Ud 4 :
Realización de Consultas

IES Maestre de Cva.
Isabel Guerrero

Objetivos

- Utilizar herramientas y sentencias para realizar consultas
- Identificar y crear consultas simples sobre una tabla
- Identificar y crear consultas que generan valores resumen
- Crear consultas con composiciones internas y externas
- Crear subconsultas
- Valorar las ventajas e inconvenientes de las distintas opciones válidas para realizar una consulta

Pag. :2

Contenidos

- La sentencia SELECT
- Consultas básicas, filtros, ordenación.
- Consultas resumen.
- Subconsultas.
- Consultas multitablas.
- Composiciones internas y externas.

Pag. :3

El lenguaje DML

- **Lenguaje de Manipulación de Datos (DML)** es parte del lenguaje SQL encargado de obtener y actualizar la información de las bases de datos.

- Las sentencias del **DML** son las siguientes:

  - La sentencia **SELECT**, que se utiliza para extraer información de la base de datos, ya sea de una tabla o de varias.
  - La sentencia **INSERT**: inserta uno o varios registros en alguna tabla.
  - La sentencia **DELETE** : borra registros de una tabla.
  - La sentencia **UPDATE:** modifica registros de una tabla.

Pag. :4

El lenguaje DML

- Cualquier ejecución de un comando en un SGBD se denomina **CONSULTA**, término derivado del anglosajón **QUERY**.
- Este término debe ser entendido más que como una *consulta* de información, como una *orden,* es decir, las QUERYS o CONSULTAS no son solo **SELECT**, sino también cualquier sentencia de tipo **UPDATE, INSERT, CREATE, DROP,** entendidas todas ellas como peticiones al SGBD para realizar una operación determinada.

Pag. :5

La sentencia SELECT

- Las **consultas** obtienen información filtrada de una o múltiples tablas, usando las relaciones entre dichas tablas e incluso tablas virtuales creadas a partir de una consulta.
- Es posible ejecutar sentencias muy sencillas, como por ejemplo: seleccionar todos los campos y mostrar todos los registros de la tabla empleados

```
SELECT * FROM empleados;
```

Pag. :6

La sentencia SELECT

- Y consultas más complicadas como la siguiente: obtener el total de los pedidos de los clientes de una tienda

```
SELECT NOMBRECLIENTE,TOTAL.CANTIDAD
FROM CLIENTES ,PEDIDOS ,
  (SELECT SUM(CANTIDAD*PRECIOUNIDAD) AS CANTIDAD,NUMEROPEDIDO
          FROM DETALLEPEDIDOS
          GROUP BY NUMEROPEDIDO) TOTAL )
WHERE
          CLIENTES.NUMEROCLIENTE=PEDIDOS.NUMEROCLIENTE
          AND PEDIDOS.NUMEROPEDIDO=TOT.NUMEROPEDIDO
ORDER BY CANTIDAD;
```

Pag. :7

SQL*Plus (Oracle)

- **SET LINESIZE 200**: ejemplo que asigna el ancho de las líneas de visualización en SQL*Plus a 200
- **SET PAGESIZE 30:** ejemplo que asigna 30 líneas por página de visualización.
- **@fichero.sql:** ejecuta el script fichero.sql
- **CLEAR SCREEN**: Borra la pantalla.

Pag. :8

SELECT: Consultas básicas

- El formato básico para hacer una consulta es el siguiente:

```
SELECT [DISTINCT] select_expr [, select_expr] ...
    [FROM tabla [alias]]
```

- **select_expr**:

```
nombre_columna [[AS] alias] | *  | expresión
```

Pag. :9

SELECT: Consultas básicas

- **nombre_columna** indica un nombre de columna.
- *** :** Seleccionar todos los campos de una tabla.
- Se puede seleccionar una o mas columnas de una tabla, indicándoles el nombre
- **Expresión:** es una expresión algebraica o de otro tipo, por ejemplo :SUELDO+COMISION. Estará compuesta por operadores, operandos y funciones.

```
SELECT APELLIDO, SALARIO+COMISION
    FROM EMPLEADOS;
```

Pag. :10

SELECT: DISTINCT

- El parámetro opcional **DISTINCT** fuerza que solo se muestren los registros con valores distintos, o, dicho de otro modo, que suprima las repeticiones.
- **[AS] alias:** sirve para darle un nombre alternativo a la columna

```
SELECT APELLIDO, SUELDO+COMISION  AS TOTAL
    FROM EMPLEADOS;
```

- **Alias** en el nombre de las tablas:

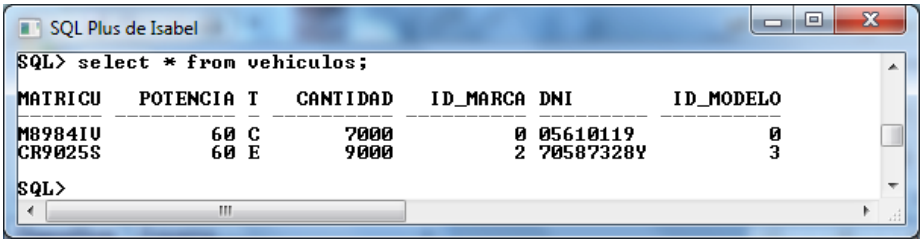
```
SELECT E.APELLIDO, E.SUELDO+E.COMISION  AS TOTAL
    FROM EMPLEADOS E;
```

Pag. :11

SELECT: Consultas básicas

- **Ejemplo Consulta :** Selecciona todas las filas y todos los campos de la tabla VEHICULOS

```
SELECT * FROM VEHICULOS;
```

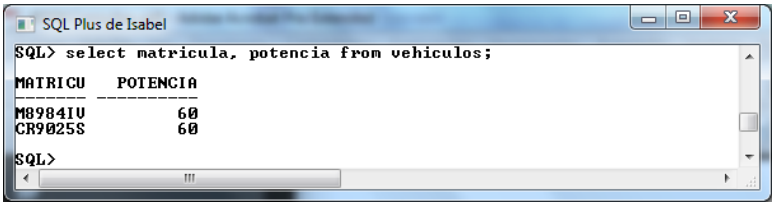


Pag. :12

SELECT: Consultas Básicas

- **Ejemplo Consulta** : Selecciona sólo las columnas matricula, y potencia de todas las filas de la tabla VEHICULOS.

SELECT MATRICULA,POTENCIA FROM VEHICULOS;

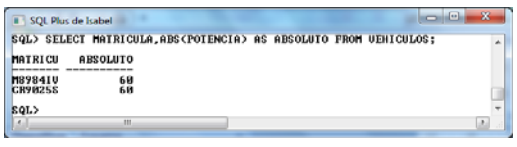


Pag. :13

SELECT: Consultas Básicas

- **Ejemplo Consulta :** Selecciona la columna matricula y hace uso de la función ABS para obtener el valor absoluto de la potencia. Selecciona todos los registros. Se le ha puesto el **alias (AS)** ABSOLUTO a la columna anterior. Cuando el alias lleva algún carácter en blanco va entre comillas dobles.

SELECT MATRICULA,ABS(POTENCIA) as "VALOR ABSOLUTO"
FROM VEHICULOS;

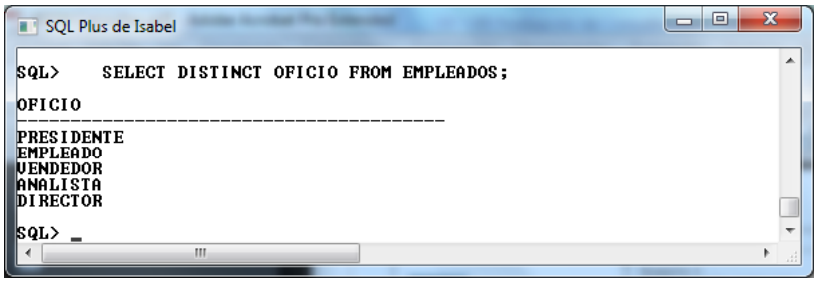


Pag. :14

SELECT: Consultas Básicas

- **Ejemplo Consulta :** Selecciona los oficios distintos de la tabla EMPLEADOS:

```
SELECT DISTINCT OFICIO FROM EMPLEADOS;
```

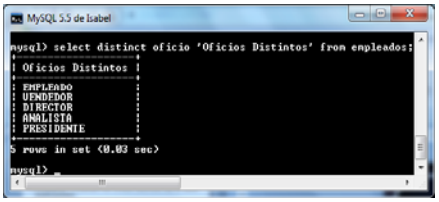


Pag. :15

SELECT: Consultas Básicas

- **Ejemplo Consulta :** Selecciona los oficios distintos de la tabla EMPLEADOS con un alias: Oficios Distintos

```
SELECT DISTINCT OFICIO AS "Oficios Distintos"
    FROM EMPLEADOS;
```

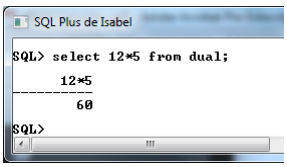


Pag. :16

SELECT: Consultas Básicas

- **Ejemplo Consulta :** Selecciona la expresión 12*5. En **MySQL** se puede poner sin FROM. En **Oracle** hay que poner la tabla ficticia **DUAL** para poder visualizar una expresión (o función) que no pertenezca a una tabla.
- **Oracle :**

```
SELECT 12*5 FROM DUAL;
```

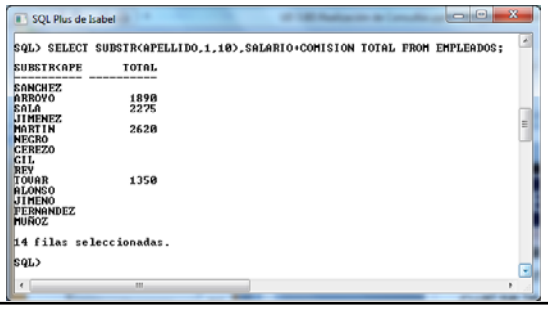


Pag. :17

SELECT: Consultas Básicas

- **Ejemplo Consulta :** Selecciona los 10 primeros caracteres de la columna APELLIDO (Función SUBSTR) y el SALARIO+COMISIÓN, a la cual se le ha asignado un alias TOTAL

```
SELECT SUBSTR(APELLIDO,1,10),SALARIO + COMISION TOTAL
    FROM EMPLEADOS;
```



Pag. :18

SELECT: IFNULL y NVL

- **Ejemplo Consulta:** En la consulta anterior vemos que en algunas filas no aparece nada en TOTAL, esto es porque en SQL cuando alguna de las expresiones o columnas que interviene en la expresión es NULL, el resultado es NULL.
- Para evitar esto se utiliza la función de convertir a 0 si es nulo.
- En **Oracle**:

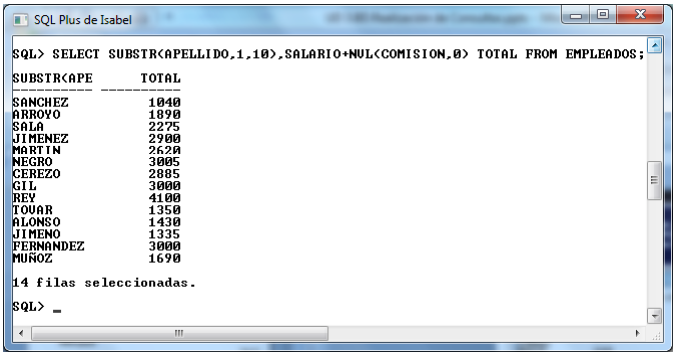
  NVL(a,0) : si a es nulo devuelve cero, si no a.

