

# **SQL-2: Consultas**

**BASES DE DATOS**

**Profesor: Héctor Gómez Gauchía**

**Materiales: Héctor Gómez Gauchía, Mercedes García Merayo**

## Consulta de datos

---

```
SELECT  $C_{j1}, \dots, C_{jr}$   
FROM tabla1, tabla2, ..., tablak  
WHERE condición
```

- ▶ La cláusula **SELECT** especifica las columnas (o expresiones) que deben aparecer en el resultado.
- ▶ La cláusula **FROM** especifica el producto cartesiano de tablas.
- ▶ La cláusula **WHERE** especifica las condiciones de selección para las tablas indicadas en la cláusula **FROM**.
- ▶ La cláusula **WHERE** es opcional, pero **SELECT** y **FROM** son obligatorias

# Consulta de datos

---

Seleccionar los códigos de proyecto del empleado con dni 27347234T.

```
SELECT codigoPr FROM distribución  
WHERE dniEmp = '27347234T'
```

**codigoPR**

---

**PRI**

**PR2**

**PR3**

## Consulta de datos

---

Seleccionar dnis de los empleados que trabajan entre 15 y 25 horas en algún proyecto junto con las horas y el código de Proyecto.

```
SELECT codigoPr,dniEmp,horas
FROM distribucion
WHERE horas>=15 AND horas<=25
```

codigoPR	dniEmp	horas
PR1	27347234T	20
PR2	27347234T	25
PR3	27347234T	25

## Consulta de datos

---

- ▶ Una consulta SQL se corresponde a una expresión del álgebra relacional que implica selecciones, proyecciones y productos cartesianos.
  - ▶ La cláusula SELECT se emplea para hacer proyecciones.
  - ▶ La cláusula FROM se emplea para hacer productos cartesianos e indicar relaciones de las que obtener los datos.
  - ▶ La cláusula WHERE se emplea para hacer selecciones.

## Consulta de datos

- ▶ Las consultas SQL trabajan con multiconjuntos en lugar de conjuntos: **se permite repetir valores.**
- ▶ La respuesta a una consulta SQL es una relación, que se corresponde con un multiconjunto de filas en SQL.
- ▶ Para trabajar con conjuntos (eliminación de tuplas duplicadas) se utiliza la cláusula **DISTINCT**

```
SELECT dniEmp  
FROM distribucion  
WHERE horas>=15 AND horas<=25
```



dniEmp
27347234T
27347234T
27347234T

```
SELECT DISTINCT dniEmp  
FROM distribucion  
WHERE horas>=15 AND horas<=25
```



dniEmp
27347234T

## Consulta de datos

---

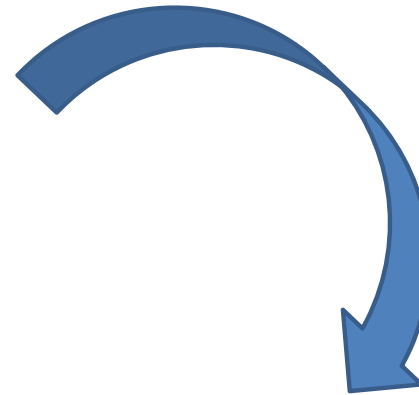
- ▶ Estrategia de evaluación de una consulta
  - ▶ Calcular el producto cartesiano de las tablas que aparecen en la cláusula FROM.
  - ▶ Eliminar las filas del producto cartesiano que no cumplen la condición que aparece en la cláusula WHERE.
  - ▶ Eliminar todas las columnas que no aparecen en la lista de atributos de la cláusula SELECT.
  - ▶ Si se especifica DISTINCT, se eliminan las filas duplicadas.
- ▶ Esta estrategia es ineficiente y no es realmente la que utilizan los SGBD, pero sirve para entender el significado de las consultas.

## Consulta de datos

---

- ▶ Para seleccionar **todos los atributos** se usa un \* en lugar de lista de atributos

```
SELECT *  
FROM Emp  
WHERE DNI='27347234T'
```



DNI	Nombre	Dirección
27347234T	Marta Sánchez	Loreto 134

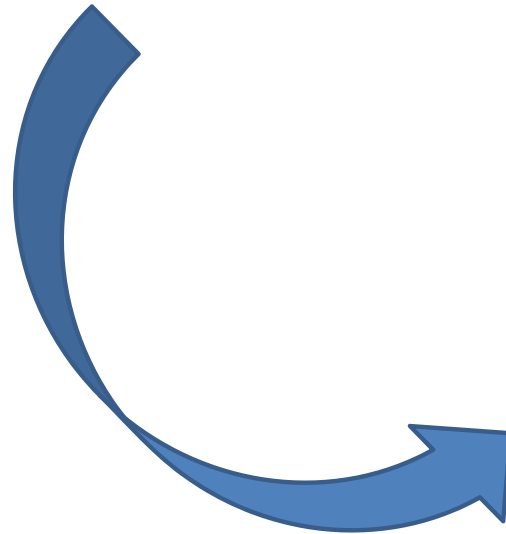


## Consulta de datos

---

- ▶ Los atributos pueden ser expresiones

```
SELECT DNIEmp, Horas/10  
FROM distribucion
```



dniEmp	Horas/10
27347234T	2
34126455Y	1
27347234T	2.5
37562365F	4.5
37562365F	1
27347234T	2.5

# Consulta de datos

---

## ► Expresiones

- Aritméticas: + (suma), - (resta), \* (producto), / (división)
- Cadenas de caracteres: || (concatenación)
- En ORACLE existe una tabla especial, **dual** que se puede usar para evaluar expresiones:

