

SQL (II)

Sentencias DML de consulta de datos

Bases de Datos

Curso 2018-2019

Jesús Correás – jcorreas@ucm.es

**Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid**

DML – Consulta de datos

- Las consultas de datos se hacen mediante la **sentencia SELECT**.
- La sintaxis básica es:

```
SELECT C j1 , ..., C jr  
FROM tabla1, tabla2, ..., tablak  
WHERE condicion
```
- La **cláusula SELECT** especifica las columnas (o expresiones) que deben aparecer en el resultado (equivalente a la **proyección** π del álgebra relacional).
- La **cláusula FROM** especifica las tablas de las que obtiene la consulta (el **producto cartesiano**).
- La **cláusula WHERE** es opcional y especifica las condiciones de selección (equivalente a la **selección** σ).
 - Si no se incluye cláusula **WHERE** se seleccionan **todas las filas**.

DML – Consulta de datos

- **Ejemplos:**

codigoPr	DNI	horas
PR1	27347234T	20
PR2	27347234T	25
PR3	27347234T	25

- ▶ Seleccionar los códigos de proyecto del empleado con dni 27347234T:
**SELECT codigoPr FROM Distribucion
WHERE DNI = '27347234T';**
- ▶ 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,DNI,horas FROM Distribucion
WHERE horas >= 15 AND horas <= 25;**
- ▶ **¿Qué devuelve la consulta siguiente?**
SELECT DNI FROM Distribucion WHERE horas > 20;

DML – Consulta de datos

- Las consultas SQL trabajan con **multiconjuntos** en lugar de conjuntos: **se pueden repetir valores.**

```
SELECT DNI FROM Distribucion WHERE horas > 20;
```

```
DNI
```

```
-----
```

```
27347234T
```

```
27347234T
```

- Para trabajar con conjuntos (eliminar tuplas duplicadas) se utiliza la cláusula **DISTINCT**:

```
SELECT DISTINCT DNI FROM Distribucion WHERE horas > 20;
```

```
DNI
```

```
-----
```

```
27347234T
```

DML – Consulta de datos

- Para seleccionar **todas las columnas** de las tablas incluidas en la cláusula **FROM** se utiliza *****:

```
SELECT * FROM Distribucion WHERE horas > 20;
```

- Los atributos de la cláusula **SELECT** pueden ser **expresiones** (aritméticas, concatenación de cadenas de caracteres, funciones):

```
SELECT codigoPr || ' - ' || DNI, Horas/8  
FROM distribucion;
```

- Se pueden cambiar el **nombre de las columnas** en el resultado:

```
SELECT DNI "DNI del empleado" FROM distribucion;
```

- En algunos casos necesitamos ejecutar una consulta que no requiere ninguna tabla. Podemos utilizar una tabla especial **DUAL** para evaluar expresiones.

- ▶ Es una tabla con una sola columna **DUMMY** y una sola fila
- ▶ **Ejemplo:** `SELECT SYSDATE, 2*3, sqrt(2) FROM DUAL;`

DML – Evaluación de una consulta

- La evaluación de una sentencia **SELECT** básica se puede ver como la ejecución de los siguientes pasos:
 1. Cálculo del **producto cartesiano** de las tablas de la cláusula FROM.
 2. **Eliminación de las filas** que no cumplen la condición de la cláusula **WHERE**.
 3. **Eliminación de las columnas** que no aparecen en la lista de la cláusula **SELECT**.
 4. Si se especifica **DISTINCT**, **eliminación de 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.

DML – Condición de la cláusula WHERE

- La condición de la cláusula **WHERE** debe ser una **expresión lógica** formada por
 - ▶ operadores lógicos **AND**, **OR**, **NOT** y
 - ▶ condiciones booleanas simples.
- Las **condiciones booleanas simples** son:
 - ▶ Operadores de comparación: **<**, **>**, **<=**, **>=**, **=**, **!=**
 - ▶ Comprobación de valor nulo: **atributo IS [NOT] NULL**
 - ▶ Pertenencia a un conjunto de valores:
atributo [NOT] IN (v1, ..., vn)
 - ▶ Pertenencia a un rango: **atributo [NOT] BETWEEN v1 AND v2**
 - ▶ Similitud entre cadenas de caracteres:
atributo [NOT] LIKE 'patrón'
patrón es una cadena con caracteres comodín:
 - ★ El carácter **_** representa un carácter cualquiera.
 - ★ El carácter **%** representa una cadena de caracteres cualquiera.

