# Oracle Joins

Antonio Espin Herranz

# Cualificando nombres de columnas

- Para evitar ambigüedades es recomendable cualificar los nombres de las columnas con el nombre de la tabla a la cual pertenece.
- Si no existen nombres de columna comunes en las dos tablas, no es necesario cualificar las columnas. Sin embargo, utilizar el prefijo de tabla mejora el rendimiento, ya que le indica a **Oracle Server** exactamente dónde puede encontrar las columnas.
- Utilice prefijos de tabla para cualificar nombres de columna que estén en varias tablas.
- Utilice prefijos de tabla para mejorar el rendimiento.
- Utilice alias de columna para distinguir columnas que tengan nombres idénticos pero que residan en tablas diferentes.

# Consultas varias tablas

- **UTILIZAR COMBINACIONES EXTERNAS**
- Tenemos 3 tipos de combinaciones:
  - COMBINACIONES EXTERIORES IZQUIERDAS:
    - LEFT JOIN.
    - SELECT CAMPOS FROM TB1 LEFT JOIN TB2 ON TB1.CAMPO =TB2.CAMPO
  - COMBINACIONES EXTERNAS DERECHAS:
    - RIGHT JOIN.
    - SELECT CAMPOS FROM TB1 RIGHT JOIN TB2 ON TB1.CAMPO =TB2.CAMPO
  - COMBINACIONES COINCIDENTES:
    - INNER JOIN.
    - SELECT CAMPOS FROM TB1 INNER JOIN TB2 ON TB1.CAMPO =TB2.CAMPO

# Uniones de dos tablas

- La unión de dos tablas que devuelve sólo filas con correspondencia se denomina unión interna(**INNER JOIN**).

- Una unión entre dos tablas que devuelve los resultados de la unión interna(**INNER JOIN**) y las filas sin correspondencia de las tablas a la izquierda o derecha se denomina unión externa izquierda(**LEFT OUTER JOIN**) o derecha(**RIGHT OUTER JOIN** ).

- Una unión entre dos tablas que devuelve los resultados de la unión interna y los resultados de una unión izquierda y derecha es una unión externa completa(**FULL OUTER JOIN**).

