Programación en Base de Datos Oracle

LENGUAJE DE MANIPULACIÓN DE DATOS DML I

Semana 03:

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)

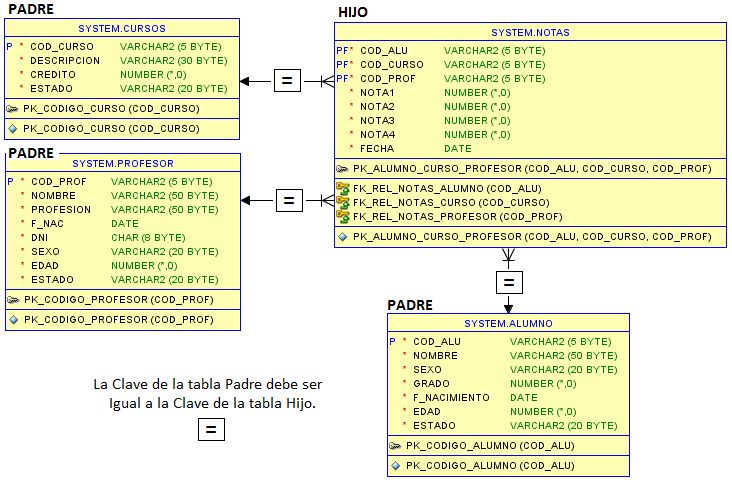
Usar dirección IP única en la máquina en la que se instala Oracle. Es decir, no usar DHCP para direccionar la IP en el servidor de Oracle. No es un requisito obligatorio, pero es muy recomendable.

MANIPULA LA INFORMACIÓN DE LA BASE DE DATOS UTILIZANDO DML INSERCIÓN, ACTUALIZACIÓN Y ELIMINACIÓN.

IMPORTANTE

**LA ESTRUCTURA Y DATOS DE LAS TABLAS, SE ENCUENTRAN EN LOS ARCHIVOS SCRIPT DE BASE DE DATOS Semana03\_DML\_1.sql, FAVOR DE DESCARGAR EL ARCHIVO.**

Es una de las partes fundamentales del lenguaje SQL. El DML (Data Manipulation Language) lo forman las instrucciones capaces de modificar los datos de las tablas. Al conjunto de instrucciones **DML** que se ejecutan consecutivamente, se las llama transacciones y se pueden anular todas ellas o aceptar, ya que una instrucción DML no es realmente efectuada hasta que no se acepta (**commit**) o rechazar/des hacer (**rollback**). En todas estas consultas, el único dato devuelto por Oracle es el número de registros que se han modificado.



Este modelo relacional servirá para realizar las pruebas necesarias, en las tareas de inserción de registros, usando diferentes métodos de acceso de datos a cada tabla de dicho modelo relacional. El Script le será proporcionado por su instructor de clases.

**Instrucciones DML y operadores**

Las instrucciones de manipulación de datos (DML) permitidas son **SELECT**, **INSERT**, **UPDATE** y **DELETE**:

* + SELECT devuelve filas de la base de datos que cumplan los criterios definidos en su cláusula WHERE.
  + INSERT añade filas a una tabla de La base de datos.
  + UPDATE DELETE, modifica las filas de una tabla que cumplan la cláusula WHERE
  + DELETE WHERE, borra las filas identificadas por la cláusula WHERE.

**INSERCION DE DATOS (INSERT).**

**Inserción de datos Método Implícito:**

Es aquella instrucción **INSERT** que omite el nombre de las columnas en el momento que se realiza la inserción de datos, esta se basa en el orden de las columnas de izquierda a derecha, es decir se asignan datos en cada columna de la tabla de manera ordenada, desde la primera columna, uno por uno, hasta llegar a la última columna. **REGLA DE INSERCION: SIEMPRE SE DEBE EMPEZAR A INSERTAR DATOS EN LAS TABLAS PADRES Y LUEGO EN LAS TABLAS HIJOS.**

**Sintaxis del Método implícito:**

**INSERT INTO <Nombre Tabla> Values (valor1, valor2, valor3, . . . , valor(n));**

**Inserción de datos, Método Explicito.**

Es aquella instrucción **INSERT** que especifica los nombres de columnas donde se va a almacenar los valores, también se puede especificar la clave NULL es la cláusula VALUES.

**Sintaxis del Método Explicito.**

**INSERT INTO <Nombre Tabla> (campo 1, Campo2, Campo3, . . . . , Campo(n) Values (valor1, valor2, valor3,………, valor(n));**

**Copiar filas de datos desde otra tabla:**

Simplemente se realiza un INSERT con una sub consulta.

**Sintaxis para copiar filas de datos desde otra tabla**

**INSERT INTO <Nombre Tabla> (campo 1, Campo2, Campo3, . . . . , Campo(n)**

**<SELECT Campo1, Campo2, Campo3, . . , Campo(n) FROM <Tabla> WHERE <Condición>>**

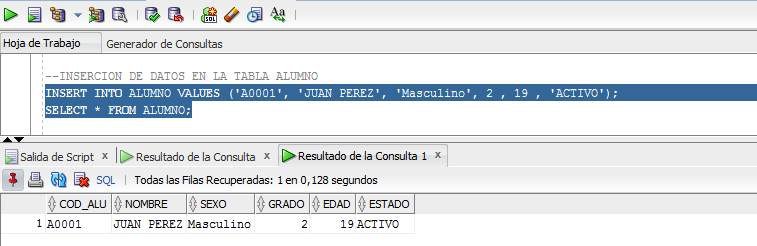
**Ejemplo de inserción de método Implícito.**

En este caso usaremos la tabla **ALUMNO** de nuestro modelo relacional, almacenaremos valores en cada columna de la tabla, sin necesidad de nombrar a las columnas, solo respetando el orden de los campos.

**INSERT INTO ALUMNO VALUES (‘A0001’, ’JUAN PEREZ’, ’Masculino’, 2, 19, ’ACTIVO’);**



Si probamos la inserción y una consulta los datos, los resultados se verán de la siguiente manera:



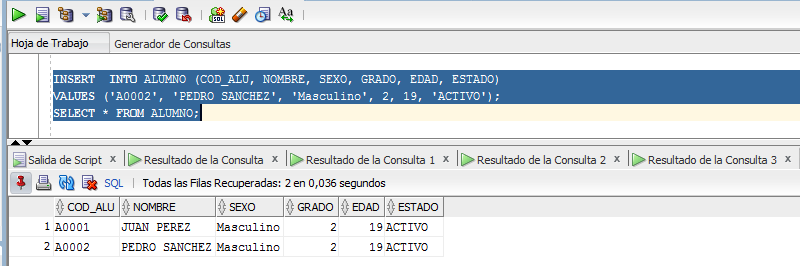
**Ejemplo de inserción de método Explicito.**

En este caso usaremos la tabla **ALUMNO** de nuestro modelo relacional, almacenaremos valores en cada columna de la tabla, nombrando las columnas.

**INSERT INTO ALUMNO (COD\_ALU, NOMBRE, SEXO, GRADO, EDAD, ESTADO)**

**VALUES (‘A0001’, ’JUAN PEREZ’, ’Masculino’, 2, 19, ’ACTIVO’);**

Si probamos la inserción y una consulta de datos, los resultados se verán de la siguiente manera:



**Ejemplo de copiar filas de datos, desde otra tabla.**

En este ejemplo las tablas origen (ALUMNO) y destino (COPIA\_ALUMNO) deben existir, y tener la misma estructura.

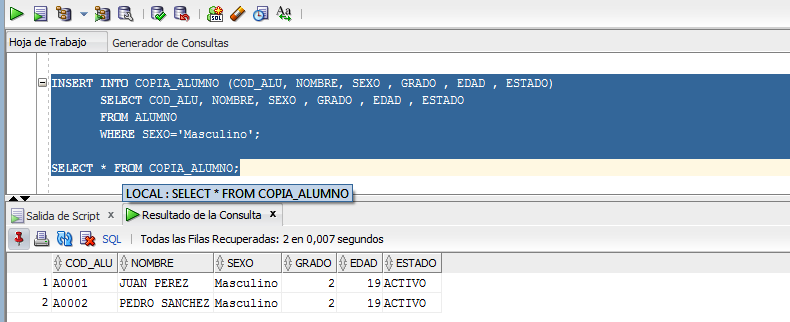
**INSERT INTO COPIA\_ALUMNO (COD\_ALU, NOMBRE, SEXO, GRADO, EDAD, ESTADO)**

**SELECT COD\_ALU, NOMBRE, SEXO, GRADO, EDAD, ESTADO**

**FROM ALUMNO**

**WHERE SEXO='Masculino';**

Realizando la Prueba:



**ACTUALIZACION DE DATOS (UPDATE).**

**Sintaxis:**

**UPDATE** <Nombre\_Tabla> SET

Nombre\_columna1 = expresion1,

Nombre\_columna2 = expresion2,

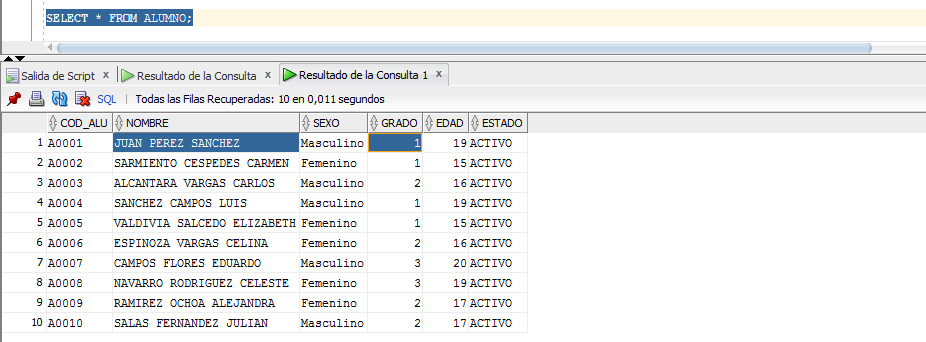
Nombre\_columna3 = expresion3,…

**WHERE** <Condicion>;

**REGLA DE ACTUALIZACION: NUNCA ACTUALICE DATOS EN COLUMNAS TENGAN VALORES REFERENCIALES CON OTRAS TABLAS.**

**Ejemplo 1:**

Se desea modificar el nombre del alumno: **JUAN PEREZ SANCHEZ** por **PEREZ SANCHEZ JUAN**, y el grado de dicho alumno será **3**.