DML – Condición de la cláusula WHERE

● Ejemplos:

- ▶ Datos de los empleados cuyo nombre comienza por 'Te':
SELECT * FROM Emp WHERE nombre LIKE 'Te%';
- ▶ Nombre de los empleados que tienen una "a" en el tercer carácter del nombre:
SELECT nombre FROM Emp WHERE nombre LIKE ' _ _a%';
- ▶ Datos de los empleados que no tienen teléfono:
SELECT * FROM Emp WHERE telefono IS NULL;
- ▶ Datos de los empleados con salario en un rango:
**SELECT * FROM Emp
WHERE salario BETWEEN 25000 AND 25500;**
- ▶ Datos de los empleados cuyo identificador está en un conjunto:
SELECT * FROM Dept WHERE DeptId IN (32,44,99);

DML – Ordenación de los resultados de una consulta

- La cláusula **ORDER BY** permite establecer el orden de presentación de las filas resultado de una consulta **SELECT**.
- Se pueden especificar varias columnas, e incluso expresiones.
- Debe ser la última cláusula de la sentencia **SELECT**.

- **Ejemplos:**

- ▶ Código de proyecto, DNI y horas trabajadas de los Empleados que trabajan más de 10 horas en algún proyecto, ordenados por horas.
**SELECT CodPr, DNI, horas FROM distribucion
WHERE horas > 10 ORDER BY horas;**
- ▶ Ordenados por horas de forma descendente:
**SELECT CodPr, DNI, horas FROM distribucion
WHERE horas > 10 ORDER BY horas DESC;**
- ▶ Ordenados por código de proyecto de forma ascendente, y dentro de cada proyecto, por horas de forma descendente:
**SELECT CodPr, DNI, horas FROM distribucion
WHERE horas > 10 ORDER BY CodPr ASC, horas DESC;**

DML – Funciones

- Se pueden utilizar expresiones y funciones en las cláusulas **SELECT** y **WHERE**.
- Existe gran número de funciones para **operaciones matemáticas**, de **cadenas de caracteres** y de **tratamiento de fechas**.

Funciones matemáticas	
ROUND (n, d)	Redondea n al valor más cercano con d decimales.
TRUNC (n, d)	Trunca n al número de decimales d indicado.
MOD (n1, n2)	Devuelve el resto resultante de dividir $n1$ entre $n2$.
POWER (v, exp)	Calcula v^{exp} : la potencia exp de v .
SQRT (n)	Raíz cuadrada de n .
SIGN (n)	Devuelve 1 si n es positivo, 0 si es cero y -1 si es negativo.
ABS (n)	Calcula el valor absoluto de n .
EXP (n)	e^n .

DML – Funciones sobre cadenas

Funciones sobre cadenas	
LOWER (texto)	Convierte el texto a minúsculas.
UPPER (texto)	Convierte el texto a mayúsculas.
INITCAP (texto)	Pone la primera letra de cada palabra en mayúsculas.
RTRIM (texto)	Elimina los espacios a la derecha de texto.
LTRIM (texto)	Elimina los espacios a la izquierda.
TRIM (texto)	Elimina espacios a izq. y der. y los espacios dobles.
TRIM (c FROM s)	Elimina de s los caracteres en c.
SUBSTR (s, n[, m])	Obtiene m caracteres de s a partir del n-ésimo.
LENGTH (texto)	Obtiene el tamaño de texto.
REVERSE (texto)	Da la vuelta a texto.

Otras funciones sobre cadenas:

- **INSTR**(texto, str [, ini [, num]]): busca str en un texto.
- **REPLACE**(texto, str1, [str2]): Reemplaza str1 por str2 en texto.
- **LPAD**(texto, max [, c]): Rellena texto a la izquierda (LPAD) con el carácter c para ocupar la anchura indicada. **RPAD** es igual, por la derecha.