Pag. :19

SELECT: Consultas Básicas

- **Oracle:**

```
SELECT SUBSTR(APELLIDO,1,10),SALARIO + COMISION TOTAL
    FROM EMPLEADOS;
```



Pag. :20

SELECT: FILTROS WHERE

- **Filtro** es una expresión que indica la condición o condiciones que deben satisfacer los registros para ser seleccionados.
- En SQL la palabra clave para realizar filtros es la cláusula ***WHERE***.
- A continuación se añade a la sintaxis de la cláusula SELECT la sintaxis de los filtros:

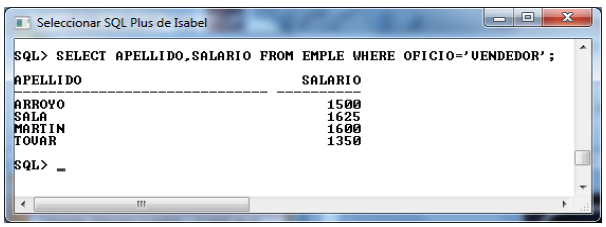
```
SELECT [DISTINCT] select_expr [,select_expr] ...
    [FROM tabla]
    [WHERE filtro]
```

Pag. :21

SELECT: FILTROS WHERE

- Ejemplos: Selecciona el apellido y el salario de los empleados cuyo oficio es VENDEDOR.

```
SELECT APELLIDO,SALARIO
    FROM EMPLE
    WHERE OFICIO='VENDEDOR';
```

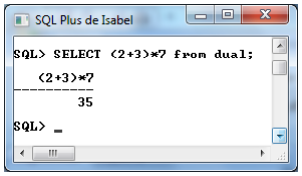


Pag. :22

SELECT: FILTROS WHERE: Expresiones para filtros

- Una **expresión**, es una combinación de operadores, operandos y funciones que producen un resultado.
- Ejemplo: expresión 1 (oracle): (2+3)*7

SQL>SELECT (2+3)*7 from dual;



Pag. :23

SELECT: FILTROS WHERE: Elementos

- Elementos que pueden formar parte de las expresiones:

  - **Operandos**
  - **Operadores aritméticos: +,-,*,/,%**
  - **Operadores relacionales :**
    =,<,>,>=,<=,<> , !=
  - **Operadores lógicos: AND, OR,NOT**
  - **Paréntesis: ()**
  - **Funciones**

Pag. :24

SELECT: FILTROS WHERE:
Elementos: Operandos y Operadores Aritméticos

- **Operandos**: Los operandos pueden ser
  - **Constantes**, por ejemplo, el número entero 3, el número real 2.3, la cadena de caracteres 'España' o la fecha '2010-01-02'. Todas las constantes numéricos ya sean reales o enteros van sin comilla simple, y cualquier otra cosa que no sea número, por ejemplo, cadenas de caracteres o fechas, van entre comillas simples.
  - **Columna:**, por ejemplo el campo *edad* o el campo *NombreMascota*.
  - Otras **expresiones**.
- **Operadores aritméticos**: +, -, *, /, %.
  - El operador **+** y el operador - se utilizan para sumar o restar dos operandos (binario) o para poner el signo positivo o negativo a un operando (unario).
  - El operador ***** : es la multiplicación de dos operandos
  - El operador **/** es para dividir.
  - El operador **% :** resto de la división entera **a %b** devuelve el resto de dividir a entre b.

Pag. :25

SELECT-FILTROS WHERE:
Elementos: Operadores Relacionales

- **Operadores relacionales**: >, <, <> o !=, >=, <=, =. Los operadores relacionales sirven para comparar dos operandos.
- Se puede preguntar si un campo es mayor que un valor, o si un valor es distinto de otro.
- Estos operadores devuelven un número entero, de tal manera que si el resultado de la expresión es *cierto el resultado será 1, y si el resultado es falso el resultado será 0.*
- Por ejemplo, la expresión a > b devuelve 1 si a es estrictamente mayor que b y 0 en caso contrario.
- La expresión d < >e devuelve 1 si d y e son valores distintos.

Pag. :26

SELECT-FILTROS WHERE: Elementos: Operadores Lógicos

- **Operadores lógicos**: **AND, OR, NOT**.
- Los operadores lógicos toman como operandos valores lógicos, esto es, cierto o falso, en SQL, 1 o 0.
- Los operadores lógicos se comportan según las siguientes tablas de verdad:

Operando1	Operando2	Oper1 AND Oper2	Oper1 OR Oper2	NOT Operando1
FALSO	FALSO	FALSO	FALSO	VERDADERO
FALSO	VERDADERO	FALSO	VERDADERO	VERDADERO
VERDADERO	FALSO	FALSO	VERDADERO	FALSO
VERDADERO	VERDADERO	VERDADERO	VERDADERO	FALSO

Pag. :27

SELECT-FILTROS WHERE: Prioridad de los Operadores

- Los operadores de mayor prioridad se realizan primero, dentro de una expresión se realizarán de izquierda a derecha.

Operador (Ordenador de Mayor a menor prioridad)
- , + (unarios)
*, /, %
+,- (binarios)
=, >, <, >=, <=, <>, !=
NOT
AND
OR, ALL, ANY, BETWEEN, IN, LIKE, SOME

Pag. :28

SELECT-FILTROS WHERE: Prioridad de los Operadores

- Por ejemplo, en la siguiente expresión:

```
10*2 - 200/10 >10 AND 30- 6*3   =   40/5 OR 25>10+10
--*---  --/-----                  --*-     ---/-
1º :20   2º:2                     3º:18    4º:8
 --------------                 --------                 ---+--
  5º:0                            6º:  12                  7º:20
 -----0-----><---10-            ----12---=--8---         --25---><--20--
       8º: 1                            9º:    0                10º: 0
  --------------AND-------------------------
       11º: 0
  ---------0--------------------------OR---------1--------------
                          12º: 1
```

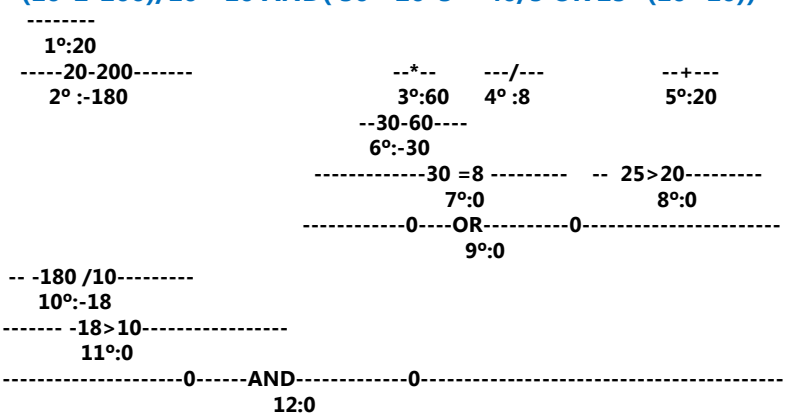

SELECT-FILTROS WHERE: Elementos: Paréntesis

- **Paréntesis**: (). Los operadores tienen una prioridad, por ejemplo, en la expresión 3+4*2, la multiplicación se aplica antes que la suma, se dice que el operador * tiene más prioridad que el operador +. El resultado sería: 3+8=11
- Para alterar esta prioridad, se puede usar el operador paréntesis, cuyo cometido es precisamente dar máxima prioridad a una parte de una expresión. Así, (3+4)*2, no es lo mismo que 3+4*2. Primero se realizaría: 3+4 y después se multiplica por 2. El resultado sería: 7*2=14

SELECT-FILTROS WHERE: Prioridad de los Operadores

- El ejemplo si colocamos algunos paréntesis quedaría:

(10*2-200)/10 >10 AND(30 - 20*3 = 40/5 OR 25>(10+10))

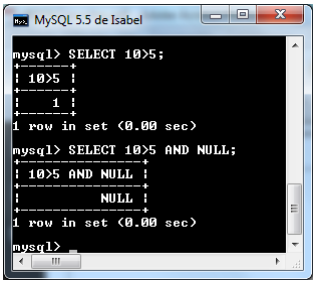


Pag. :31

SELECT-FILTROS WHERE: Elementos

- Por otro lado, se necesita un tratamiento de los valores nulos; hay que incluir como un posible operando el valor nulo.
- Si el resultado de parte de una expresión es NULL, es resultado total de dicha expresión será NULL.

10>5 AND NULL;



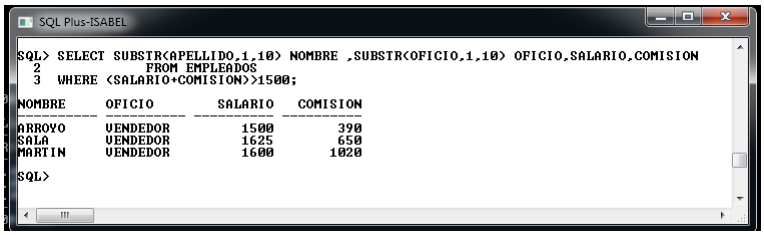
- Será NULL.

