Programación en Base de Datos Oracle

LENGUAJE DE MANIPULACIÓN DE DATOS DML II

Semana 04

OBJETIVO DEL LABORATORIO

Reconoce y manipula la información de la base de datos utilizando DML, hace uso de las funciones predefinidas en Oracle, además podrá convertir explícitamente los tipos de datos, desde el entorno de PL/SQL.

MARCO TEÓRICO

Hoy en día el manejo de información y sobre todo el almacenamiento de datos es una parte muy importante en cualquier ámbito laboral.

Esto se realiza a través de un gestor de base de datos, “Un Sistema Gestor de Bases de Datos (SGBD) o DBMA (Data Base Management System) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.”.

Para el problema que se nos presenta ocuparemos el gestor ORACLE, Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Donde desarrollaremos una base de datos para una empresa la cual requiere almacenar la información de todas las compras que hace para su reabastecimiento donde se registrara desde la solicitud, datos de los proveedores, facturas, y todos los datos necesarios para un mejor control de compras.

RECURSOS

Hardware

* Sistema. Oracle en Windows requiere un PC Intel x86, AMD64 o Intel EM64T
* Memoria. Al menos 1 GB de RAM y el doble en virtual.
* Espacio en disco duro. Al menos 6 GB para la instalación (algunas instalaciones requieren menos). Además, necesitamos poder almacenar 500 MB en la carpeta TEMP del sistema.
* Tarjeta gráfica. Debe de ser capaz de mostrar 1024 por 768 píxeles como mínimo y 256 colores.

Software

**Sistema Operativo**.

* + Windows 2003 Server y 2003 Server R2.
  + Windows XP Professional.
  + Windows Vista, pero no la versión Home Edition.
  + Windows Server 2008 y 2012. No la versión Server Core.

Compiladores. Se usan para la gente que crean aplicaciones en Oracle usando lenguajes como Pro C, Pro COBOL, JAVA.

* + Visual C++.NET 2005 8.0 o Intel 10.1 C, .Net Express.

Navegador. Para configurar algunos servicios de Oracle. Debe de ser navegador moderno (Internet Explorer 6 o superior, Firefox 2.0 o superior, Safari 3.1 o superior, Chrome 3.0 o superior)

Usar dirección IP única en la máquina en la que se instala Oracle. Es decir, no usar DHCP para direccionar la IP en el servidor de Oracle. No es un requisito obligatorio, pero es muy recomendable.

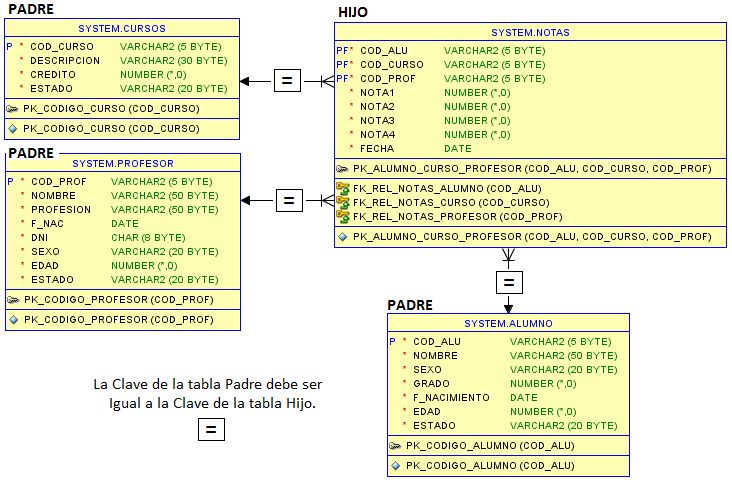
CONSULTAS AVANZADAS, USO DE COMBINACIONES (JOINS)

**FAVOR DE DESCARGAR EL SCRIPT DE BASE DE DATOS**: **Semana04\_DML\_2.sql**

**OBTENER DATOS DE MÚLTIPLES TABLAS**

Es más que habitual necesitar en una consulta datos que se encuentran distribuidos en varias tablas. Las bases de datos relacionales se basan en que los datos se distribuyen en Manual de SQL para Oracle 11g comandos internos SQL e SQL\*Plus 40 tablas que se pueden relacionar mediante un campo. Ese campo es el que permite integrar los datos de las tablas.

El Script de base de datos **Semana04\_DML\_2.sql** va generar el siguiente modelo relacional.



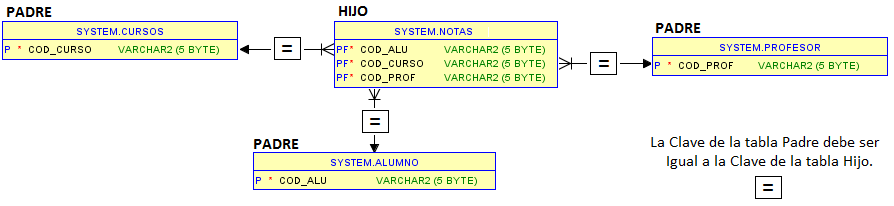
Este modelo relacional servirá para realizar las pruebas necesarias, en las tareas operaciones de multi consultas.

**APRENDE ASOCIAR TABLAS, USO EXCLUSIVO DE LA CLAUSUAL WHERE.**

**RELACIONES DE IGUALDAD.**

En este proceso solo realizaremos comparaciones entre las llaves de las tablas padre e hijos de nuestro modelo relacional.

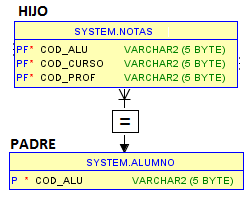
Si observamos detenidamente nuestro modelo relacional, vemos que algunas tablas tienen denominación **PADRE** y otros la denominación **HIJO**,



**La interpretación de estas relaciones, es la siguiente:**

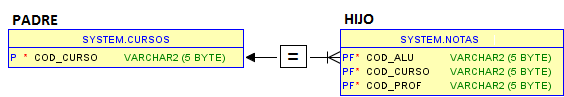
La llave del Padre ALUMNO es igual a la Llave del Hijo NOTAS.

