ArgentinaPrograma
YoPrograMo

Consultas relacionales

por Leonardo Blautzik, Federico Gasior y Lucas Videla

Julio / Diciembre 2021

Ministerio de
Desarrollo Productivo
Argentina

Consultas relacionales

El punto más fuerte de las bases de datos relacionales es bueno, justamente relacionarse

Utilizando SQL podemos realizar operaciones de junta, combinando columnas de una o más tablas

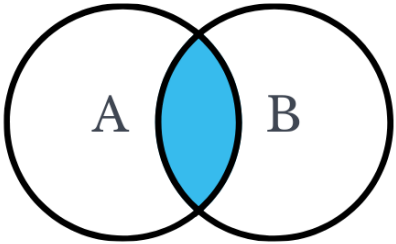
Las operaciones de junta del estandar SQL son llamados JOINS, y existen 5 tipos básicos

JOINS

Tipos de JOINS

- INNER (incluye NATURAL JOIN)
- LEFT OUTER
- RIGHT OUTER
- FULL OUTER
- CROSS

INNER JOIN

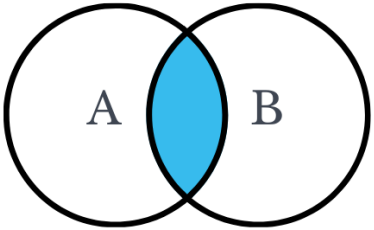



```
SELECT *
FROM tabla1 A
INNER JOIN tabla2 B
ON A.clave = B.clave
```

Figure 1: INNER JOIN

NATURAL JOIN

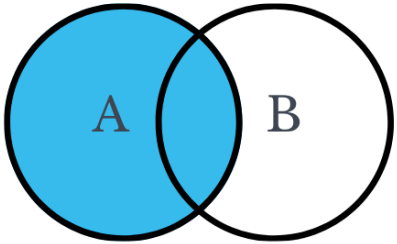
Consideramos que tenemos un atributo en común



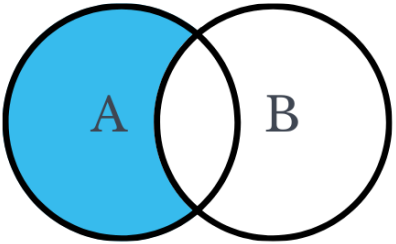

```
SELECT *
FROM tabla1 A
NATURAL JOIN tabla2 B
```

Figure 2: NATURAL JOIN

LEFT OUTER JOIN



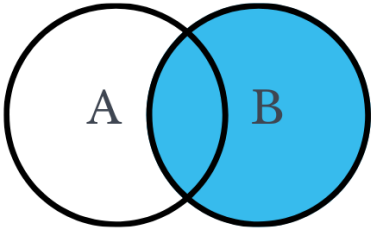

```
SELECT *
FROM tabla1 A
LEFT JOIN tabla2 B
ON A.clave = B.clave
```



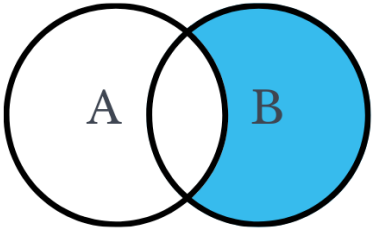

```
SELECT *
FROM tabla1 A
LEFT JOIN tabla2 B
ON A.clave = B.clave
WHERE B.clave IS NULL
```

Figure 3: LEFT OUTER JOIN

RIGHT OUTER JOIN



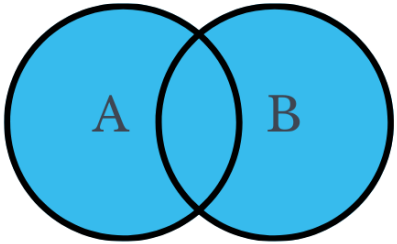

```
SELECT *
FROM tabla1 A
RIGHT JOIN tabla2 B
ON A.clave = B.clave
```



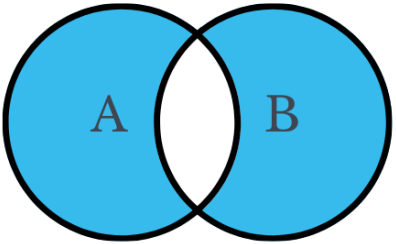

```
SELECT *
FROM tabla1 A
RIGHT JOIN tabla2 B
ON A.clave = B.clave
WHERE A.clave IS NULL
```

