GUSTAVO CORONEL
DESARROLLA SOFTWARE

TALLER DE ORACLE SQL

CLASE 3

GUSTAVO CORONEL
desarrollasoftware.com

CONTENIDO

CONSULTAS MULTITABLAS....3

INNER JOIN....3
OUTER JOIN....4
CROSS JOIN....5
SELF JOIN....6

OPERADORES ROLLUP, CUBE Y GROUPING SETS....7

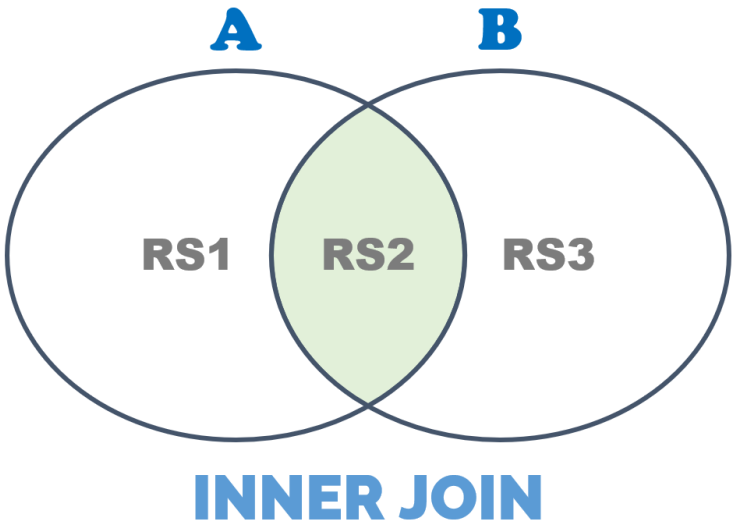
OPERADOR ROLLUP....7
OPERADOR CUBE....10
OPERADOR GROUPING SETS....10

FUNCIONES GROUPING....12

GROUPING....12
GROUPING_ID....14
GROUP_ID....15

CONSULTAS MULTITABLAS

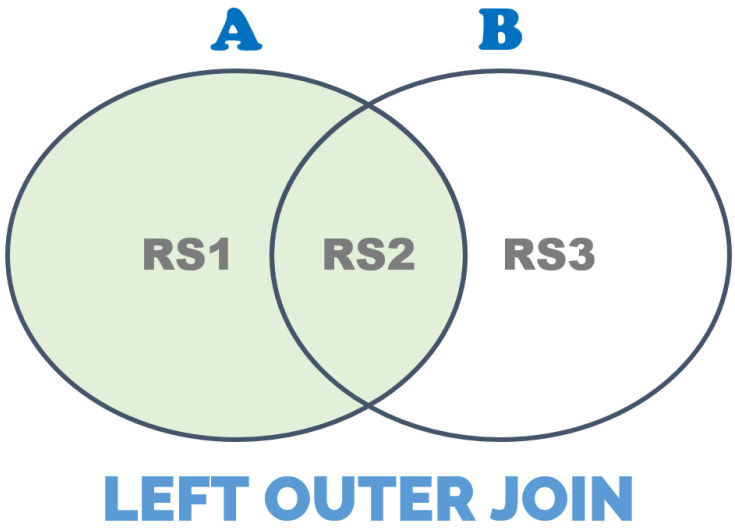
INNER JOIN



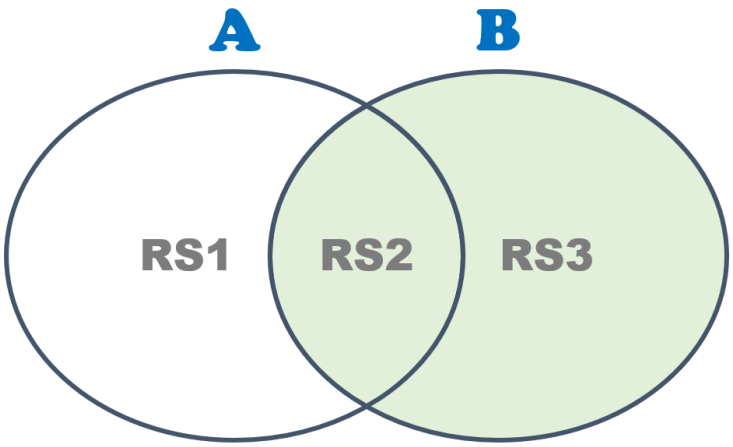

```
SELECT <lista de columnas>
FROM TablaA AS A
[INNER] JOIN TablaB AS B ON A.key = B.key
```