**ALUMNO.COD\_ALU = NOTAS.COD\_ALU**



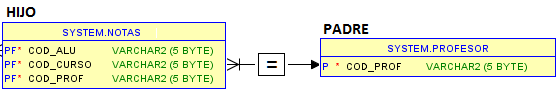
La llave del Padre CURSOS es igual a la Llave del Hijo NOTAS.

**CURSOS.COD\_CURSO = NOTAS.COD\_CURSO**



La llave del Padre PROFESOR es igual a la Llave del Hijo NOTAS.

**PROFESOR.COD\_PROF = NOTAS.COD\_PROF**



La asociación de las tablas es posible usando la cláusula **WHERE**. Solo con comparar llaves se puede establecer una relación.

**Ejemplo 1:**

Mostrar los cursos y calificaciones que tiene asignado el alumno **PEREZ SANCHEZ JUAN**.

Para dar solución a este ejemplo, analicemos que los campos solicitados en la consulta pertenecen a diferentes tablas, para ser exactos: ALUMNO, CURSOS, NOTAS y PROFESOR, para que esta consulta funcione, dichas tablas tienen que ser nombradas en la cláusula FROM, y en la cláusula WHERE comparar las llaves entre estas tablas, es decir **PADRE.LLAVE = HIJO.LLAVE**. use este ejemplo para desarrollar otros casos.

**SOLUCION:**

**SELECT CURSOS.DESCRIPCION, NOTAS.NOTA1, NOTAS.NOTA2,**

**NOTAS.NOTA3, NOTAS.NOTA4**

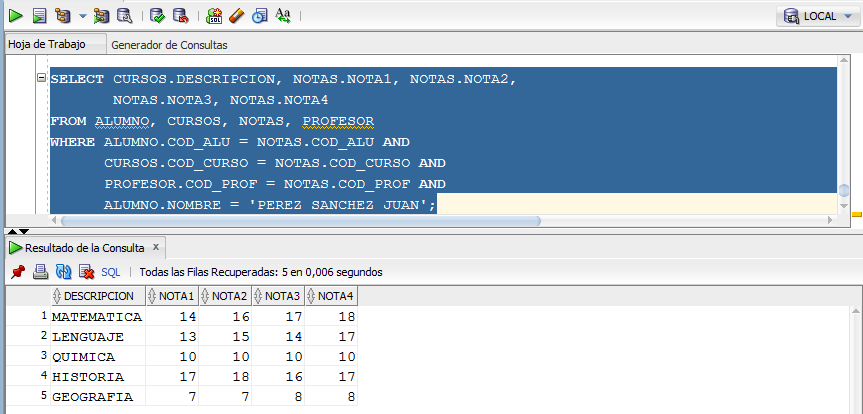
**FROM ALUMNO, CURSOS, NOTAS, PROFESOR**

**WHERE ALUMNO.COD\_ALU = NOTAS.COD\_ALU AND**

**CURSOS.COD\_CURSO = NOTAS.COD\_CURSO AND**

**PROFESOR.COD\_PROF = NOTAS.COD\_PROF AND**

**ALUMNO.NOMBRE = 'PEREZ SANCHEZ JUAN';**



**IMPORTANTE:**

SI LAS **TABLAS A USAR** ESTAN RELACIONADAS, ENTONCES EN LA **CONSULTA DEBEN ESTAR RELACIONADAS TAMBIEN**.

**JOIN ON**

Permite establecer relaciones cuya condición se establece manualmente, lo que permite realizar asociaciones más complejas o bien asociaciones cuyos campos en las tablas no tienen el mismo nombre:

**Ejemplo 2:**

Mostrar los cursos y calificaciones que tiene asignado el alumno **PEREZ SANCHEZ JUAN**.

**SELECT CURSOS.DESCRIPCION, NOTAS.NOTA1, NOTAS.NOTA2,**

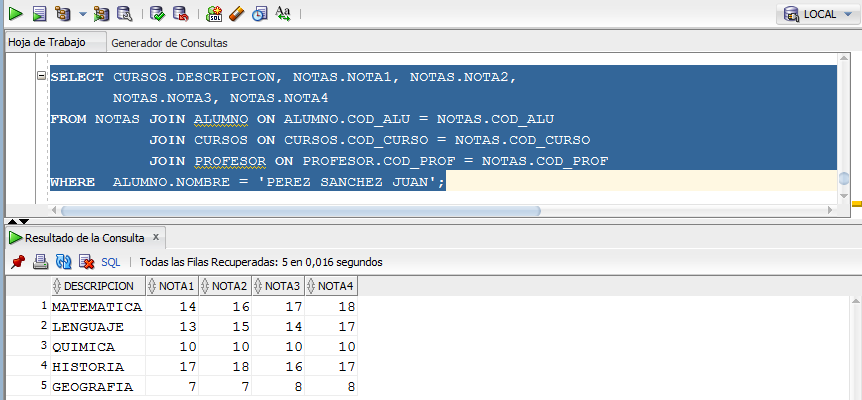
**NOTAS.NOTA3, NOTAS.NOTA4**

**FROM NOTAS JOIN ALUMNO ON ALUMNO.COD\_ALU = NOTAS.COD\_ALU**

**JOIN CURSOS ON CURSOS.COD\_CURSO = NOTAS.COD\_CURSO**

**JOIN PROFESOR ON PROFESOR.COD\_PROF = NOTAS.COD\_PROF**

**WHERE ALUMNO.NOMBRE = 'PEREZ SANCHEZ JUAN';**



**NATURAL JOIN**

Establece una relación de igualdad entre las tablas a través de los campos que tengan el mismo nombre en ambas tablas:

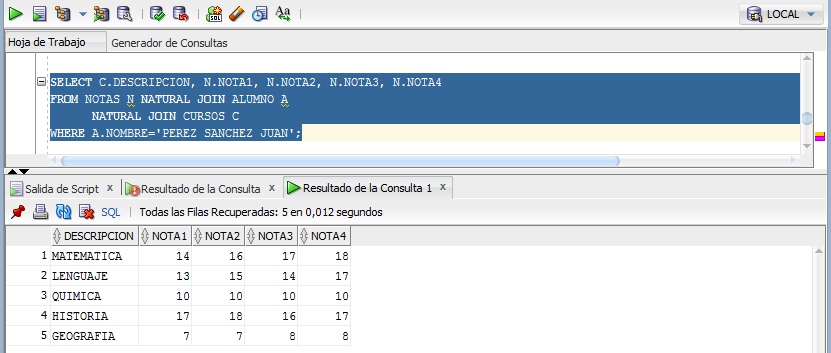
**Ejemplo 3:**

Mostrar los cursos y calificaciones que tiene asignado el alumno **PEREZ SANCHEZ JUAN**.