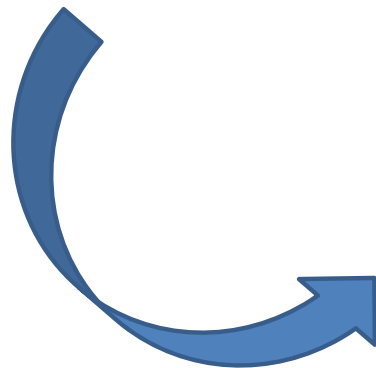
**SELECT 3 + 4 FROM DUAL**

**SELECT SQRT(5) FROM DUAL**

## Consulta de datos

- Se pueden **renombrar** las cabeceras de las columnas de la salida de una consulta

```
SELECT codigoPr "Código Proyecto",  
DniEmp "DNI Empleado"  
FROM distribucion;
```



Código Proyecto	DNI Empleado
PR1	27347234T
PR1	34126455Y
PR2	27347234T
PR3	37562365F
PR2	37562365F
PR3	27347234T

- También es posible indicar un alias para las tablas que aparecen en la cláusula **FROM**.

# Claúsula WHERE

---

- ▶ Las condiciones pueden ser
  - ▶ Conjunciones (AND) de condiciones booleanas simples
  - ▶ Disyunciones (OR) de condiciones booleanas simples
  - ▶ Negaciones (NOT) de condiciones booleanas simples
- ▶ Las condiciones booleanas simples pueden ser
  - ▶ De comparación entre valores (=,>,>=,<,<=,!=)
  - ▶ De similitud entre cadenas de caracteres

**atributo [NOT] LIKE 'patrón'**

- % Una serie cualquiera de caracteres
- \_ Un carácter cualquiera

# Claúsula WHERE

---

- ▶ Comprobación de NULL

**atributo IS NULL**

**atributo IS NOT NULL**

- ▶ Pertenencia a conjuntos de valores

**atributo [NOT] IN (v1,v2,...,vn)**

- ▶ Valores dentro de un rango

**atributo BETWEEN v1 AND v2**

# Claúsula WHERE

---

Seleccionar los datos de los empleados cuyo nombre comienza por 'Te'.

```
SELECT * FROM Emp
WHERE nombre LIKE 'Te%'
```

Seleccionar los datos de los empleados que tienen una "a" en el tercer carácter del nombre.

```
SELECT nombre FROM Emp
WHERE nombre LIKE '__a%'
```

Seleccionar los datos de los empleados que no tienen teléfono.

```
SELECT * FROM Emp
WHERE telefono IS NULL
```

## Claúsula ORDER BY

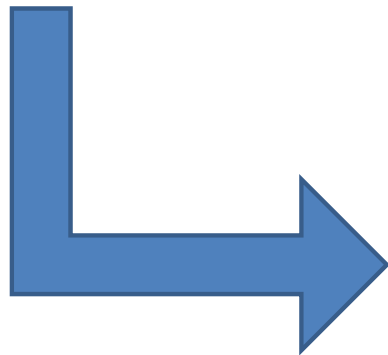
---

- ▶ Permite establecer el orden de presentación de las tuplas de una relación mediante la cláusula **ORDER BY**.
- ▶ Por defecto el orden es ascendente, pudiéndose establecer orden descendente mediante **DESC**.
- ▶ Si se indica **ORDER BY** debe ser la última cláusula de la sentencia.

## Claúsula Order By

Código de proyecto, DNI y horas trabajadas de los Empleados que trabajan más de 10 horas en algún proyecto, *ordenados por horas.*

```
SELECT CodigoPr, DNIEmp, horas  
FROM distribucion  
WHERE horas > 10  
ORDER BY horas
```



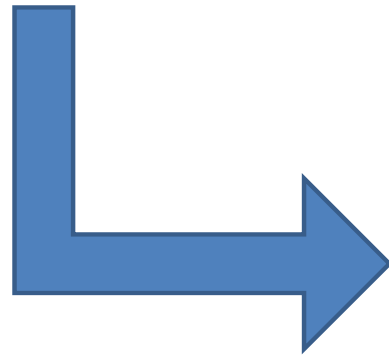
codigoPR	dniEmp	horas
PR2	37562365F	10
PR1	34126455Y	10
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45



# Claúsula Order By

*Ordenados por horas de forma descendente.*

```
SELECT CodigoPr, DNIEmp, horas  
FROM distribucion  
WHERE horas > 10  
ORDER BY horas DESC
```



codigoPR	dniEmp	horas
PR3	37562365F	45
PR2	27347234T	25
PR3	27347234T	25
PR1	27347234T	20
PR1	34126455Y	10
PR2	37562365F	10

# Funciones

**ROUND(n,decimales)** Redondea el número al siguiente número con el número de decimales indicado más cercano.

**TRUNC(n,decimales)** Los decimales del número se cortan para que sólo aparezca el número de decimales indicado .

**MOD(n1,n2)** Devuelve el resto resultante de dividir *n1* entre *n2*.

**POWER(valor,exponente)** Eleva el valor al exponente indicado.

**SQRT(n)** Calcula la raíz cuadrada de *n*.

**SIGN(n)** Devuelve *1* si *n* es *positivo*, *0* si vale *cero* y *-1* si es *negativo*.

**ABS(n)** Calcula el valor absoluto de *n*.

**EXP(n)** Calcula  $e^n$  .

**LOWER(texto)** Convierte el texto a minúsculas (funciona con los caracteres españoles).

**UPPER(texto)** Convierte el texto a mayúsculas.

**INITCAP(texto)** Coloca la primera letra de cada palabra en mayúsculas.

# Funciones

---

**RTRIM(texto)** Elimina los espacios a la derecha de texto.

**LTRIM(texto)** Elimina los espacios a la izquierda que posea texto.

**TRIM(texto)** Elimina los espacios en blanco a la izquierda y la derecha de texto y los espacios dobles del interior.

**TRIM(caracteres FROM texto)** Elimina de texto los caracteres indicados. Por ejemplo **TRIM('h' FROM nombre)** elimina las haches de la columna nombre que estén a la izquierda y a la derecha.

