Joins

Bases de Datos 1 (CC3088) - 2026

Joins

Semestre 1, 2025

Definición

Un JOIN permite combinar registros de dos o más tablas basándose en una relación entre columnas.

- Se usa cuando necesitamos consultar datos que están en diferentes tablas.
- Las relaciones suelen establecerse con claves foráneas.

Tipos de JOINs

INNER JOIN

Devuelve las filas que tienen coincidencias en ambas tablas.

```
SELECT *
FROM empleados AS e
INNER JOIN departamentos AS d ON e.departamento_id = d.id;
```

Solo se muestran empleados que están asignados a un departamento existente.

LEFT JOIN

Devuelve todas las filas de la tabla izquierda y las coincidencias de la tabla derecha. Si no hay coincidencia, muestra NULL.

```
SELECT *
FROM empleados AS e
LEFT JOIN departamentos AS d ON e.departamento_id = d.id;
```

Útil para identificar empleados sin departamento asignado.

RIGHT JOIN

Devuelve todas las filas de la tabla derecha y las coincidencias de la tabla izquierda.

```
SELECT *
FROM empleados AS e
RIGHT JOIN departamentos AS d ON e.departamento_id = d.id;
```

Podemos ver qué departamentos no tienen empleados asignados.

FULL OUTER JOIN

Devuelve todas las filas cuando hay coincidencia en una u otra tabla.

```
SELECT *
FROM empleados AS e
FULL OUTER JOIN departamentos AS d ON e.departamento_id = d.id;
```

Se incluyen empleados sin departamento y departamentos sin empleados.

CROSS JOIN

Realiza el producto cartesiano de ambas tablas.

```
SELECT *
FROM empleados CROSS JOIN turnos;
```

Genera todas las combinaciones posibles de registros.

SELF JOIN

Una tabla se une consigo misma usando alias.

```
SELECT e1.nombre AS empleado, e2.nombre AS jefe
FROM empleados AS  e1
JOIN empleados AS e2 ON e1.jefe_id = e2.id;
```

Útil para estructuras jerárquicas.

NATURAL JOIN y USING

NATURAL JOIN une las tablas automáticamente por columnas con el mismo nombre.

USING permite simplificar la cláusula ON si la columna tiene el mismo nombre en ambas tablas.

```
SELECT
    e.nombre,
    p.nombre AS proyecto
FROM empleados_proyectos
JOIN empleados AS e USING (empleado_id)
JOIN proyectos AS p USING (proyecto_id);
```

Buenas prácticas

- Usar alias para facilitar la lectura.
- Especificar siempre la condición ON (salvo USING).
- Cuidado con los NULL en LEFT/RIGHT/FULL JOINs.
- No abusar del CROSS JOIN.

Conclusiones

- Los JOINs son fundamentales para el análisis de datos.
- Cada tipo de JOIN tiene su uso dependiendo del contexto.
- La práctica constante es clave para dominarlos.