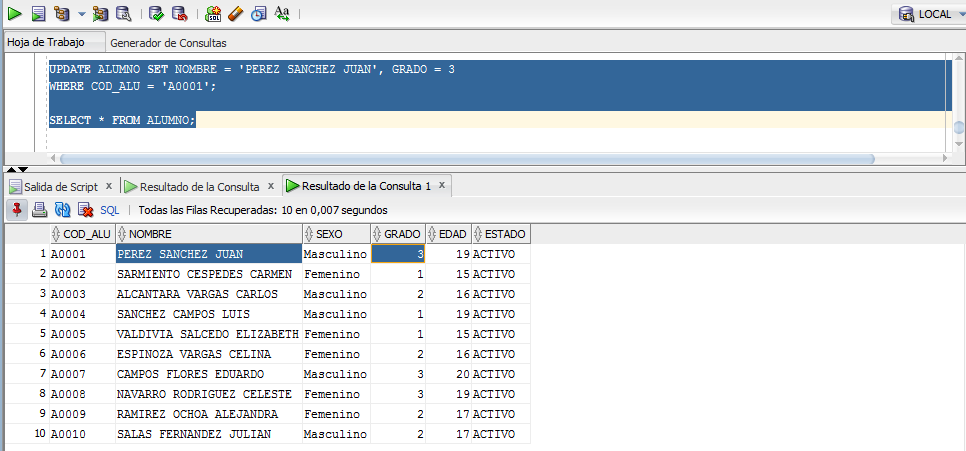


**SOLUCION:**

**UPDATE ALUMNO SET NOMBRE = 'PEREZ SANCHEZ JUAN', GRADO = 3**

**WHERE COD\_ALU = 'A0001';**

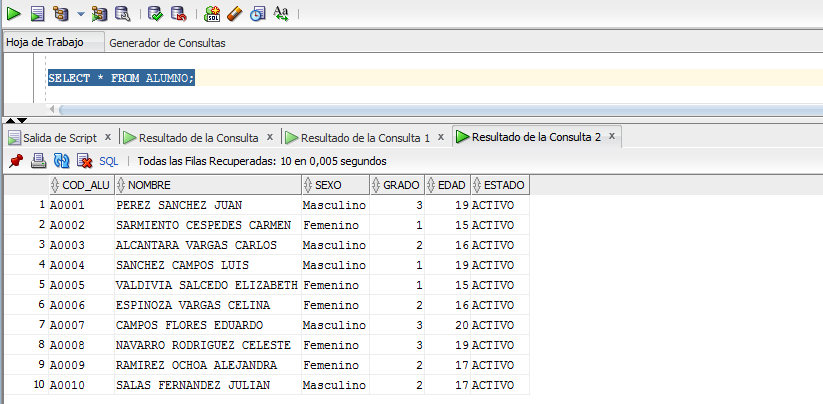
**SELECT \* FROM ALUMNO;**



**Ejemplo 2:**

Se desea bloquear a todos los alumnos.

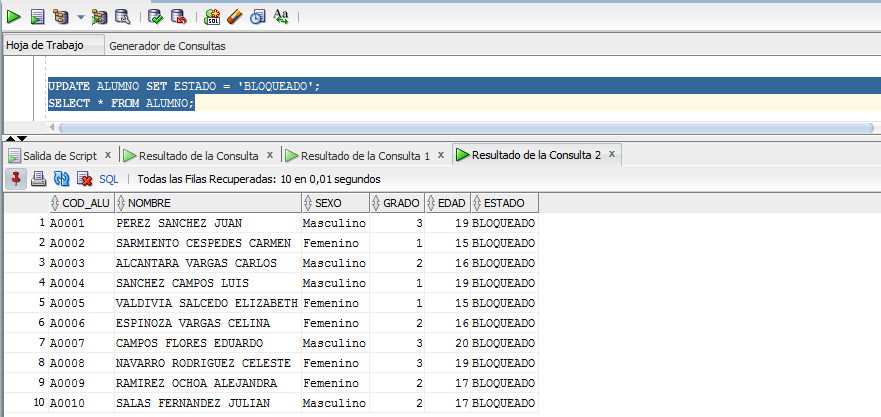
En este momento los Estados son **ACTIVOS**.



**SOLUCION:**

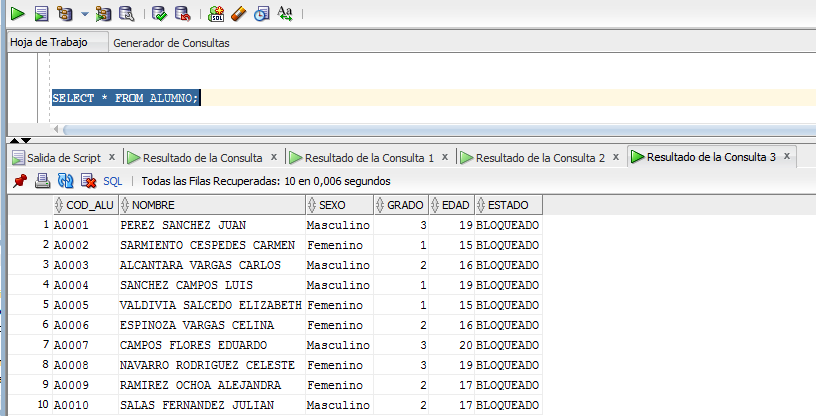
**UPDATE ALUMNO SET ESTADO = 'BLOQUEADO';**

**SELECT \* FROM ALUMNO;**

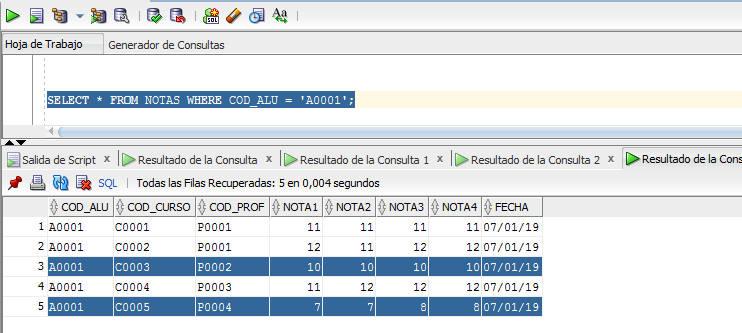


**Ejemplo 3:**

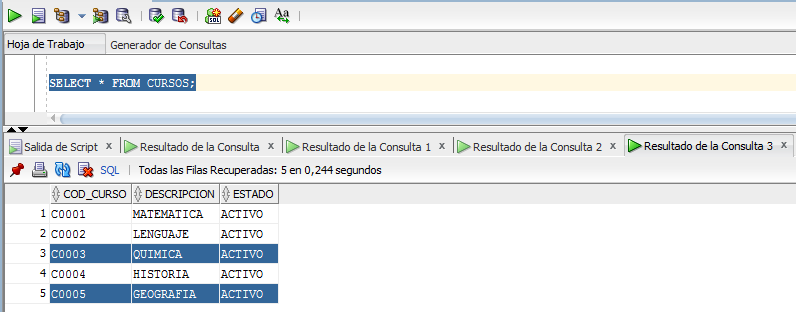
El alumno PEREZ SANCHEZ JUAN, tiene como código de alumno a A0001.



Dicho alumno tiene 2 cursos con bajas calificaciones, los códigos de curso son: **C0003** y **C0005**.



**¿Cuáles son esos Cursos? Es QUIMICA y GEOGRAFIA.**



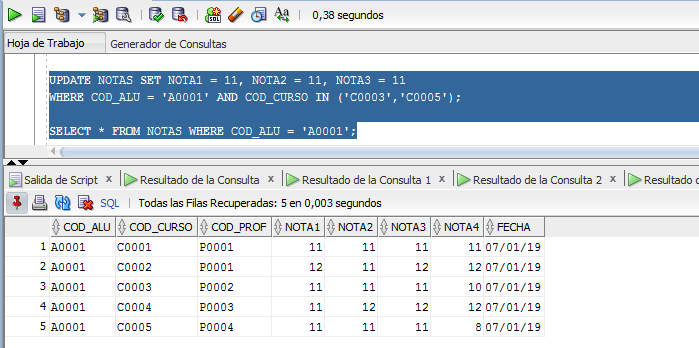
Se pide actualizar las notas de dichos cursos, asignar la calificación de 11 a todos

**SOLUCION:**

**UPDATE NOTAS SET NOTA1 = 11, NOTA2 = 11, NOTA3 = 11, NOTA4 = 11**

**WHERE COD\_ALU = 'A0001' AND COD\_CURSO IN ('C0003','C0005');**

**SELECT \* FROM NOTAS WHERE COD\_ALU = 'A0001';**



**ELIMINAR REGISTROS: DELETE FROM**

La orden DELETE elimina los registros que cumplen con la condición definida en el WHERE. La sintaxis es la siguiente:

**DELETE FROM** <Nombre\_de\_Tabla> **WHERE** <Condicion>;

Nota: Si en algún momento se eliminan registro que tengan referencias con datos de otras tablas (Relación), debe tomar una decisión importante, y estas pueden ser:

* Eliminar los registros usando la finalización CASCADE CONTRAINT, este comando rompe la relación y procede a eliminar los registros.
* Simplemente no borrar.

