SQL (IV)

DML
CONSULTAS A MÚLTIPLES TABLAS.
OPERACIONES CON JOIN

IES Valle del Jerte PLASENCIA Bilingual Section

UNION EUROPEA
Fondo Social Europeo
"Una manera de hacer Europa"

"**Una manera de hacer Europa**". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

CONSULTAS A MÚLTIPLES TABLAS

- Es muy habitual querer obtener datos que se encuentran distribuidos en varias tablas.
- Las bases de datos relacionales se basan en que los datos se distribuyen en tablas que están relacionadas mediante un campo. Ese campo es el que permite integrar los datos de las tablas en una consulta.
- En la cláusula FROM se especifica más de una tabla para consultar columnas de tablas diferentes.

PRODUCTO CRUZADO O CARTESIANO DE TABLAS

- El producto cartesiano es un tipo de composición de tablas. Al aplicarlo a dos tablas se obtiene una tabla con las **columnas** de la **primera tabla unidas** a las **columnas** de la **segunda tabla**, y las filas de la tabla resultante son **todas las posibles concatenaciones** de **filas** de la **primera tabla** con filas de la **segunda tabla**.
- Es la única manera de combinar tablas que NO están relacionadas.

La sintaxis es la siguiente:

```
SELECT *
FROM  tabla1, tabla2.
```

PRODUCTO CARTESIANO DE TABLAS. Ejemplo:

Tenemos la tabla EMPLEADOS (dni, nombre…) y una tabla TAREAS (cod_tarea, descripción, dni_empleado).

```
SELECT cod_tarea, descripcion, dni_empleado, nombre_empleado
FROM tareas,empleados;
```

- Aparecerán todos los registros de las tareas relacionados con todos los registros de los empleados.
- El producto cartesiano a veces es útil en consultas complejas, pero normalmente no lo es, porque necesitamos discriminar ese producto para que solo aparezcan los registros de las tareas asociados a sus correspondientes empleados.
- Cuando sólo obtenemos los registros que están relacionados se llama ASOCIAR TABLAS (join).

ASOCIACIÓN DE TABLAS. SINTAXIS 1992

- Para que la consulta anterior fuese correcta deberíamos escribir:

```
SELECT cod_tarea, descripcion, dni_empleado, nombre
FROM tareas, empleados
WHERE tareas.dni_empleado = empleados.dni;
```

- **La condición que aparece en la cláusula WHERE debe ser el campo común entre las dos tablas (FK).**
- En el ejemplo se utiliza la notación ***tabla.columna para evitar la ambigüedad, ya que el mismo nombre de campo se puede repetir en ambas tablas.***
- ***Es frecuente renombrar las tablas (alias) para evitar repetir continuamente el nombre de la tabla o acortar dicho nombre:***

```
SELECT a.cod_tarea, a.descripcion, b.dni_empleado, b.nombre_empleado
FROM tareas a,empleados b
WHERE a.dni_empleado = b.dni;
```

ASOCIACIÓN DE TABLAS. SINTAXIS 1992

- La cláusula WHERE contiene la condición de combinación de las tablas, pero también puede contener otras condiciones de selección de filas que se encadenarán con el operador AND. Ejemplo:

```
SELECT a.cod_tarea, a.descripcion
FROM tareas a,empleados b
WHERE a.dni_empleado = b.dni AND
b.nombre_empleado='Luis';
```

- Se pueden asociar más de dos tablas a través de los campos que las relacionan. Ejemplo:

```
SELECT a.cod_tarea, a.descripcion, b.nombre_empleado,
  c.nombre_utensilio
FROM tareas a,empleados b, utensilios_utilizados c
WHERE a.dni_empleado = b.dni AND a.cod_tarea=c.cod_tarea;
```

CLÁUSULA JOIN.

- La asociación de tablas no es el método más eficiente para extraer información de varias tablas relacionadas.
- En la versión SQL de 1999 se ideó una nueva sintaxis para consultar varias tablas.
- Se trata de separar las condiciones de asociación de las condiciones de selección de registros.
- Aunque el resultado de una consulta con JOIN o usando la sintaxis de 1992 es el mismo, se aconseja utilizar la operación JOIN y sus variantes.

CLÁUSULA JOIN

- Se utiliza **cuando** una de las **columnas de emparejamiento** está **indexada** (por ejemplo, es la PK de una de las tabla.

- En vez de hacer el producto cartesiano completo y luego s)seleccionar la filas que cumplen la condición de emparejamiento, para cada fila de una de las tablas **busca directamente** en la otra tabla **las filas que** cumplen la condición, con lo cual **se emparejan** sólo las filas que luego aparecerán en el resultado.