**SELECT C.DESCRIPCION, N.NOTA1, N.NOTA2, N.NOTA3, N.NOTA4**

**FROM NOTAS N NATURAL JOIN ALUMNO A**

**NATURAL JOIN CURSOS C**

**WHERE A.NOMBRE='PEREZ SANCHEZ JUAN';**

**INNER JOIN:**

Otra opción es utilizar la cláusula INNER JOIN. Su sintaxis es idéntica a la de una consulta SELECT habitual, con la particularidad de que en la cláusula FROM sólo aparece una tabla o vista, añadiéndose el resto de tablas a través de cláusulas INNER JOIN.

**Ejemplo 4:**

Visualice a los docentes que han impartido clases de MATEMATICA.

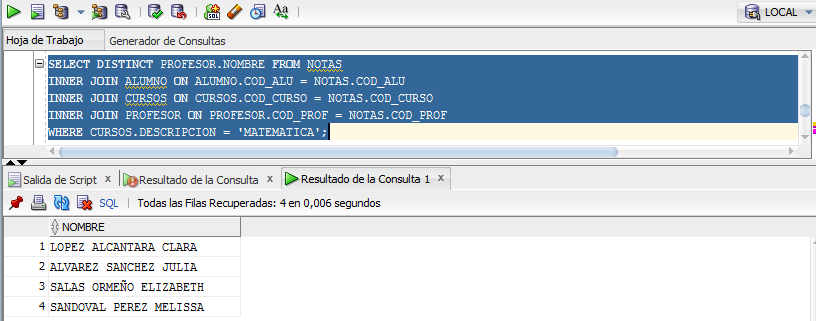
**SELECT DISTINCT PROFESOR.NOMBRE FROM NOTAS**

**INNER JOIN ALUMNO ON ALUMNO.COD\_ALU = NOTAS.COD\_ALU**

**INNER JOIN CURSOS ON CURSOS.COD\_CURSO = NOTAS.COD\_CURSO**

**INNER JOIN PROFESOR ON PROFESOR.COD\_PROF = NOTAS.COD\_PROF**

**WHERE CURSOS.DESCRIPCION = 'MATEMATICA';**



**JOIN USING.**

Unión interna usando columnas con precisión, es decir especificando en la combinación la llave a combinar.

**Ejemplo 5:**

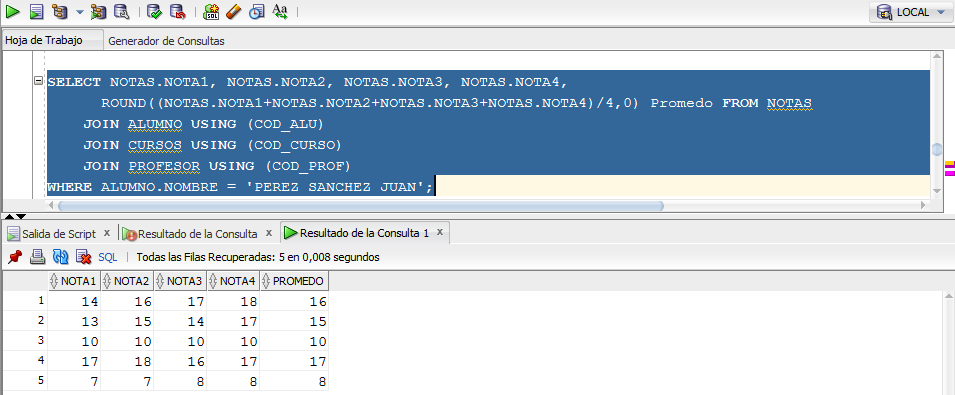
Muestre las calificaciones con promedio incluido del alumno **PEREZ SANCHEZ JUAN**

**SELECT NOTAS.NOTA1, NOTAS.NOTA2, NOTAS.NOTA3, NOTAS.NOTA4, ROUND((NOTAS.NOTA1+NOTAS.NOTA2+NOTAS.NOTA3+NOTAS.NOTA4)/4,0) Promedo FROM NOTAS**

**JOIN ALUMNO USING (COD\_ALU)**

**JOIN CURSOS USING (COD\_CURSO)**

**JOIN PROFESOR USING (COD\_PROF) WHERE ALUMNO.NOMBRE = 'PEREZ SANCHEZ JUAN';**



**CROSS JOIN**

Utilizando la opción CROSS JOIN se realiza un producto cruzado entre las tablas indicadas. Las combinaciones cruzadas (CROSS JOIN) muestran todas las combinaciones de todos los registros de las tablas combinadas. Para este tipo de join no se incluye una condición de enlace. Se genera el producto cartesiano en el que el número de filas del resultado es igual al número de registros de la primera tabla multiplicado por el número de registros de la segunda tabla, es decir, si hay 3 registros en una tabla y 4 en la otra, retorna 12 filas.

Ejemplo 6:

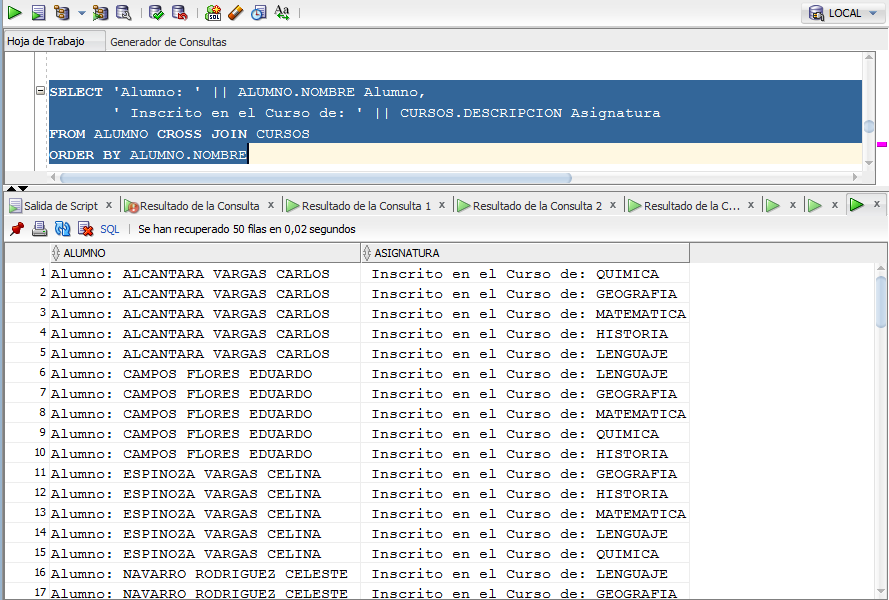
Realice una consulta donde se combine los datos de la tabla **ALUMNO** y **CURSOS**, provocando saber que asignaturas podrían cursar cada alumno.