**Ejemplo 1:**

Eliminar los registros de la tabla alumno, solo aquellos cuyo grado sea 1.

Solución:

**DELETE FROM ALUMNO WHERE GREADO = 1;**

**Ejemplo 2:**

Eliminar los registros de la tabla alumno, aquellos cuyos grados sean 3, 4, 5

Solución:

**DELETE FROM ALUMNO WHERE GREADO = 3 AND GRADO = 4 AND GRADO = 5;**

**Ejemplo 3:**

Elimine todos los registros de la tabla alumno.

Solución:

**DELETE FROM ALUMNO;**

**CONFIRMAR Y DESHACER.**

Comandos: **COMMIT** y **ROLLBACK**;

COMMIT: Comando de confirmación de datos, es decir su uso confirma la inserción, eliminación y actualización de datos en la base de datos.

ROLLBACK: Comando que deshace los datos, es decir su uso descarta toda inserción, eliminación y actualización de datos en la base de datos.

**EN QUE MOMENTO USARLO:**

Usar estos comandos después de una inserción, eliminación y actualización de datos.

RECUPERA INFORMACIÓN DE UNA BASE DE DATOS ORACLE A TRAVÉS DE SENTENCIAS SQL, CONDICIONALES, FUNCIONES AGREGADAS, AGRUPACIÓN DE DATOS Y COMBINACIONES DE TABLAS

**CONSULTA DE DATOS: COMAND SELECT.**

Una instrucción SELECT extrae datos de la base de datos. Permite consultar los valores que contienen los campos de tablas y/o vistas de la base de datos. La forma general de una instrucción SELECT es:

**SELECT [DISTINCT | ALL]** {\*, columna [alias], expresión, …}

**FROM** <tablas> [WHERE condición(es)]

[**GROUP BY** expresión [, expresión] … ]

[**HAVING** condición]

[**ORDER BY** {columna, expresión, alias} {**ASC** | **DESC**}]

**APRENDER A SELECCIONAR TODAS LAS COLUMNAS DE LA TABLA.**

Carácter comodín \*

**SELECT \* FROM ALUMNO;**

**APRENDER A SELECCIONAR COLUMNAS ESPEDIFICAS.**

Seleccione las columnas de código, nombre y sexo del alumno.

**SELECT COD\_ALU, NOMBRE, SEXO FROM ALUMNO;**

**APRENDER A CREAR CAMPOS CALCULADOS.**

**Aritméticas**

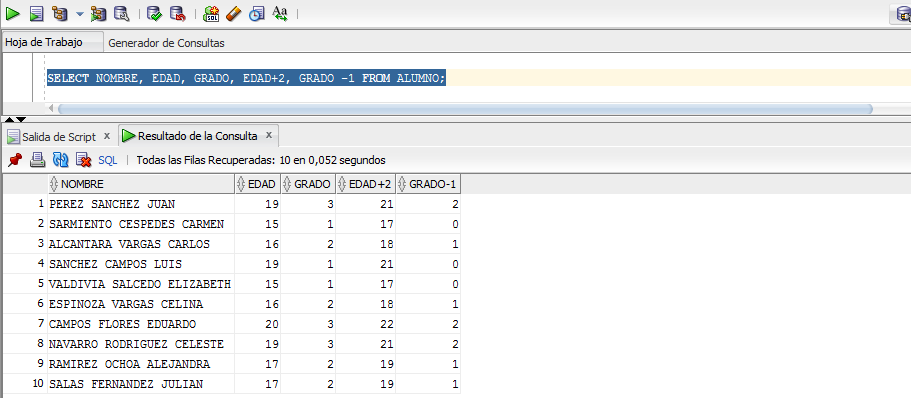
Crear expresiones con datos de fecha y números mediante operadores aritméticos.

|  |  |
| --- | --- |
| **+** | SUMA |
| **-** | RESTA |
| **\*** | MULTIPLICACION |
| **/** | DIVISION |

**Ejemplo:**

Aumentar la edad del alumno en 2 años y restar los grados en 1.

**SELECT NOMBRE, EDAD, GRADO, EDAD+2, GRADO-1 FROM ALUMNO;**



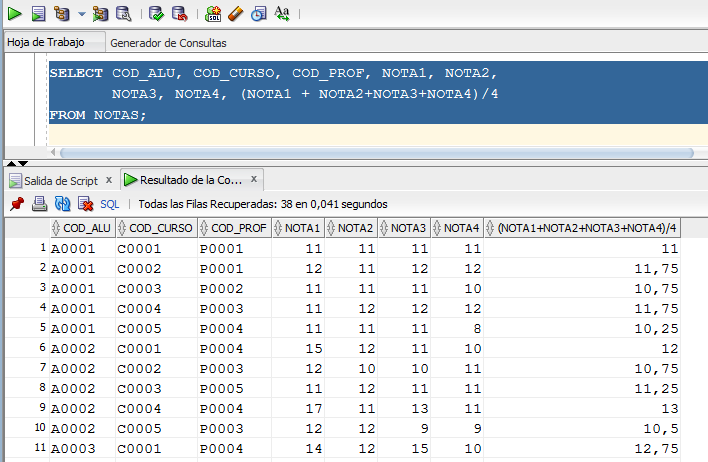
**Ejemplo 2:**

Obtener el Promedio de las calificaciones.

**SELECT COD\_ALU, COD\_CURSO, COD\_PROF, NOTA1, NOTA2,**

**NOTA3, NOTA4, (NOTA1 + NOTA2+NOTA3+NOTA4)/4**

**FROM NOTAS;**



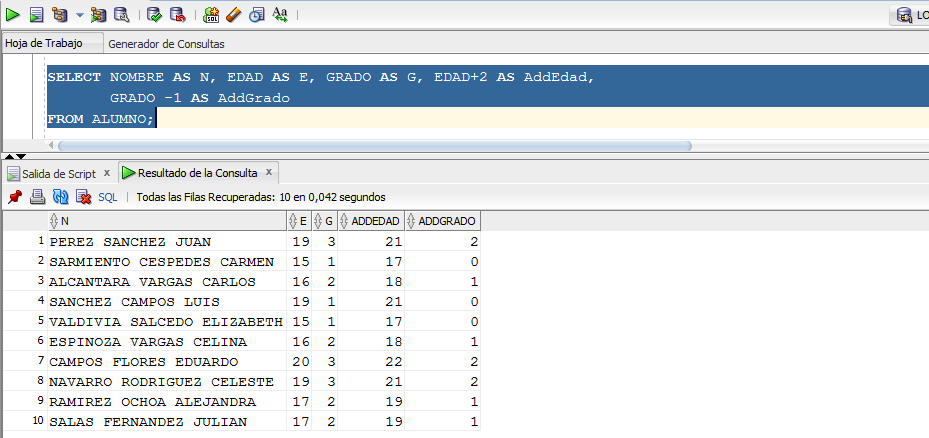
**APRENDER A CREAR ALIAS A LAS COLUMNAS.**

Asignar al campo Nombre el alias **N**, Edad el alias **E**, Grado el alias **G**, Edad+2 el alias **AddEdad**, Grado – 1 el alias **AddGrado**. Mediante una consulta realizada en la tabla ALUMNO.

**SELECT NOMBRE AS N, EDAD AS E, GRADO AS G, EDAD+2 AS AddEdad,**

**GRADO -1 AS AddGrado**

**FROM ALUMNO;**



**Operador de Concatenación**

* Un operador de concatenación:
* Enlaza columnas o cadenas de caracteres a otras columnas
* Se representa con dos barras verticales (||)
* Crea una columna resultante que es una expresión de carácter.

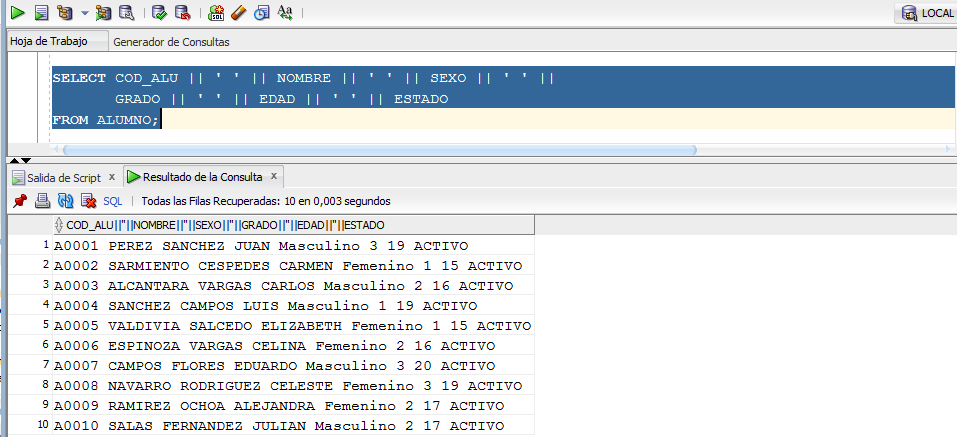
**Ejemplo 1:**

Mediante una consulta unir los datos de todas las columnas las columnas de la tabla alumno.