**SUBSTR(texto,n[,m])** Obtiene los m siguientes caracteres de texto a partir de la posición n (si m no se indica se cogen desde n hasta el final).

**LENGTH(texto)** Obtiene el tamaño de texto.

**INSTR(texto, textoBuscado [,posInicial [, nAparición]])** Obtiene la posición en la que se encuentra textoBuscado en texto. Se puede empezar a buscar a partir de una posición inicial concreta e incluso indicar el número de aparición del texto buscado. Si no lo encuentra devuelve 0.

**REPLACE(texto, textoABuscar, [textoReemplazo])** Buscar el texto a buscar en un determinado texto y lo cambia por el indicado como textoReemplazo. Si no se indica texto de reemplazo, entonces esta función elimina el textoABuscar.

# Funciones

---

**LPAD(texto, anchuraMáxima, [caracterDeRelleno])** **RPAD(texto, anchuraMáxima, [caracterDeRelleno])**  
Rellena texto a la izquierda (LPAD) o a la derecha (RPAD) con el carácter indicado para ocupar la anchura indicada.

Si texto es más grande que la anchura indicada, texto se recorta. Si no se indica carácter de relleno se rellenará el espacio marcado con espacios en blanco.

Ejemplo: **LPAD('Hola',10,'-')** da como resultado **-----Hola**

**REVERSE(texto)** Invierte texto (le da la vuelta).

**NVL(valor, sustituto)** Si *valor* es *NULL*, devuelve *sustituto*; de otro modo, devuelve *valor*.

**NVL2(valor, sustituto1, sustituto2)** Variante de la anterior, devuelve *sustituto1* si *valor* no es nulo. Si *valor* es nulo devuelve *sustituto2*

**COALESCE(listaExpresiones)** Devuelve la primera de las expresiones que no es nula.

```
CREATE TABLE test ( col1 VARCHAR2(1), col2 VARCHAR2(1), col3 VARCHAR2(1));  
INSERT INTO test VALUES (NULL, 'B', 'C');  
INSERT INTO test VALUES ('A', NULL, 'C');  
INSERT INTO test VALUES (NULL, NULL, 'C');  
INSERT INTO test VALUES ('A', 'B', 'C');  
SELECT COALESCE(col1, col2, col3) FROM test; El resultado es: B A C A
```

# Funciones

---

**SYSDATE** Obtiene la fecha y hora actuales.

**ADD\_MONTHS**(fecha,n) Añade a la fecha el número de meses indicado por *n*.

**MONTHS\_BETWEEN**(fecha1, fecha2) Obtiene la diferencia en meses entre las dos fechas (puede ser decimal).

**NEXT\_DAY**(fecha,día) Indica cual es el día que corresponde a añadir a la fecha el día indicado. El día puede ser el texto 'Lunes', 'Martes', 'Miércoles',... *(si la configuración está en español)* o el número de día de la semana (*1=lunes, 2=martes,...*) .

**LAST\_DAY**(fecha) Obtiene el último día del mes al que pertenece la fecha. Devuelve un valor DATE.

**EXTRACT**(valor FROM fecha) Extrae un valor de una fecha concreta. El valor puede ser day (día), month (mes), year (año), etc.

**GREATEST**(fecha1, fecha2,..) Devuelve la fecha más moderna de la lista.

**LEAST**(fecha1, fecha2,..) Devuelve la fecha más antigua de la lista.

## Conversión de Datos

---

- ▶ Oracle es capaz de convertir datos automáticamente a fin de que la expresión final tenga sentido. Por ello son fáciles las conversiones de texto a número y viceversa.

```
SELECT 5+'3' FROM DUAL /*El resultado es 8 */
```

```
SELECT 5 || '3' FROM DUAL /* El resultado es 53 */
```

### ▶ TO\_CHAR

- ▶ Obtiene un texto a partir de un número o una fecha.
- ▶ En el caso de las fechas se indica el formato de conversión en una cadena de texto.

# Conversión de Datos

YY (Año en formato de dos cifras )  
YYYY (Año en formato de cuatro cifras)  
MM (Mes en formato de dos cifras )  
MON (Las tres primeras letras del mes )  
MONTH (Nombre completo del mes)  
DY (Día de la semana en tres letras )  
DAY (Día completo de la semana )  
D (Día de la semana del 1 al 7)  
DD (Día en formato de dos cifras)  
DDD (Día del año)

Q (Semestre )  
WW (Semana del año )  
AM (Indicador AM )  
PM (Indicador PM )  
HH12 (Hora de 1 a 12 )  
HH24 (Hora de 0 a 23 )  
MI (Minutos 0 a 59)  
SS (Segundos 0 a 59)  
SSSS (Segundos desde medianoche )

`/ . , :: ; '`  Posición de los separadores, donde se pongan estos símbolos aparecerán en el resultado

```
SELECT TO_CHAR(SYSDATE, 'DD/MONTH/YYYY, DAY HH:MI:SS') FROM DUAL
```

```
/* 6/AGOSTO /2004, LUNES 08:35:15*/
```

# Conversión de Datos

---

## ▶ TO\_NUMBER

- ▶ Convierte textos en números, indicando el formato de la conversión

**9** Posición del número  
**0** Posición del número (muestra ceros)  
**\$** Formato dólar  
**L** Símbolo local de la moneda  
**S** Aparece el símbolo del signo  
**D** Posición del símbolo decimal (en español, la coma)  
**G** Posición del separador de grupo (en español el punto)

## ▶ TO\_DATE

- ▶ Convierte textos en fechas. Como segundo parámetro se utilizan los códigos de formato de fechas comentados anteriormente.