Pag. :32

SELECT-FILTROS: Elementos

- Ejemplo con **NULL** en una expresión. Tenemos el siguiente ejemplo en el que es resultado de una expresión puede ser NULL: SALARIO+COMISION:

```
SELECT SUBSTR(APELLIDO,1,10) NOMBRE,
    SUBSTR(OFICIO,1,10) OFICIO,SALARIO, COMISION
    FROM EMPLEADOS
    WHERE (SALARIO+COMISION)>1500;
```

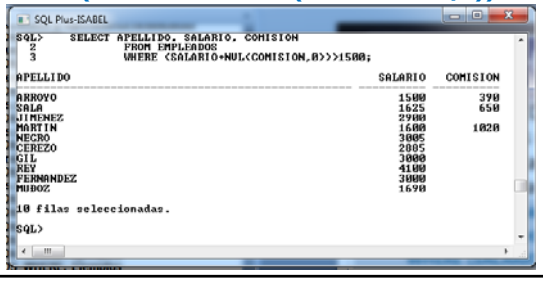


Pag. :33

SELECT-FILTROS: Elementos

- Si a la columna COMISION le ponemos la función de convertir a 0 si es nulo (NVL en Oracle y IFNULL en MySQL), el resultado cambia porque también contará cuando la comisión sea nula:

```
SELECT APELLIDO, SALARIO, COMISION
    FROM EMPLEADOS
    WHERE (SALARIO+NVL(COMISION,0))>1500;
```



Pag. :34

SELECT-FILTROS WHERE: Elementos

- Resultado de algunas expresiones:

Operación	Resultado
7+2*3	13
(7-2)*3	15
7>2	1
9<2	0
7>2 AND 4<3	0
7>2 OR 4<3	1
(10>=10 AND 0<=1)+2	3

Pag. :35

SELECT-FILTROS WHERE: Sensibilidad a mayúsculas y minúsculas

- **ORACLE:** si es sensible a mayúsculas o minúsculas, es decir, las siguientes condiciones pueden o no cumplirse, depende del contenido de las tablas y no son lo mismo. Si la tabla tiene por ejemplo VENDEDOR, sólo sería verdadera la 1ª:

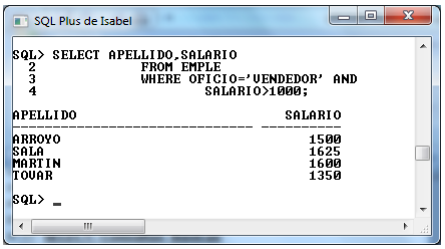
OFICIO='VENDEDOR'
OFICIO='vendedor'
OFICIO='Vendedor'

Pag. :36

SELECT-FILTROS WHERE: Ejemplos

- Selecciona el apellido y el salario de los empleados cuyo oficio sea VENDEDOR y el salario sea mayor de 1000

```
SELECT APELLIDO,SALARIO
    FROM EMPLE
    WHERE OFICIO='VENDEDOR' AND
        SALARIO>1000;
```

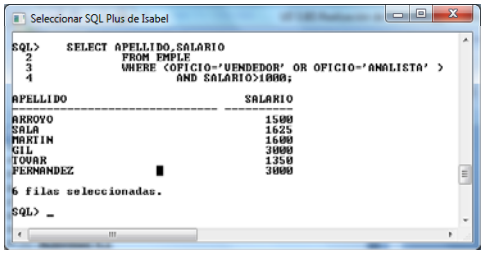


Pag. :37

SELECT-FILTROS WHERE: Ejemplos

- Selecciona el apellido, el salario y el oficio de los empleados cuyo oficio sea VENDEDOR o ANALISTA y el salario sea mayor de 1000

```
SELECT APELLIDO,SALARIO ,OFICIO
    FROM EMPLE
    WHERE (OFICIO='VENDEDOR' OR OFICIO='ANALISTA' )
        AND SALARIO>1000;
```

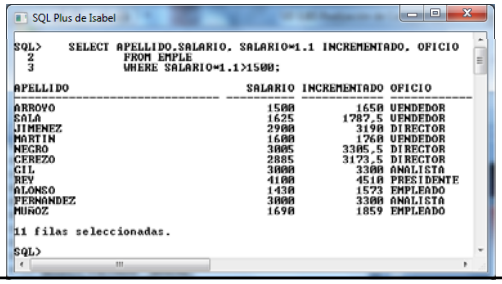


Pag. :38

SELECT-FILTROS WHERE: Ejemplos

- Selecciona el apellido, el salario y el oficio de los empleados cuyos salarios incrementados en el 10% sean mayores de 1000

```
SELECT APELLIDO,SALARIO, SALARIO*1.1 INCREMENTADO, OFICIO
    FROM EMPLE
    WHERE SALARIO*1.1>1500;
```



Pag. :39

SELECT-FILTROS WHERE: Operador de pertenencia a conjuntos :IN

- Además de los operadores presentados anteriormente (aritméticos, lógicos, etc.) se puede hacer uso del operador de pertenencia a conjuntos **IN**, cuya sintaxis es la siguiente:

```
nombre_columna IN (Valuel, Value2, ... )
```

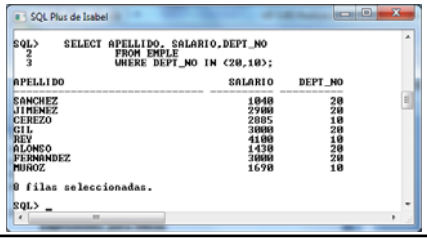
- Este operador permite comprobar si una columna tiene un valor igual que cualquier de los que están incluidos dentro del paréntesis.

Pag. :40

SELECT-FILTROS WHERE: Operador de pertenencia a conjuntos: IN

- Ejemplo, seleccionar el apellido y el salario de los empleados de los departamentos 20, 40 y 50.

```
SELECT APELLIDO, SALARIO,DEPT_NO
        FROM EMPLE
        WHERE DEPT_NO IN (20,10);
-- Es equivalente a la anterior
SELECT APELLIDO, SALARIO,DEPT_NO
        FROM EMPLE
        WHERE DEPT_NO =20 OR DEPT_NO=10;
```



Pag. :41

SELECT-FILTROS WHERE: Operador de rango: BETWEEN

- El operador de rango **BETWEEN** permite seleccionar los registros que estén incluidos en un rango.
- Su sintaxis es:

  nombre_columna BETWEEN Value1 AND Value2

- Ejemplo: selecciona el apellido y el salario de los empleados cuyo salario esté entro 1000 y 2000

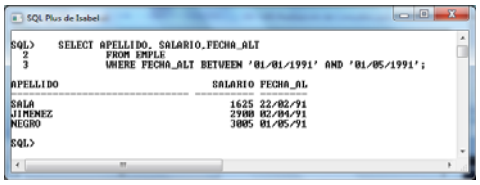
```
SELECT APELLIDO, SALARIO
        FROM EMPLE
        WHERE SALARIO BETWEEN 1000 AND 2000;
-- Es equivalente a la anterior
SELECT APELLIDO, SALARIO
        FROM EMPLE
        WHERE SALARIO >=1000 AND SALARIO<=2000;
```

Pag. :42

SELECT-FILTROS WHERE: Operador de rango: BETWEEN

- Ejemplo: selecciona el apellido y el salario de los empleados cuya fecha de alta esté entre '1991-01-01' y 1991-05-01'

```
SELECT APELLIDO, SALARIO
    FROM EMPLE
    WHERE FECHA_ALT BETWEEN '01/01/1991' AND '01/05/1991';
```

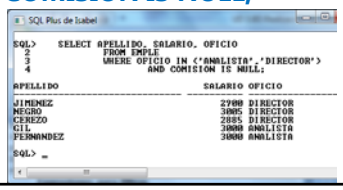


Pag. :43

SELECT-FILTROS WHERE: Test de valor nulo:IS /IS NOT

- Los operadores **IS** e **IS NOT** permiten verificar si un campo es *o no es nulo* respectivamente.
- Ejemplo, seleccionar el apellido, el salario y el oficio de los empleados que tengan como OFICIO: VENDEDOR o ANALISTA y la comisión sea nula.

```
SELECT APELLIDO, SALARIO, OFICIO
    FROM EMPLE
    WHERE OFICIO IN ('ANALISTA','DIRECTOR')
        AND COMISION IS NULL;
```



Pag. :44

SELECT-FILTROS WHERE: Test de patrón: %, y LIKE _

- Operador **LIKE**: este operador permite buscar un patrón en una cadena. Los filtros con test patrón seleccionan los registros que cumplan una serie de características.
- Se pueden usar los caracteres comodines **%** y _ para buscar una cadena de caracteres.
  - **'%'** : representa cualquier carácter de 0 o más de longitud
  - '_' : representa un solo carácter
- Ejemplos:

```
WHERE OFICIO LIKE 'M%'
        que oficio empiece por M
WHERE NOMBRE LIKE '%LUIS%'
        que nombre contenga  LUIS en cq. Posición
WHERE APELLIDO LIKE '_R%'
        que NOMBRE tenga como segundo carácter una R
```

Pag. :45

SELECT-FILTROS WHERE: Oracle : Límite de número de registros

- Oracle limita el número de filas apoyándose en una pseudocolumna, de nombre **ROWNUM**.
- Selecciona el apellido y el salario de los 5 primeros empleados.

```
SELECT APELLIDO,SALARIO
      FROM EMPLE
      WHERE ROWNUM<5;
```

- Selecciona el apellido y el salario empleados ordenados por apellido ,en el intervalo 3 al 6.

```
SELECT * FROM
  (SELECT APELLIDO, ROWNUM RNUM
   FROM  (SELECT APELLIDO,SALARIO FROM EMPLE ORDER BY APELLIDO)
            WHERE ROWNUM <= 6)
 WHERE RNUM >= 3
```

Pag. :46

SELECT: Ordenación: ORDER BY

- Para mostrar ordenados un conjunto de registros se utiliza la cláusula **ORDER BY** de la sentencia SELECT.

```
SELECT [DISTINCT] select_expr [,select_expr] ...
[FROM tabla]
[WHERE filtro]
[ORDER BY
    {nombre_columna | expr | posición} [ASC | DESC] , ... ]
```

Pag. :47

SELECT: Ordenación: ORDER BY

- Esta cláusula permite ordenar el conjunto de resultados de forma ascendente (**ASC**) o descendente (**DESC**) por una o varias columnas.
- Si no se indica ASC o DESC por defecto es ASC.
- La columna por la que se quiere ordenar se puede expresar por el nombre de la columna, una expresión o bien la posición numérica dentro de la SELECT del campo que se quiere ordenar.