CONSULTAS INTERNAS: INNER JOIN

- Se llaman consultas internas porque devuelven únicamente aquellos registros/filas que tienen valores idénticos en los dos campos que se comparan para unir ambas tablas.
- Es decir, sólo se devolverán los registros o filas en los que coincidan los valores en las dos tablas.

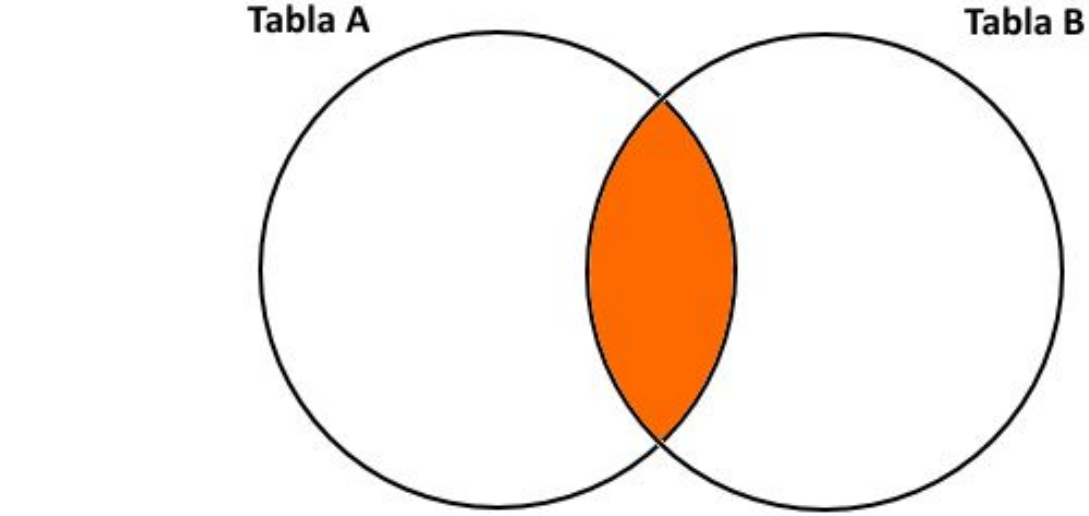
INNER JOIN.- SINTAXIS

```
SELECT [* |  LISTA DE COLUMNAS]
 FROM TablaA  INNER JOIN TablaB
 ON TablaA.columna=TablaB.columna;
```

Cuando el nombre de la columna es el mismo en las dos tablas, podemos usar USING(columna). (no válido para todos los SGBD)

```
SELECT [* |  LISTA DE COLUMNAS]
FROM TablaA INNER JOIN TablaB
USING (columna);
```

INNER JOIN (representación gráfica de la operación)



Ejemplo:

```
SELECT *
    FROM pedidos INNER JOIN productos
     ON (pedidos.idproducto = productos.idproducto)

SELECT *
    FROM pedidos INNER JOIN productos
     USING (idproducto)
```

COMBINACIONES EXTERNAS
LEFT JOIN y RIGHT JOIN

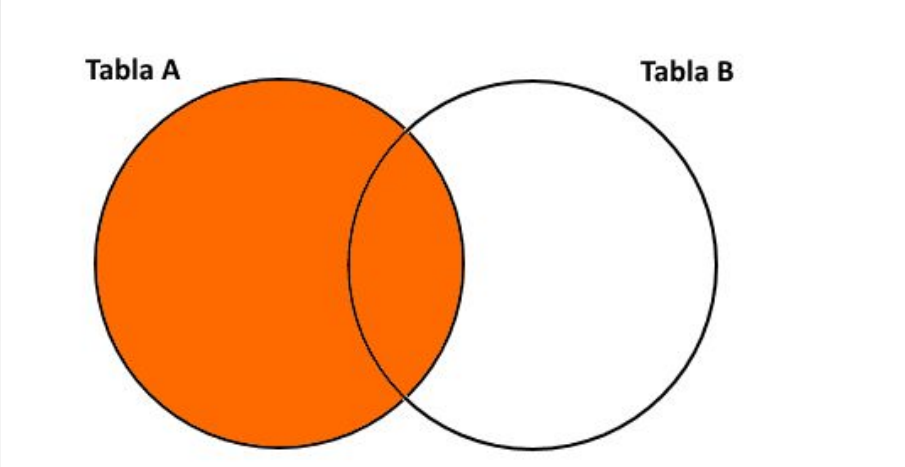
- Son una extensión del **INNER JOIN**.
- Las composiciones vistas hasta ahora son **composiciones internas** ya que todos los valores de las filas del resultado son valores que están en las tablas que se combinan.
- Con una composición interna sólo se obtienen las filas que tienen al menos una fila de la otra tabla que cumpla la condición. Un ejemplo:
- Queremos combinar los empleados con las oficinas para saber la ciudad de la oficina donde trabaja cada empleado, si utilizamos un producto cartesiano o un INNER JOIN tenemos:

```
SELECT empleados.*,ciudad
    FROM empleados INNER JOIN oficinas
    ON empleados.oficina = oficinas.cod_oficina
```