# Conversión de Datos

---

## ► DECODE

```
DECODE(expresión, valor1,resultado1  
[,valor2, resultado2,...]  
[,valorPordefecto])
```

```
SELECT DECODE(cotizacion,1, salario*0.85,  
2,salario * 0.93, 3,salario * 0.96, salario)  
FROM empleados;
```

```
SELECT DECODE(codtienda,101, 'Centro',  
102, 'CC.La vaguada', 103, 'Callao', 'Otra')  
FROM tiendas;
```

# Operaciones sobre Conjuntos

---

## ▶ UNION, INTERSECT y MINUS

- ▶ Los **SELECTS** necesitan tener el mismo número de columnas
  - ▶ Las columnas tienen que ser del mismo tipo.
- 
- ▶ Las filas no se repiten, si se desea obtener todas las filas se debe usar **UNION ALL**.

# Operadores de Agregación

---

- ▶ Consultas en las que se agrupan datos de distintos registros a fin de realizar cálculos.
- ▶ Para ello se utiliza la cláusula **GROUP BY** que permite indicar en base a qué registros se realiza la agrupación.
- ▶ **Funciones de cálculo con grupos**
  - ▶ **SUM([DISTINCT]col|expr)**: suma de todos los valores (distintos) de la columna col (numérica) o la expresión.
  - ▶ **AVG([DISTINCT]col|expr)**: promedio de los valores (distintos) de la columna col (numérica) o la expresión.
  - ▶ **MAX(col|expr)**: el valor máximo de la columna col (expresión).
  - ▶ **MIN(col|expr)**: el valor mínimo de la columna col (expresión).
  - ▶ **COUNT([DISTINCT]col|expr)**: numero de valores (distintos) de la columna col (expresión).

# Operadores de Agregación

CodigoPr	Dni	Horas
P1	I24I542G	14
P2	8754545P	22
P7	I546856L	
P4	2I78454W	16
P1	5754555C	10

- ▶ **COUNT(\*)** cuenta todas las filas incluidas duplicados y nulos.
- ▶ **COUNT(col|expr)** cuenta las filas cuyo valor de col (expr) no sea nulo.

<b>SELECT COUNT( * ) FROM distribucion;</b>	<b>5</b>	
SELECT COUNT( CodigoPr ) FROM distribucion	5	
SELECT COUNT( DISTINCT CodigoPr ) FROM distribucion	4	
SELECT COUNT( Horas ) FROM distribucion	4	
SELECT SUM( Horas ) FROM distribucion	62	
SELECT CodigoPr , SUM( Horas ) FROM distribucion GROUP BY CodigoPr	<b>P1</b>	<b>24</b>
	<b>P2</b>	<b>22</b>
	<b>P7</b>	
	<b>P4</b>	<b>16</b>

## cláusula Select Extendida

---

```
SELECT [DISTINCT] lista_atributos
FROM lista_tablas
WHERE condición
GROUP BY lista_atributos_grupos
HAVING condición_sobre_grupos
```

- ▶ Si se **omite** **GROUP BY** se considera a toda la tabla como un único grupo.
- ▶ La cláusula **HAVING** determina los grupos que aparecerán en el resultado.
- ▶ Las expresiones de la cláusula **HAVING** deben tener un solo valor por grupo.
- ▶ Las columnas que aparecen en *condición\_sobre\_grupos* deben aparecer como argumentos del operador de agregación o en la cláusula **GROUP BY**.
- ▶ La lista de atributos de la cláusula **SELECT** consta de:
  - ▶ Lista de nombres de columnas. Estas deben aparecer también en *lista\_atributos\_grupos*.
  - ▶ Lista de operaciones de agregación.

## cláusula Select Extendida

---

- ▶ Pasos en la ejecución de una instrucción de agrupación por parte del gestor de bases de datos:
  - ▶ Seleccionar las filas deseadas utilizando **WHERE**. Esta cláusula eliminará columnas en base a la condición indicada.
  - ▶ Se establecen los grupos indicados en la cláusula **GROUP BY**.
  - ▶ Se calculan los valores de las funciones de totales (**COUNT**, **SUM**, **AVG**, ...)
  - ▶ Se filtran los registros que cumplen la cláusula **HAVING**.
  - ▶ El resultado se ordena en base al apartado **ORDER BY**.