1. Desarrolle una sentencia SELECT para obtener un listado que incluya el nombre del curso con sus respectivos nombres de alumnos. Esquema: EDUCA.
2. Desarrolle una sentencia SELECT que muestre el nombre del alumno y la suma de todos sus pagos. Esquema: EDUCA.
3. Desarrolle una sentencia SELECT que muestre el nombre del curso y el importe de todos sus pagos. Esquema: EDUCA.
4. Desarrolle una sentencia SELECT que muestre el nombre del departamento y el importe de su planilla. Esquema: RECURSOS y/o RH.
5. Desarrolle una sentencia SELECT para encontrar la cantidad de trabajadores en cada ciudad. Esquema: RECURSOS.

OUTER JOIN



LEFT OUTER JOIN

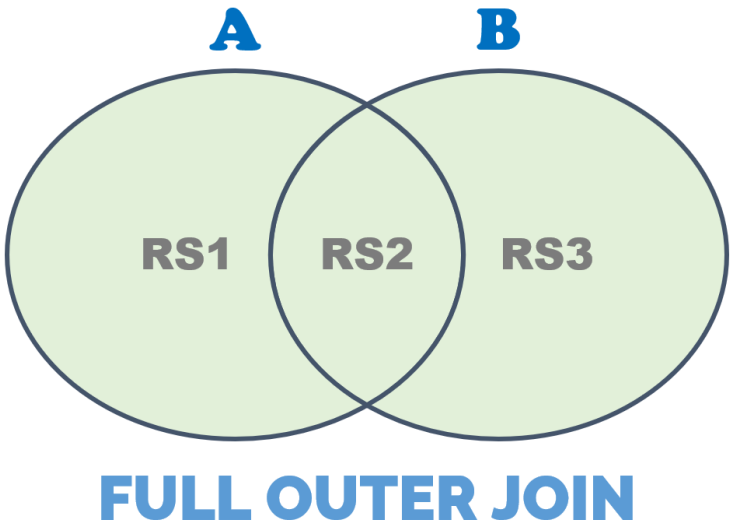


RIGHT OUTER JOIN

```
SELECT <lista de columnas>
FROM TablaA AS A
LEFT|RIGHT [OUTER] JOIN TablaB AS B ON A.key = B.key
```

6. Desarrolle una sentencia SELECT para obtener un listado de todos los departamentos y la cantidad de trabajadores en cada uno de ellos. Debe incluir a todos los departamentos incluso aquellos que no tienen empleados. Esquema: RECURSOS y/o HR.
7. Desarrolle una sentencia SELECT para encontrar la cantidad de alumnos matriculados en cada curso, debe incluir en el listado todos los cursos incluso los que aún no tienen matriculas. Esquema: EDUCA.

CROSS JOIN

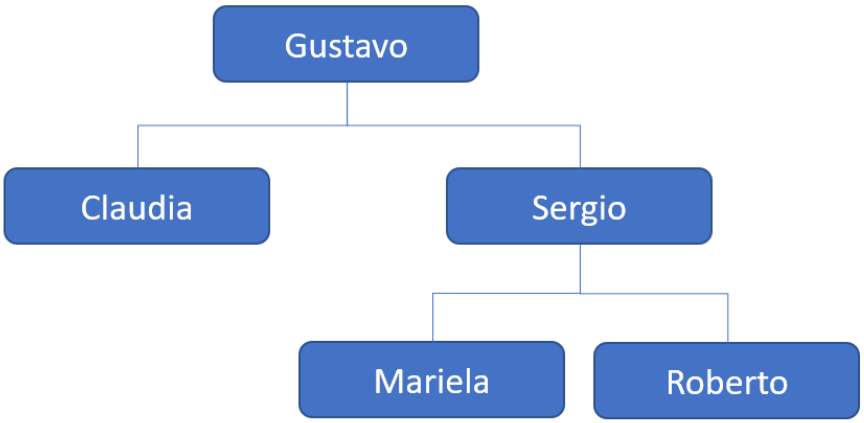



```
SELECT <lista de columnas>
FROM TablaA AS A
CROSS [OUTER] JOIN TablaB AS B ON A.key = B.key
```

8. Desarrolle una sentencia SELECT para obtener todas las posibles combinaciones entre las tablas departamento y cargo. Esquema: RECURSOS y/o RH.

