Lenguaje de Manipulación de Datos - DML

1.Lenguaje de manipulación de datos (LMD)

Los comandos de manipulación de datos o lenguaje de modificación de datos (DML) son comandos SQL que recuperan, insertan, actualizan y eliminan las filas de una tabla en una Base de Datos de Oracle. Los cuatro comandos DML básicos son SELECT, INSERT, UPDATE y DELETE.

Para realizar consultas a la base de datos utilizaremos la orden SELECT de SQL. De la consulta se puede obtener: cualquier unidad de datos, todos los datos, cualquier subconjunto de datos, cualquier subconjunto de subconjuntos de datos.

1.1.Base de datos de Alumnos y Profesores

Para el desarrollo de este tema vamos a utilizar una Base de Datos sobre Alumnos, Profesores y las relaciones de Alumnos matriculados en Asignaturas. Estas tablas se detallan a continuación. Aunque en la web del curso es posible descargarse esta Base de Datos e Instalarla sobre Oracle 18c.

Tabla Alumnos

DNI	NOMBRE	APELLIDOS	FECHA_NA
22345678L	JORDI	SANCHEZ	10/10/80
44345675J	PACO	PEREZ	10/11/80
22345676N	RAFA	ROMERO	01/12/81
22345677M	JAVI	PONCE	12/01/82
22345678B	MANOLI	ALVAREZ	10/02/81
22356679F	MANUEL	PEREZ	13/03/80
22378670S	ANA	SANCHEZ	10/10/83
22312678W	ISABEL	PEREZ	15/04/86
23245671Q	ESTEFANIA	SANCHEZ	10/10/85

Tabla Profesores

DNI	NOMBRE	APELLIDOS
44345678K	ALVARO	PEREZ
43456778H	PEDR0	SANCHEZ
44312348J	MAITE	PEREZ
41345668L	PACO	ALMAGRO
42347498W	FERMINA	AGUILAR

Tabla Asignaturas

NOMBRE	PROFESOR
DAC	44345678K
DEG	43456778H
FOL	42347498W
ADA	44312348J
PLE	44345678K
RAL	

Tabla Matriculado

ALUMNO	ASIGNATURA	CURSO_ACADEMICO	NOTA
44345675J	DAC	2010	7
44345675J	DEG	2010	5
44345675J	ADA	2010	7
22345676N	DAC	2009	8
22345677M	FOL	2010	4
22345677M	DEG	2010	3
22345678B	FOL	2010	9
22345678B	DEG	2007	6
23245671Q	DAC	2010	9
23245671Q	DEG	2000	3
22312678W	FOL	2010	4

El comando SELECT

Sintaxis General de la orden SELECT:

```
SELECT [ALL/DISTINCT] <expre_column1, expre_column2…, expre_column | * >
FROM <lista_de_tablas>
[WHERE <condición>]
[GROUP BY <lista_de_atributos>
[HAVING <condición_de_grupo> ]]
[ORDER BY <lista_de_atributos> [ASC/DESC] ];
```

Donde **expre_column** puede ser una columna de una tabla, una constante, una expresión aritmética, una función o varias funciones anidadas.

Consideraremos la siguiente base de datos formada por cuatro tablas:

ALUMNOS (DNI, NOMBRE, APELLIDOS, FECHA_NAC)
PK: DNI

PROFESORES (DNI, NOMBRE, APELLIDOS)
PK: DNI

ASIGNATURAS (NOMBRE, PROFESOR)
PK: **NOMBRE**
FK: PROFESOR → PROFESORES

MATRICULADO (ALUMNO, ASIGNATURA, CURSO_ACADEMICO)
PK: (ALUMNO, ASIGNATURA)
FK: ALUMNO → ALUMNOS
FK: ASIGNATURA → ASIGNATURAS

NOTA: Este ejemplo de Base de datos se puede descargar de la web del curso.

Cláusula FROM

La única cláusula de la sentencia SELECT que es obligatoria es la cláusula FROM, el resto son opcionales.

FROM: From [nombre_tabla1, nombre_tabla2,…..,nombre_tabla_n]

Especifica la tabla o lista de tablas de las que se recuperarán los datos**.** ***Con esta orden obtenemos la información de las columnas requeridas de toda la tabla***.

Ejemplo: consultamos los nombres de los ALUMNOS.

```
SELECT NOMBRE FROM ALUMNOS;
```

NOMBRE
JORDI
PACO
RAFA
JAVI
MANOLI
MANUEL
ANA
ISABEL
ESTEFANIA

Es posible asociar un nuevo nombre a las tablas mediante **alias**.

Ejemplo: si la tabla profesores le damos el nombre P, las columnas de la tabla irán acompañadas de P

```
SELECT   P.DNI, P.NOMBRE FROM PROFESORES   P;
```

DNI	NOMBRE
44345678K	ALVARO
43456778H	PEDR0
44312348J	MAITE
41345668L	PACO
42347498W	FERMINA

Así por ejemplo podremos diferenciar columnas que se llamen igual, por ejemplo entre PROFESORES y ALUMNOS, donde el campo *nombre* coincide.

