



**DBD**


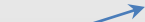


Diseño de  
Bases de Datos

# TRABAJANDO CON ORACLE – SENTENCIA SELECT



# Tipos de datos

Los que veremos en la asignatura

- Char(n) 
- Varchar2(n) 
- Date 
- Number(p,s) 

Integer = number(38)

# Funciones para datos tipo DATE en ORACLE

- **TO\_CHAR(fecha [, formato])**

Convierte la fecha de tipo DATE a un valor VARCHAR2 en el formato especificado en "formato"

- **TO\_DATE(cadena [, formato])**

Convierte la cadena de caracteres "cadena" de tipo CHAR a un valor de tipo DATE con el formato especificado en "formato"

- **SYSDATE**

Devuelve la fecha actual del sistema

- **ADD\_MONTHS(fecha,n)**

Devuelve la fecha especificada con n meses más

- **MONTHS\_BETWEEN(fecha1,fecha2)**

Devuelve los meses transcurridos entre fecha1 y fecha2

# Para las funciones TO\_CHAR y TO\_DATE

## ELEMENTO

## SIGNIFICADO

- - / ' . ; : 'texto' Marcas de puntuación y texto fijo que se reproduce en el resultado
- D Día de la semana (1-7)
- DAY Nombre del día de la semana
- DD Día del mes (1-31)
- DDD Día del año (1-366)
- DY Nombre reducido del día de la semana (LUN, MAR, MIE...)
- MM Mes (1-12)
- MON Nombre abreviado del mes
- MONTH Nombre completo del mes
- Q Trimestre del año (1-4)
- YYYY Año con 4 dígitos
- Y,YYY Año con punto de millar
- YY Año con 2 dígitos

# Ejemplos

- Profesores que ingresaron en el primer semestre de cualquier año

```
select * from profesores  
where to_char(ingreso,'MM') <= '06'
```

- Profesores que han ingresado hoy.

```
select * from profesores  
where to_char(ingreso,'dd-mm-yyyy') = to_char(sysdate,'dd-mm-yyyy')
```

- Profesores que han ingresado en el primer trimestre de 2010.

```
select * from profesores  
where to_char(ingreso,'MM') <= '03' and  
      to_char(ingreso, 'YYYY') = '2010'
```

Contenidos ya vistos en FBD

# Sentencia SELECT

**SELECT [ DISTINCT ] listaColumnas**  
**FROM** listaTablas  
[ **WHERE** condición para filas]  
[ **GROUP BY** listaColumnas por las que se quiere agrupar  
[ **HAVING** condición para los grupos ] ]  
[ **ORDER BY** listaColumnas [ **ASC** | **DESC** ] ]

[ ] significa que es opcional

# Repaso JOIN

Al hablar de **JOIN** nos referimos a combinar en una consulta filas de dos o más tablas concatenándolas atendiendo a algún criterio. Dependiendo de la condición que se utilice existen **distintos tipos de join**.



# Repaso JOIN

- Si las tablas las relacionamos sin condición, Oracle devuelve el **producto cartesiano** de esas tablas.
- Cuando existe una condición que relaciona ambas tablas y se devuelven sólo las filas que cumplen la condición se habla de **inner join** (o **simple join**). Esta es la forma más usual de vincular las tablas.
- En ocasiones es útil extender el resultado de estos joins, y entonces trabajaremos con **outer joins**. Al trabajar con outer joins, Oracle devolverá todas las filas que cumplan la condición del join más aquellas filas de la tabla marcada (LEFT, RIGHT) para las que no hemos encontrado filas que hayan hecho que se cumpla la condición del join  
Dependiendo de por cual de las tablas queremos extender el resultado, podemos hablar de
  - extender el resultado con las filas de la tabla que aparece en primer lugar en el FROM, para ello usaremos **LEFT [OUTER] JOIN en la cláusula FROM**.
  - extender el resultado con las filas de la tabla que aparece en segundo lugar, para esto podemos usar **RIGHT [OUTER] JOIN en la cláusula FROM**
  - extender el resultado con las filas de ambas tablas, habrá que usar **FULL [OUTER] JOIN en la cláusula FROM**.

# Repaso JOIN

Veamos unos ejemplos con la tabla USUARIO y con la tabla PEDIDO.

- **usuario** (email, nombre, apellidos ... )

Clave primaria: email



- **pedido** (numPedido, usuario, fecha date )

Clave primaria: numpedido

Clave ajena: usuario?  usuario



Se refiere a la tabla con la que está relacionada, en este caso "usuario". La columna con la que haya que relacionar será la clave primaria de esa tabla, es decir, email

# Repaso JOIN

## select email from usuario

EMAIL

-----  
acdLv@bitoben.mus.es  
acg@hotmail.com  
acl@dlsi.ua.es  
acn@hotmail.com  
adf@lolipop.com  
adlmm@ua.es  
adrm@dlsi.ua.es  
aeb@colegas.com  
afg@colegas.com  
agg@gmail.com  
agl@dlsi.ua.es  
agt@lamail.ar  
alm@lolipop.com  
ama@lolipop.com  
.  
-  
-  
-

**270 filas seleccionadas.**

## Select usuario from pedido

USUARIO

-----  
deg@lamail.ar  
jccf@eps.ua.es  
svv@colegas.com  
rbc@bitoben.mus.es  
mav@colegas.com  
jme@lolipop.com  
pge@colegas.com  
bmm@agwab.com  
amd@colegas.com  
jmem@colegas.com  
mps@agwab.com  
adlmm@ua.es  
hrdcj@colegas.com  
acl@dlsi.ua.es  
.  
.  
.