DML – Funciones sobre valores nulos

Funciones sobre valores nulos	
NVL(v, s)	Si v es NULL, devuelve s; si no, devuelve v.
NVL2(valor, s1, s2)	devuelve s1 si v no es nulo. Si v es nulo devuelve s2
COALESCE(listaExpr)	Devuelve la primera expresión no nula.

Ejemplos:

```
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;
```

DML – Funciones de fecha

Funciones de fecha	
SYSDATE	Obtiene la fecha y hora actuales.
ADD_MONTHS (<i>fecha</i> , <i>n</i>)	Añade a <i>fecha</i> el número de meses <i>n</i> .
MONTHS_BETWEEN (<i>f1</i> , <i>f2</i>)	Obtiene la diferencia en meses entre dos fechas.
NEXT_DAY (<i>fecha</i> , <i>D</i>)	Devuelve la fecha correspondiente al siguiente <i>D</i> después de <i>fecha</i> . <i>D</i> debe ser un día de la semana (en el idioma de la sesión: 'LUNES', 'MARTES' ...)
LAST_DAY (<i>fecha</i>)	Obtiene el último día del mes indicado en <i>fecha</i> .
EXTRACT (<i>v</i> FROM <i>fecha</i>)	Extrae el componente <i>v</i> de <i>fecha</i> . <i>v</i> puede ser day, month, year, minute...
GREATEST (<i>f1</i> , <i>f2</i> , ...)	Devuelve la fecha más moderna de la lista.
LEAST (<i>f1</i> , <i>f2</i> , ...)	Devuelve la fecha más antigua de la lista.

DML – Conversión de tipos de datos

- Oracle intenta convertir datos automáticamente para que la expresión final tenga sentido.
- 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.
```

- Existen dos funciones de **conversión explícita** entre texto y número:
 - ▶ **TO_NUMBER(texto, [fmt,] [nlsparams])**
 - ▶ **TO_CHAR(exp, [fmt,] [nlsparams])**
- El formato es un texto con caracteres que representan cada dígito:

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)

DML – Conversión de tipos de datos

- **Conversión explícita** entre **texto y fecha**:

- ▶ `TO_DATE(texto, [fmt,] [nlsparms])`
- ▶ `TO_CHAR(fecha, [fmt,] [nlsparms])`

- El formato es un texto con caracteres que representan cada dígito:

YY	Año en formato de dos cifras	Q	Semestre
YYYY	Año en formato de cuatro cifras	WW	Semana del año
MM	Mes en formato de dos cifras	AM	Indicador AM
MON	Las tres primeras letras del mes	PM	Indicador PM
MONTH	Nombre completo del mes	HH12	Hora de 1 a 12
DY	Día de la semana en tres letras	HH24	Hora de 0 a 23
DAY	Día completo de la semana	MI	Minutos 0 a 59
D	Día de la semana del 1 al 7	SS	Segundos 0 a 59
DD	Día en formato de dos cifras	SSSS	Segundos desde medianoche
DDD	Día del año		

- Los separadores entre elementos de fecha: / . , : ; '.

- **Ejemplo:**

```
SELECT TO_CHAR(SYSDATE, 'DD/MONTH/YYYY, DAY HH:MI:SS')
FROM DUAL      -- 21/NOVIEMBRE/2016, LUNES 08:35:15
```

DML – Consultas con operaciones sobre conjuntos

- En SQL se pueden combinar los resultados de distintas sentencias **SELECT** utilizando los operadores de teoría de conjuntos **UNION**, **INTERSECT** y **MINUS**.
- Las columnas de las dos consultas **deben ser similares: mismo número y tipo** (el nombre puede ser diferente).
- **Se eliminan las filas duplicadas** para poder realizar las operaciones sobre conjuntos.
 - ▶ Para mostrar todas las filas: **UNION ALL**.

- **Ejemplos:**

```
create table mitabla (c1 integer, c2 varchar2(20));  
insert into mitabla values (1, 'dato uno');  
insert into mitabla values (2, 'dato dos');  
insert into mitabla values (3, 'dato tres');
```