Figure 4: RIGHT OUTER JOIN

FULL OUTER JOIN



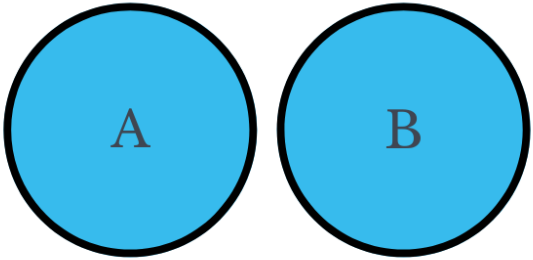

```
SELECT *
FROM tabla1 A
FULL JOIN tabla2 B
ON A.clave = B.clave
```




```
SELECT *
FROM tabla1 A
FULL JOIN tabla2 B
ON A.clave = B.clave
WHERE A.clave IS NULL
  OR B.clave IS NULL
```

Figure 5: FULL OUTER JOIN

CROSS JOIN




```
SELECT *
FROM tabla1 A
CROSS JOIN tabla2 B
```

```
SELECT *
FROM tabla1 A,
    tabla2 B
```

Figure 6: CROSS JOIN

Estudiantes y cursos

Vamos a mostrar un ejemplo trabajando con una base de datos de estudiantes y cursos

Tablas

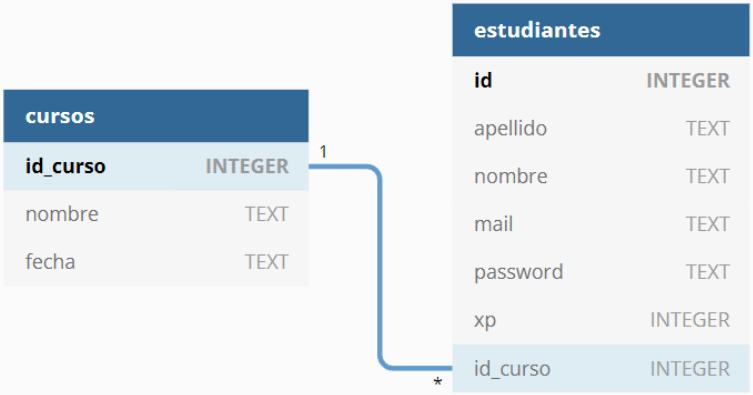


Figure 7: Tablas estudiantes y cursos

INNER JOIN

```
SELECT *
FROM estudiantes
/*INNER */JOIN cursos
  ON cursos.id_curso = estudiantes.id_curso
```

NATURAL JOIN

```
SELECT *
FROM estudiantes
CROSS JOIN cursos -- id_curso
```

LEFT OUTER JOIN

```sql
SELECT *
FROM estudiantes
LEFT /*OUTER */ JOIN cursos
  ON cursos.id_curso = estudiantes.id_curso
```

RIGHT OUTER JOIN

```
SELECT *
FROM estudiantes
RIGHT JOIN cursos
  ON cursos.id_curso = estudiantes.id_curso
```

RIGHT OUTER JOIN EN SQLITE

```
SELECT *
FROM cursos
LEFT JOIN estudiantes
  ON estudiantes.id_curso = cursos.id_curso
```

FULL OUTER JOIN

```sql
SELECT *
FROM estudiantes
FULL JOIN cursos
  ON cursos.id_curso = estudiantes.id_curso
-- ORDER BY id_curso
```

FULL OUTER JOIN EN SQLITE

```sql
SELECT *
FROM estudiantes
LEFT JOIN cursos
  ON cursos.id_curso = estudiantes.id_curso
UNION --ALL
SELECT *
FROM cursos
LEFT JOIN estudiantes
  ON estudiantes.id_curso = cursos.id_curso
WHERE estudiantes.id_curso IS NULL
```

CROSS JOIN

```sql
SELECT *
FROM estudiantes
CROSS JOIN cursos
-- LIMIT 100
```

USING

Se puede simplificar la clausula ON del JOIN en caso de que ambas columnas se llamen igual

```
SELECT *
FROM estudiantes
JOIN cursos
  -- ON cursos.id_curso = estudiantes.id_curso
  USING(id_curso)
```

Multiples tablas en FROM

```
SELECT *
FROM estudiantes, cursos
WHERE cursos.id_curso = estudiantes.id_curso
```

Auto JOIN o self JOIN

```
SELECT *
FROM cursos
JOIN cursos curso_padre
  ON curso_padre.id_curso_hijo = cursos.curso_hijo
```

Relación muchos a muchos

Creamos una tabla intermedia para manejar la relación que llamaremos **estudiantes_cursos**. La misma estara relacionada de 1 a muchos tanto con estudiantes como con cursos

```
SELECT estudiantes.*, cursos.*, estudiantes_cursos.fecha
FROM estudiantes
JOIN estudiantes_cursos
  ON estudiantes_cursos.id_estudiante = estudiantes.id
JOIN cursos
  ON cursos.id_curso = estudiantes_cursos.id_curso
```

¡Muchas Gracias!

continuará...

Ministerio de
Desarrollo Productivo
Argentina