Pag. :48

SELECT: Ordenación: ORDER BY

- Ejemplo: selecciona el departamento, el apellido y el salario de los empleados ordenador por departamento y dentro de éste por apellidos.

```
SELECT DEPT_NO,APELLIDO,SALARIO
    FROM EMPLE
    WHERE  DEPT_NO IN (10,20)
    ORDER BY DEPT_NO,APELLIDO;
```

Pag. :49

SELECT: Ordenación: ORDER BY

- Ejemplo: selecciona el departamento, el apellido y el salario de los empleados ordenador por departamento (DESC)y dentro de éste por apellidos(DESC).

```
SELECT DEPT_NO,APELLIDO,SALARIO
    FROM EMPLE
    WHERE SALARIO >1500
    ORDER BY
      DEPT_NO DESC,APELLIDO DESC;
```

Pag. :50

SELECT: Ordenación: ORDER BY

- Ejemplo: selecciona el departamento, el apellido y el salario de los empleados ordenador por departamento (ASC) y dentro de éste por apellido(DESC).

```
SELECT DEPT_NO,APELLIDO,SALARIO
    FROM EMPLE
    WHERE SALARIO >=1500
      AND DEPT_NO IN (10,20)
    ORDER BY  1,2 DESC;
```

Pag. :51

Actividad 4.1

- Crea una tabla con la siguiente estructura:

  MASCOTAS

  - IdMascota : Numérico de 4 caracteres.
  - Nombre : Alfanumérico de 50 caracteres
  - Especie : Alfanum de 40 car
  - Raza : Alfanum de 30 car.
  - Edad : numerico de 3 car.
  - Sexo : char(1) (M=Macho/H=Hembre)

- Introduce 6 registros (con el Insert o con el SQLDeveloper):

```
INSERT INTO MASCOTAS VALUES(1,'TOBI','CANINA','CHUCHO',5,'M');
```

Pag. :52

Actividad 4.1

- Codifica las siguientes consultas:

  4.1. Muestra el nombre y la especie de todas las mascotas .

  4.2. Muestra el nombre y el sexo de las mascotas poniendo un alias a los campos, sólo las de una especie en concreto, ordenado por raza y dentro de ésta por Nombre

  4.3. Muestra el nombre y la fecha de nacimiento aproximada de las mascotas(Hay que restar a la fecha actual la edad de la mascota). Sólo se visualizarán las Hembras cuya edad sea menor de 5 años. Ordenado por nombre.

- **Oracle** : Utiliza las funciones SYSDATE o CURRENT_DATE (fecha actual) y ADD_MONTHS(fecha,nm) suma nm meses a fecha, si es negativo resta.

Pag. :53

Práctica 1: Consultas Básicas

- Realizar la prácticas:

1.Ud4.PRACTICA 1-CONSULTAS BÁSICAS

(CREDITOS, NOMINAS Y EMPLE-DEPART)

Pag. :54

SELECT: Consultas de resumen

- En SQL se pueden generar consultas más complejas que resuman cierta información, extrayendo información calculada de varios conjuntos de registros.
- Por ejemplo para obtener la media o la suma del salario de todos los empleados, cuente el nº de empleados de un departamento, etc..
- Un ejemplo de consulta resumen seria la siguiente: que visualiza el nº total re filas de la tabla EMPLE:

```
SELECT COUNT(*) FROM EMPLE;
```

Pag. :55

SELECT: Consultas de resumen (Funciones)

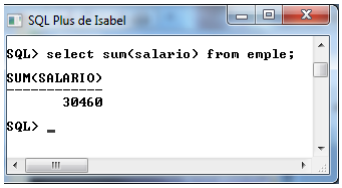
Función	Significado
SUM (Expresión)	Suma los valores indicados en el argumento
AVG (Expresión)	Calcula la media de los valores
MIN (Expresión)	Calcula el mínimo
MAX (Expresión)	Calcula el máximo
COUNT (Columna)	Cuenta el número de valores de una columna, excepto los nulos
COUNT (*)	Cuenta el número de valores de una fila, incluyendo los nulos.

Pag. :56

SELECT: Consultas de resumen

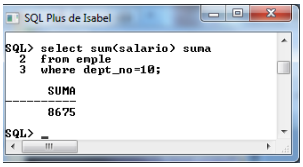
- Ejemplo: Visualiza la suma de los salarios de todos los empleados:

```
SELECT SUM(SALARIO)
    FROM EMPLE;
```



- Ejemplo: Visualizar la suma de los salarios de los empleados del departamento 10.

```
SELECT SUM(SALARIO) SUMA
    FROM EMPLE
    WHERE DEPT_NO=10;
```



Pag. :57

SELECT: Consultas de resumen

- Ejemplo: Visualiza la media de los salarios de todos los empleados:

```
SELECT AVG(SALARIO) MEDIA
    FROM EMPLE;
```

- Ejemplo: Visualizar la MEDIA de los salarios de los empleados del departamento 10.

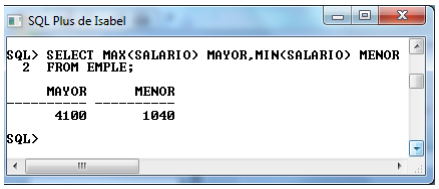
```
SELECT AVG(SALARIO) "MEDIA DEPT.10"
        FROM EMPLE
        WHERE DEPT_NO=10;
```

Pag. :58

SELECT: Consultas de resumen

- Ejemplo: Visualiza el salario mayor y el menor de todos los empleados:

```
SELECT MIN(SALARIO) MENOR, MAX(SALARIO) MAYOR
    FROM EMPLE;
```

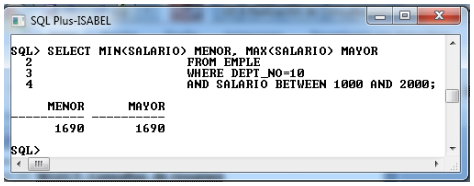


Pag. :59

SELECT: Consultas de resumen

- Ejemplo: Visualizar la MEDIA de los salarios de los empleados del departamento 10.

```
SELECT MIN(SALARIO) MENOR, MAX(SALARIO) MAYOR
        FROM EMPLE
        WHERE DEPT_NO=10
        AND SALARIO BETWEEN 1000 AND 2000;
```

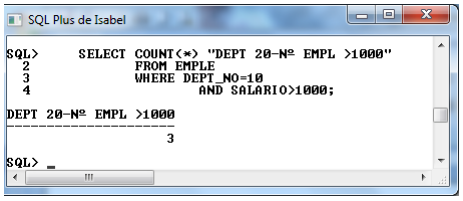


Pag. :60

SELECT: Consultas de resumen

- Ejemplo: Visualizar el nº de empleados del departamento 10 que tengan un salario mayor de 1000 .

```
SELECT COUNT(*) "DEPT 10-Nº EMPL >1000"
    FROM EMPLE
    WHERE DEPT_NO=10
        AND SALARIO>1000;
```



Pag. :61

SELECT: Consultas de resumen

- Ejemplo: Visualizar el nº de empleados del departamento 30 que tengan comisión .

```
SELECT COUNT(COMISION) "EMPLE CON COMISION"
    FROM EMPLE
    WHERE DEPT_NO=30;
```

- Si cambiamos COUNT(COMISION) por COUNT(*) saldrán contará todos los empleados, incluyendo aquellos que no tengan nada en comisión.(serán todos)

```
SELECT COUNT(*)
    FROM EMPLE
    WHERE DEPT_NO=30;
```

Pag. :62

SELECT: Agrupación : GROUP BY

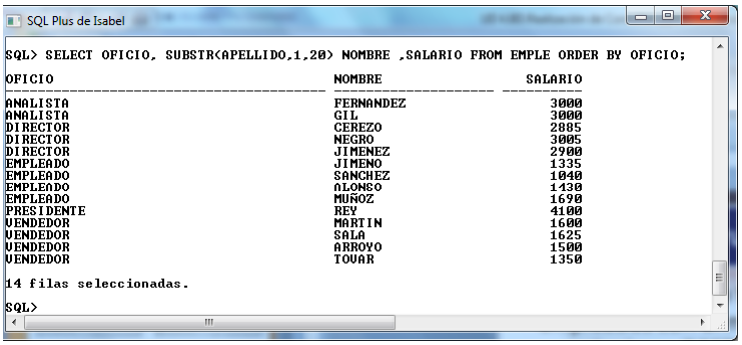
- Con las consultas de resumen se pueden realizar agrupaciones de registros.
- Se denomina agrupación de registros a un conjunto de registros que cumplen que tienen una o varias columnas con el mismo valor.
- La sintaxis para la sentencia **SELECT** con **GROUP BY** queda como sigue:

```
SELECT [DISTINCT] select_expr [,select_expr] ...
    [FROM tabla]
    [WHERE filtro]
    [GROUP BY expr [, expr] .... ]
    [ORDER BY {nombre_columna | expr |posición} [ASC | DESC] , ... ]
```

Pag. :63

SELECT: Agrupación : GROUP BY

- Por ejemplo, en la tabla EMPLE hay varios empleados con el mismo OFICIO, varios empleados con el mismo DEPT_NO, etc.