**SELECT COD\_ALU || ' ' || NOMBRE || ' ' || SEXO || ' ' ||**

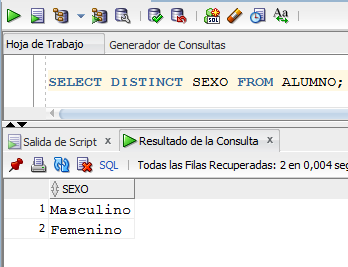
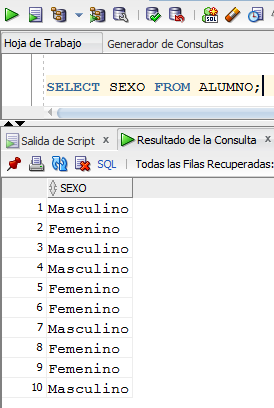
**GRADO || ' ' || EDAD || ' ' || ESTADO**

**FROM ALUMNO;**



**APRENDER A SEPARAR FILAS DUPLICADAS. DISTINCT.**

La visualización por defecto de las consultas incluye todas las filas, también las filas duplicadas.



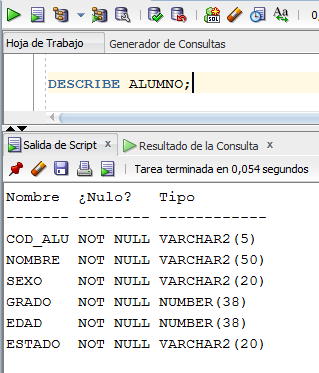
**APRENDER A VISUALIZAR LA ESTRUCTURA DE UNA TABLA: COMANDO DESCRIBE.**

Sintaxis:

**DESCRIBE** <Nombre\_Tabla>**;**

Ejemplo:

**DESCRIBE ALUMNO;**



**APRENDE A GENERAR CONSULTAS BASADAS EN CONDICIONALES: CLAUSULA WHERE.**

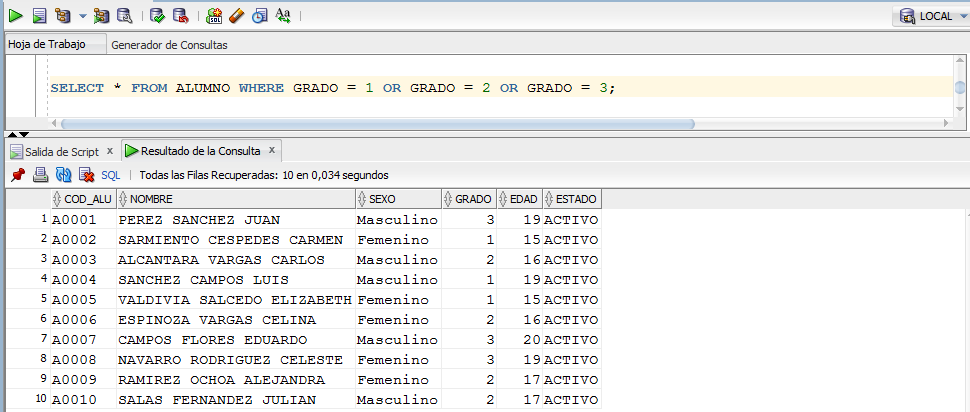
**Ejemplo 1:**

Mostrar aquellos alumnos cuyo grado sea 1, 2 ó 3.

**SELECT \* FROM ALUMNO WHERE GRADO = 1 OR GRADO = 2 OR GRADO = 3;**

**ó**

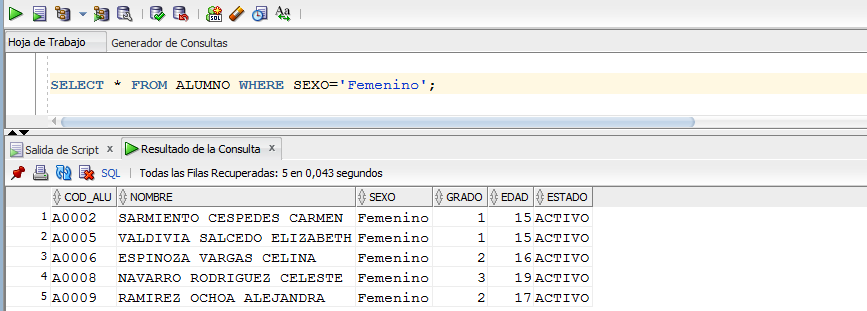
**SELECT \* FROM ALUMNO WHERE GRADO IN (1, 2, 3);**



**Ejemplo 2:**

Mostrar solo a los alumnos del sexo Femenino.

**SELECT \* FROM ALUMNO WHERE SEXO='Femenino';**



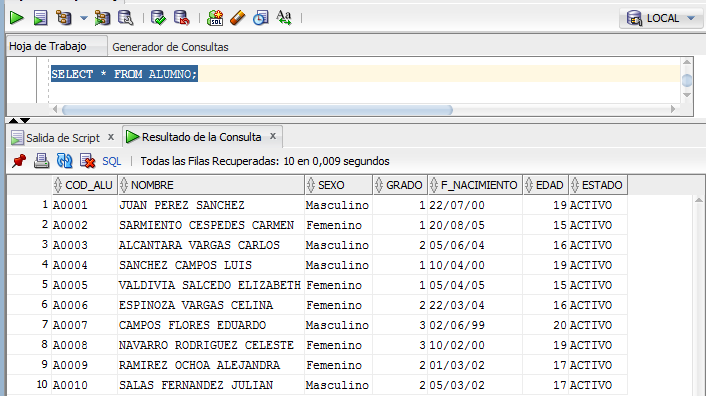
**Fechas y cadenas de caracteres**

* Las cadenas de caracteres y valores de fecha se incluyen entre comillas simples.
* Los valores de caracteres son sensibles a mayúsculas / minúsculas y los valores de datos son sensibles a formato.
* El formato de visualización de la fecha por defecto es DD-MON-RR

**Ejemplo:**

Mostrar un listado de alumnos que nacieron a partir del 01 de enero del año 2005.

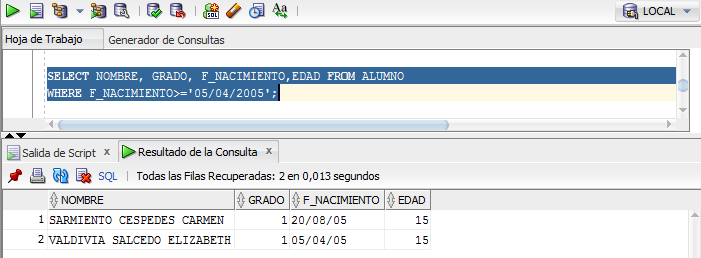
**Consulta preliminar:**



**SOLUCION:**

**SELECT NOMBRE, GRADO, F\_NACIMIENTO, EDAD FROM ALUMNO**

**WHERE F\_NACIMIENTO>='05/04/2005';**

**RESULTADOS:**

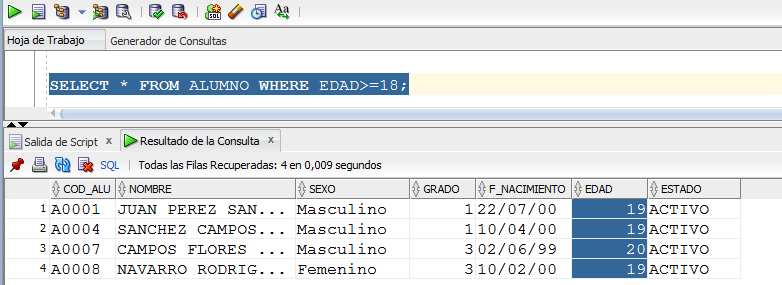
**OPERADORES DE COMPARACION:**

|  |  |
| --- | --- |
| **Operador** | **Descripción** |
| **=** | Igual que |
| **>** | Mayor que |
| **<** | Menor que |
| **>=** | Mayor igual que |
| **<=** | Menor igual que |
| **<>** | Diferente |
| **BETWEEN - AND** | Entre dos valores. |
| **IN ( lista de valores)** | Incluido en la lista de valores |
| **LIKE** | Que coincida con un patrón de valores. |
| **IS NULL** | Es un valor nulo. |

**Uso del operador de comparación:**

**Ejemplo 1: USO DEL OPERADOR >=**

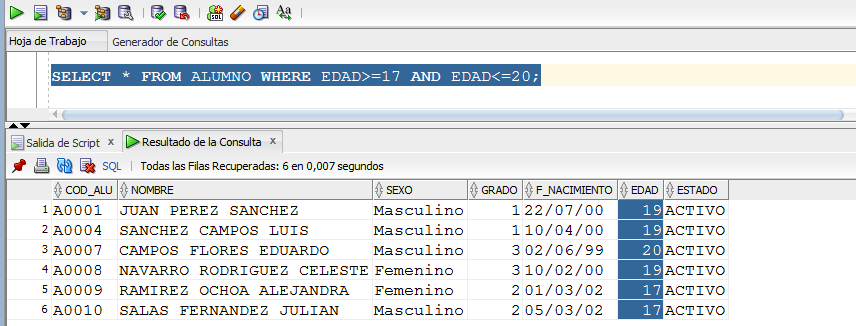
Visualice a los alumnos que son mayores de edad.

**SELECT \* FROM ALUMNO WHERE EDAD >= 18;**

**Ejemplo 2: USO DEL OPERADOR >=, <=**

Visualice aquellos alumnos cuyas edades estén comprendidos entre 17 7 20 años.

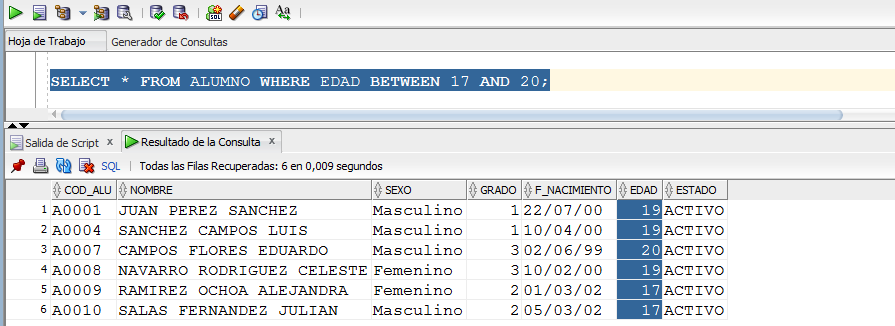
**SELECT \* FROM ALUMNO WHERE EDAD>=17 AND EDAD<=20;**



**Ejemplo 3: USO DEL OPERADOR BETWEEN AND.**

El ejemplo anterior, puede ser resuelto usando el comando BETWEEN.