**51 filas seleccionadas**

# Repaso JOIN

```
select email, nombre, numpedido
from usuario, pedido
where email=usuario
```

EMAIL	NOMBRE	NUMPEDIDO
amd@colegas.com	ALEJANDRA	1
rpv@hotmail.com	RAMIRO	2
jmem@colegas.com	JUAN MANUEL	7
jptg@colegas.com	JUAN PABLO	9
jccf@eps.ua.es	JUAN CARLOS	11
mrar@colegas.com	MARIA ROSA	14
rbc@bitoben.mus.es	RUTH	15
.		
.		
.		

**51** filas seleccionadas

```
select email, nombre, numpedido
from usuario left join pedido
on email=usuario
```

EMAIL	NOMBRE	NUMPEDIDO
amd@colegas.com	ALEJANDRA	1
rpv@hotmail.com	RAMIRO	2
.		
.		
jmem@colegas.com	INES	
ihdlh@lamail.ar	ILOVENY	
.		
ieq@colegas.com	JUAN PABLO	9
jccf@eps.ua.es	JUAN CARLOS	11
mrar@colegas.com	MARIA ROSA	14
.		
..		

**270** filas seleccionadas

# Repaso JOIN

```
select email, nombre, numpedido
from usuario, pedido
where email=usuario
```

EMAIL	NOMBRE	NUMPEDIDO
-----	-----	-----
amd@colegas.com	ALEJANDRA	1
rpv@hotmail.com	RAMIRO	2
jmem@colegas.com	JUAN MANUEL	7
jptg@colegas.com	JUAN PABLO	9
jccf@eps.ua.es	JUAN CARLOS	11
mraj@colegas.com	MARIA ROSA	14
rbc@bitoben.mus.es	RUTH	15
.		
.		
.		

**51 filas seleccionadas**

```
select email, nombre, numpedido
from usuario right join pedido
on email=usuario
```

EMAIL	NOMBRE	NUMPEDIDO
-----	-----	-----
amd@colegas.com	ALEJANDRA	1
rpv@hotmail.com	RAMIRO	2
jmem@colegas.com	JUAN MANUEL	7
jptg@colegas.com	JUAN PABLO	9
jccf@eps.ua.es	JUAN CARLOS	11
mraj@colegas.com	MARIA ROSA	14
rbc@bitoben.mus.es	RUTH	15
.		
.		
.		

**51 filas seleccionadas**

# Repaso COUNT

EMPLEADO (DNI, NOMBRE, ESPECIALIDAD)

clave primaria: DNI

DNI	NOMBRE	ESPECIALIDAD
11111111A	Juan Martínez	1
22222222B	María Pérez	
33333333C	Ana Escudero	1
44444444D	Pedro Pérez	2
55555555M	Arturo Álvarez	3
66666666L	Luisa Olmo	3

```
SELECT count(*)  
FROM empleado
```

count(\*)  
6

```
SELECT count(dni)  
FROM empleado
```

count(dni)  
6

```
SELECT count(especialidad)  
FROM empleado
```

count(especialidad)  
5

```
SELECT count(distinct especialidad)  
FROM empleado
```

count(distinct especialidad)  
3

```
SELECT count(distinct dni)  
FROM empleado
```

count(distinct dni)  
6

# Repaso COUNT

No se debe confundir COUNT con SUM

DNI	NOMBRE	ESPECIALIDAD
11111111A	Juan Martínez	1
22222222B	María Pérez	
33333333C	Ana Escudero	1
44444444D	Pedro Pérez	2
55555555M	Arturo Álvarez	3
66666666L	Luisa Olmo	3

```
SELECT count(especialidad)
FROM empleado
```

count(especialidad)  
5

```
SELECT sum(especialidad)
FROM empleado
```

sum(especialidad)  
10

# Repaso GROUP BY - HAVING

DNI	NOMBRE	ESPECIALIDAD
11111111A	Juan Martínez	1
22222222B	María Pérez	
33333333C	Ana Escudero	1
44444444D	Pedro Pérez	2
55555555M	Arturo Álvarez	3
66666666L	Luisa Olmo	3

```
SELECT especialidad, count(*)  
FROM empleado  
GROUP BY especialidad
```

<u>especialidad</u>	<u>count(*)</u>
1	2
2	1
3	2
(null)	1

```
SELECT especialidad, count(*)  
FROM empleado  
GROUP BY especialidad  
HAVING count(*) >=2
```