Ejemplo:

```
SELECT   P.NOMBRE, A.NOMBRE FROM PROFESORES   P, ALUMNOS A;
```

Cláusula WHERE

WHERE: [WHERE condición]
Obtiene las filas que cumplen la condición expresada .La complejidad de la condición es ilimitada. El formato de la condición es:

Expresión operador expresión

Los ***operadores de comparación*** y ***operadores lógicos*** que se utilizan son:

Operadores de comparación		**Operadores lógicos**	
OPERADOR	**FUNCIÓN**	**OPERADOR**	**FUNCIÓN**
< **>** **< >** **<=** **>=** **=**	Menor que Mayor que Distinto de Menor ó Igual que Mayor ó Igual que Igual a	**AND**	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
BETWEEN	Especifica un intervalo de valores.	**OR**	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
LIKE	Comparación de un modelo	**NOT**	Negación lógica. Devuelve el valor contrario de la expresión

Se pueden construir condiciones múltiples usando los ***operadores lógicos booleanos estándares***: **AND, OR, NOT** .Está permitido emplear paréntesis para forzar el orden de evaluación:

Ejemplos:

SELECT * FROM MATRICULADO WHERE CURSO_ACADEMICO > 2008;

ALUMNO	ASIGNATURA	CURSO_ACADEMICO
44345675J	DAC	2010
44345675J	DEG	2010
44345675J	ADA	2010
22345677M	FOL	2010
22345677M	DEG	2010
22345678B	FOL	2010
23245671Q	DAC	2010
22312678W	FOL	2010

SELECT * FROM MATRICULADO WHERE ASIGNATURA = 'DAC';

ALUMNO	ASIGNATURA	CURSO_ACADEMICO
44345675J	DAC	2010
22345676N	DAC	2010
23245671Q	DAC	2010

SELECT NOMBRE FROM PROFESORES WHERE NOMBRE LIKE 'A%';

NOMBRE
ALVARO

SELECT * FROM MATRICULADO WHERE CURSO_ACADEMICO > 2008 AND CURSO_ACADEMICO <=2010;

ALUMNO	ASIGNATURA	CURSO_ACADEMICO
44345675J	DAC	2010
44345675J	DEG	2010
44345675J	ADA	2010
22345676N	DAC	2009
22345677M	FOL	2010
22345677M	DEG	2010
22345678B	FOL	2010
23245671Q	DAC	2010
22312678W	FOL	2010

Cláusula ORDER BY

ORDER BY:[Order By expre_columna [Desc | Asc] [,expre columna [Desc | Asc]]…]

Podemos ordenar la salida producida por la orden Select por valores ascendentes (Asc) o descendentes (Desc) de una columna en particular. Si no se indica nada la ordenación será ascendente.

Ejemplos

```
SELECT   * FROM ALUMNOS ORDER BY APELLIDOS;
```

DNI	NOMBRE	APELLIDOS	FECHA_NA
22345678B	MANOLI	ALVAREZ	10/02/81
44345675J	PACO	PEREZ	10/11/80
22356679F	MANUEL	PEREZ	13/03/80
22312678W	ISABEL	PEREZ	15/04/86
22345677M	JAVI	PONCE	12/01/82
22345676N	RAFA	ROMERO	01/12/81
23245671Q	ESTEFANIA	SANCHEZ	10/10/85
22378670S	ANA	SANCHEZ	10/10/83
22345678L	JORDI	SANCHEZ	10/10/80

Cláusula ALL/DISTINCT

- **ALL**: Con la cláusula ALL, *recuperamos todas las filas* aunque estén repetidas .Es la opción por omisión
- **DISTINCT***: Elimina las filas repetidas*

Ejemplos

```
SELECT  ALL APELLIDOS FROM ALUMNOS ORDER BY APELLIDOS;
```

APELLIDOS
ALVAREZ
PEREZ
PEREZ
PEREZ
PONCE
ROMERO
SANCHEZ
SANCHEZ
SANCHEZ

```
SELECT  DISTINCT APELLIDOS FROM ALUMNOS ORDER BY APELLIDOS;
```

APELLIDOS
ALVAREZ
PEREZ
PONCE
ROMERO
SANCHEZ

Crear y utilizar Alias de columnas

Cuando se consulta la base de datos, los nombres de las columnas se usan como cabeceras de presentación. Si el nombre resulta demasiado largo, corto o complicado, existe la posibilidad de cambiarlo utilizando un Alias en la propia sentencia Select.

Ejemplo:

```
SELECT ALUMNO AS DNI_ALUMNO FROM MATRICULADO;
```

DNI_ALUMNO
22312678W
22345676N
22345677M
22345677M
22345678B
22345678B
23245671Q
23245671Q
44345675J
44345675J
44345675J

Operadores aritméticos

Los operadores Aritméticos sirven para formar expresiones con constantes, valores de columnas y funciones de valores de columna

Operador aritmético	Operación
+	Suma
-	Resta
*	Multiplicación
/	división

Ejemplos:

```
SELECT 5+3 AS CALCULO, CURSO_ACADEMICO, CURSO_ACADEMICO+10 FROM MATRICULADO;
```

CALCULO	CURSO_ACADEMICO	CURSO_ACADEMICO+10
8	2010	2020
8	2010	2020
8	2010	2020
8	2009	2019
8	2010	2020
8	2010	2020
8	2010	2020
8	2007	2017
8	2010	2020
8	2000	2010
8	2010	2020

Operadores de comparación de cadenas de caracteres

Para comparar cadenas enteras de caracteres se puede hacer con el operador de comparación =. Sin embargo se utiliza el operador LIKE cuando queremos comparar algunos caracteres de esa cadena.