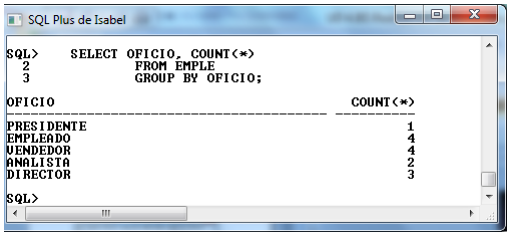


Pag. :64

SELECT: Agrupación : GROUP BY

- Como en la tabla EMPLE hay varios registros por cada OFICIO, se pueden agrupar por la columna OFICIO y después aplicar una función de grupo, por ejemplo CONTAR los empleados por oficio.

```
SELECT OFICIO, COUNT(*)
    FROM EMPLE
    GROUP BY OFICIO;
```

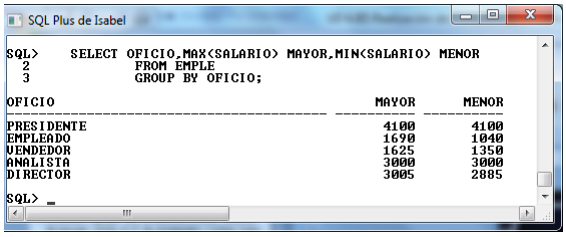


Pag. :65

SELECT: Agrupación : GROUP BY

- Ejemplo: seleccionar por cada oficio el salario mayor y menor.

```
SELECT OFICIO,MAX(SALARIO) MAYOR,MIN(SALARIO) MENOR
    FROM EMPLE
    GROUP BY OFICIO;
```

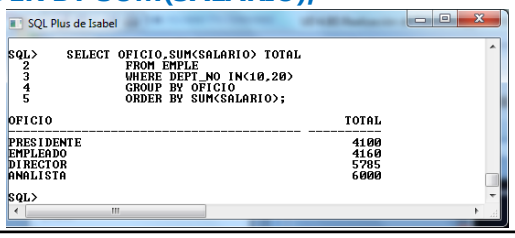


Pag. :66

SELECT: Agrupación : GROUP BY

- Ejemplo: seleccionar por cada oficio la suma del salario , sólo de los departamentos 10 y 20.

```
SELECT OFICIO,SUM(SALARIO) TOTAL
    FROM EMPLE
    WHERE DEPT_NO IN(10,20)
    GROUP BY OFICIO
    ORDER BY SUM(SALARIO);
```

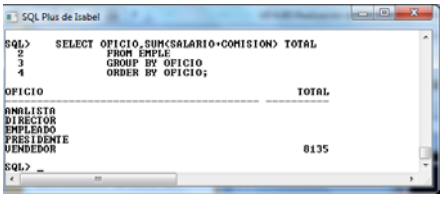


Pag. :67

SELECT: Agrupación : GROUP BY

- Ejemplo: seleccionar por cada oficio la suma del salario mas la comisión (que pude ser nula en algunos casos).

```
SELECT OFICIO,SUM(SALARIO+COMISION) TOTAL
    FROM EMPLE
    GROUP BY OFICIO
    ORDER BY OFICIO;
```



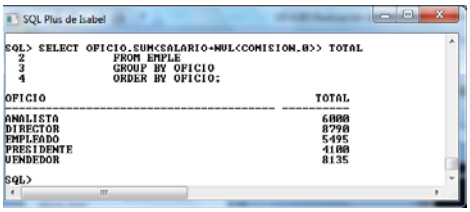
- Como vemos algunas filas no sale nada, es porque en cuanto que alguna comisión es nula ya todo el resultado es nulo.

Pag. :68

SELECT: Agrupación : GROUP BY

- Tendremos que cambiar por la siguiente SELECT:

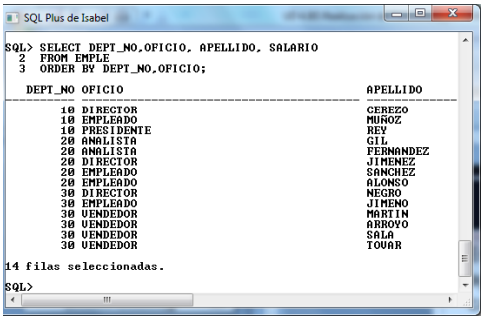
```
SELECT OFICIO,SUM(SALARIO+NVL(COMISION,0)) TOTAL
    FROM EMPLE
    GROUP BY OFICIO
    ORDER BY OFICIO;
```



Pag. :69

SELECT: Agrupación : GROUP BY

- En una agrupación puede haber varios niveles de agrupamiento, por ejemplo: si queremos contar por cada departamento el nº de empleados que hay en cada oficio, tendremos que agrupar por DEPT_NO y dentro de éste por OFICIO. El contenido de la tabla es:

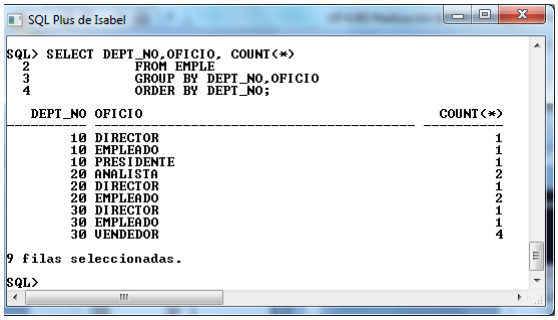


Pag. :70

SELECT: Agrupación : GROUP BY

- La SELECT sería:

```
SELECT DEPT_NO,OFICIO, COUNT(*)
    FROM EMPLE
    GROUP BY DEPT_NO,OFICIO
    ORDER BY DEPT_NO;
```

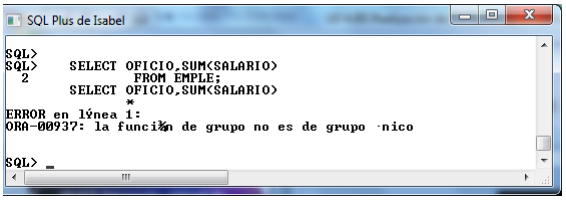


Pag. :71

SELECT: Agrupación : GROUP BY

- En las agrupaciones sólo es posible poner una columna si se incluye ésta en el **GROUP BY** , además podemos poner funciones de resumen**.**
- En **Oracle** esta sentencia daría error:

```
SELECT OFICIO,SUM(SALARIO)
    FROM EMPLE;
```



Pag. :72

SELECT: Agrupación GROUP BY: Filtros de Grupos HAVING

- Los filtros **HAVING** se aplican después de agrupar.
- Los filtros **WHERE** seleccionan las filas **antes** de agrupar los registros.
- Antes de agrupar se aplican los filtros del WHERE y una vez agrupados se aplican los del HAVING.
- Si se desea filtrar resultados calculados mediante agrupaciones se debe usar la siguiente sintaxis:

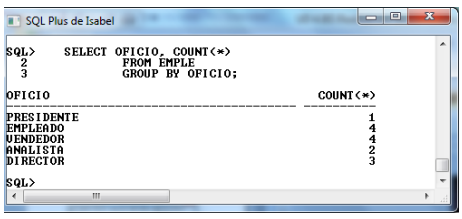
```
SELECT [DISTINCT] select_expr [,select_expr] ...
    [FROM tabla]
    [WHERE filtro]
    [GROUP BY expr [, expr] .... ]
    [HAVING filtro_grupos]
    [ORDER BY {nombre_columna | expr |posición} [ASC | DESC], ...]
```

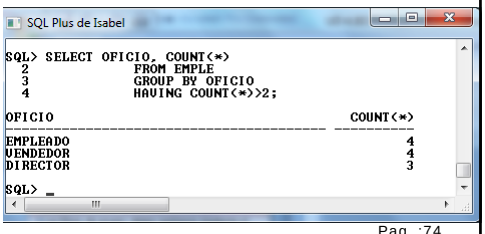
Pag. :73

SELECT: Agrupación : GROUP BY Filtros de Grupos HAVING

- Ejemplo: visualizar el nº de empleados por cada oficio, sólo se visualizarán aquellos oficios que tengan mas de 2 empleados.

```
SELECT OFICIO, COUNT(*)
    FROM EMPLE
    GROUP BY OFICIO
    HAVING COUNT(*)>2;
```



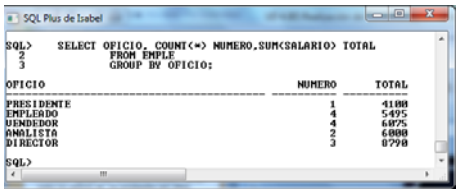


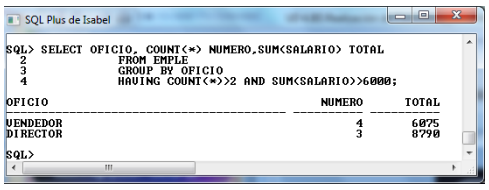
Pag. :74

SELECT: Agrupación : GROUP BY Filtros de Grupos HAVING

- Ejemplo: Por cada oficio, visualizaremos el nº de empleados y la suma de los salarios, sólo aquellos oficios que tengan mas de 2 empleados y la suma de sus salario sea >6000.

```
SELECT OFICIO, COUNT(*) NUMERO,SUM(SALARIO) TOTAL
    FROM EMPLE
    GROUP BY OFICIO
    HAVING COUNT(*)>2 AND SUM(SALARIO)>6000;
```





Pag. :75

SELECT: Agrupación GROUP BY: Filtros de Grupos HAVING

- Las condiciones que no sean del grupo no van en el HAVING sino en el WHERE, porque al realizarlo antes, son registros que no se tiene en cuenta y así la agrupación tardará menos en realizarse.
- Por ejemplo, queremos contar los empleados por oficio, si además queremos sólo contar aquellos empleados que tengan hijos, esta condición no es del grupo es individual para cada empleado, luego irá en el WHERE:

```
SELECT OFICIO, COUNT(*)
    FROM EMPLE
    WHERE  NRO_HIJOS>0
    GROUP BY OFICIO;
```

Pag. :76

SELECT: Agrupación : GROUP BY

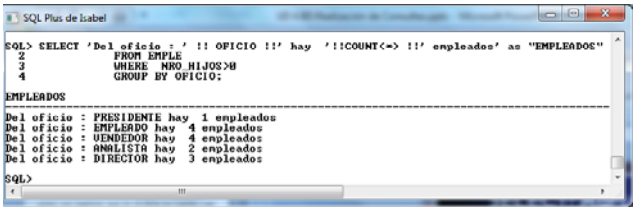
- En **Oracle**, los datos que pongamos en la **SELECT** que lleve un **GROUP BY** deben ser: una **constante** , una **función** de **grupo** (AVG, SUM,...), una **columna** expresada en el **GROUP BY .** Si no es así, da error.