**SELECT 'Alumno: ' || ALUMNO.NOMBRE Alumno,**

**' Inscrito en el Curso de: ' || CURSOS.DESCRIPCION Asignatura**

**FROM ALUMNO CROSS JOIN CURSOS**

**ORDER BY ALUMNO.NOMBRE**



Nota:

Este ejemplo es semejante a: combinar comidas y postres, cada comida tendrá un postre.

**LEFT OUTER JOIN – RIGHT OUTER JOIN**

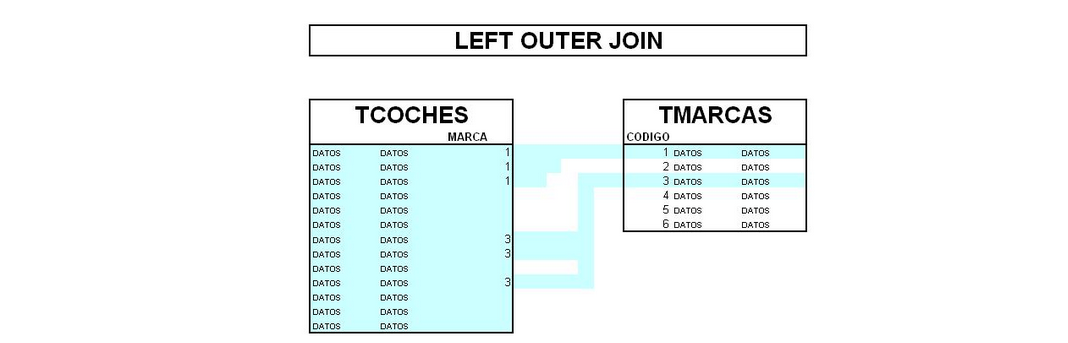
La combinación interna es excluyente. Esto quiere decir que si un registro no cumple la condición de combinación no se incluye en los resultados. De este modo en el ejemplo anterior si un coche no tiene grabada la marca no se devuelve en mi consulta.

Según la naturaleza de nuestra consulta esto puede ser una ventaja, pero en otros casos significa un serio problema. Para modificar este comportamiento SQL pone a nuestra disposición la combinación externa. La combinación externa no es excluyente. La sintaxis es muy parecida a la combinación interna,

La combinación externa puede ser diestra o siniestra, LEFT OUTER JOIN o RIGHT OUTER JOIN.  Con LEFT OUTER JOIN obtenemos todos los registros de la tabla que situemos a la **izquierda** de la cláusula **JOIN**, mientras que con RIGHT OUTER JOIN obtenemos el efecto contrario. Como mejor se ve la combinación externa es con un ejemplo

**SELECT** tCoches.matricula,   
       tMarcas.marca,   
       tCoches.modelo,   
       tCoches.color,   
       tCoches.numero\_kilometros,   
       tCoches.num\_plazas   
FROM tCoches  
**LEFT OUTER JOIN** tMarcas ON tCoches.marca = tMarcas.codigo

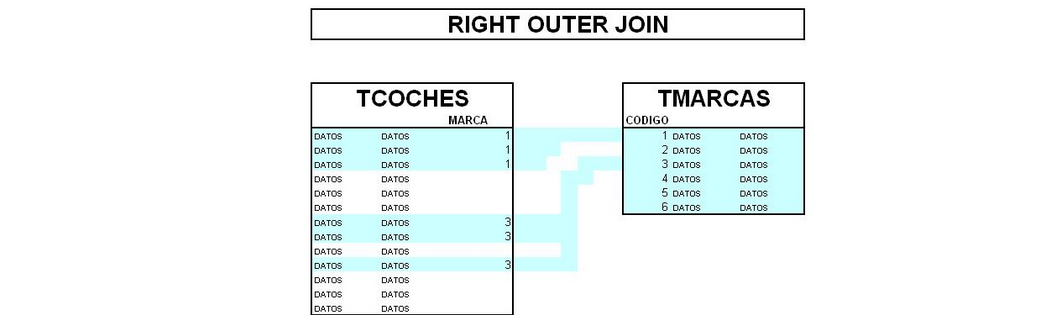
Esta consulta devolverá todos los registros de la tabla **TCOCHES**, independientemente de que tengan marca o no.  En el caso de que el coche no tenga **marca** se devolverá el valor null para los campos de la tabla **TMARCAS**. Visualmente (la consulta devuelve los datos en celeste).



Esta consulta devolverá los registros de la tabla **TCOCHES** que tengan marca relacionada y todos los registros de la tabla **TMARCAS**, tengan algún registro en **TCOCHES** o no. Visualmente (la consulta devuelve los datos en azul)...

El mismo ejemplo con RIGHT OUTER JOIN.