El operador LIKE se utiliza con dos caracteres especiales:

% comodín: sustituye cualquier cadena de 0 o más caracteres
'_' Marcador de posición: sustituye un carácter cualquiera

También se puede utilizar NOT LIKE

Ejemplos:

A partir de la tabla profesores obtén los nombres que empiecen por una M

```
SELECT NOMBRE FROM PROFESORES WHERE NOMBRE LIKE 'M%';
```

NOMBRE
MAITE

Obtén aquellos nombres que tengan una L en segunda posición:

```
SELECT NOMBRE FROM PROFESORES WHERE NOMBRE LIKE '_L%';
```

NOMBRE
ALVARO

Obtén aquellos nombres que empiecen por P y tenga una A en su interior

```
SELECT NOMBRE FROM PROFESORES WHERE NOMBRE LIKE 'P%A%';
```

NOMBRE
PACO

Null y Not Null

Una columna de una fila es Null si está completamente vacía. Para comprobar si el valor de una columna es nulo empleamos la expresión: columna IS NULL. Para saber si el valor de esa columna no es nulo, utilizamos la expresión: columna IS NOT NULL. Cuando comparamos con valores nulos o no nulos **NO** podemos utilizar los operadores de igualdad, mayor o menor.

Ejemplos:

```
SELECT  * FROM ASIGNATURAS WHERE PROFESOR IS NULL;
```

NOMBRE	PROFESOR
RAL	

```
SELECT  * FROM ASIGNATURAS WHERE PROFESOR IS NOT NULL;
```

NOMBRE	PROFESOR
DAC	44345678K
DEG	43456778H
FOL	42347498W
ADA	44312348J
PLE	44345678K

Comprobaciones con conjuntos de valores

Podemos comparar una columna o una expresión con una lista de valores utilizando los operadores IN y BETWEEN.

Operador IN: Nos permite comprobar si una expresión pertenece o no a un conjunto de valores, pudiendo realizar comparaciones múltiples.

Formato:
<EXPRESIÓN> NOT IN (LISTA DE VALORES SEPARADOS POR COMAS)

(La lista de valores puede estar formada por números o por cadenas de caracteres)

Ejemplo: consulta en la tabla MATRICULADO donde el alumno esté matriculado en DAC o FOL.

```
SELECT * FROM MATRICULADO WHERE ASIGNATURA IN ('DAC','FOL')
ORDER BY ASIGNATURA;
```

ALUMNO	ASIGNATURA	CURSO_ACADEMICO
23245671Q	DAC	2010
22345676N	DAC	2009
44345675J	DAC	2010
22345678B	FOL	2010
22312678W	FOL	2010
22345677M	FOL	2010

Ejemplo: consulta en la tabla MATRICULADO donde el alumno NO esté matriculado ni en DAC ni en FOL.

```
SELECT * FROM MATRICULADO WHERE ASIGNATURA NOT IN
('DAC','FOL') ORDER BY ASIGNATURA;
```

ALUMNO	ASIGNATURA	CURSO_ACADEMICO
44345675J	ADA	2010
23245671Q	DEG	2000
22345678B	DEG	2007
44345675J	DEG	2010
22345677M	DEG	2010

Operador BETWEEN... AND: establece una comparación dentro de un intervalo. También se puede utilizar NOT BETWEEN.

Formato:

EXPRESSION [NOT] BETWEEN VALOR_INICIAL AND VALOR_FINAL

Ejemplo:
Muestra los alumnos matriculados entre 2007 y 2011

```
SELECT * FROM MATRICULADO WHERE CURSO_ACADEMICO BETWEEN 2007 AND 2011;
```

ALUMNO	ASIGNATURA	CURSO_ACADEMICO
44345675J	DAC	2010
44345675J	DEG	2010
44345675J	ADA	2010
22345676N	DAC	2009
22345677M	FOL	2010
22345677M	DEG	2010
22345678B	FOL	2010
22345678B	DEG	2007
23245671Q	DAC	2010
22312678W	FOL	2010

1.1.Relaciones entre tablas

Normalmente nos va a surgir la necesitar de asociar una tabla con otra. En este punto vamos a ver qué tipos de relaciones podemos hacer.

Producto cartesiano

Por ejemplo, saber el nombre de los alumnos que están en el curso 2010. Si vemos la tabla de MATRICULADO, allí no está el nombre del alumno sino que está su DNI, el nombre del alumno aparece en la tabla ALUMNOS, por eso tenemos que asociar la tabla MATRICULADO con ALUMNOS y relacionarla. Para ello habría que hacer lo siguiente:

Si realizamos la siguiente consulta:

```
SELECT  * FROM MATRICULADO, ALUMNOS;
```

Obtenemos:

ALUMNO	ASIGNATURA	CURSO_ACADEMICO	DNI	NOMBRE	APELLIDOS	FECHA_NA
44345675J	DAC	2010	22345678L	JORDI	SANCHEZ	10/10/80
44345675J	DEG	2010	22345678L	JORDI	SANCHEZ	10/10/80
44345675J	ADA	2010	22345678L	JORDI	SANCHEZ	10/10/80
22345676N	DAC	2009	22345678L	JORDI	SANCHEZ	10/10/80
22345677M	FOL	2010	22345678L	JORDI	SANCHEZ	10/10/80
22345677M	DEG	2010	22345678L	JORDI	SANCHEZ	10/10/80
22345678B	FOL	2010	22345678L	JORDI	SANCHEZ	10/10/80
22345678B	DEG	2007	22345678L	JORDI	SANCHEZ	10/10/80
23245671Q	DAC	2010	22345678L	JORDI	SANCHEZ	10/10/80
23245671Q	DEG	2000	22345678L	JORDI	SANCHEZ	10/10/80
22312678W	FOL	2010	22345678L	JORDI	SANCHEZ	10/10/80
44345675J	DAC	2010	44345675J	PACO	PEREZ	10/11/80
44345675J	DEG	2010	44345675J	PACO	PEREZ	10/11/80
44345675J	ADA	2010	44345675J	PACO	PEREZ	10/11/80
22345676N	DAC	2009	44345675J	PACO	PEREZ	10/11/80
22345677M	FOL	2010	44345675J	PACO	PEREZ	10/11/80
22345677M	DEG	2010	44345675J	PACO	PEREZ	10/11/80
22345678B	FOL	2010	44345675J	PACO	PEREZ	10/11/80
22345678B	DEG	2007	44345675J	PACO	PEREZ	10/11/80
23245671Q	DAC	2010	44345675J	PACO	PEREZ	10/11/80
23245671Q	DEG	2000	44345675J	PACO	PEREZ	10/11/80
22312678W	FOL	2010	44345675J	PACO	PEREZ	10/11/80
44345675J	DAC	2010	22345676N	RAFA	ROMERO	01/12/81
44345675J	DEG	2010	22345676N	RAFA	ROMERO	01/12/81

Página Siguiente

Habiendo varias páginas en la consulta.

Como se aprecia ha hecho una operación X (producto cartesiano) del algebra relacional. Es decir, ha hecho todas las combinaciones de cada tupla de MATRICULADO con cada tupla de ALUMNOS.

Para realizar la consulta de forma correcta tendríamos que hacer lo siguiente:

```
SELECT  ALUMNO, NOMBRE, APELLIDOS, CURSO_ACADEMICO
 FROM MATRICULADO, ALUMNOS
 WHERE ALUMNO=DNI
AND CURSO_ACADEMICO=2010;
```

ALUMNO	NOMBRE	APELLIDOS	CURSO_ACADEMICO
44345675J	PACO	PEREZ	2010
44345675J	PACO	PEREZ	2010
44345675J	PACO	PEREZ	2010
22345677M	JAVI	PONCE	2010
22345677M	JAVI	PONCE	2010
22345678B	MANOLI	ALVAREZ	2010
22312678W	ISABEL	PEREZ	2010
23245671Q	ESTEFANIA	SANCHEZ	2010

Como se puede apreciar aparecen repetidos, esto es porque PACO, por ejemplo, está matriculado en las asignaturas de DAC, DEG, ADA.

Para solucionarlo y que solo nos apareciesen una tupla por cada alumno tendríamos que hacer lo siguiente:

```
SELECT DISTINCT ALUMNO, NOMBRE, APELLIDOS,
CURSO_ACADEMICO
 FROM MATRICULADO, ALUMNOS
 WHERE ALUMNO=DNI AND CURSO_ACADEMICO=2010;
```

ALUMNO	NOMBRE	APELLIDOS	CURSO_ACADEMICO
22345677M	JAVI	PONCE	2010
23245671Q	ESTEFANIA	SANCHEZ	2010
44345675J	PACO	PEREZ	2010
22345678B	MANOLI	ALVAREZ	2010
22312678W	ISABEL	PEREZ	2010

Veamos un ejemplo más:

Imaginemos las siguientes tablas Tabla1 y Tabla2:

Tabla 1
b1
b2
b3

Tabla 2
a1
a2

Si realizamos la consulta: **SELECT * FROM TABLA1, TABLA2**
Obtendríamos lo siguiente:

b1	a1
b2	a1
b3	a1
b1	a2
b2	a2
b3	a2

Reunión de relaciones

Inner join	Con esta operación se calcula el producto cruzado de todos los registros; así cada registro en la tabla A es combinado con cada registro de la tabla B. Es necesario tener especial cuidado cuando se combinan columnas con valores nulos NULL ya que el valor nulo no se combina con otro valor o con otro nulo.
Natural inner join	En este caso se comparan todas las columnas que tengan el mismo nombre en ambas tablas.
Left outer join	El resultado de esta operación siempre contiene todos los registros de la tabla de la izquierda (la primera tabla que se menciona en la consulta), aun cuando no exista un registro correspondiente en la tabla de la derecha, para uno de la izquierda.
Right outer join	Esta operación inversa a la anterior; el resultado de esta operación siempre contiene todos los registros de la tabla de la derecha (la segunda tabla que se menciona en la consulta), aun cuando no exista un registro correspondiente en la tabla de la izquierda, para uno de la derecha.