SELF JOIN

IDEMPLEADO	NOMBRE	JEFE
E0001	Gustavo	NULL
E0002	Claudia	E0001
E0003	Sergio	E0001
E0004	Mariela	E0003
E0005	Roberto	E0003



9. Desarrolle una sentencia SELECT para obtener un listado de los empleados con el respectivo nombre de su superior inmediato. Esquema: RECURSOS y/o HR.

OPERADORES ROLLUP, CUBE Y GROUPING SETS

Operador ROLLUP

Además de los resultados de agregación regulares que esperamos de la cláusula GROUP BY, el operador ROLLUP genera subtotales de grupo de derecha a izquierda y un total general.

Ejemplo 1

Este ejemplo se desarrolla con el esquema **RECURSOS**.

Consultando la planilla por departamento:

```
SELECT
     d.nombre DEPARTOAMENTO,
     SUM(e.sueldo) PLANILLA
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
GROUP BY d.nombre;
```

A continuación, se tiene el resultado:

	DEPARTAMENTO	PLANILLA
1	Gerencia	33000
2	Contabilidad	25000
3	Investigacion	41800
4	Ventas	37000
5	Sistemas	26500

Ahora se solicita también tener el total general, para lo cual se debe aplicar el operador ROLLUP de la siguiente manera:

```
SELECT
     d.nombre DEPARTOAMENTO,
     SUM(e.sueldo) PLANILLA
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
GROUP BY ROLLUP ( d.nombre );
```

En el resultado se puede apreciar que se agrega una fila con el total, en la columna DEPARTAMENTO se tiene valor NULL.

	DEPARTAMENTO	PLANILLA
1	Gerencia	33000
2	Contabilidad	25000
3	Investigacion	41800
4	Ventas	37000
5	Sistemas	26500
6	(null)	163300

Ejemplo 2

Este ejemplo se desarrolla con el esquema **RECURSOS**.

Consultando la planilla por departamento y cargo o puesto de trabajo:

```
SELECT
     d.nombre DEPARTAMENTO,
     c.nombre CARGO,
     SUM(e.sueldo) PLANILLA
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
JOIN cargo C ON e.idcargo = c.idcargo
GROUP BY d.nombre, c.nombre
ORDER BY 1, 2;
```

El resultado es el siguiente:

	DEPARTAMENTO	CARGO	PLANILLA
1	Contabilidad	Empleado	8000
2	Contabilidad	Gerente	15000
3	Contabilidad	Oficinista	2000
4	Gerencia	Empleado	8000
5	Gerencia	Gerente General	25000
6	Investigacion	Gerente	15000
7	Investigacion	Investigador	21500
8	Investigacion	Oficinista	1800
9	Investigacion	Programador	3500
10	Sistemas	Analista	6000
11	Sistemas	Gerente	15000
12	Sistemas	Programador	5500
13	Ventas	Empleado	7500
14	Ventas	Gerente	15000
15	Ventas	Oficinista	2000
16	Ventas	Vendedor	12500

Ahora se solicita también tener el total general y un total por departamento, para lo cual se debe aplicar el operador ROLLUP de la siguiente manera:

```
SELECT
     d.nombre DEPARTAMENTO,
     c.nombre CARGO,
     SUM(e.sueldo) PLANILLA
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
JOIN cargo C ON e.idcargo = c.idcargo
GROUP BY ROLLUP ( d.nombre, c.nombre )
ORDER BY 1, 2;
```

En el resultado se puede apreciar que se agrega una fila con el total por departamento y una fila para el total general respectivamente.

	DEPARTAMENTO	CARGO	PLANILLA
1	Contabilidad	Empleado	8000
2	Contabilidad	Gerente	15000
3	Contabilidad	Oficinista	2000
4	Contabilidad	(null)	25000
5	Gerencia	Empleado	8000
6	Gerencia	Gerente General	25000
7	Gerencia	(null)	33000
8	Investigacion	Gerente	15000
9	Investigacion	Investigador	21500
10	Investigacion	Oficinista	1800
11	Investigacion	Programador	3500
12	Investigacion	(null)	41800
13	Sistemas	Analista	6000
14	Sistemas	Gerente	15000
15	Sistemas	Programador	5500
16	Sistemas	(null)	26500
17	Ventas	Empleado	7500
18	Ventas	Gerente	15000
19	Ventas	Oficinista	2000
20	Ventas	Vendedor	12500
21	Ventas	(null)	37000
22	(null)	(null)	163300

Operador CUBE

Además de los subtotales generados por el operador ROLLUP, el operador CUBE genera subtotales para todas las combinaciones posibles.