**SELECT** tCoches.matricula,   
       tMarcas.marca,   
       tCoches.modelo,   
       tCoches.color,   
       tCoches.numero\_kilometros,   
       tCoches.num\_plazas   
FROM tCoches  
**RIGHT OUTER JOIN** tMarcas **ON** tCoches.marca = tMarcas.codigo

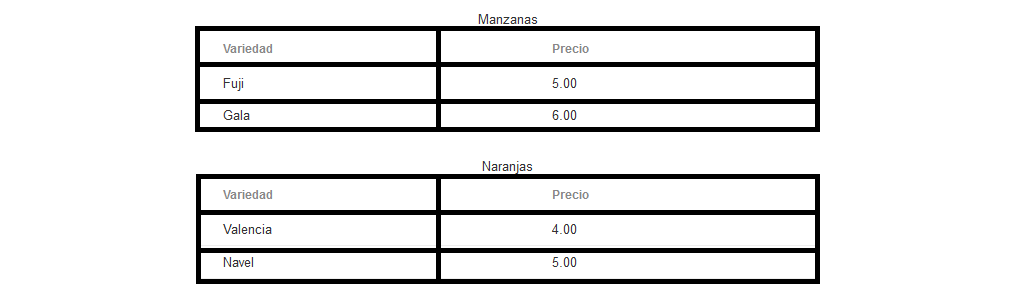


**FULL OUTER JOIN**

Para empezar recordemos que un FULL OUTER JOIN es así como un LEFT JOIN y RIGHT JOIN a la vez. O bien, como vimos en [Inner, Outer, Natural & Cross JOINs](http://luauf.com/2008/05/31/inner-outer-natural-cross-joins/), FULL OUTER JOIN combina los resultados de dos o más tablas, tengan o no coincidencia entre sí.

**Veamos un ejemplo demostrativo:**

Disponemos de dos tablas, **MANZANAS** y **NARANJAS**: y sus contenidos.



El Script es el siguiente:

**CREATE TABLE** MANZANAS (

Variedad Varchar2 (30),

Precio Numeric (10,2));

**CREATE TABLE** NARANJAS (

Variedad Varchar2 (30),

Precio Numeric (10,2));

**INSERT INTO MANZANAS** VALUES ('FUJI', 5.00);

**INSERT INTO MANZANAS** VALUES ('GALA', 6.00);

**INSERT INTO NARANJAS VALUES** ('VALENCIA',4.00);

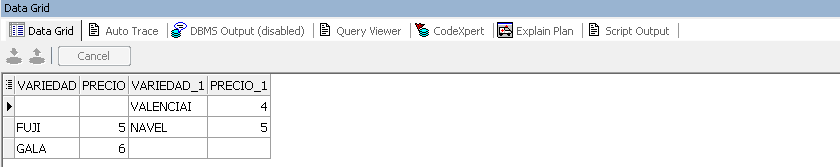
**INSERT INTO NARANJAS VALUES** ('NAVEL',5.00);

Si realizamos un FULL OUTER JOIN por precio, tenemos:

SELECT \* FROM MANZANAS a

FULL OUTER JOIN NARANJAS o ON a.precio = o.precio

**Resultado: Analice bien el ejemplo.**



**COMBINACIONES ESPECIALES, Uniones**

La palabra UNION permite añadir el resultado de un SELECT a otro SELECT. Para ello ambas instrucciones tienen que utilizar el mismo número y tipo de columnas. Ejemplo:

**SELECT** nombre **FROM** provincias

**UNION**

**SELECT** nombre **FROM** comunidades

El resultado es una tabla que contendrá nombres de provincia y de comunidades. Es decir, UNION crea una sola tabla con registros que estén presentes en cualquiera de las consultas. Si están repetidas sólo aparecen una vez, para mostrar los duplicados se utiliza UNION ALL en lugar de la palabra UNION.

**INTERSECCIONES**

De la misma forma, la palabra INTERSECT permite unir dos consultas SELECT de modo que el resultado serán las filas que estén presentes en ambas consultas.

**DIFERENCIA**

Con MINUS también se combinan dos consultas SELECT de forma que aparecerán los registros del primer SELECT que no estén presentes en el segundo.

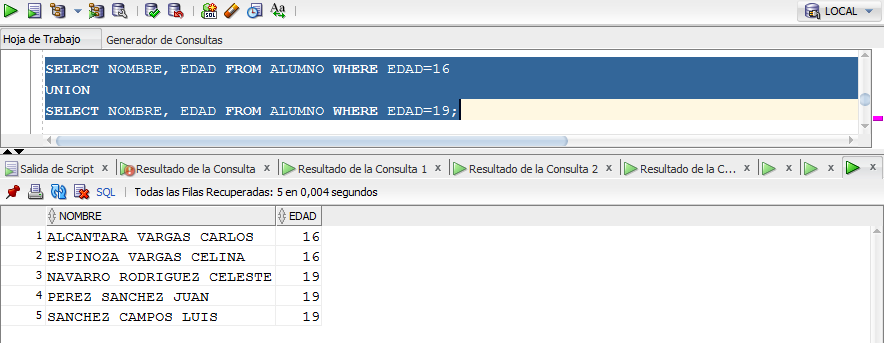
**Ejemplo usando UNION.**

Unir los nombre de alumnos que tengan las edades de 18 y 20 años.

**SELECT NOMBRE, EDAD FROM ALUMNO WHERE EDAD=16**

**UNION**

**SELECT NOMBRE, EDAD FROM ALUMNO WHERE EDAD=19;**



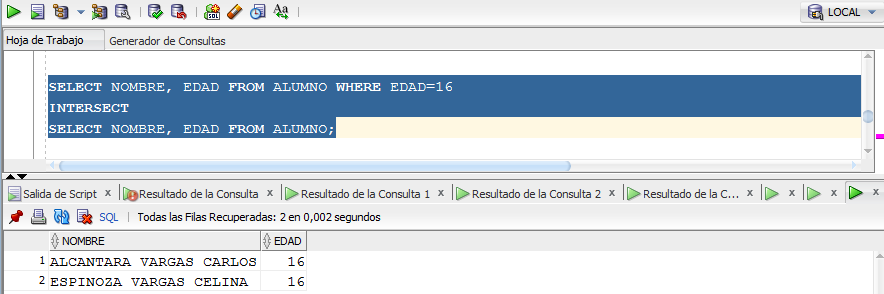
**Ejemplo usando INSTERSECT.**

Este ejemplo muestra a los alumnos cuya edad sea de 16 años, pero debe pasar un filtro dependiente de la segunda consulta.