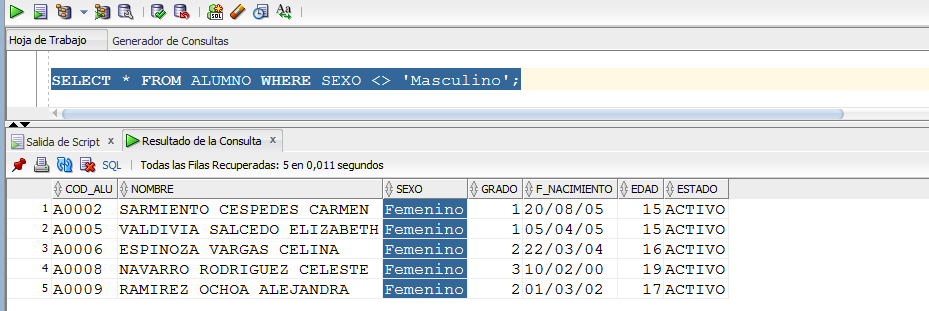
**SELECT \* FROM ALUMNO WHERE EDAD BETWEEN 17 AND 20;**



**Ejemplo 4: USO DEL OPERADOR < >**

Visualice aquellos alumnos que sean diferente a varones.

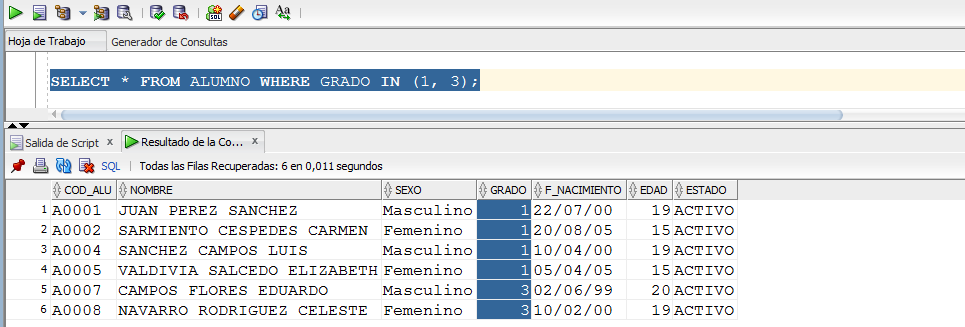
**SELECT \* FROM ALUMNO WHERE SEXO <> 'Masculino';**



**Ejemplo 5: USO DEL OPERADOR IN.**

Visualice aquellos alumnos de los grados 1 y 3.

**SELECT \* FROM ALUMNO WHERE GRADO IN (1, 3);**



**USO DEL OPERADOR LIKE.**

Antes de usar este operador, debe tener en cuenta lo siguiente:

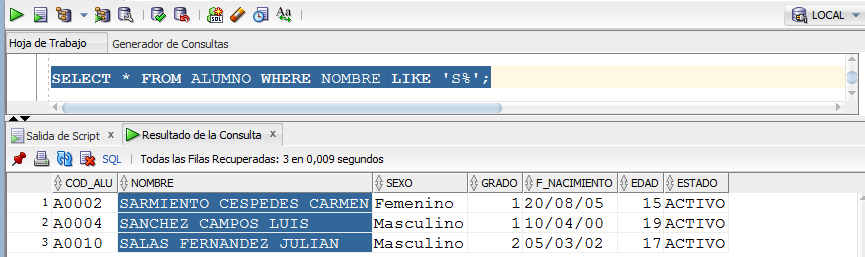
Utilizar el operador LIKE para realizar búsquedas con comodines de valores de cadena de búsqueda válidos.

* Las condiciones de búsqueda pueden contener caracteres literales o números
* **%** indica cero o varios caracteres
* **\_** indica un carácter

**Ejemplo 6: USO DEL OPERADOR LIKE.**

Visualice aquellos alumnos cuyos nombre empiecen con la letra “S”.

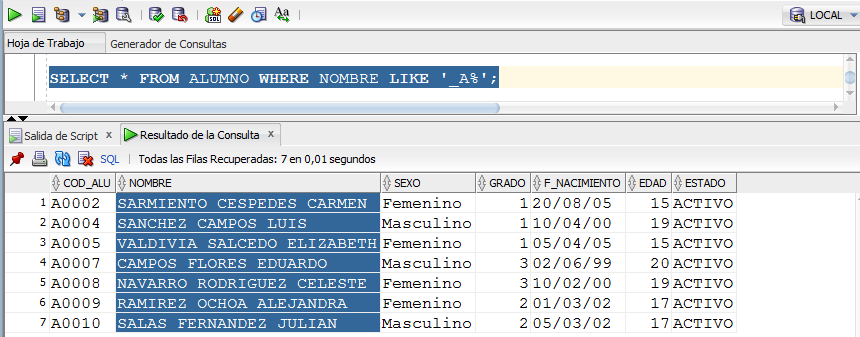
**SELECT \* FROM ALUMNO WHERE NOMBRE LIKE 'S%';**



**Ejemplo 7: USO DEL OPERADOR LIKE.**

Visualice aquellos alumnos cuya segunda letra sea “A”.

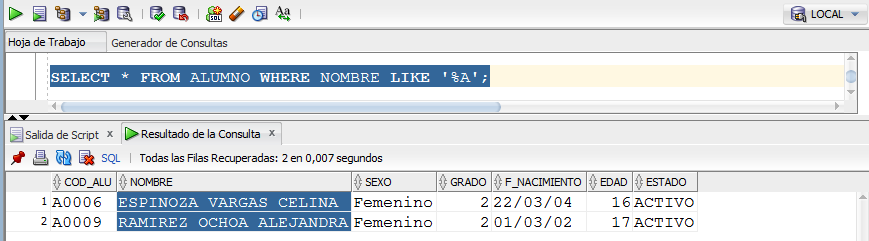
**SELECT \* FROM ALUMNO WHERE NOMBRE LIKE '\_A%';**



**Ejemplo 8: USO DEL OPERADOR LIKE.**

Visualice aquellos alumnos, que en el nombre finalice con la letra “A”.

**SELECT \* FROM ALUMNO WHERE NOMBRE LIKE '%A';**

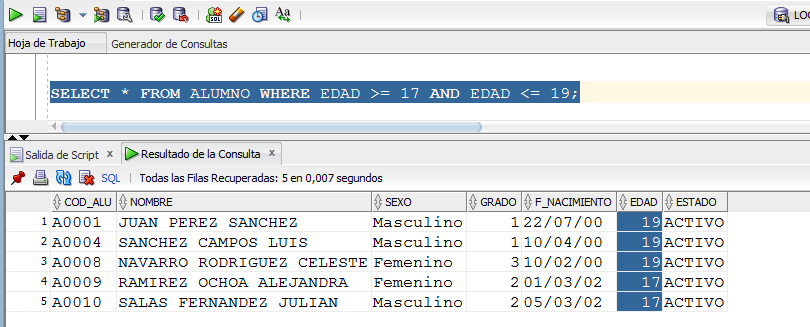


**Condicionales mediante Operadores Lógicos.**

**Ejemplo 9: USO DEL OPERADOR AND**

Visualice a los alumnos cuyas edades se encuentren entre 17 y 19 años.

**SELECT \* FROM ALUMNO WHERE EDAD >= 17 AND EDAD <= 19;**



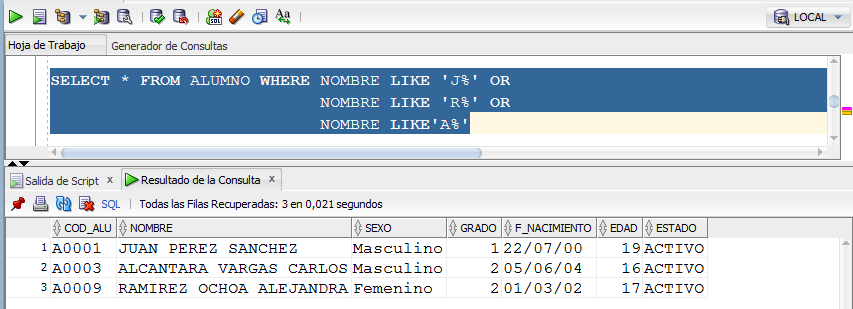
**Ejemplo 10: USO DEL OPERADOR OR.**

Visualice a los alumnos cuyos nombres empiecen con los caracteres J, R, o A.

**SELECT \* FROM ALUMNO WHERE NOMBRE LIKE 'J%' OR**

**NOMBRE LIKE 'R%' OR**

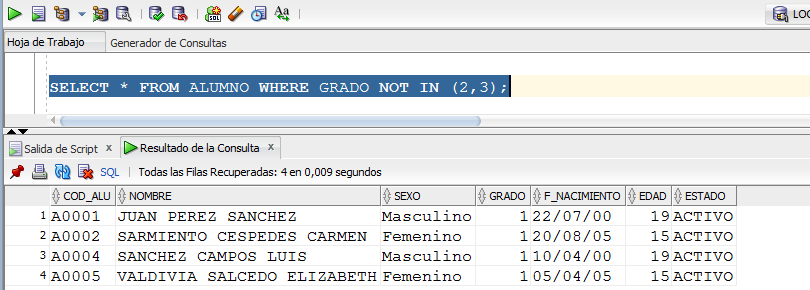
**NOMBRE LIKE'A%'**



**Ejemplo 11: USO DEL OPERADOR NOT IN.**

Visualice a los alumnos que no sean de los grados 2 y 3.