```
SELECT * FROM mitabla UNION SELECT 2*c1, c2 || ' 2' FROM mitabla;  
SELECT c1 FROM mitabla INTERSECT SELECT 2*c1 FROM mitabla;
```


DML – Reuniones de tablas (JOIN)

- En SQL existen varios tipos de **reuniones de tablas**, de forma similar al álgebra relacional.
- Las reuniones se especifican en la cláusula **FROM**:
 - ▶ *tabla1* **CROSS JOIN** *tabla2* (equivalente a *tabla1, tabla2*)
 - ▶ *tabla1* **NATURAL JOIN** *tabla2*
 - ▶ *tabla1* **JOIN** *tabla2* **USING** (*col1*, ..., *colk*)
 - ▶ *tabla1* **JOIN** *tabla2* **ON** *joinCond*
 - ▶ *tabla1* **LEFT OUTER JOIN** *tabla2* **ON** *joinCond*
 - ▶ *tabla1* **RIGHT OUTER JOIN** *tabla2* **ON** *joinCond*
 - ▶ *tabla1* **FULL OUTER JOIN** *tabla2* **ON** *joinCond*
- *joinCond* contiene las condiciones para combinar *tabla1* con *tabla2*.
- La condición de la cláusula **WHERE** puede contener nombres de columnas de las dos tablas.
- Si varias columnas tienen el mismo nombre, se puede utilizar *tabla.col* o alias de tablas.
- Varios *joins* pueden combinar más de dos tablas en la misma sentencia.

DML – CROSS JOIN

- Hay dos formas de combinar dos tablas con un *cross join*:

```
SELECT c1...cm FROM tabla1,tabla2 WHERE cond;
```

```
SELECT c1...cm FROM tabla1 CROSS JOIN tabla2 WHERE cond;
```

- Corresponde con el **producto cartesiano del álgebra relacional**.
- Se pueden seleccionar subconjuntos de las filas del producto cartesiano con condiciones en la cláusula **WHERE**.
- **Ejemplo:** Las siguientes consultas son idénticas y producen el producto cartesiano **Emp**×**Distribucion**:

```
SELECT Nombre, DNI, CodPr FROM Emp CROSS JOIN Distribucion;
```

```
SELECT Nombre, DNI, CodPr FROM Emp, distribucion;
```

DML – CROSS JOIN

- Con **CROSS JOIN** se puede implementar la **reunión condicional**:

```
SELECT Nombre, DNI, CodPr FROM Emp, Distribucion
WHERE distribucion.DNI = Emp.DNI
```

Tabla Emp

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

Tabla Proyecto

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla Distribucion

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla Dpto

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- ¿Cómo sería la consulta siguiente?:

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

DML – CROSS JOIN

- Se pueden renombrar tablas con alias en la cláusula **SELECT**, a continuación del nombre de la tabla.
- Si renombra una tabla, **el alias debe utilizarse en lugar del nombre de la tabla** para referirse a las columnas.

```
SELECT Nombre, e.DNI, a.Horas FROM Emp e, Distribucion d  
WHERE e.DNI = d.DNI;
```

- Es obligatorio utilizar alias cuando se hace la reunión de una tabla consigo misma en la misma consulta.
- **Ejemplo:** “Nombre de los empleados y de los directores de los proyectos en los que trabajan”.

DML – CROSS JOIN

- Se pueden renombrar tablas con alias en la cláusula **SELECT**, a continuación del nombre de la tabla.
- Si renombra una tabla, **el alias debe utilizarse en lugar del nombre de la tabla** para referirse a las columnas.

```
SELECT Nombre, e.DNI, a.Horas FROM Emp e, Distribucion d
WHERE e.DNI = d.DNI;
```

- Es obligatorio utilizar alias cuando se hace la reunión de una tabla consigo misma en la misma consulta.
- **Ejemplo:** “Nombre de los empleados y de los directores de los proyectos en los que trabajan”.