**SELECT NOMBRE, EDAD FROM ALUMNO WHERE EDAD=16**

**INTERSECT**

**SELECT NOMBRE, EDAD FROM ALUMNO;**



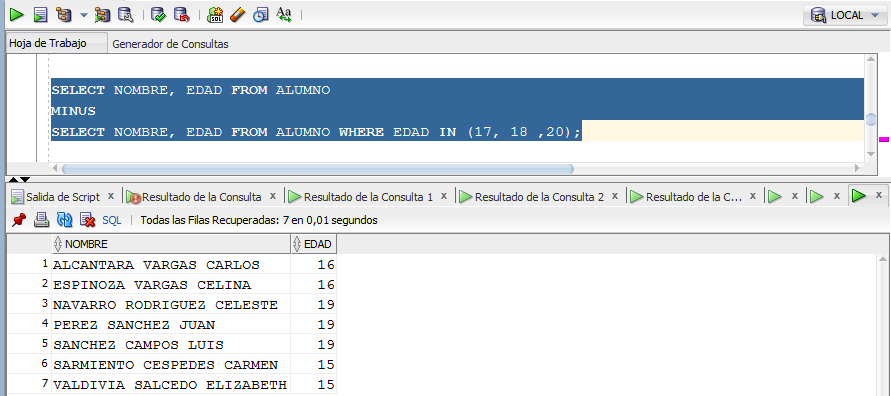
**Ejemplo usando MINUS.**

Este ejemplo muestra a toso los alumnos menos aquellos cuyas edades sea de 17, 18 y 20 años..

**SELECT NOMBRE, EDAD FROM ALUMNO**

**MINUS**

**SELECT NOMBRE, EDAD FROM ALUMNO WHERE EDAD IN (17, 18, 20);**



SUB CONSULTAS.

Se trata de una técnica que permite utilizar el resultado de una tabla SELECT en otra consulta SELECT. Permite solucionar problemas en los que el mismo dato aparece dos veces.

La sintaxis es:

**SELECT** lista\_Expresiones

**FROM** tabla

**WHERE** expresión **operador** (**SELECT** listaEx\_presiones **FROM** tabla);

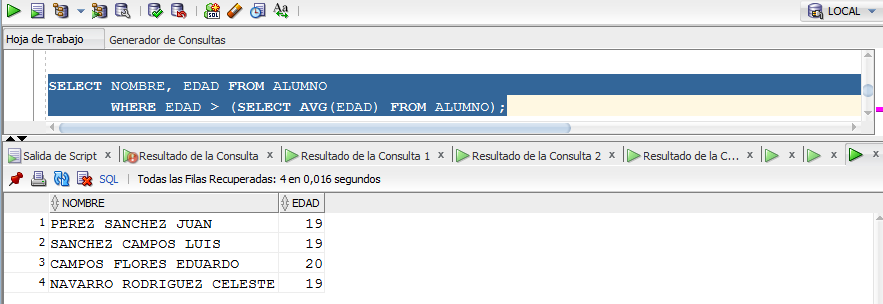
Se puede colocar el SELECT dentro de las cláusulas WHERE, HAVING o FROM. El operador puede ser: >, <, >=, <=, <>, = o IN.

**Ejemplo 1:**

Muestre a los alumnos que superen el promedio de edades.

**SELECT NOMBRE, EDAD FROM ALUMNO**

**WHERE EDAD > (SELECT AVG (EDAD) FROM ALUMNO);**



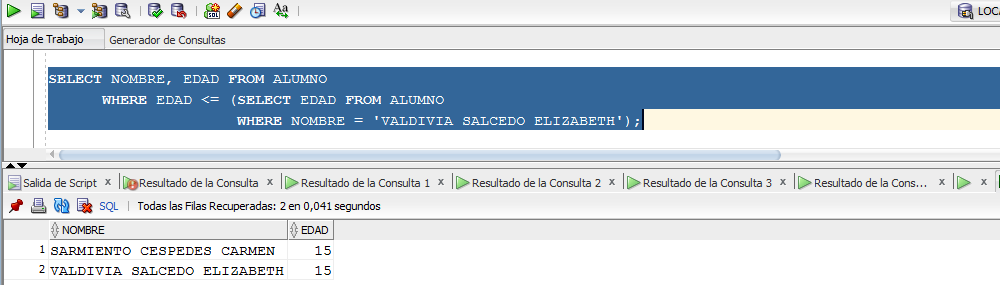
**Ejemplo 2:**

Se desea saber quiénes son los alumnos que no superan la edad de VALDIVIA SALCEDO ELIZABETH.

**SELECT NOMBRE, EDAD FROM ALUMNO**

**WHERE EDAD <= (SELECT EDAD FROM ALUMNO**

**WHERE NOMBRE = 'VALDIVIA SALCEDO ELIZABETH');**



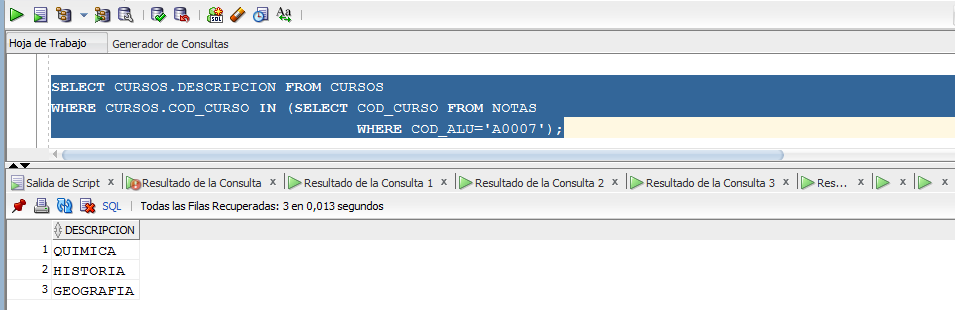
**Ejemplo 3:**

Se desea saber los cursos que tiene asignado el alumno **CAMPOS FLORES EDUARDO**, cuyo código de alumno es: **A0007**.