**SELECT \* FROM ALUMNO WHERE GRADO NOT IN (2, 3);**



**Ejemplo 12: USO DEL OPERADOR IN.**

Visualice a los alumnos que sean de los grados 2 y 3.

**SELECT \* FROM ALUMNO WHERE GRADO NOT IN (2, 3);**



**APRENDER A USAR LA CLAUSULA ORDER BY.**

1. Ordenar las filas recuperadas con la cláusula **ORDER BY**:

**ASC**: ordena los registros de manera ascendente, Caracteres(A..Z, a...z) y Numéricos (1..n), si no se especifica será valor por defecto

**DESC**: ordena los registros de manera descendente, Caracteres(Z..A, z...a) y Numéricos (n..1), su especificación es obligatoria, si se decide usarlo.

1. La cláusula ORDER BY es la última en una sentencia SELECT:

**Ejemplo 1: NO ES NECESARIO USAR ASC, ES EL VALOR POR DEFECTO SI NO SE USA.**

Visualice a los alumnos, debidamente ordenados por su nombre.

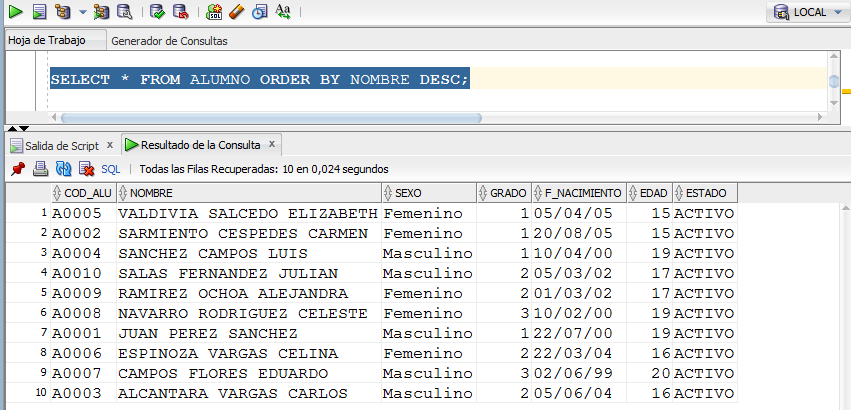
**SELECT \* FROM ALUMNO ORDER BY NOMBRE;**



**Ejemplo 2: USO DE DESC.**

Visualice a los alumnos, debidamente ordenados de manera descendente por su nombre.

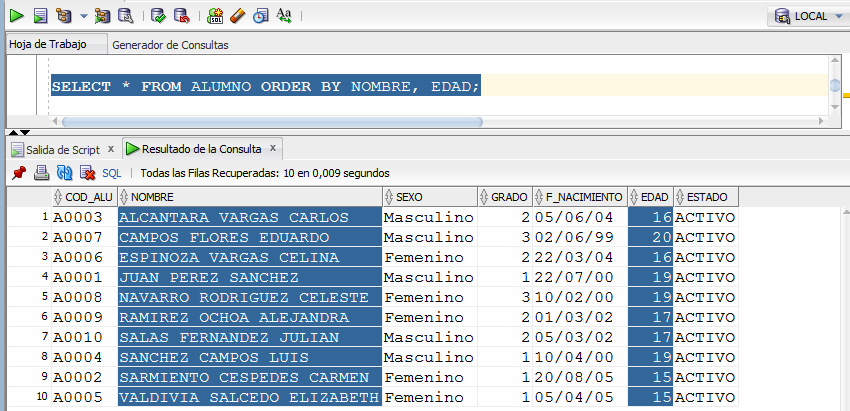
**SELECT \* FROM ALUMNO ORDER BY NOMBRE DESC;**



**Ejemplo 3: ORDENAR POR VARIAS COLUMNAS.**

Visualice a los alumnos, debidamente ordenados de manera ascendente por su nombre y edad.

**SELECT \* FROM ALUMNO ORDER BY NOMBRE, EDAD;**



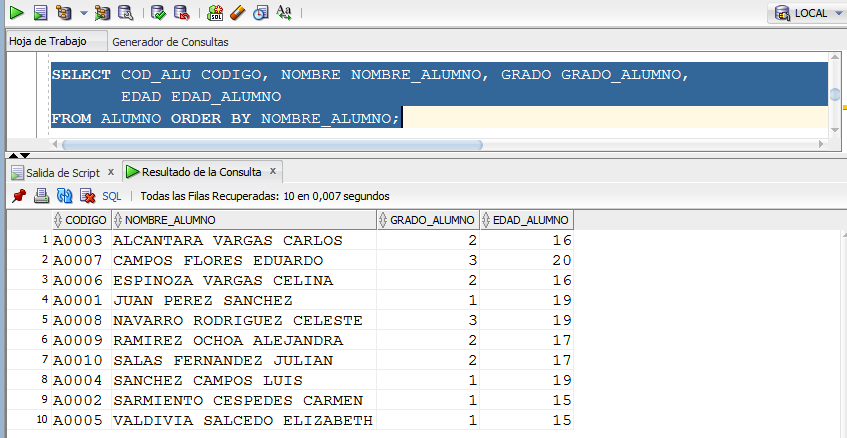
**Ejemplo 4: ORDENAR POR ALIAS DE COLUMNA.**

Visualice código, nombre, grado y edad del alumno, cada uno con un alias a su elección, ordenar dichos datos por el alias del nombre del alumno.

**SELECT COD\_ALU CODIGO, NOMBRE NOMBRE\_ALUMNO, GRADO GRADO\_ALUMNO,**

**EDAD EDAD\_ALUMNO**

**FROM ALUMNO ORDER BY NOMBRE\_ALUMNO;**

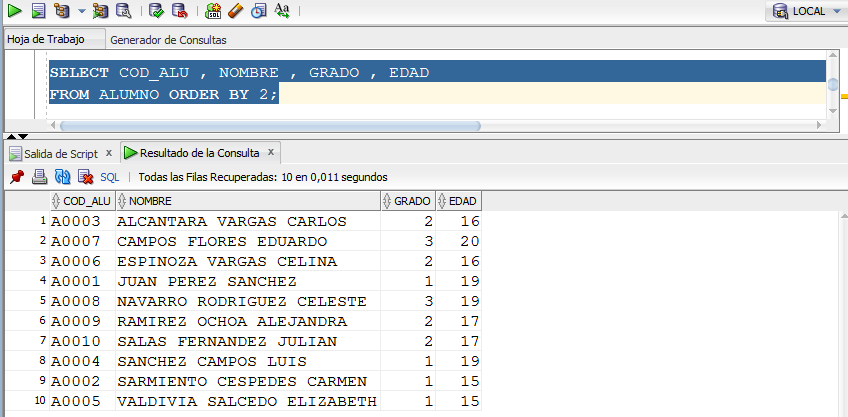


**Ejemplo 5: ORDENAR POR NUMERO DE COLUMNA.**

Visualice código, nombre, grado y edad del alumno, ordenar dichos registros por el número de columna 2 que corresponde al nombre del alumno.

**SELECT COD\_ALU, NOMBRE, GRADO, EDAD**

**FROM ALUMNO ORDER BY 2;**



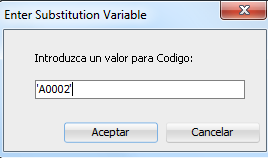
**APRENDER EL USO DE VARIABLES DE SUSTITUCIÓN EN NUESTRAS CONSULTAS &.**

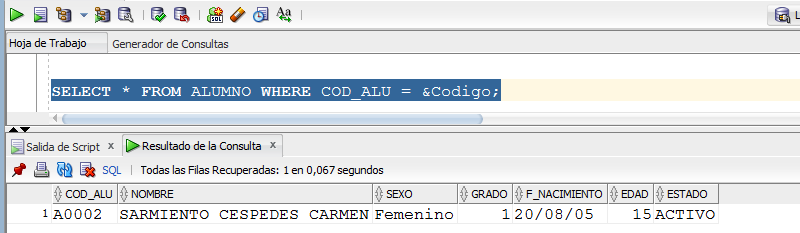
Utilizar una variable prefijada con un **ampersand** (**&**) para solicitar al usuario un valor:

**Ejemplo 1:**

Visualice a cualquier alumno usando como dato de búsqueda al código de alumno, este debe ser ingresado desde teclado a través de una ventana de acceso, en este ejemplo usaremos al código A0002.

**SELECT \* FROM ALUMNO WHERE COD\_ALU = &Codigo;**





**Ejemplo 2:**

Visualice a los alumnos de cualquier sexo, grado, y estado, ordenarlos por cualquier columna.