# Ejemplos

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e LEFT OUTER JOIN departments d
ON (e.department_id = d.department_id)
ORDER BY d.department_name DESC;
/*Este es un ejemplo de un LEFT OUTER JOIN, en el cual se muestran todos los empleados y sus respectivos departamentos, también se muestran los empleados que no tienen departamento asignado.*/
---
SELECT e.last_name, e.department_id, d.department_name
FROM employees e RIGHT OUTER JOIN departments d
ON (e.department_id = d.department_id)
ORDER BY d.department_name;
/*Este es un ejemplo de un RIGHT OUTER JOIN, en el cual se muestran todos los departamentos y sus respectivos empleados, también se muestran los departamentos sin empleados.*/
---
SELECT e.last_name, d.department_id, d.department_name
FROM employees e FULL OUTER JOIN departments d
ON (e.department_id = d.department_id)
ORDER BY d.department_name DESC;
/*Este es un ejemplo de un FULL OUTER JOIN, en el cual se muestran todos los empleados y departamentos de la empresa, independientemente de que haya o no correspondencia entre ellos.*/
---
```

# Ejemplos 2

- ***SELECT*** *e.last_name, e.department_id, d.department_name*
- ***FROM*** *employees e, departments d*
- ***WHERE*** *e.department_id = d.department_id(+)*
- ***ORDER BY*** *d.department_name* ***DESC****;*
- **/*Este ejemplo es el mismo del LEFT OUTER JOIN, pero sin la cláusula.*/**
- **---**
- ***SELECT*** *e.last_name, e.department_id, d.department_name*
- ***FROM*** *employees e, departments d*
- ***WHERE*** *e.department_id(+) = d.department_id*
- ***ORDER BY*** *d.department_name;*
- **/*Este ejemplo es el mismo del RIGHT OUTER JOIN, pero sin la cláusula.*/**

# Consultas varias tablas

- Cuando una condición de unión no es válida o se omite por completo, el resultado es un producto cartesiano, en el que se muestran todas las combinaciones de filas. Todas las filas de la primera tabla se unen a todas las filas de la segunda tabla.

  Un producto cartesiano tiende a generar gran cantidad de filas, con lo que el resultado no suele ser de utilidad. Debería incluir siempre una condición de unión válida a menos que tenga la necesidad específica de combinar todas las filas de todas las tablas.

Los productos cartesianos resultan útiles si necesita generar gran número de filas para simular una cantidad aceptable de datos.

- Aunque para generar uno solo es necesario omitir una condición de union, existe la cláusula CROSS JOIN que hace lo mismo.

# Consultas varias tablas

- **Producto cartesiano:**
  - **Select columnas from tb1, tb2, … tbN**
  - Asocia cada fila de una tabla con cada fila de la otra tabla.
  - Puede haber mas de dos tablas, pero el número de combinaciones se dispara.
  - Cuando coincidan las columnas utilizaremos el nombre de la tabla para quitar la ambigüedad.
  - El número de filas que devuelve es:
    - Num.Filas Tb1 X Num. Filas Tb2 X … Num.Filas TbN.
  - **Select clientes.nombre, empleados.nombre from clientes, empleados;**

# Consultas varias tablas

- **Combinaciones:**
  - **Select columnas from tb1, tb2 where condición;**
  - Se establece un criterio sobre el producto cartesiano.
  - Normalmente en la condición se liga la clave primaria de una tabla con la clave externa de la otra.

  **Select numpedido from pedidos, clientes**

  **Where pedidos.idcliente = clientes.id;**

# Cross join

- ***SELECT*** *last_name, department_name*
- ***FROM*** *employees*
- ***CROSS JOIN*** *departments;*

- Pruebas:
  - select count(*) from tbpedidos **cross join** tbclientes;
  - select count(*) from tbpedidos;
  - select count(*) from tbclientes;

# Consultas varias tablas

- (+): Cuando tenemos dos tablas relacionadas y hay registros que no tienen su equivalente en la tabla secundaria y queremos también que aparezcan podemos utilizar el + al indicar la relación.
- Podemos forzar a que seleccione todos los registros de una u otra tabla, es similar al left / right join.

# Ejemplo

- **Sintaxis**:

  SELECT tabla1.columna1, tabla1.columna2,....  tabla2.columna1, tabla2.columna2,...

  FROM tabla1, tabla2  WHERE tabla1.columnaRelacionada**(+)**=tabla2.columnaRelacionada

- Eso obtiene los registros relacionados entre las tablas y además los registros no relacionados de la tabla2.
- Se podría usar esta otra forma:

  SELECT tabla1.columna1, tabla1.columna2,....

  tabla2.columna1, tabla2.columna2,...

  FROM tabla1, tabla2

  WHERE tabla1.columnaRelacionada=tabla2.columnaRelacionada(+)

# Ejemplo

- En el ejemplo tenemos una relación 1 a N entre tbcompañiasenvios y tbpedidos.

- En la consulta si hubiera alguna compañía de envío que NO mandara ningún pedido, con el (+) aparecería en el resultado.

```
select com.nombre, count(ped.id) as cuenta
from tbcompañiasenvios com, tbpedidos ped
where com.id = ped.idcompanya(+)
group by com.nombre;
```

# Autojoin

- Cuando relacionamos una tabla consigo misma.
- Indicar la tabla dos veces y poner dos alias distintos.

- *Obtener los viajeros que tienen el mismo perfil que un viajero dado.*
  - SELECT v1.nombre, v1.apellidos, v1.perfil_viajero FROM **viajero** v1, **viajero** v2 WHERE v1.perfil_viajero = v2.perfil_viajero AND v2.nombre = 'Beatriz'  AND v2.apellidos = 'González';

# Resumen

- **CROSS JOIN** devuelve un producto cartesiano de las dos tablas.
- **NATURAL JOIN** une dos tablas basándose en que tengan el mismo nombre de columna. ***OJO NO UTILIZARLO***
- **JOIN** realiza una unión entre las dos tablas especificando el nombre de la columna que las une. Se puede expresar como **INNER JOIN**.
- **LEFT / RIGHT / FULL OUTER JOIN** realiza una unión externa entre las dos tablas especificando el nombre de la columna de la unión