**SELECT CURSOS.DESCRIPCION FROM CURSOS**

**WHERE CURSOS.COD\_CURSO IN (SELECT COD\_CURSO FROM NOTAS**

**WHERE COD\_ALU='A0007');**



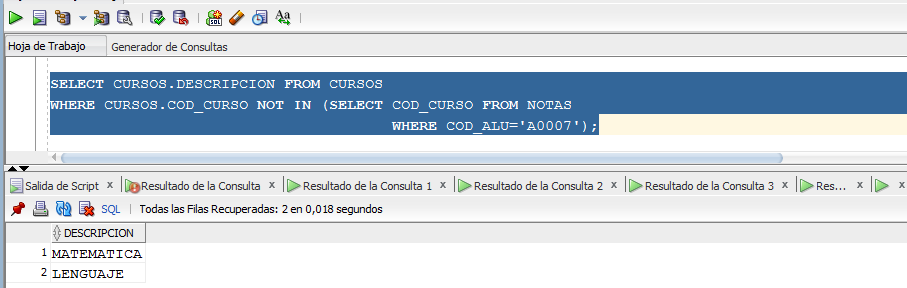
**Ejemplo 4:**

Se desea saber los cursos que no tiene asignado el alumno **CAMPOS FLORES EDUARDO**, cuyo código de alumno es: **A0007**.

**SELECT CURSOS.DESCRIPCION FROM CURSOS**

**WHERE CURSOS.COD\_CURSO NOT IN (SELECT COD\_CURSO FROM NOTAS**

**WHERE COD\_ALU='A0007');**



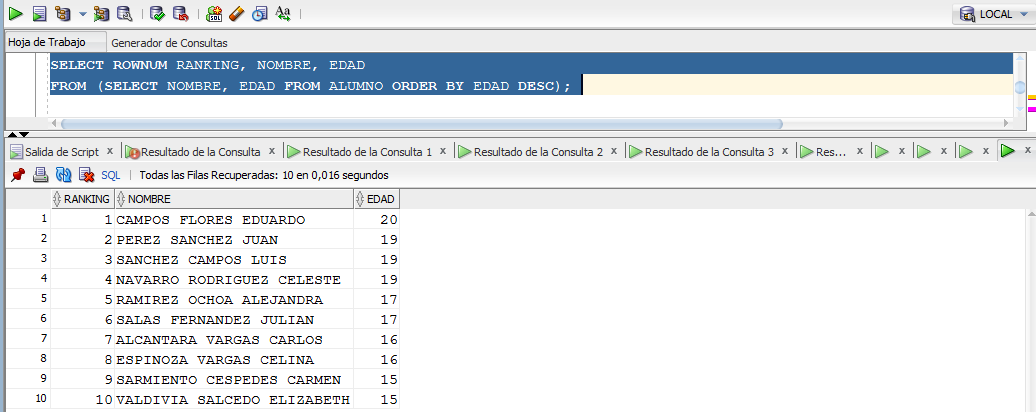
**Consultas con ROWNUM**

La función **ROWNUM** devuelve el número de la fila de una consulta. Por ejemplo:

**Ejemplo:** Mostrar el ranking de edades entre los alumnos.

**SELECT ROWNUM RANKING, NOMBRE, EDAD**

**FROM (SELECT NOMBRE, EDAD FROM ALUMNO ORDER BY EDAD DESC);**



IMPLEMENTA FUNCIONES PREDEFINIDAS DE UNA BD ORACLE A TRAVÉS DE LAS FUNCIONES DE TEXTO, MATEMÁTICAS Y FECHAS.

**FUNCIONES DE TEXTO.**

**CONCAT (cadena1, cadena2)**

Devuelve "cadena1" concatenada con "cadena2".

Por Ejemplo:

**SELECT CONCAT ('REY','ARTURO') FROM DUAL;**

Devuelve

**REYARTURO**

**Tener en cuenta que CONCAT solo permite 2 argumentos es decir solo 2 cadenas**

**¿Si quiere concatenar más caracteres?**

**Usamos “||” - (alt+124) =|**

Ejemplo:

**SELECT 'LLAVES PRINCIPALES'||' '||' Y '||' '||'FORANEAS' FROM DUAL;**

Devuelve:

**LLAVES PRINCIPALES Y FORANEAS**

**SUBSTR (char, m [, n])**

Devuelve una **subcadena** de char de n caracteres a partir de la posición m.

Ejemplo:

**SELECT SUBSTR ('CONSTANTINOPLA', 5, 5) FROM DUAL;**

Devuelve:

**TANTI**

¿Porque?-- si ubicamos la posición 5(m) desde la izquierda de la cadena, y contamos a partir de dicha posición 5 caracteres(n) a la derecha obtenemos la sub-cadena “**TANTI**”

**LENGTH (char)**

Devuelve un entero que representa la longitud de una cadena, es decir cuenta carácter por carácter incluido espacios en blanco dentro de una cadena. (los espacios en blanco deben estar entre los caracteres).

Ejemplo:

**SELECT LENGTH (‘MJOLNIER') FROM DUAL;**

Devuelve:

**8**

**INSTR (char1, char2)**

Devuelve la posición de inicio de la enésima aparición de char2 en Char1.

Ejemplo:

**SELECT INSTR ('REY ARTURO','AR') FROM DUAL;**

Devuelve:

**5**

**LPAD (char1, n [, char2])**

Rellena a una cadena por el lado izquierdo, hasta la longitud n con el carácter definido (char2)

Ejemplo:

**SELECT LPAD (‘PEPE GRILLO', 20,'-') FROM DUAL;**

Devuelve:

**----------PEPE GRILLO**

**RPAD (char1, n [, char2])**

Rellena la cadena por la derecha hasta la longitud n con el carácter definido (char2)

Ejemplo:

**SELECT RPAD ('PEPE GRILLO', 20,'-') FROM DUAL;**

Devuelve:

**PEPE GRILLO----------**

**TRIM (char)**

Esta función permite quitar los espacios en blanco a los extremos de una cadena.