# Operadores de Agregación

**DNIIs y total de horas de los empleados que trabajan en 2 o más proyectos diferentes.**

```
SELECT dniEmp, SUM(Horas)
FROM distribucion
GROUP BY dniEmp
HAVING COUNT(dniEmp)>= 2
ORDER BY SUM(Horas)
```

codigoPR	dniEmp	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

dniEmp	horas
27347234T	20
27347234T	25
27347234T	25
37562365F	45
37562365F	10
34126455Y	10

dniEmp	Sum(horas)
27347234T	70
37562365F	55

# Operadores de Agregación

Mostrar proyectos en los que trabajan 2 o más empleados.

```
SELECT codigoPr, count(*)  
FROM distribucion  
GROUP BY codigoPr  
HAVING COUNT(*)>= 2  
ORDER BY codigoPr
```

codigoPR	dniEmp	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

codigoPR	dniEmp
PR1	27347234T
PR1	37562365F
PR1	34126455Y
PR3	27347234T
PR3	37562365F
PR2	27347234T

codigoPR	Count(*)
PR1	3
PR3	2



## Join

---

```
SELECT tabla1.columna1, tabla1.columna2,...
tabla2.columna1, tabla2.columna2,...
FROM tabla1
[CROSS JOIN tabla2]|
[NATURAL JOIN tabla2]|
[JOIN tabla2
  USING(columna[,columna,...columna])]|
[JOIN tabla2 ON
  (tabla1.columna=tabla2.columna)]|
[LEFT|RIGHT|FULL OUTER JOIN tabla2 ON
  (tabla1.columna=tabla2.columna)]
WHERE ...
```

## Join

---

- ▶ Aparece en la cláusula **FROM** de las consultas SQL y permiten establecer relaciones entre las tuplas de diferentes tablas.
- ▶ Estas relaciones también pueden establecerse en la cláusula **WHERE** mediante expresiones que incluyan atributos de las diferentes tablas.

## Join

---

- ▶ Utilizando la opción **CROSS JOIN** se realiza un producto cartesiano entre las tablas indicadas.
- ▶ No es necesario indicar la cláusula **CROSS JOIN**, basta separar los nombres de las tablas mediante comas.
- ▶ Es posible indicar restricciones en la cláusula **WHERE**.
- ▶ Estas consultas son equivalentes
  - ▶ **SELECT Nombre, DNI,CodigoPr  
FROM Emp CROSS JOIN distribucion;**
  - ▶ **SELECT Nombre, DNI, CodigoPr FROM Emp, distribucion;**

# Join

Nombre de los empleados, códigos de proyectos y horas que trabajan en cada proyecto.

```
SELECT Nombre, CodigoPr, Horas  
FROM Emp, distribucion  
WHERE distribucion.dniEmp = Emp.DNI
```

DNI	Nombre
27347234T	Marta Sánchez
85647456W	Alberto San Gil
37562365F	María Puente
34126455Y	Juan Panero

codigoPR	dniEmp	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Nombre	CodigoPr	Horas
Marta Sánchez	PR1	20
Marta Sánchez	PR3	25
Marta Sánchez	PR2	25
María Puente	PR3	45
María Puente	PR1	10
Juan Panero	PR1	10

# Join

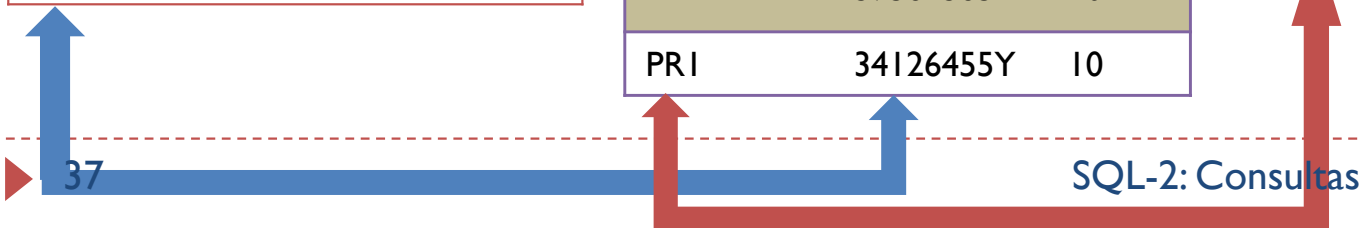
Nombre de los empleados y los proyectos en los que trabajan.

```
SELECT nombre, desc  
FROM Emp, distribucion, proyectos  
WHERE distribucion.dniEmp = Emp.DNI AND  
proyectos.codigo = distribucion.codigoPr;
```

DNI	Nombre
27347234T	Marta Sánchez
85647456W	Alberto San Gil
37562365F	María Puente
34126455Y	Juan Panero

codigoPR	dniEmp	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

CodigoPr	DNIDir	Desc
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística



## Join

---

Nombre de los empleados y los directores de los proyectos en los que trabajan.

```
SELECT Trabajador.nombre, Director.nombre  
FROM Emp Trabajador, distribucion, Emp Director,  
     proyectos  
WHERE distribucion.dniEmp = Trabajador.DNI AND  
     proyectos.codigo = distribucion.codigoPr  
AND Director.DNI = proyectos.dniDir
```

# Join

---

## ▶ NATURAL JOIN

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

## ▶ JOIN USING

Permite establecer relaciones indicando qué columna (o columnas) común a las dos tablas hay que utilizar.