Ejemplos:

```
SELECT *
FROM   empleado
       INNER JOIN departamento
         ON empleado.IDdepartamento = departamento.IDdepartamento
```

Daría el mismo resultado que:

```
SELECT *
FROM   empleado, departamento
WHERE  empleado.IDdepartamento = departamento.IDDepartamento
```

Empleado.Apellido	Empleado.IDdepartamento	departamento.NombreDepartamento	departamento.IDDepartamento
Smith	34	Producción	34
Jordán	33	Ingeniería	33
Róbinson	34	Producción	34
Steinberg	33	Ingeniería	33
Rafferty	31	Ventas	31

```
SELECT *
FROM   empleado NATURAL JOIN departamento
```

Empleado.Apellido	IDDepartamento	Departamento.NombreDepartamento
Smith	34	Producción
Jordán	33	Ingeniería
Róbinson	34	Producción
Steinberg	33	Ingeniería
Rafferty	31	Ventas

```
SELECT distinct *
FROM   empleado
       LEFT OUTER JOIN departamento
         ON empleado.IDDepartamento = departamento.IDDepartamento
```

Empleado.Apellido	Empleado.IDDepartamento	Departamento.NombreDepartamento	Departamento.IDDepartamento
Jordán	33	Ingeniería	33
Rafferty	31	Ventas	31
Róbinson	34	Producción	34
Smith	34	Producción	34
Gaspar	36	NULL	NULL
Steinberg	33	Ingeniería	33

NOTA: Mirar la tupla Gaspar que se ha rellenado con valores nulos

```
SELECT *
FROM   empleado
      RIGHT OUTER JOIN departamento
        ON empleado.IDDepartamento = departamento.IDDepartamento
```

Empleado.Apellido	Empleado.IDDepartamento	Departamento.NombreDepartamento	Departamento.IDDepartamento
Smith	34	Producción	34
Jordán	33	Ingeniería	33
Róbinson	34	Producción	34
Steinberg	33	Ingeniería	33
Rafferty	31	Ventas	31
NULL	NULL	Marketing	35

1.2.Subconsultas

A veces, para realizar alguna operación de consulta, necesitamos los datos devueltos por otra consulta. Por ejemplo saber el nombre del Alumno que está matriculado en DAC, como vemos están en diferentes tablas. Una posible solución es usar el producto cartesiano de dos tablas. O también decir que se nos muestren los alumnos que pertenezcan al subconjunto de alumnos que tienen DAC.

Este problema se puede resolver usando una subconsulta, que no es mas que una sentencia SELECT dentro de otra sentencia SELECT Las subconsultas son aquellas sentencias SELECT que forman parte de una cláusula WHERE de una sentencia SELECT anterior. Una subconsulta consistirá en incluir una declaración SELECT como parte de una cláusula WHERE.
Formato de la subconsulta:

```
SELECT....
FROM.....
WHERE columna  operador _ comparativo ( SELECT .....
                                        FROM.....
                                        WHERE....)
```

Primero se resolverá el SELECT del paréntesis y después el SELECT de la sentencia principal.

Ejemplos
Para el ejemplo anterior la subconsulta quedaría:

```
SELECT * FROM ALUMNOS WHERE DNI IN (SELECT ALUMNO FROM
MATRICULADO WHERE ASIGNATURA='DAC');
```

DNI	NOMBRE	APELLIDOS	FECHA_NA
44345675J	PACO	PEREZ	10/11/80
22345676N	RAFA	ROMERO	01/12/81
23245671Q	ESTEFANIA	SANCHEZ	10/10/85

Como se puede apreciar es una consulta que dentro tiene otra subconsulta.

Condiciones de búsqueda en subconsultas

- **Comparación de subconsulta** (>, < ,<>, <=, >= , =) . Compara el valor de una expresión con **un valor único** producido por una subconsulta.

  Ejemplo: Nombre y apellidos del profesor que imparte la asignatura 'FOL'

```
SELECT NOMBRE,APELLIDOS FROM PROFESORES WHERE DNI =
   (
      SELECT PROFESOR FROM ASIGNATURAS
      WHERE NOMBRE ='FOL'
   );
```

NOMBRE	APELLIDOS
FERMINA	AGUILAR

- **Pertenencia a un conjunto devuelto por una subconsulta (IN):** Comprueba si el valor de una expresión coincide con uno del conjunto de valores producido por una subconsulta.

  Ejemplo: obtener aquellos alumnos que tengan clase con ALVARO.

```
SELECT NOMBRE, APELLIDOS FROM ALUMNOS WHERE DNI IN
(
        SELECT ALUMNO FROM MATRICULADO WHERE ASIGNATURA IN
        (
                SELECT NOMBRE FROM ASIGNATURAS WHERE PROFESOR IN
                (
                        SELECT DNI FROM PROFESORES WHERE
                NOMBRE='ALVARO'
                )
        )
);
```

NOMBRE	APELLIDOS
PACO	PEREZ
RAFA	ROMERO
ESTEFANIA	SANCHEZ

- **Test de existencia (Exists ,not exists).**Examina si una subconsulta produce alguna fila de resultados. La consulta es True si devuelve filas, si no es False.