Ejemplo:

**SELECT TRIM (' JOJOJO ') FROM DUAL;**

Devuelve:

**JOJOJO**

**RESUMEN DE LAS FUNCIONES DE MANIPULACIÓN DE CARACTERES:**



**FUNCIONES MATEMÁTICAS:**

**ROUND:** Redondea el valor a los decimales especificados.

Ejemplo:

**SELECT ROUND (45.926, 2) FROM DUAL;**

Devuelve:

**45.93**

**TRUNC**: Trunca el valor a los decimales especificados.

**SELECT TRUNC (45.926, 2) FROM DUAL;**

Devuelve:

**45.92**

**MOD**: Devuelve el resto de la división.

**SELECT MOD (1600, 300) FROM DUAL;**

Devuelve:

**100**

**ABS**: Calcula el valor absoluto de n.

**SELECT ABS (-16) FROM DUAL;**

**16**

**CEIL**: Calcula el menor número entero mayor o igual que n.

**SELECT CEIL (16.7) FROM DUAL;**

Devuelve:

**17**

**FLOOR**: Calcula el mayor número entero menor o igual que n.

**SELECT FLOOR (16.7) FROM DUAL;**

Devuelve:

**16**

**POWER**: Devuelve m elevado a la n potencia, n debe ser entero

**SELECT POWER (3, 2) FROM DUAL;**

Devuelve:

**9**

**FUNCIONES DE FECHA:**

|  |  |
| --- | --- |
| **Función** | **Descripción** |
| MONTHS\_BETWEEN | Numero de meses entre dos fechas. |
| ADD\_MONTHS | Suma meses de calendario a una fecha. |
| NEXT\_DAY | Siguiente día de la fecha especificada. |
| LAST\_DAY | Ultimo día del mes. |
| ROUND | Redondea la fecha. |
| TRUNC | Trunca la fecha. |

Cuantos meses existen entre el 11 de Enero del 2019 y el 01 de Septiembre del 2019

**SELECT MONTHS\_BETWEEN ('01/09/2019','11/01/2019') FROM DUAL;**

Resultado: **7,67741935483870967741935483870967741935**

Se puede redondear la fecha al igual que las funciones numéricas de la siguiente forma

**SELECT ROUND (MONTHS\_BETWEEN ('01/09/2019','11/01/2019'), 1) FROM DUAL;**

Resultado: **7,7**

Aumentar 6 meses al 12 de Agosto del 2019

**SELECT ADD\_MONTHS (‘12/08/2019’, 6) FROM DUAL;**

Resultado**: 12/02/2020**

Qué fecha cae el siguiente viernes después del 12 de Agosto del 2019

**SELECT NEXT\_DAY ('12/08/2019','VIERNES') FROM DUAL;**

Resultado**: 16/08/2019**

Cuál es la fecha del último día del mes correspondiente a Febrero del 2019

**SELECT LAST\_DAY ('01/02/2019') FROM DUAL;**

**Resultado: 28/02/2019**

Como obtener la fecha del sistema.

**SELECT SYSDATE FROM DUAL;**

Resultado: **12/08/2019**

Sumarle 3 días a la fecha actual

**SELECT SYSDATE + 3 FROM DUAL;**

Resultado: **15/08/2019**

**CONVERSION A TIPOS DE DATOS.**

**DATE a CHAR (TO\_CHAR):**

Convertir la fecha del día de hoy al formato DD/MM/YYYY HH24:MI:SS

**SELECT TO\_CHAR (SYSDATE,'DD/MM/YYYY HH24:MI:SS') FROM DUAL;**

Resultado: **12/08/2019 01:43:11**

Los formatos pueden ser:

|  |  |
| --- | --- |
| **YYYY** | **Año completo en números** |
| YEAR | Años en letra |
| MM | Valor de dos dígitos para el mes |
| MONTH | Nombre completo del mes. |
| MON | Abreviatura de tres letras del mes. |
| DY | Abreviatura de tres letras del día de la semana. |
| DAY | Nombre completo del día de la semana. |
| DD | Dia del mes en número. |

**CHAR a DATE (TO\_DATE):**

Convertir la cadena 23/10/2012 al formato fecha para realizar operaciones tipo fecha

**SELECT TO\_DATE ('12/08/2019','DD/MM/YYYY') FROM DUAL;**

Resultado: **12/08/2019** --(tipo de dato date)

**CHAR a NUMBER (TO\_NUMBER):**

Convertir el texto '1567,7' a formato numérico para realizar operaciones matemáticas

**SELECT TO\_NUMBER ('1567, 7’) FROM DUAL;**

Resultado: **1567,7** --(tipo de dato number)

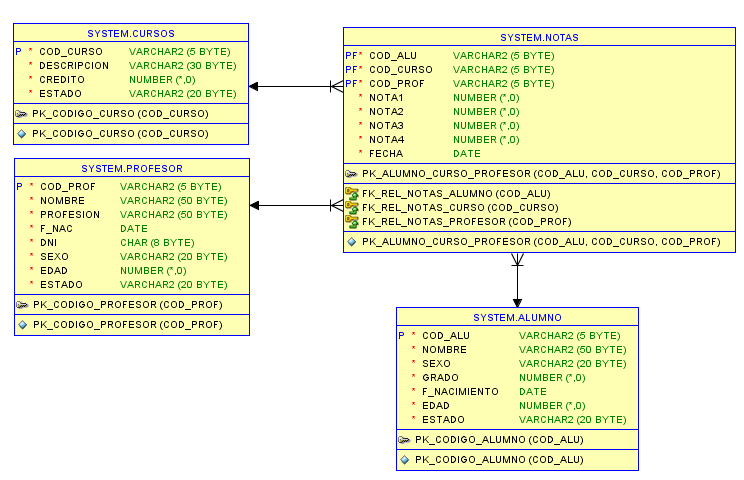
**CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA**

El objetivo de este capítulo es comprender el uso de consultas avanzadas, de esta manera el estudiante se compenetra más con la base de datos de Oracle, al analizar la información, resultado de las búsquedas realizadas de manera exitosa. Mediante el uso de las herramientas SQL \*PLUS y **PL/SQL**.

ACTIVIDAD VIRTUAL

* Ingresa a la plataforma virtual, revisa y analiza el material presentado en esta semana. Luego sigue las instrucciones para realizar la actividad propuesta.

De acuerdo al siguiente modelo relacional**,** desarrolla los siguientes enunciados:



1. **FECILITE UNA CONSULTA, USANDO LA CLAUSULA WHERE**
2. **FACILITE UNA CONSULTA, USANDO LA CLAUSULA JOIN**
3. **FACILITE UNA CONSULTA, USANDO LA CLAUSULA INNER JOIN.**
4. **FACILITE UNA CONSULTA, USANDO LA CLAUSULA NATURAL JOIN.**
5. **FACILITE UNA CONSULTA, USANDO LA CLAUSULA CROSS JOIN.**
6. **FACILITE UNA SUB CONSULTA.**