```
SELECT trbj.nombre Trabajador, dir.nombre Director, p.CodPr
FROM Emp trbj, Distribucion d, Emp dir, Proyecto p
WHERE d.DNI = trbj.DNI
      AND p.CodPr = d.CodPr
      AND dir.DNI = p.DNIDir;
```

DML – NATURAL JOIN / JOIN USING

- La **reunión natural** \bowtie **del álgebra relacional** está implementada de dos formas:

```
SELECT c1...cm FROM tabla1 NATURAL JOIN tabla2 WHERE cond;
```

- ▶ Realiza la reunión por igualdad de **todas las columnas con el mismo nombre** en ambas tablas.
- ▶ No repite las columnas con el mismo nombre (como la reunión natural del álgebra relacional).
- ▶ Las columnas con el mismo nombre pierden el prefijo de la tabla.

```
SELECT c1...cm FROM tbl1 JOIN tbl2 USING (cj,...,ck)  
WHERE cond;
```

- ▶ Realiza la reunión por igualdad de **las columnas con el mismo nombre en ambas tablas que aparecen en la lista USING.**
- ▶ No repite las columnas de la cláusula **USING**.
- ▶ Las columnas de la cláusula **USING** pierden el prefijo de la tabla. Las columnas restantes aparecen como en las tablas originales.

DML – NATURAL JOIN / JOIN USING

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Ejemplo:** “Nombre de los empleados y código de los proyectos en que trabajan más de 22 horas”.

```
SELECT Nombre, CodPr FROM Emp NATURAL JOIN Distribucion
WHERE horas > 22;
```

DML – NATURAL JOIN / JOIN USING

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Otro ejemplo:** “Nombre de los empleados y descripción de los proyectos en los que trabajan”.

```
SELECT Nombre, Descr
FROM Emp NATURAL JOIN Distribucion NATURAL JOIN Proyecto;
```


DML – NATURAL JOIN / JOIN USING

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Otro ejemplo más:** “DNI de los empleados y nombre de los departamentos a los que pertenecen”.

```
SELECT Emp.DNI, Dpto.Nombre  
FROM Emp JOIN Dpto USING (CodDp);
```

DML – NATURAL JOIN / JOIN USING

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Otro ejemplo más:** “DNI de los empleados y nombre de los departamentos a los que pertenecen”.

```
SELECT Emp.DNI, Dpto.Nombre  
FROM Emp JOIN Dpto USING (CodDp);
```

- ¿Se podría hacer esta consulta con NATURAL JOIN?

DML – JOIN ON

- La **reunión condicional del álgebra relacional** también se puede representar mediante una reunión **JOIN ON**:

```
SELECT c1...cm FROM tabla1 JOIN tabla2 ON rels WHERE cond;
```

- ▶ Utiliza la condición **rels** para relacionar columnas de ambas tablas.
- ▶ **rels** puede contener **otros operadores**: {<, >, <=, >=, !=, =}.
- ▶ Es **similar a CROSS JOIN**, pero separa las **condiciones de relación** de las **condiciones de selección** (cláusula **WHERE**).
- ▶ A diferencia de **NATURAL JOIN**, repite las columnas con el mismo nombre en ambas tablas.

- **Ejemplos:**

```
SELECT Nombre,CodPr FROM Emp JOIN Distribucion  
ON Emp.DNI = Distribucion.DNI WHERE horas > 22;
```

```
SELECT Nombre,CodPr FROM Emp JOIN Proyecto  
ON Emp.DNI = Proyecto.DNIDir;
```

```
SELECT Nombre,CodPr FROM Emp e JOIN Proyecto p  
ON e.DNI = p.DNIDir;
```

DML – JOIN ON

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- Se pueden **combinar** varios tipos de reuniones.

- **¿Qué devuelve la siguiente consulta?**