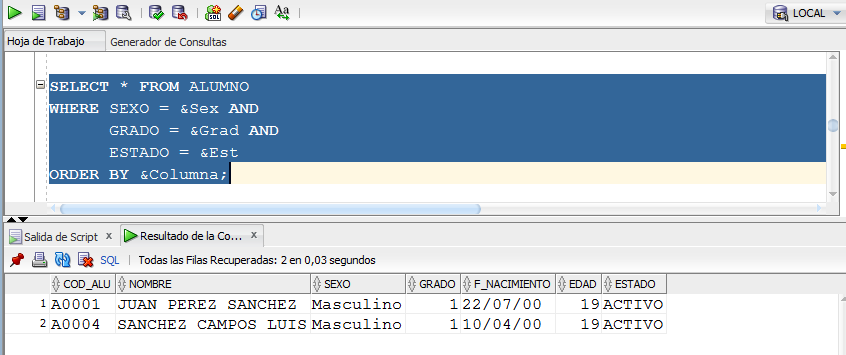
**SELECT \* FROM ALUMNO**

**WHERE SEXO = &Sex AND**

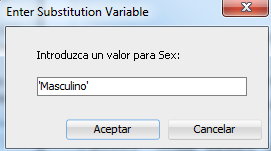
**GRADO = &Grad AND**

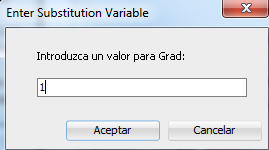
**ESTADO = &Est**

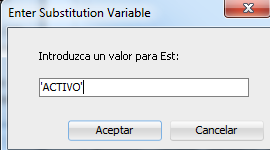
**ORDER BY &Columna;**

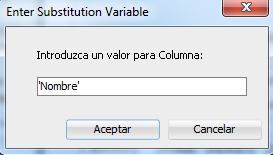


La ejecución de esta consulta implica el uso de ventanas interactivas que permitan el ingreso manual de: Sexo, Grado, Estado y la columna a ordenar.





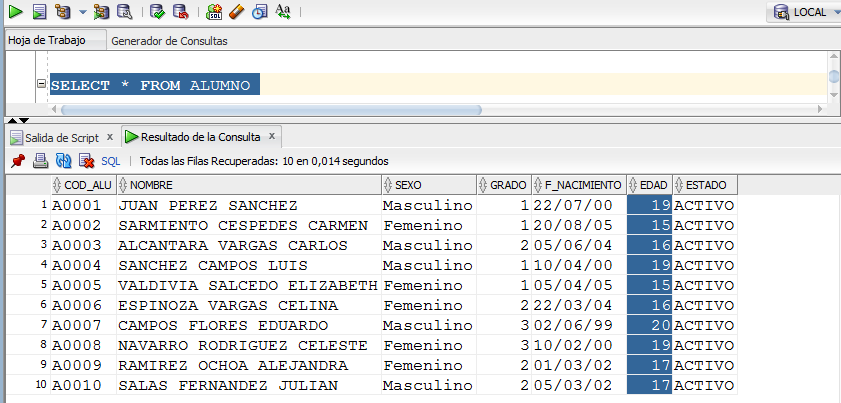




**APRENDER A USAR FUNCIONES AGREGADAS. SUM, MAX, MIN, AVG, COUNT**

Las funciones de grupo funcionan en juegos de filas para proporcionar un resultado por grupo.

Estos son los datos almacenados en la tabla alumno, se observa la columna Edad.

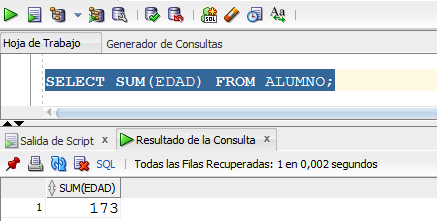


**Ejemplo 1: USANDO FUNCIÓN AGREGADA SUM = SUMATORIA DE DATOS.**

Realice la suma de edad de la tabla alumno.

Para saber cuál es la suma de las edades, se debe realizar la siguiente consulta:

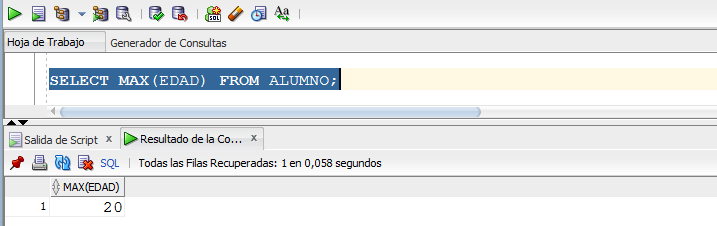
**SELECT SUM (EDAD) FROM ALUMNO;**



**Ejemplo 2: USANOD FUNCIÓN AGREGADA MAX = SUMATORIA DE DATOS.**

Para saber cuál es la edad máxima es, se debe realizar la siguiente consulta:

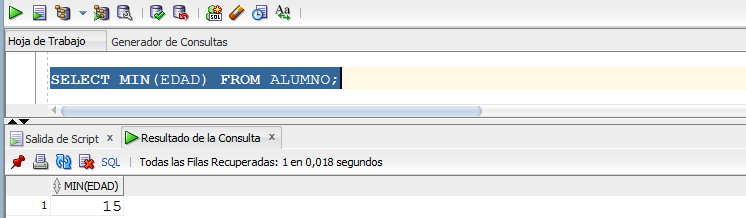
**SELECT MAX (EDAD) FROM ALUMNO;**



**Ejemplo 3: USANDO FUNCIÓN AGREGADA MIN = OBTENER EL MINIMO.**

Para saber cuál es la edad máxima es, se debe realizar la siguiente consulta:

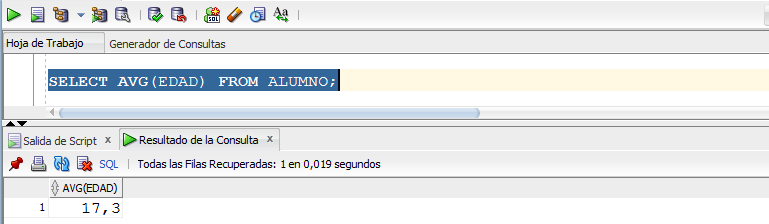
**SELECT MIN (EDAD) FROM ALUMNO;**



**Ejemplo 4: USANDO FUNCIÓN AGREGADA AVG = OBTENER PROMEDIO ARIMETICO.**

Para saber cuál es el promedio de edades, se debe realizar la siguiente consulta:

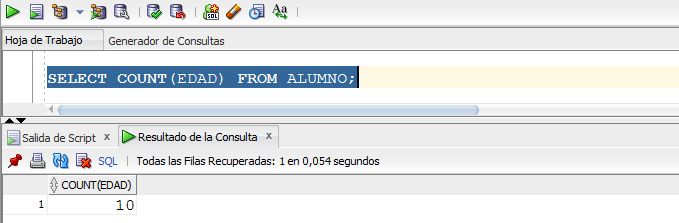
**SELECT AVG (EDAD) FROM ALUMNO;**



**Ejemplo 5: USAND FUNCIÓN AGREGADA COUNT = CONTEOS**

Para saber cuál es el conteo de edades, se debe realizar la siguiente consulta:

**SELECT COUNT (EDAD) FROM ALUMNO;**



**APRENDER EL USO DE LAS CLAUSULAS GROUP BY y HAVING.**

Ejemplo recreativo, para entender mejor, ¡QUE ES UN GRUPO?

**Tenemos un mundo de esferas de colores y estos tienen un valor referencial.**

**Para entender mejor lo que es un GRUPO, debemos separarlos, es decir clasificarlos y obtendremos algo así:**



**Tenemos 5 grupos de esferas, ¿QUE PODEMOS HACER CON ESOS GRUPOS?**

**Voy a tomar un Grupo: EN ESTE CASO EL AMARILLO.**



**¿Me pregunto qué puedo hacer con este grupo?**

1. Sumarlos: **10 + 20 + 20 + 5 + 50 + 15 + 50 = 170.**
2. Contarlos: **10, 20, 20, 5, 50, 15, 50 = Son 7 Elementos**.
3. Hallar el Máximo: **10, 20, 20, 5, 50, 15, 50 = El Mayor es 50.**
4. Hallar el Mínimo: **10, 20, 20, 5, 50, 15, 50 = El Menor es: 5.**
5. Promedio Aritmético: (**10 + 20 + 20 + 5 + 50 + 15 + 50) / 7 = 24.28571428571729**

**¿Me pregunto qué puedo hacer con todos los grupos?**



**Sumar los elementos de cada grupo.**

**AQUÍ ENTRA EL USO DE LA CLAUSULA GROUP BY.**

GRUPO AMARILLO : **170 = 10+20+20+5+50+15+50**

GRUPO ROJO : **179 = 60+10+5+20+16+7+40+21**

GRUPO VERDE : **218 = 30+5+12+10+6+42+24+12+55+22**

GRUPO AZUL : **247 = 11+10+5+90+51+20+10+30+20**

GRUPO MORADO : **15 = 15**

**Promediar los elementos de cada grupo.**

**AQUÍ ENTRA EL USO DE LA CLAUSULA GROUP BY.**

GRUPO AMARILLO : **24.2857 = (10+20+20+5+50+15+50) / 7**

GRUPO ROJO : **22.375 = (60+10+5+20+16+7+40+21) / 8**

GRUPO VERDE : **21.8 = (30+5+12+10+6+42+24+12+55+22) / 10**

GRUPO AZUL : **27.4 =** **(11+10+5+90+51+20+10+30+20) / 9**

GRUPO MORADO : **15 =** **(15) / 1**

**Contar los elementos por cada grupo.**

**AQUÍ ENTRA EL USO DE LA CLAUSULA GROUP BY.**

GRUPO AMARILLO : **7 Elementos = (10, 20, 20, 5, 50, 15, 50)**

GRUPO ROJO : **8 Elementos = (60, 10, 5, 20, 16, 7, 40, 21)**

GRUPO VERDE : **10 Elementos = (30, 5, 12, 10, 6, 42, 24, 12, 55, 22)**

GRUPO AZUL : **9 Elementos =** **(11+10+5+90+51+20+10+30+20)**

GRUPO MORADO : **1 Elemento =** **(15)**

**Saber Quién es el Mayor de todos.**

**AQUÍ ENTRA EL USO DE LA CLAUSULA GROUP BY.**

GRUPO AMARILLO : **El Mayor es 50**  **= (10, 20, 20, 5, 50, 15, 50)**

GRUPO ROJO : **El Mayor es 60 = (60, 10, 5, 20, 16, 7, 40, 21)**

GRUPO VERDE : **El Mayor es 55 = (30, 5, 12, 10, 6, 42, 24, 12, 55, 22)**

GRUPO AZUL : **El Mayor es 90 =** **(11, 10, 5, 90, 51, 20, 10, 30, 20)**

GRUPO MORADO : **El Mayor es 15 =** **(15)**

**Saber Quién es el Menor de todos.**

**AQUÍ ENTRA EL USO DE LA CLAUSULA GROUP BY.**

GRUPO AMARILLO : **El Menor es 5 = (10, 20, 20, 5, 50, 15, 50)**

GRUPO ROJO : **El Menor es 5 = (60, 10, 5, 20, 16, 7, 40, 21)**

GRUPO VERDE : **El Menor es 5 = (30, 5, 12, 10, 6, 42, 24, 12, 55, 22)**

GRUPO AZUL : **El Menor es 5 =** **(11, 10, 5, 90, 51, 20, 10, 30, 20)**

GRUPO MORADO : **El Menor es 15 =** **(15)**

**¿Podríamos ser más específicos en los grupos? CLARO QUE SI.**

**AQUÍ ENTRA EL USO DE LA CLAUSULA HAVING DEL GROUP BY.**

Sumar del grupo amarillo, solo aquellos valores que coincidan con 20.