<u>especialidad</u>	<u>count(*)</u>
1	2
3	2



# Repaso GROUP BY - HAVING

empleado

DNI	NOMBRE	DIRECCIÓN
11111111A	Juan Martínez	Federico Soto 5
22222222B	María Pérez	Formentera 1
33333333C	María Pérez	Calderón de la Barca 4
44444444D	Pedro Pérez	Doctor Casanova 20
55555555M	Arturo Álvarez	
66666666L	Luisa Olmo	Avda. de Novelda 12

participar

EMPLEADO	PROYECTO
11111111A	PROY1
11111111A	PROY2
22222222B	PROY1
33333333C	PROY3
66666666L	PROY1
66666666L	PROY3

Dni de los empleados que participan en proyectos junto con el total de proyectos en los que participan

```
SELECT empleado, count(*)
```

```
FROM participar
```

```
GROUP BY empleado
```

mostrar también el nombre del empleado ?

empleado	count(*)
11111111A	2
22222222B	1
33333333C	1
66666666L	2

```
SELECT empleado, nombre, count(*)
```

```
FROM participar, empleado
```

```
WHERE empleado=dni
```

```
GROUP BY empleado, nombre
```

```
empleado nombre count(*)
```

11111111A	Juan Martínez	2
22222222B	María Pérez	1
33333333C	María Pérez	1
66666666L	Luisa Olmo	2

# Repaso GROUP BY - HAVING

empleado

DNI	NOMBRE	DIRECCIÓN
11111111A	Juan Martínez	Federico Soto 5
22222222B	María Pérez	Formentera 1
33333333C	María Pérez	Calderón de la Barca 4
44444444D	Pedro Pérez	Doctor Casanova 20
55555555M	Arturo Álvarez	
66666666L	Luisa Olmo	Avda. de Novelda 12

participar

EMPLEADO	PROYECTO
11111111A	PROY1
11111111A	PROY2
22222222B	PROY1
33333333C	PROY3
66666666L	PROY1
66666666L	PROY3

Nombre de los empleados que participan en proyectos junto con el total de proyectos en los que participan

~~SELECT nombre, count(\*)  
FROM empleado, participar  
WHERE empleado=dni  
**GROUP BY** nombre  
nombre      count(\*)  
Juan Martínez      2  
María Pérez      2  
Luisa Olmo      2~~

SELECT nombre, count(\*)  
FROM empleado, participar  
WHERE empleado=dni  
**GROUP BY** empleado, nombre  
nombre      count(\*)  
Juan Martínez      2  
María Pérez      1  
María Pérez      1  
Luisa Olmo      2

# Subquery

- Introducida con [NOT] IN

Select \* from usuario  
where email IN (select usuario from pedido);

- Introducida con [NOT] EXISTS

Select \* from usuario u  
where EXISTS(select 1 from pedido p where u.email=p.usuario);

- Introducida con operadores aritméticos ([ALL|ANY])

Select \* from articulo where pvp = (select max(pvp) from articulo);  
Select \* from articulo where pvp >= ALL  
(Select pvp from articulo where pvp is not null);

# UNION

Select cod from tv

UNION

Select cod from camara

# ¡¡ATENCIÓN al trabajar con NULOS!!

En algunos casos es posible que podamos tener problemas al trabajar con valores nulos. Podemos evitar problemas usando la función NVL

NVL (expresión1, expresión2)

Si **expresión1** es NULO la función NVL devuelve **expresión2**

Si **expresión1** NO es NULO la función NVL devuelve **expresión1**

*Expresión1 y expresión2 pueden ser de cualquier tipo de datos.*

*Si no coinciden, Oracle convierte implícitamente expresión2 al tipo de datos de expresión1, si no puede devuelve un error.*

# ¡¡ATENCIÓN al trabajar con NULOS!!

## Ejemplos

```
SELECT email, nombre, NVL(direccion, 'desconocida')  
FROM usuario
```

```
SELECT *  
FROM articulo  
WHERE NVL(pvp, 0) > (select AVG(pvp) FROM articulo)
```

Os recomendamos repasar los apuntes de  
Fundamentos de las Bases de Datos  
(operadores, subconsultas, ...)  
**<https://sites.google.com/view/fbddocs/>**

Las tablas de la base de datos TiendaOnLine usada en FBD no han sido creadas por el usuario dbd\_alumno, por lo que no las vemos al ejecutar `SELECT * FROM USER_TABLES` ni en el desplegable de las Tablas del usuario.

Vuestros usuarios tienen permisos para acceder a las tablas de TiendaOnLine, por lo que podremos ejecutar sentencias `SELECT` sobre esas tablas, sin necesidad de ejecutar el script disponible en UACloud.

**Select \* from dbd\_alumnoCAST.articulo;**

La estructura de TiendaOnLine la podéis recordar en <https://sites.google.com/view/fbddocs/materiales/tienda-online-relacional>





**DBD**

Diseño de  
Bases de Datos

**¿DUDAS,  
PREGUNTAS?**



Universitat d'Alacant  
Universidad de Alicante

**lsi** Departamento  
de Lenguajes  
y Sistemas  
Informáticos



**DBD**

Diseño de  
Bases de Datos

# TRABAJANDO CON ORACLE – SENTENCIA SELECT