```
SELECT e.Nombre, CodPr, e2.Nombre FROM Emp e
NATURAL JOIN Distribucion
NATURAL JOIN Proyecto p
JOIN Emp e2 ON p.DNIDir=e2.DNI;
```

DML – LEFT / RIGHT / FULL OUTER JOINs

- Las reuniones vistas hasta ahora son **reuniones internas**: producen filas resultado de combinar **filas existentes** en las tablas de origen.
- También se pueden utilizar **reuniones externas**:
- **LEFT OUTER JOIN**: implementa la **reunión externa izquierda** \Join .
 - ▶ Obtiene todas las filas de la primera tabla, combinando con las filas correspondientes de la segunda tabla, o rellenando con valores nulos si no hay correspondencia.
- **RIGHT OUTER JOIN**: implementa la **reunión externa derecha** \Join .
 - ▶ Obtiene todas las filas de la segunda tabla, combinando con las filas correspondientes de la primera tabla, o rellenando con valores nulos si no hay correspondencia.
- **FULL OUTER JOIN**: implementa la **reunión externa completa** \Join .
 - ▶ Obtiene todas las filas de las dos tablas, rellenando con nulos si no hay correspondencia.
- En los tres casos **se debe utilizar una cláusula ON o USING para indicar las columnas de la reunión.**

DML – LEFT / RIGHT / FULL OUTER JOINs

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Ejemplos:** Datos de todos los empleados, con la información de los que son directores de proyectos:

```
SELECT * FROM Emp e LEFT OUTER JOIN Proyecto p
ON p.DNIDir=e.DNI;
```

DML – Funciones de agregación

- En SQL también se pueden realizar consultas en las que se **agrupan** las filas resultado.
- Las **funciones de agregación** permiten calcular resultados sobre **grupos de filas** de una consulta **SELECT**:
 - ▶ **COUNT ([DISTINCT] col | expr)** : devuelve el **numero de valores** de la columna *col* (o la expresión *expr*). No incluye las filas con valor **NULO**.
 - ▶ **SUM ([DISTINCT] col | expr)** : devuelve la suma de todos los valores de la columna *col* (numérica) o la expresión *expr*.
 - ▶ **AVG ([DISTINCT] col | expr)** : Calcula el valor medio de los valores de la columna *col* (numérica) o la expresión *expr*.
 - ▶ **MAX (col | expr)** : Devuelve el valor máximo de la columna o expresión.
MIN Devuelve el valor mínimo.
- En algunos casos, se pueden calcular los valores agregados de los datos que sean **distintos** con la palabra **DISTINCT**.
- Se puede utilizar **COUNT (*)** para contar todas las filas, incluyendo duplicados y nulos.

DML – Funciones de agregación

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Ejemplo:** Cálculo de la dedicación total de empleados a proyectos: número de asignaciones de empleados a proyectos, número de horas totales, dedicación media:

```
SELECT count (Horas) , sum (Horas) , avg (Horas) FROM
Distribucion;
```

- **¿Cuál sería el resultado si utilizamos DISTINCT?**

DML – Agrupaciones: Sentencia `SELECT` extendida

- Las funciones de agregación consideran las filas de una consulta como **un grupo** sobre el que se calcula una sola fila resultado.
- Esta noción se puede **extender a múltiples grupos**:

```
SELECT [DISTINCT] listaExpr FROM tablas WHERE condW  
GROUP BY colsGrupos  
HAVING condiciónGrupos  
[ORDER BY lista];
```

- `GROUP BY` produce tantas filas como valores diferentes de *colsGrupos*
- Si se omite `GROUP BY`, toda la tabla es un único grupo.
- La cláusula `HAVING` selecciona **qué grupos** aparecen en el resultado.
- En las cláusulas `SELECT` y `HAVING` solo pueden aparecer expresiones **disponibles para las filas de grupo**:
 - ▶ **Nombres de columnas**: solo aquellas que también aparezcan en la cláusula `GROUP BY`.
 - ▶ Funciones de agregación.

DML – Evaluación de una consulta extendida

- Los pasos que se realizan para la ejecución de una consulta extendida son:
 1. Se seleccionan las filas deseadas de las tablas de la cláusula **FROM** utilizando la condición de la cláusula **WHERE**.
 2. Se establecen los grupos indicados en la cláusula **GROUP BY**.
 3. Se calculan los valores de las funciones de agregación (**COUNT**, **SUM**, **AVG**, ...) para cada grupo. **Se genera una sola fila por cada grupo.**
 4. Se filtran **las filas de grupo** que cumplen la cláusula **HAVING**.
 5. El resultado se ordena como se indique en la cláusula **ORDER BY**.
- Esta estrategia es ineficiente y no es realmente la que utilizan los SGBD, pero sirve para entender el significado de las consultas extendidas.

DML – Sentencia `SELECT` extendida

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Ejemplo:** DNI y número total de horas de los empleados que trabajan en 2 o más proyectos diferentes, en orden decreciente de dedicación.

DML – Sentencia SELECT extendida

Tabla Emp

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

Tabla Proyecto

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla Distribucion

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla Dpto

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Ejemplo:** DNI y número total de horas de los empleados que trabajan en 2 o más proyectos diferentes, en orden decreciente de dedicación.