Pag. :77

SELECT: Concatenar o Unir en Oracle

- En **Oracle** se utiliza el operando **||** para concatenar una o más expresiones (También se puede utiliza **concat,** pero sólo se pueden poner 2 expresiones)

```
SELECT 'Del oficio : ' || OFICIO ||' hay  '||COUNT(*) ||' empleados' as "EMPLEADOS"
    FROM EMPLE
    WHERE  NRO_HIJOS>0
    GROUP BY OFICIO;
```



Pag. :78

Práctica 2 y 3

- Realiza las consultas de la siguiente prácticas:

2.UD04.PRACTICA 2.JARDINERIA-BASICAS
3.UD04.PRACTICA 3.AGRUPACION

Pag. :79

Subconsultas

- Las **subconsultas** se utilizan para realizar filtrados con los datos de otra consulta.
- Estos filtros pueden ser aplicados tanto en la cláusula WHERE para filtrar registros como en la cláusula HAVING para filtrar grupos.
- Una **subconsulta** es una consulta encerrada entre paréntesis que después incluiremos en la consulta principal.

Pag. :80

Subconsultas

- Por ejemplo, con la base de datos Empleados (Tablas EMPLE/DEPART) queremos seleccionar el DNI, APELLIDOS de los empleados que trabajen en el departamento cuyo nombre es VENTAS.

```
SELECT DNI,APELLIDO
FROM EMPLE
WHERE DEPT_NO =(Código de Departamento cuyo NOMBRE es VENTAS);
```

- En la tabla EMPLE está el código del departamento, y necesitamos el nombre del mismo, que está en DEPART, luego necesitamos hacer una consulta para averiguarlo.
- **Código de VENTAS :** Para saber el código del departamento cuyo nombre es VENTAS tendremos que buscarlo primero en DEPART, realizaríamos una consulta que después incluiríamos en la consulta, construyendo una **subconsulta**

```
SELECT DEPT_NO
  FROM DEPART
  WHERE UPPER(DNOMBRE)='VENTAS';
```

- Quedaría:

```
SELECT DNI,APELLIDO
FROM EMPLE
WHERE DEPT_NO =(SELECT DEPT_NO FROM DEPART
                WHERE UPPER(DNOMBRE)='VENTAS' );
```

Pag. :81

Subconsultas

- Ejemplo: Obtener el apellido de los empleados con el mismo oficio que "GIL":

```
SELECT APELLIDO FROM EMPLE
 WHERE OFICIO = (El oficio de GIL)
```

- Primero debemos saber cual es el OFICIO de GIL:

```
SELECT OFICIO FROM EMPLE WHERE APELLIDO = 'GIL';
```

- A continuación lo incluimos en la consulta principal:

```
SELECT APELLIDO FROM EMPLE
WHERE OFICIO =
        (SELECT OFICIO FROM EMPLE
                WHERE APELLIDO='GIL');
```

Pag. :82

Subconsultas: Test de Comparación

- Consiste en usar los **operadores de comparación =, >=, <=, <>, >y <** para comparar el valor producido con un **valor único** generado por una **subconsulta**.
- Si la subconsulta devuelve mas de un valor la consulta principal fallaría.
- Una restricción importante es que la **subconsulta** debe estar siempre al lado **derecho** del **operador** de comparación.

```
campo >= (subconsulta)
```

Pag. :83

Subconsultas: Test de Comparación

- **Por ejemplo para saber el nombre del o los empleados con mayor salario.**
- 1º Hay que realizar una consulta para saber el salario máximo:

```
SELECT MAX(SALARIO) FROM EMPLE;
```

- TRUCO: Antes de insertarla en la consulta, ejecutarla para comprobar que devuelve un **único** valor.
- Esta consulta la insertaremos en la consulta principal

```
SELECT APELLIDO
   FROM EMPLE
   WHERE SALARIO =
      (SELECT MAX(SALARIO) FROM  EMPLE);
```

Pag. :84

Subconsultas: Test de Comparación

- **Para saber el nombre del o los empleados cuyo salario sea mayor o igual que la media de todos los salarios.**

- 1º Para saber la media de todos los salarios :

```
SELECT AVG(SALARIO) FROM EMPLE;
```

- Esta consulta la insertaremos en la consulta principal

```
SELECT APELLIDO
   FROM EMPLE
   WHERE SALARIO >=
      (SELECT AVG(SALARIO) FROM  EMPLE);
```

Pag. :85

Subconsultas: Test de pertenencia a conjunto: IN

- Consiste en usar el operador **IN** para filtrar los registros cuya expresión coincida **con algún** valor producido por la subconsulta, que puede devolver más de un valor.

- Visualizar el nombre de los empleados que pertenezcan a los departamentos de VENTAS y CONTABILIDAD.

```
SELECT APELLIDO FROM EMPLE
      WHERE DEPT_NO IN
            ( SELECT DEPT_NO FROM DEPART
                  WHERE DNOMBRE IN('VENTAS','CONTABILIDAD');
```

Pag. :86

Subconsultas: Test cuantificados ALL y ANY

- Los test cuantificados sirven para comparar una expresión con **todos** los registros de la subconsulta (**ALL**) o **algunos** de los registros de la subconsulta (**ANY**).
- La subconsulta puede devolver mas de un resultado.
- Para seleccionar el nombre de los empleados cuyo sueldo sea mayor que los sueldos de TODOS los empleados del departamento cuyo código es 30:

```
SELECT APELLIDO
    FROM EMPLE
    WHERE SALARIO >ALL
                (SELECT SALARIO
                        FROM  EMPLE
                        WHERE DEPT_NO=30);
```

Pag. :87

Subconsultas: Test cuantificados ALL y ANY

- Para seleccionar el nombre de los empleados cuyo sueldo sea mayor que los sueldos de ALGUNO de los empleados del departamento cuyo código es 30:

```
SELECT APELLIDO
    FROM EMPLE
    WHERE SALARIO >ANY
            (SELECT SALARIO
                    FROM  EMPLE
                    WHERE DEPT_NO=30);
```

Pag. :88

Subconsultas anidadas

- Se puede usar una **subconsulta** para filtrar los resultados de **otra subconsulta**.
- De esta manera se **anidan** subconsultas.
- Por ejemplo, en el ejemplo anterior si se quisiera saber el nombre de los empleados cuyo sueldo sea mayor que los sueldos de ALGUNO de los empleados del departamento cuyo nombre es VENTAS.

Pag. :89

Subconsultas anidadas

- 1º Subconsulta para saber el código del departamento VENTAS:

```
SELECT DEPT_NO FROM EMPLE WHERE DNOMBRE='VENTAS';
```

- 2º Los salarios de los empleados del departamento de VENTAS:

```
SELECT SALARIO FROM EMPLE
    WHERE DEPT_NO =
        (SELECT DEPT_NO FROM DEPART
            WHERE DNOMBRE='VENTAS');
```

- 3º Consulta completa:

```
SELECT APELLIDO FROM EMPLE
    WHERE SALARIO>ALL
    (SELECT SALARIO FROM EMPLE
        WHERE DEPT_NO =
            (SELECT DEPT_NO FROM DEPART
                WHERE DNOMBRE='VENTAS')
    );
```

Pag. :90

Subconsultas correlacionadas o consultas con referencias externas

- Una consulta es correlacionada cuando necesitas algún valor de la consulta principal para poder resolver la subconsulta.
- Por ejemplo: para saber los nombres de los departamentos que no tengan empleados, se puede hacer de varias formas, pero una de ellas es contar el número de empleados de ese departamento:

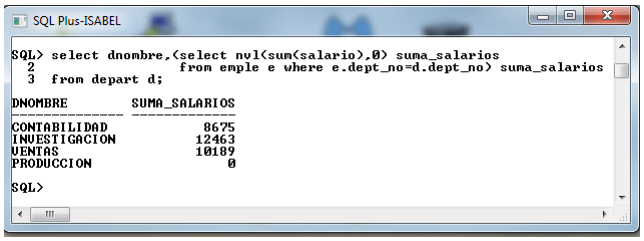
```
SELECT DNOMBRE FROM DEPART D
    WHERE 0=
        (SELECT COUNT(*) FROM EMPLE E
            WHERE  D.DEPT_NO=E.DEPT_NO);
```

Pag. :91

Subconsultas correlacionadas o consultas con referencias externas

- Ejemplo: seleccionar el nombre de los departamentos y la suma de los salarios de sus empleados:

```
SELECT DNOMBRE,
  (SELECT NVL(SUM(SALARIO),0)
    FROM EMPLE E WHERE    E.DEPT_NO=D.DEPT_NO)
FROM DEPART D;
```

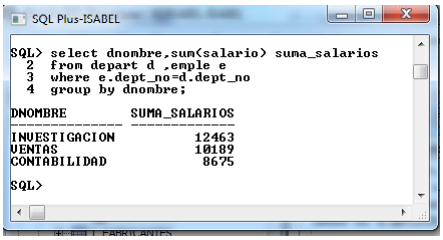


Pag. :92

Subconsultas correlacionadas o consultas con referencias externas

- Se podría poner también así, pero la diferencia con la anterior es que los departamentos que no tengan empleados no salen:

```
SELECT DNOMBRE,SUM(SALARIO)
   FROM DEPART D ,EMPLE E
   WHERE E.DEPT_NO=D.DEPT_NO
   GROUP BY DNOMBRE;
```



Pag. :93

Subconsultas: Test de existencia

- El test de existencia permite filtrar los resultados de una consulta si existen filas en la subconsulta asociada, esto es, si la subconsulta genera un número de filas distinto de O.
- Para usar el test de existencia se utiliza el operador **EXISTS**:

```
SELECT columnas
   FROM tabla
   WHERE [NOT] EXISTS (subconsulta)
```

- Para visualizar **los departamentos que tengan no tengan empleados :**

```
SELECT DNOMBRE
   FROM DEPART D
   WHERE NOT EXISTS
      ( SELECT DISTINCT  DEPT_NO  FROM EMPLE E
            WHERE E.DEPT_NO=D.DEPT_NO);
```