Ejemplo 3

Analice el resultado de la siguiente consulta y compararlo con el resultado del Ejemplo 2.

```
SELECT
     d.nombre DEPARTAMENTO,
     c.nombre CARGO,
     SUM(e.sueldo) PLANILLA
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
JOIN cargo C ON e.idcargo = c.idcargo
GROUP BY CUBE ( d.nombre, c.nombre )
ORDER BY 1, 2;
```

Operador GROUPING SETS

GROUPING SETS le permite definir de forma selectiva uno o más conjuntos de agrupación en una consulta.

Esta es la sintaxis de la expresión GROUPING SETS:

```
GROUP BY
    GROUPING SETS(grouping_set_list);
```

En esta sintaxis, **`grouping_set_list`** es una lista de conjuntos de agrupación separados por comas, por ejemplo:

```
GROUP BY
    GROUPING SETS(
        (),
        (c1),
        (c2),
        (c1,c2),
        (c1,c2,c3)
    )
```

Ejemplo 4

Desarrolle una sentencia SELECT para encontrar el importe de la planilla por cada departamento, el importe de la planilla por cargo en cada departamento y el importe total de la planilla. Esquema RECURSOS.

Analizar el resultado de la siguiente consulta:

```
SELECT
     NVL(d.nombre,'TOTAL') DEPARTAMENTO,
     NVL(c.nombre,'TOTAL') CARGO,
     SUM(e.sueldo) PLANILLA
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
JOIN cargo C ON e.idcargo = c.idcargo
GROUP BY GROUPING SETS(
     (d.nombre,c.nombre),
     (d.nombre)
     )
ORDER BY 1, 2;
```

FUNCIONES GROUPING

GROUPING

Puede ser bastante fácil identificar visualmente los subtotales generados por ROLLUP y CUBE, pero para hacerlo programáticamente se necesita algo más preciso que la presencia de valores nulos en las columnas de agrupación.

Aquí es donde entra la función GROUPING. Acepta una sola columna como parámetro y devuelve "1" si la columna contiene un valor nulo generado como parte de un subtotal por una operación ROLLUP o CUBE o "0" para cualquier otro valor.

Ejemplo 5

La siguiente consulta es una repetición del CUBE del Ejemplo 3, pero se ha agregado la función GROUPING para cada una de las dimensiones.

```
SELECT
     d.nombre DEPARTAMENTO,
     c.nombre CARGO,
     SUM(e.sueldo) PLANILLA,
     GROUPING(d.nombre) AS FLG1,
     GROUPING(c.nombre) AS FLG2
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
JOIN cargo C ON e.idcargo = c.idcargo
GROUP BY CUBE ( d.nombre, c.nombre )
ORDER BY 1, 2;
```

El resultado de la consulta anterior se puede analizar de la siguiente manera:

FLG1	FLG2	ANALISIS
0	**0**	Representa una fila que contiene un subtotal regular que puede obtenerse con GROUP BY.
0	**1**	Representa una fila que contiene un subtotal para un departamento, generado por las operaciones ROLLUP y CUBE.
1	**0**	Representa una fila que contiene un subtotal para un CARGO, que solo se podría obtener con el operador CUBE.
1	**1**	Representa una fila que contiene un total general para la consulta, generado por los operadores ROLLUP y CUBE.

Con estos flags sería mucho más fácil programar estos resultados.

Las columnas GROUPING se pueden utilizar para ordenar o filtrar el resultado, como se puede apreciar en el siguiente ejemplo:

```
SELECT
     d.nombre DEPARTAMENTO,
     c.nombre CARGO,
     SUM(e.sueldo) PLANILLA,
     GROUPING(d.nombre) AS FLG1,
     GROUPING(c.nombre) AS FLG2
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
JOIN cargo C ON e.idcargo = c.idcargo
GROUP BY CUBE ( d.nombre, c.nombre )
HAVING GROUPING(d.nombre)=1 OR GROUPING(c.nombre)=1
ORDER BY GROUPING(d.nombre), GROUPING(c.nombre);
```

A continuación, tienes el resultado:

	DEPARTAMENTO	CARGO	PLANILLA	FLG1	FLG2
1	Contabilidad	(null)	25000	0	1
2	Investigacion	(null)	41800	0	1
3	Ventas	(null)	37000	0	1
4	Sistemas	(null)	26500	0	1
5	Gerencia	(null)	33000	0	1
6	(null)	Investigador	21500	1	0
7	(null)	Oficinista	5800	1	0
8	(null)	Programador	9000	1	0
9	(null)	Gerente General	25000	1	0
10	(null)	Gerente	60000	1	0
11	(null)	Empleado	23500	1	0
12	(null)	Analista	6000	1	0
13	(null)	Vendedor	12500	1	0
14	(null)	(null)	163300	1	1

GROUPING_ID

La función GROUPING_ID proporciona una forma alternativa y más compacta de identificar filas de subtotales. Al pasar las columnas de dimensión como argumentos, devuelve un número que indica el nivel GROUP BY.

Ejemplo 6

Analice el resultado de la siguiente consulta:

```
SELECT
     NVL(d.nombre,'TOTAL') DEPARTAMENTO,
     NVL(c.nombre,'TOTAL') CARGO,
     SUM(e.sueldo) PLANILLA,
     GROUPING(d.nombre) AS FLG1,
     GROUPING(c.nombre) AS FLG2,
     GROUPING_ID(d.nombre,c.nombre) AS ID
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
JOIN cargo C ON e.idcargo = c.idcargo
GROUP BY CUBE ( d.nombre, c.nombre )
ORDER BY 1,2;
```

GROUP_ID

Es posible escribir consultas que devuelvan los subtotales duplicados, lo que puede resultar un poco confuso. La función GROUP_ID asigna el valor "0" al primer conjunto y a todos los conjuntos posteriores se les asigna un número más alto.

La siguiente consulta obliga a tener duplicados en el resultado y mostrar la función GROUP_ID en acción.

```
SELECT
      NVL(d.nombre,'TOTAL') DEPARTAMENTO,
      NVL(c.nombre,'TOTAL') CARGO,
      SUM(e.sueldo) PLANILLA,
      GROUPING_ID(d.nombre,c.nombre) AS GROUPING_ID,
      GROUP_ID() GROUP_ID
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
JOIN cargo C ON e.idcargo = c.idcargo
GROUP BY GROUPING SETS ( d.nombre, CUBE(d.nombre,c.nombre) )
ORDER BY 1,2;
```

A continuación, se tiene parte del resultado, se puede apreciar que existen subtotales duplicados, por ejemplo, el de contabilidad.

	DEPARTAMENTO	CARGO	PLANILLA	GROUPING_ID	GROUP_ID
1	Contabilidad	Empleado	8000	0	0
2	Contabilidad	Gerente	15000	0	0
3	Contabilidad	Oficinista	2000	0	0
4	Contabilidad	TOTAL	25000	1	0
5	Contabilidad	TOTAL	25000	1	1
6	Gerencia	Empleado	8000	0	0
7	Gerencia	Gerente General	25000	0	0
8	Gerencia	TOTAL	33000	1	0
9	Gerencia	TOTAL	33000	1	1
10	Investigacion	Gerente	15000	0	0
11	Investigacion	Investigador	21500	0	0
12	Investigacion	Oficinista	1800	0	0
13	Investigacion	Programador	3500	0	0
14	Investigacion	TOTAL	41800	1	0
15	Investigacion	TOTAL	41800	1	1
16	Sistemas	Analista	6000	0	0
17	Sistemas	Gerente	15000	0	0
18	Sistemas	Programador	5500	0	0
19	Sistemas	TOTAL	26500	1	0
20	Sistemas	TOTAL	26500	1	1

Para eliminar las filas duplicadas debes utilizar HAVING para aplicar filtro de grupos:

```
SELECT
     NVL(d.nombre,'TOTAL') DEPARTAMENTO,
     NVL(c.nombre,'TOTAL') CARGO,
     SUM(e.sueldo) PLANILLA,
     GROUPING_ID(d.nombre,c.nombre) AS GROUPING_ID,
     GROUP_ID() GROUP_ID
FROM empleado E
JOIN departamento D ON e.iddepartamento = d.iddepartamento
JOIN cargo C ON e.idcargo = c.idcargo
GROUP BY GROUPING SETS ( d.nombre, CUBE(d.nombre,c.nombre) )
HAVING GROUP_ID() = 0
ORDER BY 1,2;
```