## ▶ JOIN ON


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

# Join Natural

```
SELECT CodigoPr, NombreDept FROM  
Proyectos NATURAL JOIN Departamentos
```

CodigoPr	DNIDir	Desc	CodDept
PR1	27347234T	Ventas	SMP
PR2	37562365F	Personal	RRHH
PR3	37562365F	Logística	SMP

CodDept	NombreDept
SMP	Servicios Múltiples
RRHH	Recursos Humanos



CodigoPr	NombreDept
PR1	Servicios Múltiples
PR2	Recursos Humanos
PR3	Servicios Múltiples



# Join On

```
SELECT Nombre, codigoPr, Horas FROM  
Emp JOIN Distribucion  
ON Emp.DNI = Distribucion.DniEmp
```

DNI	Nombre
27347234T	Marta Sánchez
85647456W	Alberto San Gil
37562365F	María Puente
34126455Y	Juan Panero

codigoPR	dniEmp	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Nombre	CodigoPr	Horas
Marta Sánchez	PR1	20
Marta Sánchez	PR3	25
Marta Sánchez	PR2	25
María Puente	PR3	45
María Puente	PR1	10
Juan Panero	PR1	10

# Join ON

```
SELECT Emp.Nombre, Deptos.Nombre FROM  
Emp JOIN Deptos  
ON Emp.CodDept = Deptos.CodDept  
Where Salario >1500
```

DNI	Nombre	CodDept	Salario
27347234T	Marta Sánchez	SMP	2301
85647456W	Alberto San Gil	SMP	1569
37562365F	María Puente	RRHH	1236
34126455Y	Juan Panero	SMP	2500




CodDept	Nombre
SMP	Servicios Múltiples
RRHH	Recursos Humanos

Emp.Nombre	Deptos.Nombre
Marta Sánchez	Servicios Múltiples
Alberto San Gil	Servicios Múltiples
Juan Panero	Servicios Múltiples

# Join USING

```
SELECT Emp.Nombre, Deptos.Nombre FROM  
Emp JOIN Deptos  
USING (CodDept)  
Where Salario >1500
```

DNI	Nombre	CodDept	Salario
27347234T	Marta Sánchez	SMP	2301
85647456W	Alberto San Gil	SMP	1569
37562365F	María Puente	RRHH	1236
34126455Y	Juan Panero	SMP	2500



CodDept	Nombre
SMP	Servicios Múltiples
RRHH	Recursos Humanos

Emp.Nombre	Deptos.Nombre
Marta Sánchez	Servicios Múltiples
Alberto San Gil	Servicios Múltiples
Juan Panero	Servicios Múltiples

# Join

---

- ▶ Utilizando las formas anteriores de relacionar tablas solo aparecen en el resultado de la consulta filas presentes en las tablas relacionadas.

**inner join**

**left outer join**

**right outer join**

**full outer join**

- ▶ Interna (**inner**) : en el resultado solo participan tuplas que satisfacen la condición.
- ▶ Externa (**outer**): también participan las tuplas que *no satisfacen la condición*.

# Reunión de Relaciones

---

## ▶ LEFT OUTER JOIN

- ▶ El resultado de esta operación siempre contiene todas las tuplas de la tabla de la izquierda, aun cuando no exista una tupla correspondiente en la tabla de la derecha.
- ▶ La sentencia **LEFT OUTER JOIN** devuelve las tuplas formadas por todos los valores de la tabla izquierda con los valores de la tabla de la derecha correspondientes, o retorna un valor nulo NULL, en caso de que no exista correspondencia.

# Join On

```
SELECT Nombre, codigoPr, Horas FROM  
Emp LEFT OUTER JOIN Distribucion  
ON Emp.DNI = Distribucion.DniEmp
```

DNI	Nombre
27347234T	Marta Sánchez
85647456W	Alberto San Gil
37562365F	María Puente
34126455Y	Juan Panero

codigoPR	dniEmp	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Nombre	CodigoPr	Horas
Marta Sánchez	PR1	20
Marta Sánchez	PR3	25
Marta Sánchez	PR2	25
Alberto San Gil	Null	Null
María Puente	PR3	45
María Puente	PR1	10
Juan Panero	PR1	10

# Reunión de Relaciones

---


- ▶ **RIGHT OUTER JOIN**
- ▶ El resultado de esta operación siempre contiene todas las tuplas de la tabla de la derecha, aun cuando no exista una tupla correspondiente en la tabla de la izquierda.
- ▶ La sentencia **RIGHT OUTER JOIN** retorna las tuplas formadas por todos los valores de la tabla derecha con los valores de la tabla de la izquierda correspondientes, o retorna un valor nulo NULL, en caso de que no exista correspondencia.

## Join On

```
SELECT codigoPr, NombreDept FROM
Proyectos RIGHT OUTER JOIN Deptos
ON Proyectos.CodDept = Deptos.CodDept
```

CodigoPr	DNIDir	Desc	CodDept
PR1	27347234T	Ventas	SMP
PR2	37562365F	Personal	RRHH
PR3	37562365F	Logística	SMP

CodDept	NombreDept
SMP	Servicios Múltiples
RRHH	Recursos Humanos
COM	Comercial



CodigoPr	NombreDept
PR1	Servicios Múltiples
PR2	Recursos Humanos
PR3	Servicios Múltiples
Null	Comercial



# Reunión de Relaciones

---

## ▶ FULL OUTER JOIN

- ▶ Esta operación presenta los resultados de tabla izquierda y tabla derecha, aunque no tengan correspondencia en la otra tabla.
- ▶ La tabla combinada contendrá, entonces, todas las filas de ambas tablas y presentará valores nulos para las filas sin correspondencia.