```
SELECT DNI, SUM(Horas) FROM distribucion
GROUP BY DNI HAVING COUNT(*) >= 2
ORDER BY SUM(Horas) DESC;
```

DML – Sentencia `SELECT` extendida

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Ejemplo:** Descripción de los proyectos en los que trabajan 2 o más empleados.

DML – Sentencia `SELECT` extendida

Tabla **Emp**

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

Tabla **Proyecto**

CodPr	DNIDir	Descr
PR1	27347234T	Ventas
PR2	37562365F	Personal
PR3	37562365F	Logística

Tabla **Distribucion**

CodPr	DNI	horas
PR1	27347234T	20
PR3	27347234T	25
PR2	27347234T	25
PR3	37562365F	45
PR1	37562365F	10
PR1	34126455Y	10

Tabla **Dpto**

CodDp	Nombre
SMP	Servicios Múltiples
RH	Recursos Humanos

- **Ejemplo:** Descripción de los proyectos en los que trabajan 2 o más empleados.

```
SELECT Descr, count(*)  
FROM Distribucion JOIN Proyecto USING (CodPr)  
GROUP BY codPr, Descr  
HAVING COUNT(*) >= 2;
```

DML – Consultas anidadas

- Es posible utilizar una consulta dentro de otra consulta: son **consultas anidadas**.
- A la consulta interna se le denomina **subconsulta**.
- Normalmente se incluyen subconsultas en la cláusula **WHERE**, pero también pueden estar en las cláusulas **FROM** o **HAVING**.
- Normalmente se utiliza para comprobar la **pertenencia** a un conjunto, su **cardinalidad** o hacer **comparaciones**.
- Si la subconsulta devuelve **una sola fila con una sola columna**, se pueden hacer **comparaciones directamente**: **>, <, >=, <=, != y =**.
- **Ejemplo:** Empleados con salario inferior a la media:

```
SELECT Nombre, Salario FROM Emp
WHERE Salario < (SELECT avg(Salario) FROM Emp);
```

DML – Consultas anidadas

- También se pueden utilizar operadores de comparación $op \in \{>, <, >=, <=, !=, =\}$ con subconsultas cuando **devuelven varias filas** (de un solo valor):
 - ▶ **expr op ANY (subconsulta)**: cuando se cumple la comparación op para **alguna fila** de la subconsulta.
 - ▶ **expr op ALL (subconsulta)**: cuando se cumple la comparación op para **todas las filas** de la subconsulta.
- **Ejemplo:** Empleados que dedican más horas a alguno de sus proyectos que cualquiera de los empleados del proyecto PR1:

```
SELECT DNI FROM Distribucion WHERE Horas > ALL  
(SELECT Horas FROM Distribucion WHERE CodPr='PR1');
```


DML – Consultas anidadas

- Otros operadores sobre subconsultas:
 - ▶ El operador **expr [NOT] IN (subconsulta)** comprueba la **pertenencia** o no de *expr* al conjunto resultante de una subconsulta.
 - ▶ El operador **unario EXISTS (subconsulta)** devuelve **cierto** si la subconsulta devuelve algún resultado.
 - ▶ **NOT EXISTS (subconsulta)** devuelve **cierto** si la subconsulta no devuelve ningún resultado.
 - ▶ En estos tres casos la subconsulta puede devolver filas con más de una columna.
- Las consultas anidadas pueden estar **correlacionadas**: cuando la subconsulta **depende de cada fila de la consulta exterior**. Por ejemplo:
 - ▶ Empleados que dirigen y trabajan en un mismo proyecto:

```
SELECT DNIDir FROM Proyectos p WHERE CodPr IN  
(SELECT CodPr FROM Distribucion d  
WHERE d.DNI=p.DNIDir);
```

DML – Consultas anidadas

- Las **subconsultas correlacionadas** se evalúan **para cada fila** de la consulta externa (consulta principal).

- **Ejemplo:** Empleados que no trabajan en ningún proyecto.