Ejemplo: Listar las asignaturas que no tengan profesores

```
SELECT * FROM ASIGNATURAS A WHERE NOT EXISTS (
            SELECT DNI FROM PROFESORES P
            WHERE A.PROFESOR = P.DNI
);
```

NOMBRE	PROFESOR
RAL	

Ejemplo: Listar los profesores que no den clase

```
SELECT * FROM PROFESORES P WHERE NOT EXISTS (SELECT * FROM ASIGNATURAS A WHERE A.PROFESOR = P.DNI );
```

DNI	NOMBRE	APELLIDOS
41345668L	PACO	ALMAGRO

- **Test de comparación cuantificada (any ,all).** Se usan en unión con los operadores de comparación (>, < ,<>, <=, >= , =).

  → **Any:** Compara el valor de una expresión con cada uno del conjunto de valores producido por una subconsulta. Si alguna de las comparaciones individuales da como resultado TRUE, ANY devuelve TRUE, si la subconsulta no devuelve nada, devolverá FALSE.

  Ej. Obtener las notas de los alumnos y asignaturas, que tengan mayor nota que alguna de las notas de DAC

```
SELECT * FROM MATRICULADO WHERE NOTA > ANY (SELECT NOTA FROM MATRICULADO WHERE ASIGNATURA='DAC')
```

ALUMNO	ASIGNATURA	CURSO_ACADEMICO	NOTA
23245671Q	DAC	2010	9
22345678B	FOL	2010	9
22345676N	DAC	2009	8

→ **All:** Compara el valor de una expresión con cada uno del conjunto de valores producido por una subconsulta, si todas las comparaciones individuales da como

resultado TRUE, all devuelve true, en caso contrario devuelve false

Ej. Obtén un listado de la nota más alta de cada una de las asignaturas.

```
SELECT * FROM MATRICULADO M1 WHERE NOTA >= ALL
(SELECT NOTA FROM MATRICULADO M2 WHERE
M1.ASIGNATURA=M2.ASIGNATURA);
```

ALUMNO	ASIGNATURA	CURSO_ACADEMICO	NOTA
44345675J	ADA	2010	7
22345678B	FOL	2010	9
22345678B	DEG	2007	6
23245671Q	DAC	2010	9

Ejemplo: Mostrar de cada asignatura el alumno que tiene la máxima nota.

```
SELECT NOMBRE, APELLIDOS, ASIGNATURA, NOTA FROM
MATRICULADO M, ALUMNOS A WHERE M.ALUMNO=A.DNI AND
NOTA >= ALL
(SELECT NOTA FROM MATRICULADO M2 WHERE
M.ASIGNATURA=M2.ASIGNATURA
);
```

NOMBRE	APELLIDOS	ASIGNATURA	NOTA
PACO	PEREZ	ADA	7
MANOLI	ALVAREZ	FOL	9
MANOLI	ALVAREZ	DEG	6
ESTEFANIA	SANCHEZ	DAC	9

1.1.Funciones

Las funciones se usan dentro de expresiones y actúan con los valores de las columnas variables o constantes. Se utilizan en: cláusulas select, cláusulas where y cláusulas order by.

Es posible el anidamiento de funciones. Existen cinco tipos de funciones: aritméticas, de cadena de caracteres, de manejo de fechas de conversión y otras funciones.

Funciones aritméticas

Trabajan con datos de tipo numérico NUMBER. Incluye los dígitos de 0 al 9, el punto decimal el signo menos, si es necesario. Los literales numéricos no se encierran entre comillas. Ej :-123.32

Estas funciones trabajan con tres clases de números: valores simples, grupos de valores y lista de valores. Algunas modifican los valores sobre las que actúan; otras informan de algo sobre los valores. Podemos dividir las funciones aritméticas en tres grupos:

***Funciones de valores simples**:* son funciones sencillas que trabajan con valores simples. Un valor simple es: un número, una variable o una columna de una tabla. Las funciones de valores simples son:

Función	Propósito
Abs(n)	Devuelve el valor absoluto de “n”.El valor absoluto es siempre positivo
Mod(m,n)	Devuelve el resto resultante de dividir “m” entre “n”
Power(n,exponente)	Calcula la potencia de un número. devuelve el valor de n elevado a un exponente
Round(número[,m])	Devuelve el valor de número redondeado a “m” decimales. Si “m” es negativo, el redondeo de dígitos se lleva a cabo a la izquierda del punto decimal. Si se omite”m” ,devuelve número con 0 decimales y redondeado
Sqrt(n)	Devuelve la raíz cuadrada de “n”.El valor de “n” no puede ser negativo.
Trunc(número,[n])	Trunca los números para que tengan un cierto número de dígitos de precisión .devuelve “número” truncado a “m” decimales;”m” puede ser negativo; si lo es, trunca por la izquierda del punto decimal. Si se omite “m” devuelve “número” con 0 decimales
NVL(valor, expresión)	Sustituye un valor NULO por otro valor

Funciones de grupos de valores: Actúan sobre un grupo de filas para obtener un valor. Los valores nulos son ignorados por este tipo de funciones y los cálculos se realizan sin contar con ellos. Las cláusulas DISTINCT y ALL se pueden utilizar con todas ellas, aunque generalmente se usa con COUNT.