Pag. :94

Subconsultas: GROUP BY ... HAVING

- También se pueden utilizar subconsultas en las condiciones del HAVING.
- Ejemplo: seleccionar el departamento y el nº de empleados del mismo cuya media de los salarios de sus empleados sea mayor que la media total de todos los empleados.

```
SELECT DEPT_NO,COUNT(*)
  FROM EMPLE
  GROUP BY DEPT_NO
  HAVING AVG(SALARIO)>
        (SELECT AVG(SALARIO) FROM EMPLE);
```

Pag. :95

Subconsultas: GROUP BY ... HAVING

- Ejemplo: **Seleccionar el departamento con mayor número de empleados.**
- La siguiente consulta nos devuelve el nº de empleados de cada departamento:

```
SELECT COUNT(*) FROM EMPLE GROUP BY DEPT_NO;
```

- Si le aplicamos el MAX nos devolverá el nº de empleados del departamento con mayor nº de empleados
- En Oracle:

```
SELECT MAX(COUNT(*)) FROM EMPLE GROUP BY DEPT_NO
```

Pag. :96

Subconsultas: GROUP BY ... HAVING

- Si lo aplicamos a la consulta principal, nos devolverá el departamento con mayor nº de empleados:
- En **Oracle**:

```
SELECT DEPT_NO
  FROM EMPLE
  GROUP BY DEPT_NO
  HAVING COUNT(*)=
     (SELECT MAX(COUNT(*)) FROM EMPLE  GROUP BY DEPT_NO);
```

- En **MySQL**:

```
SELECT DEPT_NO
  FROM EMPLE
  GROUP BY DEPT_NO
  HAVING COUNT(*)=
     (SELECT MAX(NUMERO) FROM
                 (SELECT COUNT(*) NUMERO FROM EMPLE
                 GROUP BY DEPT_NO) AS A_EMPLE
     );
```

Pag. :97

Práctica 4

- Realizar la actividad:

4.UD04.PRACTICA 4-SUBCONSULTAS JARDINERÍA

Pag. :98

Consultas Multitabla

- Una **consulta multitabla** es aquella en la que se puede consultar información de más de una tabla.
- Se aprovechan los campos relacionados de las tablas para **unirlas** (**join**).
- Para poder realizar este tipo de consultas hay que utilizar la siguiente sintaxis:

```
SELECT [DISTINCT] select_expr [,select_expr] ...
[FROM referencias_tablas]
[WHERE filtro]
[GROUP BY expr [, expr] .... ]
[HAVING filtro_grupos]
[ORDER BY {nombre_columnas | expr | posición} [ASC |DESC] , ... ]
```

Pag. :99

Consultas Multitabla

- La diferencia con las consultas sencillas se halla en la cláusula FROM.
- **referencias_tablas**:

```
nombre_tabla [ alias][, nombre_tabla [ alias]]
| nombre_tabla [alias] INNER  JOIN nombre_tabla [ alias]
      [ON condición]
| nombre_tabla [alias] LEFT [OUTER] JOIN nombre_tabla [ alias]
      ON condición
| nombre_tabla [alias] RIGHT [OUTER] JOIN nombre_tabla [alias]
      ON condición
```

Pag. :100

Consultas Multitabla

- La primera opción, (referencia_tabla[,referencia_tabla] ...) es típica de SQL 1 (SQL-86) para las uniones, que consisten en un producto cartesiano más un filtro por las columnas relacionadas

  nombre_tabla [alias][, nombre_tabla [alias]]

- El resto de opciones son propias de SQL 2 (SQL-92 y SQL-2003). (**INNER JOIN, LEFT [OUTER] JOIN, RIGHT [OUTER] JOIN)**

Pag. :101

Consultas Multitabla en SQL1

- La siguiente consulta realiza lo que se llama el producto cartesiano es decir cruzar las filas de la primera tabla con todas las filas de la segunda:

```
SELECT APELLIDO, DNOMBRE
  FROM EMPLE, DEPART ;
```

Pag. :102

Consultas Multitabla en SQL1

- Ejemplo SQL1: seleccionar el APELLIDO y el nombre de su departamento. La columna APELLIDO está en la tabla EMPLE y la columna nombre del departamento está en la tabla DEPART.
- Para poder visualizar ambos en la misma consulta hay que combinar las tablas EMPLE y DEPART, para unirlas se utiliza la columna DEPT_NO que relaciona ambas tablas:

```
SELECT E.APELLIDO, D.DNOMBRE
  FROM EMPLE E, DEPART D
  WHERE E.DEPT_NO=D.DEPT_NO;
```

Pag. :103

Consultas Multitabla SQL 1

- Por ejemplo en la BD Jardinería: para seleccionar de todos los pedidos :el Nombre del empleado que ha realizado el pedido, el nombre del cliente al que se le ha vendido y el código de pedido tendremos que cruzar las tablas CLIENTES (es donde está el nombre del cliente), EMPLEADOS(es donde está el nombre del empleado) y PEDIDOS(porque es lo que queremos listar :

```
SELECT E.NOMBRE,C.NOMBRECLIENTE,P.CODIGOPEDIDO
  FROM CLIENTES C, PEDIDOS P, EMPLEADOS E
  WHERE C.CODIGOCLIENTE=P.CODIGOCLIENTE
    AND C.CODIGOEMPLEADOREPVENTAS=E.CODIGOEMPLEADO
  ORDER BY E.NOMBRE;
```

Pag. :104

Consultas Multitablas

- Ejemplo: se quiere visualizar de todos los pedidos las siguientes columnas: CodigoPedido, FechaPedido, CódigoProducto, NombreProducto, CantidadPedida, Unidades, Total(que será la CantidadPedida x Unidades). Necesitaremos las tablas: PEDIDOS , DETALLESPEDIDO y PRODUCTOS.

```
SELECT PE.CODIGOPEDIDO, PE.FECHAPEDIDO,
       PR.CODIGOPRODUCTO, PR.NOMBRE, DP.CANTIDAD,
       DP.PRECIOUNIDAD,
       DP.CANTIDAD*DP.PRECIOUNIDAD TOTAL
FROM PEDIDOS PE, DETALLESPEDIDO DP, PRODUCTOS PR
WHERE DP.CODIGOPEDIDO=PE.CODIGOPEDIDO
      AND DP.CODIGOPRODUCTO=PR.CODIGOPRODUCTO;
```

Pag. :105

Consultas Multitablas

- Ejemplo: visualizar el Apellido de los empleados y el Nombre de su departamento de todos los empleados cuyo salario sea mayor que la media de los salarios de todos los empleados.

```
SELECT E.APELLIDO, D.DNOMBRE
    FROM EMPLE E,DEPART D
    WHERE
        D.DEPT_NO=E.DEPT_NO AND
        SALARIO>
            (SELECT AVG(SALARIO)
                FROM EMPLE)
```

Pag. :106

Consultas Multitabla SQL 1

- Ejemplo: seleccionar el código de departamento, el nombre del departamento y el nº de empleados de los departamentos que tengan mas de 3 empleados.

```
SELECT D.DEPT_NO,D.DNOMBRE,COUNT(*)
  FROM EMPLE E,DEPART D
  WHERE D.DEPT_NO=E.DEPT_NO
  GROUP BY E.DEPT_NO, D.DNOMBRE
  HAVING COUNT(*) >2;
```

Pag. :107

Consultas Multitabla SQL 1

- Ejemplo: seleccionar los nombres de los departamentos y el nº de empleados ,de aquellos departamentos en los que la media de los salarios de sus empleados sea mayor que la media de todos los empleados.

```
SELECT D.DNOMBRE,COUNT(*)
  FROM EMPLE E,DEPART D
  WHERE D.DEPT_NO=E.DEPT_NO
  GROUP BY D.DNOMBRE
  HAVING AVG(E.SALARIO)>
        (SELECT AVG(SALARIO)FROM EMPLE);
```

Pag. :108

Practica 5

- Realizar la actividad:

5.UD04.PRACTICA 5-MULTITABLAS JARDINERÍA

Pag. :109

Consultas Multitabla

- Como hemos dicho, SQL 2 introduce otra sintaxis para los siguientes tipos de consultas multitablas: los joins (o composiciones) internas, externas y productos cartesianos (también llamadas composiciones cruzadas):
    1. Join Interna:
        - De equivalencia (**INNER JOIN**)
        - Natural (**NATURAL JOIN**)
    2. Producto Cartesiano (**CROSS JOIN**)
    3. Join Externa
        - De tabla derecha (**RIGHT OUTER JOIN**)
        - De tabla izquierda (**LEFT OUTER JOIN**)
        - Completa (**FULL OUTER JOIN)**

Pag. :110

Composiciones internas: INNER JOIN

- Con la operación **INNER JOIN** se calcula el producto cartesiano de todos los registros, después, cada registro en la primera tabla es combinado con cada registro de la segunda tabla, y solo se seleccionan aquellos registros que satisfacen las condiciones que se especifiquen. Hay que tener en cuenta que los valores Nulos no se combinan.

```
| nombre_tabla [alias] INNER  JOIN nombre_tabla [ alias]
  [ON condición]
```

Pag. :111

Composiciones internas: INNER JOIN

- Ejemplo: para visualizar el apellido y el nombre del departamento de todos los empleados . Si algún empleado no tuviera DEPT_NO no saldría.

```
SELECT E.APELLIDO, D.DNOMBRE
  FROM EMPLE E INNER JOIN  DEPART D
  ON  E.DEPT_NO=D.DEPT_NO;
```

Pag. :112

Composiciones internas: INNER JOIN

- Por ejemplo en la BD Jardinería: para seleccionar de todos los pedidos :el Nombre del empleado que ha realizado el pedido, el nombre del cliente al que se le ha vendido y el código de pedido:

```
SELECT E.NOMBRE,C.NOMBRECLIENTE,P.CODIGOPEDIDO
   FROM CLIENTES C INNER JOIN  PEDIDOS P
         ON C.CODIGOCLIENTE=P.CODIGOCLIENTE
    INNER JOIN  EMPLEADOS E
         ON  E.CODIGOEMPLEADO=C.CODIGOEMPLEADOREPVENTAS
   WHERE  lower(LOCALIDAD)='ciudad real'
   ORDER BY E.NOMBRE;
```