**Es decir 20 + 20 = 40** (aquí entra el uso de la cláusula HAVING)

La cláusula HAVING condiciona al grupo

**EJEMPLOS DE GRUPOS Y CONDICIONES DE GRUPO.**

**GROUP BY – HAVING.**

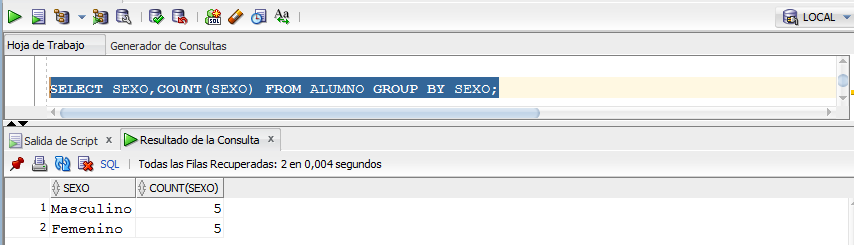
**Ejemplo 1: uso de solo GROUP BY**

Tenemos los siguientes datos, en la tabla ALUMNO, aquí observamos que existen tanto alumnos varones como damas, se pide una consulta que muestre el conteo y los diferencie.



La consulta seria así:

**SELECT SEXO, COUNT (SEXO) FROM ALUMNO GROUP BY SEXO;**

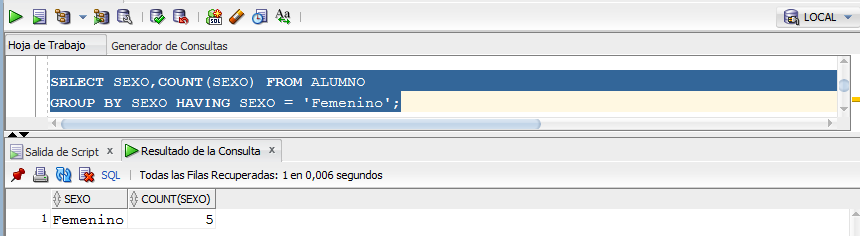


**Ejemplo 2: uso de GROUP BY – HAVING.**

De la consulta anterior (**Ejemplo 1),** solo muestre al sexo Femenino.

**SELECT SEXO, COUNT (SEXO) FROM ALUMNO**

**GROUP BY SEXO HAVING Sexo = ‘Femenino’;**

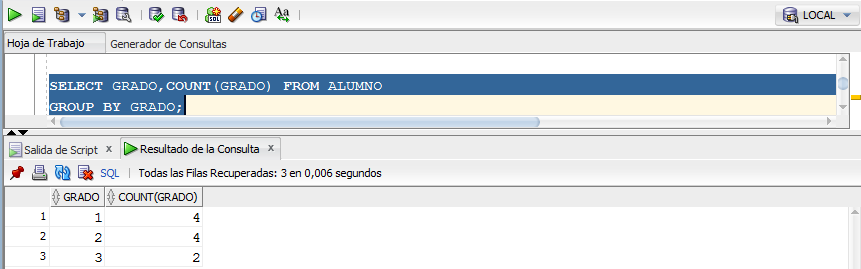


**Ejemplo 3:**

Muestre el conteo de alumnos por grados, diferenciándolos.

**SELECT GRADO, COUNT (GRADO) FROM ALUMNO**

**GROUP BY GRADO;**

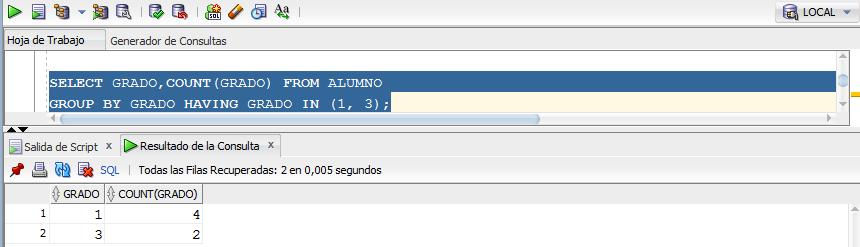


**Ejemplo 4:**

Del Ejemplo anterior (**Ejemplo 3**), muestre solo a los alumnos de los grados 1 y 3.

**SELECT GRADO, COUNT (GRADO) FROM ALUMNO**

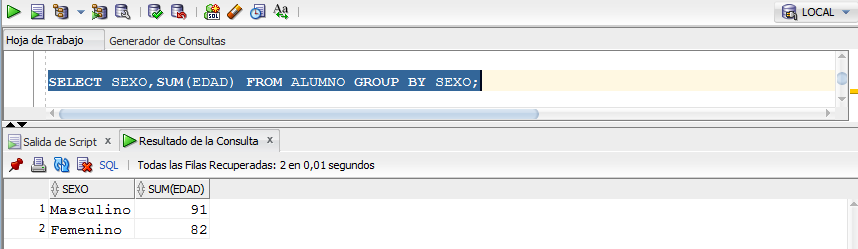
**GROUP BY GRADO HAVING GRADO IN (1, 3);**



**Ejemplo 5:**

Sume las edades de los varones y damas, muéstrelos de forma separada.

**SELECT SEXO, SUM (EDAD) FROM ALUMNO GROUP BY SEXO;**

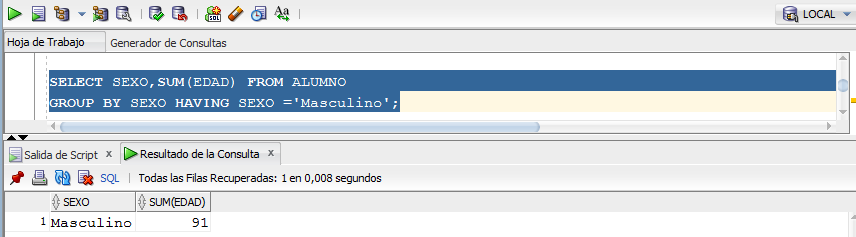


**Ejemplo 6:**

Del ejemplo anterior (**Ejemplo 5**), solo muestre la suma de edades de los varones.

**SELECT SEXO, SUM (EDAD) FROM ALUMNO**

**GROUP BY SEXO HAVING SEXO ='Masculino';**



**CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA**

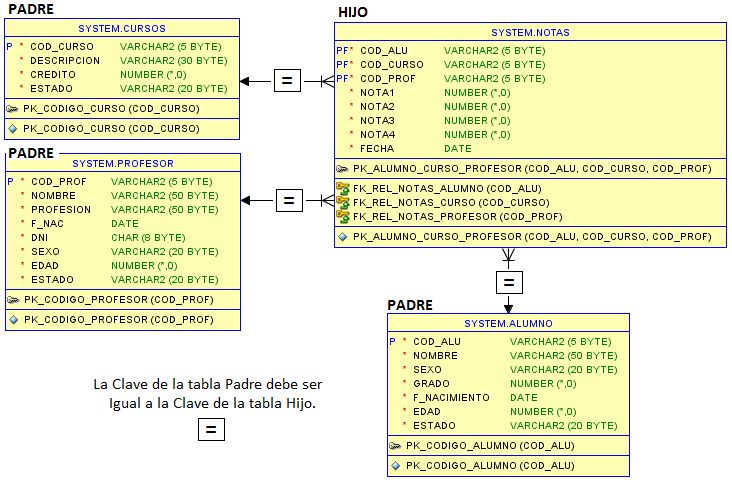
* El objetivo de este capítulo es comprender el uso de consultas, de esta manera el estudiante se compenetra más con la base de datos de Oracle, al analizar la información, resultado de las búsquedas realizadas de manera exitosa. Mediante el uso de las herramientas SQL \*PLUS y **PL/SQL**.

ACTIVIDAD VIRTUAL

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

Responda a las siguientes Preguntas:

**DE ACUERDO AL SIGUIENTE MODELO RELACIONAL.**



REALICE LAS SIGUIENTES CONSULTAS:

1. FECILITE UNA CONSULTA, USANDO LA CLAUSULA WHERE, LIKE, AND, OR, NOT IN, IN
2. FACILITE UNA CONSULTA, USANDO CAMPOS CALCULADOS
3. FACILITE UNA CONSULTA, CON FUNCIONES AGREGADAS.
4. FACILITE UNA CONSULTA, USANDO VARIABLES DE SUSTITUCION.
5. FACILITE UNA CONSULTA, USANDO GRUPOS.
6. FACILITE UNA CONSULTA, USANDO GRUPOS CONDICIONADOS.