Estas funciones son:

Función	Propósito
AVG(n)	Calcula el valor medio de "n" ignorando los valores nulos
COUNT (* | [DISTINCT | ALL] expresión)	Cuenta el número de veces que la expresión evalúa algún dato con valor no nulo. La opción "*" cuenta todas las filas seleccionadas
MAX (expresión)	Calcula el máximo valor de la "expresión "
MIN (expresión)	Calcula el mínimo valor de la "expresión"
SUM (expresión)	Obtiene la suma de valores de la "expresión" distintos de nulos

Funciones de listas: Trabajan sobre un grupo de columnas dentro de una misma fila. Comparan los valores de cada una de las columnas en el interior de una fila para obtener el mayor o el menor valor de la lista.

Estas funciones son:

Función	Propósito
Greatest (valor1,valor2,...)	Obtiene el mayor valor de la lista.
Least(valor1,valor2,....)	Obtiene el menor valor de la lista.

Funciones de cadenas de caracteres

Trabajan con datos de tipo char o varchar2.Permiten manipular cadenas de letras u otros caracteres. Estas funciones pueden calcular el número de caracteres de una cadena, convertir una cadena a mayúsculas o a minúsculas o añadir caracteres a la izquierda o a la derecha,..etc

- ***Funciones que devuelven valores carácter***

Función	Propósito
Chr(n)	Devuelve el carácter cuyo valor en binario es

	equivalente a “n”
Concat(cad1,cad2..)	Devuelve “cad1” concatenada con “cad2” es equivalente al operador ||
Lower(cad)	Devuelve la cadena “cad” con todas sus letras convertidas a minúsculas
Upper(cad)	Devuelve la cadena "cad" en mayúsculas.
Lpad(texto, anchuraMáxima, [carácter_relleno])	Rellena el texto a la izquierda con el carácter indicado hasta obtener la anchura indicada
Rpad(texto, anchuraMáxima, [carácter_relleno])	Rellena el texto a la derecha con el carácter indicado hasta obtener la anchura indicada
Ltrim(cad,[set])	Suprime un conjunto de caracteres a la izquierda de la cadena
Rtrim(cad,[set])	Suprime un conjunto de caracteres a la derecha de la cadena
REPLACE(cad, cadena_busqueda [,cadena_sustitucion])	Sustituye un carácter o caracteres de una cadena con 0 o más caracteres.
SUBSTR (cad, m [,n])	Obtiene parte de una cadena. Desde la posición m hasta n (si n no se indica, hasta el final)

- ***Funciones que devuelven valores numéricos***

Función	**Propósito**
ASCII(cad)	Devuelve el valor ASCII de la primera letra de la cadena “cad”
Instr(cad1,cad2[,comienzo [,m]])	Busca un conjunto de caracteres en una cadena. Devuelve la posición de la “m_ésima “ ocurrencia de cad2 en cad1,empezando la búsqueda en la posición comienzo. Por omisión empieza buscando en la posición 1
Length(cad)	Devuelve el número de caracteres de “cad”

C) Funciones para el manejo de fechas

Oracle puede almacenar datos de tipo fecha (date), tiene un formato por omisión: ”DD/ MM/YY”, pero con la función TO_CHAR es posible mostrar las fechas de cualquier modo. Los literales de fecha deben encerrarse entre comillas simples. EJEMPLO: ‘18/11/05’

Función	**Propósito**
SYSDATE	Devuelve la fecha del sistema

ADD_MONTHS(fecha,n)	Devuelve la fecha “fecha” incrementada en ‘n’ meses.
LAST_DAY(fecha)	Devuelve la fecha del último día del mes que contiene “fecha”
MONTHS_BEETWEN(fecha1,fecha2)	Devuelve la diferencia en meses entre las fechas “fecha1” y “fecha2”
NEXT_DAY(fecha,cad)	Devuelve la fecha del primer día de la semana indicado por cad, después de la fecha indicada por fecha .El día de la semana en ‘cad’ se indica con su nombre, es decir, lunes (Monday)…

FUNCIONES DE CONVERSION

Función	Propósito
TO_CHAR(campo,formato)	Transforma un tipo DATE o NUMBER en una cadena de caracteres. Obtiene un texto a partir de un número o una fecha.
TO_DATE(campo,formato_FECHA)	Transforma un tipo NUMBER o CHAR en DATE. Convierte textos en fechas.
TO_NUMBER(campo,formato)	Transforma una cadena de caracteres en NUMBER.Convierte textos en números.

FUNCION ESPECIAL

Función	Propósito
DECODE(expresión,valor1,resultado1, [,valor2,resultado2,…,[valorPorDefecto]	Función que permite evaluar condiciones en una consulta
Ejemplo: DECODE(nota,10,’Sobresaliente’,9,’Sobresaliente’,8,’Notable’…5,’Aprobado’, ’Suspendido’)	

Cláusulas avanzadas de selección

Vamos a estudiar las órdenes que nos permiten agrupar filas de una tabla según alguna condición, o sin ninguna condición, para obtener algún resultado referente a ese grupo de filas agrupadas: GROUP by

También utilizaremos las órdenes que nos permiten seleccionar algunas filas de una tabla, aunque estas no tengan su correspondencia con otra tabla.: **HAVING**

Group by

La sentencia select posibilita utilizar uno o más conjuntos de filas. El agrupamiento se lleva a cabo mediante la cláusula **group by** por las columnas especificadas y en el orden especificado. La sentencia select tendrá el formato:

```
SELECT…
FROM…..
GROUP BY columna1, columna2, columna3,….
HAVING condición
ORDER BY.....
```

Los datos seleccionados en la sentencia select que lleva el group by deben ser: una constante, una función de grupo (SUM, COUNT, AVG,..), una columna expresada en el Group BY.

