombre, m.id_matricula, m.id_curso
FROM alumnos a
LEFT JOIN matriculas m ON m.id_alumno = a.id_alumno
UNION
SELECT a.id_alumno, a.nombre, m.id_matricula, m.id_curso
FROM alumnos a
RIGHT JOIN matriculas m ON m.id_alumno = a.id_alumno
```

```sql
WHERE a.id_alumno IS NULL
ORDER BY 1 IS NULL, a.id_alumno, m.id_matricula;
```

8.
```sql
SELECT c.id_curso, c.nombre_curso,
       COUNT(a.id_alumno) AS n_alumnos
FROM cursos c
LEFT JOIN matriculas m ON m.id_curso = c.id_curso
LEFT JOIN alumnos a ON a.id_alumno = m.id_alumno
GROUP BY c.id_curso, c.nombre_curso
ORDER BY c.id_curso;
```

Plantilla de corrección del tipo test

1	2	3	4	5	6	7	8	9	10
B	B	B	C	B	B	B	B	B	B