## Consultas Anidadas

---

- ▶ Es una de las características mas potentes de SQL
- ▶ Una **consulta anidada** es aquella que tiene otra consulta incorporada en su interior a la que se llama **subconsulta**
- ▶ Las subconsultas suelen aparecer en la cláusula **WHERE** pero también pueden aparecer en la cláusula **FROM** o **HAVING**
- ▶ Habitualmente se utilizan para comprobar la pertenencia a conjuntos, cardinalidad de los mismos o establecer comparaciones

## Consultas Anidadas

---

- ▶ Si la subconsulta devuelve un único valor se podrán utilizar los operadores de comparación >, <, >=, <=, != y =.

```
SELECT Nombre, salario
FROM Empleados
WHERE salario <
      (SELECT salario FROM Empleados
       WHERE DNI='08235723R')
```

- ▶ Las subconsultas pueden devolver un conjunto de filas. En este caso, no podremos utilizar un operador de comparación directamente.

## Consultas Anidadas

---

- ▶ **>, >=, <, <=, =, <> ANY**
- ▶ Permite comprobar si un valor es mayor (mayor o igual, menor,...) que algún elemento del resultado de la subconsulta.
- ▶ **>, >=, <, <=, =, <> ALL**
- ▶ Permite comprobar si un valor es mayor (mayor o igual, menor,...) que todos los elementos del resultado de la subconsulta.
- ▶ La conectiva **(NOT)IN** comprueba la pertenencia o no al conjunto de valores resultado de una cláusula **SELECT**.
- ▶ Las funciones de agrupación se pueden aplicar a subconsultas siempre y cuando estas tengan una sola expresión en su parte select y esta sea de tipo numérico.

## Consultas Anidadas

---

Todos los empleados excepto el que menos horas trabaja.

```
SELECT Nombre, DNI, sum(horas)
FROM Emp, distribucion
WHERE DNIEmp = DNI
GROUP BY Nombre, DNI
HAVING sum(horas) > ANY
        (SELECT SUM(Horas) FROM distribucion
         GROUP BY distribucion.DNIEmp)
```


## Consultas Anidadas

---

- ▶ Las **consultas anidadas** pueden ser **correlacionadas**, es decir, consultas en las que la subconsulta depende de la fila que se está examinando en cada momento en la consulta exterior.

**Seleccionar los empleados que dirigen y trabajan en un mismo proyecto**

```
SELECT DISTINCT DNIDIr FROM proyectos P
WHERE codigo IN
  (SELECT codigoPr
   FROM distribucion D
   WHERE D.DNIEmp= P.DNIDIr)
```



# Consultas Anidadas

---

## ▶ EXISTS, NOT EXISTS

- ▶ Son operadores para realizar comparaciones de conjuntos.
- ▶ Permite comprobar si un conjunto no esta vacio, es decir, se trata como una comparación de igualdad con el conjunto vacio.

### □ EXISTS (subconsulta)

- Cierto si la subconsulta devuelve alguna fila
- Falso en otro caso

### □ NOT EXISTS (subconsulta)


- Cierto si la consulta no devuelve ningún valor
- Falso en otro caso

## Consultas Anidadas

---

Empleados que trabajan mas de 10 horas en algún proyecto

```
SELECT Nombre, DNI
FROM Emp E
WHERE EXISTS (SELECT CodigoPr FROM distribucion D
               WHERE D.DNIEmp = E.DNI AND horas > 10)
```



Para cada la fila de empleados, se evalúa la subconsulta. Si la subconsulta produce algún resultado, entonces la fila aparece en la respuesta.



# Consultas Anidadas

```
SELECT Nombre, DNI  
FROM Emp E
```

```
WHERE EXISTS (SELECT CodigoPr FROM distribucion D  
WHERE D.DNIEmp = E.DNI AND horas > 10)
```

DNI	Nombre
27347234T	Marta Sánchez
85647456W	Alberto San Gil
37562365F	María Puente
34126455Y	Juan Panero

DNI	Nombre
27347234T	Marta Sánchez
37562365F	María Puente

codigoPR
PR1
PR3
PR2

codigoPR
PR3


codigoPR	dniEmp	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

# Consultas Anidadas

---

Empleados que no trabajan en ningún proyecto

```
SELECT Nombre, DNI
FROM Emp E
WHERE NOT EXISTS (SELECTCodigoPr FROMdistribucion D
                  WHERE D.DNIEmp = E.DNI)
```



Para cada la fila de empleados, se evalúa la subconsulta. Si la subconsulta NO produce ningún resultado, entonces la fila aparece en la respuesta. En caso contrario la fila no aparecerá.

# Consultas Anidadas

```
SELECT Nombre, DNI  
FROM Emp E
```

```
WHERE NOT EXISTS (SELECT CodigoPr FROM distribucion D  
WHERE D.DNIEmp = E.DNI AND horas > 10)
```

DNI	Nombre
27347234T	Marta Sánchez
85647456W	Alberto San Gil
37562365F	María Puente
34126455Y	Juan Panero

codigoPR

codigoPR

codigoPR	dniEmp	horas
PRI	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PRI	37562365F	10
PRI	34126455Y	10

DNI	Nombre
85647456W	Alberto San Gil
34126455Y	Juan Panero

# Vistas

---

- ▶ Una vista es una **tabla virtual**, es decir, una relación que no forma parte del modelo lógico pero que aparece como tal ante el usuario.
- ▶ Definición de una vista

```
create view view_name [lista_columnas]
as <expresión de consulta>
[with read only|with check option]
```
- ▶ El SGBD guarda la definición de la vista no el resultado.
- ▶ Los nombres de las vistas pueden aparecer en cualquier lugar en el que pueda aparecer un relación.