La cláusula group by sirve para calcular propiedades de uno o mas conjuntos de filas. Además si se selecciona más de un conjunto de filas, GROUP by controla que las filas de la tabla original sean agrupadas en una temporal.

Ejemplos

Ejemplo: Mostrar la nota media de cada asignatura.

```
SELECT ASIGNATURA, AVG(NOTA) AS MEDIA FROM MATRICULADO
GROUP BY ASIGNATURA ORDER BY ASIGNATURA;
```

ASIGNATURA	MEDIA
ADA	7
DAC	8
DEG	4,25
FOL	5,66666667

Ejemplo: Mostrar la nota media de los alumnos que han aprobado.

```
SELECT ASIGNATURA, AVG(NOTA) AS MEDIA FROM MATRICULADO
WHERE NOTA >=5 GROUP BY ASIGNATURA ORDER BY ASIGNATURA;
```

ASIGNATURA	MEDIA
ADA	7
DAC	8
DEG	5,5
FOL	9

Ejemplo: Mostrar la asignatura y cuántos alumnos hay matriculados para el curso 2010.

```
SELECT ASIGNATURA, COUNT(*) AS NUMERO_ALUMNOS FROM
MATRICULADO WHERE CURSO_ACADEMICO=2010 GROUP BY
ASIGNATURA ORDER BY NUMERO_ALUMNOS DESC;
```

ASIGNATURA	NUMERO_ALUMNOS
FOL	3
DEG	2
DAC	2
ADA	1

Having

HAVING es similar a WHERE, determina qué registros se seleccionan una vez agrupados. Una vez que los registros se han agrupado utilizando GROUP BY, HAVING determina cuáles de ellos se van a mostrar.

Se evalúa sobre la tabla que devuelve el group by. No puede existir sin group by.

Ejemplos

Ejemplo: Mostrar la nota media de cada asignatura que sea mayor o igual que 5.

```
SELECT ASIGNATURA, AVG(NOTA) AS MEDIA FROM MATRICULADO GROUP BY ASIGNATURA HAVING AVG(NOTA)>=5 ORDER BY ASIGNATURA;
```

ASIGNATURA	MEDIA
ADA	7
DAC	8
FOL	5,66666667

Ejemplo: Muestra la asignatura con la nota más baja.

```
SELECT ASIGNATURA, MIN(NOTA) FROM MATRICULADO GROUP BY ASIGNATURA HAVING MIN(NOTA) <= ALL(SELECT MIN(NOTA) FROM MATRICULADO GROUP BY ASIGNATURA);
```

ASIGNATURA	MIN(NOTA)
DEG	3

1.1.Operaciones sobre conjuntos

Union

Une varios conjuntos, pero ambas tuplas a unir deben tener el mismo número y tipo. Union elimina duplicados, para conservar los duplicados union all

Ejemplo

Muestra un listado con alumnos y profesores, mostrando su DNI, Nombre y Apellidos

```
SELECT DNI,NOMBRE,APELLIDOS FROM ALUMNOS
UNION
SELECT DNI,NOMBRE,APELLIDOS FROM PROFESORES;
```

DNI	NOMBRE	APELLIDOS
22312678W	ISABEL	PEREZ
22345676N	RAFA	ROMERO
22345677M	JAVI	PONCE
22345678B	MANOLI	ALVAREZ
22345678L	JORDI	SANCHEZ
22356679F	MANUEL	PEREZ
22378670S	ANA	SANCHEZ
23245671Q	ESTEFANIA	SANCHEZ
41345668L	PACO	ALMAGRO
42347498W	FERMINA	AGUILAR
43456778H	PEDR0	SANCHEZ
44312348J	MAITE	PEREZ
44345675J	PACO	PEREZ
44345678K	ALVARO	PEREZ

Intersect

Da el resultado de la intersección de varios conjuntos, pero ambas tuplas a unir deben tener el mismo número y tipo. Intersect elimina duplicados, para conservar los duplicados intersect all

Ejemplo

Muestra un listado de nombres donde se muestre que profesores se llaman igual que los alumnos

```
SELECT NOMBRE FROM ALUMNOS
INTERSECT
SELECT NOMBRE FROM PROFESORES;
```

NOMBRE
PACO

Minus

Se relaciona con la operación '-' sobre conjuntos.

Ejemplo

Muestra qué alumnos están matriculados en DEG y, de ellos, quita los que están matriculados también en DAC.

```
SELECT DNI, NOMBRE, APELLIDOS FROM ALUMNOS WHERE DNI IN(
(SELECT ALUMNO FROM MATRICULADO WHERE ASIGNATURA='DEG')
MINUS
(SELECT ALUMNO FROM MATRICULADO WHERE ASIGNATURA='DAC')
);
```

DNI	NOMBRE	APELLIDOS
22345677M	JAVI	PONCE
22345678B	MANOLI	ALVAREZ