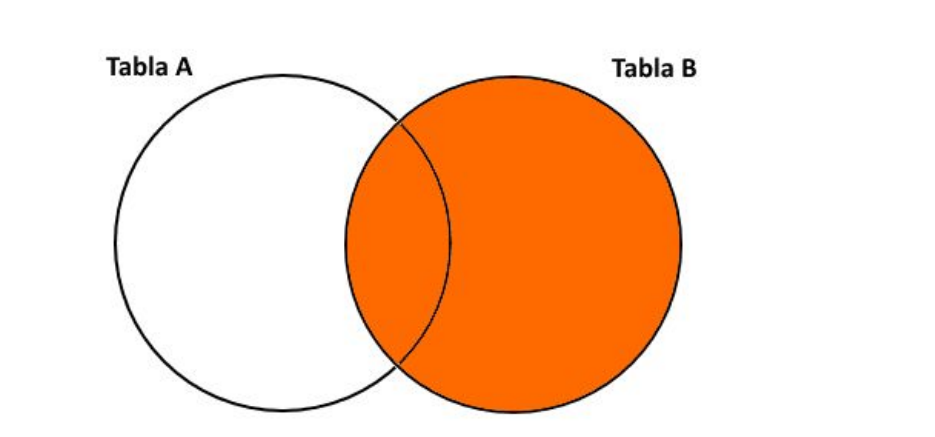
- los **empleados** que **no tienen** una **oficina** asignada (un valor nulo en el campo oficina de la tabla empleados) **no aparecen en el resultado** ya que la condición **empleados.oficina = oficinas.oficina** será siempre nula para esos empleados.

LEFT JOIN y RIGHT JOIN
COMBINACIONES EXTERNAS

LEFT JOIN



RIGHT JOIN



```
SELECT *
FROM empleados INNER JOIN oficinas
 ON empleados.oficina = oficinas.cod_oficina
```

emp r	nom_emp character varying(30)	edad integer	oficina integer	titulo character varying(20)	fecha_contrato date	jefe integer	cuota money	ventas money	ciudad character vary
	ANTONIO GARCIA	45	12	COMERCIAL	1986-10-20	104	€5.000	€7.500	ALICANTE
	ALVARO LOPEZ	48	21	COMERCIAL	1986-12-10	108	€5.500	€10.00	BADAJOZ
	JUAN ROVIRA	29	12	COMERCIAL	2000-03-01	104	€5.000	€7.500	ALICANTE
	JOSE GONZALEZ	33	12	DIRECTOR VENTAS	1995-05-19	106	€3.000	€7.500	ALICANTE
	VICENTE MARTINEZ	37	13	COMERCIAL	2000-10-20	104	€5.000	€7.500	CASTELLON
	LUIS MORENO	52	11	DIRECTOR GENERAL	2001-01-14		€5.000	€7.500	VALENCIA
	JORGE GARCIA	49	22	COMERCIAL	2000-01-02	108	€5.000	€7.500	LA CORUÑA
	ANA MARTIN	40	21	DIRECTOR VENTAS	1999-02-12	106	€5.000	€7.500	BADAJOZ
	MARIA PEREZ	31	11	COMERCIAL	2000-10-20	106	€5.000	€7.500	VALENCIA

- En los casos en que **queremos** que **también aparezcan** las **filas que no tienen una fila coincidente** en la otra tabla, **utilizaremos** el **LEFT** o **RIGHT JOIN**.

  La sintaxis del **LEFT JOIN** es la siguiente:

```
SELECT * FROM –tabla1- LEFT JOIN tabla2—
ON tabla1.columna1=tabla2.columna2
```

- La sintaxis es la **misma** que la del **INNER JOIN** cambiando la palabra **INNER** por **LEFT** (**izquierda**).
- Esta operación consiste en **añadir al resultado** del **INNER JOIN** las **filas** de la **tabla** de la **izquierda** que **no tienen correspondencia** en la otra tabla, y **rellenar** en esas filas los **campos** de la **tabla** de la **derecha** con **valores nulos**. Ejemplo:

```
SELECT *
FROM empleados LEFT JOIN oficinas
ON empleados.oficina = oficinas.cod_oficina
```

Obtenemos una lista de los empleados con los datos de su oficina, y el empleado que no tiene oficina aparece con sus datos y los datos de su oficina a nulos.