# Vistas

---

- ▶ **with read only:** la vista no permite actualizaciones.

```
CREATE VIEW clientes_ro (nombre, direccion, saldo)
  AS SELECT nombre || ' ' || apellidos, direccion, saldo
  FROM clientes
WITH READ ONLY;
```

- ▶ **with check option:** solo permite la inserción y actualización de filas que cumplan la subconsulta de la definición.

```
CREATE VIEW emp_SMP (DNI,nombre, salario)
  AS SELECT DNI, nombre, salario
  FROM EMP WHERE CodDept='SMP' and salario < 1500
WITH CHECK OPTION;
```

# Vistas

---

- ▶ Las vistas actualizables deben cumplir **VARIAS** restricciones. Algunas de ellas son:
  - ▶ Todas las columnas obligatorias (**NOT NULL**) deben aparecer en la definición de la vista.
  - ▶ La consulta no contiene operadores de conjunto como **UNION**, **MINUS** o **INTERSECT**.
  - ▶ La cláusula **DISTINCT** tampoco está permitida.
  - ▶ Ninguna función de agregación puede estar incluida en la cláusula **SELECT**.
  - ▶ No se puede utilizar las cláusulas **GROUP BY** y **ORDER BY**.

# Vistas

---

**Nombres de los empleados que trabajan mas horas que la media**

```
CREATE VIEW DNIHoras (DNI, Horas) AS
    SELECT DNIEMP, SUM(Horas)
    FROM distribucion
    GROUP BY DNIEMP;
```

```
CREATE VIEW NombreHoras (Nombre, Horas) AS
    SELECT Emp.nombre, DNIHoras.Horas
    FROM Emp, DNIHoras
    WHERE Emp.DNI = DNIHoras.DNI;
```

```
CREATE VIEW MediaHoras (media) AS
    SELECT AVG(Horas) FROM NombreHoras;
```

```
SELECT Nombre FROM NombreHoras
WHERE Horas > (SELECT media FROM MediaHoras);
```

# Vistas

---

**DNI del empleado (o empleados) que trabaja(n) mas horas**

```
CREATE VIEW DNIHoras (DNI, Horas) AS
SELECT DNIEMP, SUM(Horas)
FROM distribucion
GROUP BY DNIEMP
```

```
CREATE VIEW Resultado(DNI, Horas) AS
    SELECT * FROM DNIHORAS
    WHERE Horas >= ALL (SELECT Horas FROM DNIHoras)
```

```
CREATE VIEW Resultado(DNI, Horas) AS
    SELECT * FROM DNIHORAS
    WHERE Horas = (SELECT MAX(HORAS) FROM DNIHoras)
```



## Consultas Jerárquicas

---

- ▶ En muchas ocasiones necesitamos recuperar información de la base de datos que esta recursivamente relacionada. Jefes-Empleados, Padres-Hijos, Piezas-Componentes...
- ▶ Habitualmente estas relaciones se establecen mediante atributos de una misma tabla.
- ▶ La cláusula **CONNECT BY** permite recuperar filas basándonos en su orden jerárquico.

```
[START WITH start_expression]  
CONNECT BY  
{PRIOR parent_expr = child_expr |  
  PRIOR child_expr = parent_expr}
```

## Consultas Jerárquicas

---

- ▶ **START WITH** : Especifica la fila raíz de la jerarquía
- ▶ **CONNECT BY**: permite establecer si el recorrido jerárquico será ascendente o descendente en base a las columnas en la relacionadas
  - ▶ Recorrido descendente  
**PRIOR parent\_expr = child\_expr**
  - ▶ Recorrido ascendente  
**PRIOR child\_expr = parent\_expr**

# Consultas Jerárquicas

idFamilia	Nombre	IdPadre
1	Paula	
2	Elena	1
3	Margarita	1
4	Eduardo	3
5	Carolina	2
6	Marta	3
7	Alberto	6
8	Sandra	6

## Descendientes de Margarita

```
Select * from familia
START WITH IdFamilia=3
CONNECT BY idPadre= PRIOR idFamilia
```

IDFAMILIA	NOMBRE	IDPADRE
3	Margarita	1
4	Eduardo	3
6	Marta	3
7	Alberto	6
8	Sandra	6

# Consultas Jerárquicas

idFamilia	Nombre	IdPadre
1	Paula	
2	Elena	1
3	Margarita	1
4	Eduardo	3
5	Carolina	2
6	Marta	3
7	Alberto	6
8	Sandra	6

## Ascendientes de Marta

```
Select * from familia
START WITH IdFamilia=7
CONNECT BY PRIOR idPadre = idFamilia
```

IDFAMILIA	NOMBRE	IDPADRE
7	Alberto	6
6	Marta	3
3	Margarita	1
1	Paula	

## Consultas Jerárquicas

---

- ▶ Se puede explícitamente mostrar el nivel de una fila en la jerarquía con el uso de la pseudo columna LEVEL.

```
Select Nombre, Level from familia  
START WITH IdFamilia=3  
CONNECT BY idPadre= PRIOR idFamilia
```

NOMBRE	LEVEL
-----	
Margarita	1
Eduardo	2
Marta	2
Alberto	3
Sandra	3

## Consultas Jerárquicas

---

- ▶ Se puede explícitamente mostrar el nivel de una fila en la jerarquía con el uso de la pseudo columna LEVEL.

```
Select Lpad(nombre,Length(Nombre)+LEVEL * 10 - 10,'-')  
from familia  
START WITH IdFamilia=3  
CONNECT BY idPadre= PRIOR idFamilia
```

```
Margarita  
-----Eduardo  
-----Marta  
-----Alberto  
-----Sandra
```