```
SELECT Nombre, DNI FROM Emp E WHERE NOT EXISTS  
(SELECT CodPr FROM distribucion D WHERE D.DNI=E.DNI);
```

- ▶ Se evalúa la subconsulta para cada fila de la tabla de empleados.
 - ▶ Si la subconsulta NO produce ningún resultado, entonces la fila aparece en la respuesta.
 - ▶ En caso contrario la fila no aparecerá.
- **Otro ejemplo:** Empleados asignados a un proyecto más horas que la media de empleados asignados a ese proyecto.

DML – Consultas anidadas

- Las **subconsultas correlacionadas** se evalúan **para cada fila** de la consulta externa (consulta principal).

- Ejemplo:** Empleados que no trabajan en ningún proyecto.

```
SELECT Nombre, DNI FROM Emp E WHERE NOT EXISTS  
(SELECT CodPr FROM distribucion D WHERE D.DNI=E.DNI);
```

- ▶ Se evalúa la subconsulta para cada fila de la tabla de empleados.
 - ▶ Si la subconsulta NO produce ningún resultado, entonces la fila aparece en la respuesta.
 - ▶ En caso contrario la fila no aparecerá.
- Otro ejemplo:** Empleados asignados a un proyecto más horas que la media de empleados asignados a ese proyecto.

```
SELECT e.Nombre, d.CodPr, d.horas  
FROM Distribucion d join Emp e on e.DNI = d.DNI  
where d.Horas > (select AVG(d2.Horas)  
                  from Distribucion d2  
                  where d2.CodPr = d.CodPr);
```

DDL – Vistas

- Una **vista** es una **tabla virtual**: una relación que no forma parte del modelo lógico pero que aparece como tal ante el usuario.
- El SGBD solo guarda la definición de la vista, no el resultado, que **se calcula cada vez que se utiliza**.
- A (casi) todos los efectos una vista es como una tabla.
- Para definir una vista se utiliza la siguiente sintaxis:

```
CREATE VIEW vista [(listaColumnas)] AS consulta  
[WITH READ ONLY | WITH CHECK OPTION];
```

- ▶ **WITH READ ONLY**: La vista no permite modificaciones de los datos.
- ▶ **WITH CHECK OPTION**: Permite la inserción y actualización de filas que cumplan **las condiciones de la consulta**.
- ▶ Por defecto, las vistas son actualizables: se modifican **los datos de las tablas subyacentes** (con restricciones).

DDL – Vistas

- Nombres de los empleados que trabajan para proyectos más horas que la media.
- Podemos resolverlo creando las siguientes vistas para la consulta:

- ▶ DNI y total de horas que trabaja cada empleado:

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

- ▶ Nombre y total de horas de cada empleado:

```
CREATE VIEW NombreHoras (Nombre, Horas) AS  
SELECT nombre, Horas FROM Emp NATURAL JOIN DNIHoras;
```

- ▶ Promedio de horas trabajadas por cada empleado:

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

- La consulta final es la siguiente:

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

DML – Vistas actualizables

- Las vistas actualizables deben cumplir VARIAS restricciones. Algunas de ellas son:
 - ▶ Todas las columnas obligatorias deben aparecer en la definición de la vista.
 - ▶ La consulta no puede contener operadores de conjunto (**UNION**, **MINUS** o **INTERSECT**).
 - ▶ No se puede utilizar **DISTINCT**, **GROUP BY** y **ORDER BY**.
 - ▶ No se pueden utilizar funciones de agregación
- Además, algunas vistas que contienen reuniones pueden tener solo algunas columnas actualizables.