Resultado de la consulta con left join

num_emp integer	nom_emp character varying(30)	edad integer	oficina integer	titulo character varying(20)	fecha_contrato date	jefe integer	cuota money	ventas money
101	ANTONIO GARCIA	45	12	COMERCIAL	1986-10-20	104	€5.000	€7.500
102	ALVARO LOPEZ	48	21	COMERCIAL	1986-12-10	108	€5.500	€10.00
103	JUAN ROVIRA	29	12	COMERCIAL	2000-03-01	104	€5.000	€7.500
104	JOSE GONZALEZ	33	12	DIRECTOR VENTAS	1995-05-19	106	€3.000	€7.500
105	VICENTE MARTINEZ	37	13	COMERCIAL	2000-10-20	104	€5.000	€7.500
106	LUIS MORENO	52	11	DIRECTOR GENERAL	2001-01-14		€5.000	€7.500
107	JORGE GARCIA	49	22	COMERCIAL	2000-01-02	108	€5.000	€7.500
108	ANA MARTIN	40	21	DIRECTOR VENTAS	1999-02-12	106	€5.000	€7.500
109	MARIA PEREZ	31	11	COMERCIAL	2000-10-20	106	€5.000	€7.500
110	JUAN PEREZ	45		COMERCIAL	2000-10-20	104		€7.500

Observamos un registro más que en el caso anterior, es un empleado que no tiene asignada oficina y que en la consulta anterior no aparecía.
La consulta muestra todos los registros de la tabla que está a la izquierda de la instrucción join, aunque no haya coincidencia con los datos de la otra tabla.

- La sintaxis del **RIGHT JOIN** es la siguiente:
  SELECT * FROM –tabla1- RIGHT JOIN tabla2—ON tabla1.columna1=tabla2.columna2
- La sintaxis es la **misma** que la del **INNER JOIN** cambiando la palabra **INNER** por **RIGHT** (derecha).
- Esta operación consiste en **añadir al resultado** del **INNER JOIN** las **filas** de la **tabla** de la **derecha** que **no tienen correspondencia** en la otra tabla, y **rellenar** en esas filas los **campos** de la **tabla** de la **izquierda** con **valores nulos**.

Ejemplo:

```
SELECT *
FROM empleados RIGHT JOIN oficinas
ON empleados.oficina = oficinas.cod_oficina
```

- Obtenemos una lista de los empleados con los datos de su oficina, y además aparece una fila por cada oficina que no está asignada a ningún empleado con los datos del empleado a nulos.

num_emp integer	nom_emp character varying(30)	edad integer	oficina integer	titulo character varying(20)	fecha_contrato date	jefe integer	cuota money	ventas money	ciudad character v
101	ANTONIO GARCIA	45	12	COMERCIAL	1986-10-20	104	€5.000	€7.500	ALICANTE
102	ALVARO LOPEZ	48	21	COMERCIAL	1986-12-10	108	€5.500	€10.00	BADAJOZ
103	JUAN ROVIRA	29	12	COMERCIAL	2000-03-01	104	€5.000	€7.500	ALICANTE
104	JOSE GONZALEZ	33	12	DIRECTOR VENTAS	1995-05-19	106	€3.000	€7.500	ALICANTE
105	VICENTE MARTINEZ	37	13	COMERCIAL	2000-10-20	104	€5.000	€7.500	CASTELLON
106	LUIS MORENO	52	11	DIRECTOR GENERAL	2001-01-14		€5.000	€7.500	VALENCIA
107	JORGE GARCIA	49	22	COMERCIAL	2000-01-02	108	€5.000	€7.500	LA CORUÑA
108	ANA MARTIN	40	21	DIRECTOR VENTAS	1999-02-12	106	€5.000	€7.500	BADAJOZ
109	MARIA PEREZ	31	11	COMERCIAL	2000-10-20	106	€5.000	€7.500	VALENCIA
									PAMPLONA
									MADRID
									VALENCIA
									MADRID

Ahora me aparecen ciudades donde hay oficinas que no tienen trabajadores asignados.
La consulta me muestra todos los registros de la tabla que está a la derecha en la instrucción join, aunque no haya filas coincidentes con los registros de la otra tabla.