Pag. :113

Composiciones internas: INNER JOIN

- Ejemplo: seleccionar el nombre del departamento y el nº de empleados del mismo de los departamentos que tengan mas de 3 empleados.

```
SELECT D.DNOMBRE,COUNT(*)
  FROM EMPLE E INNER JOIN DEPART D
         ON D.DEPT_NO=E.DEPT_NO
  GROUP BY D.DNOMBRE
  HAVING COUNT(*) >2;
```

Pag. :114

Composiciones internas: INNER JOIN

- Ejemplo: seleccionar el nombre del departamento y el nº de empleados del mismo cuya media de los salarios de sus empleados sea mayor que la media total de todos los empleados.

```
SELECT E.DNOMBRE,COUNT(*)
 FROM EMPLE E INNER JOIN DEPART D
        ON  D.DEPT_NO=E.DEPT_NO
 GROUP BY E.DNOMBRE
 HAVING AVG(E.SALARIO)>
        (SELECT AVG(SALARIO)FROM EMPLE);
```

Pag. :115

Composiciones naturales: NATURAL JOIN

- Es una especialización de la **INNER JOIN**. En este caso se comparan todas las columnas que tengan el mismo nombre en ambas tablas. La tabla resultante contiene solo una columna por cada par de columnas con el mismo nombre.

```
SELECT E.APELLIDO, D.DNOMBRE
   FROM EMPLE E NATURAL JOIN  DEPART D;
```

Pag. :116

Producto cartesiano: CROSS JOIN

- Este tipo de sintaxis devuelve el producto cartesiano de dos tablas:

```
SELECT APELLIDO, DNOMBRE
   FROM EMPLE  CROSS JOIN  DEPART ;
```

- Es equivalente a:

```
SELECT APELLIDO, DNOMBRE
   FROM EMPLE , DEPART ;
```

Pag. :117

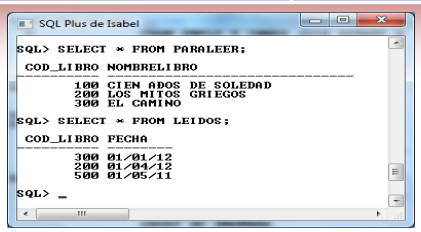
Composiciones externas: OUTER JOIN

- En este tipo de operación, las tablas relacionadas no requieren que haya una equivalencia.
- El registro es seleccionado para ser mostrado aunque no haya otro registro que le corresponda.
- **OUTER JOIN** se subdivide dependiendo de la tabla a la cual se le admitirán los registros que no tienen correspondencia, ya sean de tabla izquierda (**LEFT**) de tabla derecha **(RIGTH**), o combinación completa (**FULL**).

```
| nombre_tabla [ alias] LEFT [OUTER] JOIN nombre_tabla [alias]
      ON condición
| nombre_tabla [alias] RIGHT [OUTER] JOIN nombre_tabla [alias]
      ON condición
```

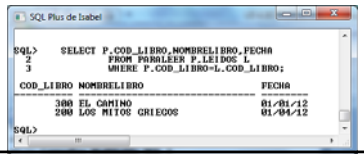
Pag. :118

Composiciones externas: OUTER JOIN



- Si tecleamos la siguiente SELECT aparecen sólo los libros leídos:

```
SELECT P.COD_LIBRO,P.NOMBRELIBRO,FECHA
    FROM PARALEER P,LEIDOS L
    WHERE P.COD_LIBRO=L.COD_LIBRO;
```

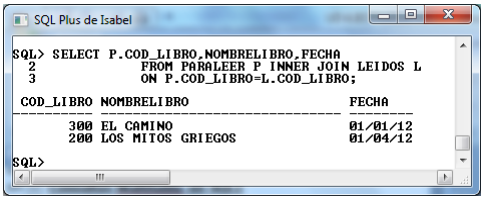


Pag. :119

Composiciones externas: OUTER JOIN

- Con **INNER JOIN**:

```
SELECT P.COD_LIBRO,P.NOMBRELIBRO,FECHA
    FROM PARALEER P INNER JOIN LEIDOS L
    ON P.COD_LIBRO=L.COD_LIBRO;
```



Pag. :120

Composiciones externas: LEFT OUTER JOIN

- Si los registros que no tienen correspondencia son los que aparecen en la tabla de la izquierda se llama composición de tabla izquierda o **LEFT JOIN** (o **LEFT OUTER JOIN**).
- Es decir, que la tabla que NO TIENE NULOS es la de la izquierda.

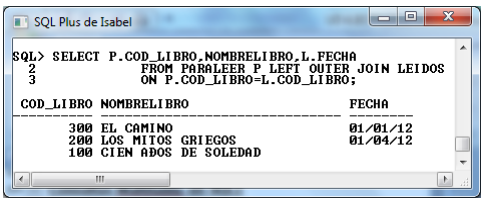
```
nombre_tabla [alias] LEFT [OUTER] JOIN nombre_tabla [alias]
    ON condición
```

Pag. :121

Composiciones externas: LEFT OUTER JOIN

- Si queremos que aparezcan TODOS los libros se hayan leído o no:

```
SELECT P.COD_LIBRO,P.NOMBRELIBRO,L.FECHA
    FROM PARALEER P LEFT OUTER JOIN LEIDOS L
    ON P.COD_LIBRO=L.COD_LIBRO;
```



Pag. :122

Composiciones externas: RIGHT OUTER JOIN

- Si los registros que no tienen correspondencia son los que aparecen en la tabla de la derecha, se llama composición de tabla derecha **RIGHT JOIN** (o **RIGHT OUTER JOIN**):
- Es decir, que la tabla que NO TIENE NULOS es la de la derecha.

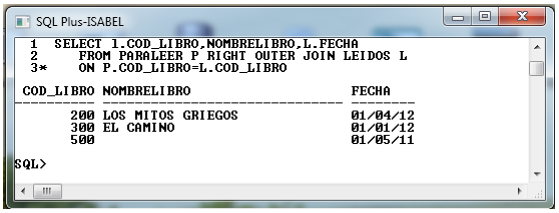
```
nombre_tabla [ alias] RIGHT [OUTER] JOIN nombre_tabla [alias]
    ON condición
```

Pag. :123

Composiciones externas: RIGHT OUTER JOIN

- Si queremos que aparezcan TODOS los libros se hayan leído aunque ese libro no esté ya

```
SELECT L.COD_LIBRO,P.NOMBRELIBRO,L.FECHA
    FROM PARALEER P RIGHT OUTER JOIN LEIDOS L
    ON P.COD_LIBRO=L.COD_LIBRO;
```



Pag. :124

Composiciones externas: FULL OUTER JOIN

- Si los registros que no tienen correspondencia pueden estar en ambas tablas se llama composición de tabla derecha **FULL JOIN** (o **FULL OUTER JOIN**):
- Es decir, que la tabla que NO TIENE NULOS es la de la derecha y en la izquierda.

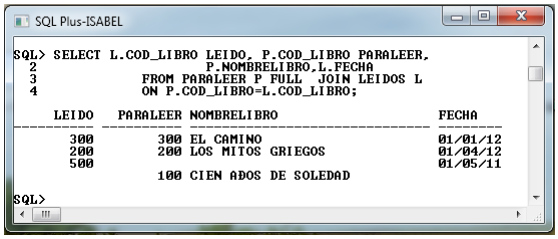
```
nombre_tabla [alias] FULL [OUTER] JOIN nombre_tabla [alias]
    ON condición
```

Pag. :125

Composiciones externas: FULL OUTER JOIN

- Si queremos que aparezcan TODOS los libros leídos o no

```
SELECT L.COD_LIBRO LEIDO, P.COD_LIBRO PARALEER,
            P.NOMBRELIBRO,L.FECHA
    FROM PARALEER P FULL  JOIN LEIDOS L
    ON P.COD_LIBRO=L.COD_LIBRO;
```



Pag. :126

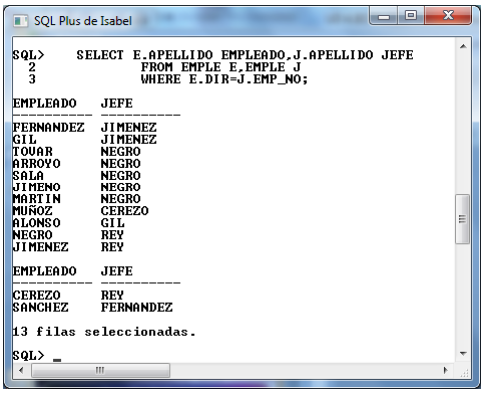
Consultas Reflexivas

- A veces es necesario obtener información de relaciones reflexivas, por ejemplo, un informe de empleados donde junto a su nombre y apellidos apareciera el nombre y apellidos de su jefe.
- Para ello, es necesario hacer una JOIN entre registros de la misma tabla: EMPLE.

Pag. :127

Consultas Reflexivas

- **SELECT E.APELLIDO EMPLEADO,J.APELLIDO JEFE**
  FROM EMPLE E,EMPLE J
  WHERE E.DIR=J.EMP_NO;



Pag. :128

Consultas Derivadas

- Las consultas con tablas derivadas son aquellas que utilizan sentencias SELECT en la cláusula FROM en lugar de nombres de tablas. (En MySQL deben de llevar un alias), por ejemplo:

```
SELECT *
FROM (SELECT CODIGOEMPLEADO, NOMBRE
        FROM EMPLEADOS
        WHERE CODIGOOFICINA='OF1')
    AS TABLA_DERIVADA;
```

Pag. :129

Consultas Derivadas

- Por ejemplo en la base de datos JARDINERIA, si se desea sacar el importe del pedido de menor coste de todos los pedidos hay que pensar primero como sacar el total de todos los pedidos y de ahí el pedido con menor coste con la función de columna MIN.

Pag. :130

Consultas Derivadas

1: Para calcular el total de cada pedido hay que Codificar

```
SELECT SUM(CANTIDAD*PRECIOUNIDAD) AS TOTAL,CODIGOPEDIDO
FROM DETALLEPEDIDOS
GROUP BY CODIGOPEDIDO;
```

2: Para calcular el menor pedido, se puede hacer una tabla derivada de la consulta anterior y con la función MIN obtener el menor de ellos:

```
SELECT MIN(TOTAL) , CODIGOPEDIDO FROM
   (SELECT SUM(CANTIDAD*PRECIOUNIDAD) AS TOTAL, CODIGOPEDIDO
     FROM DETALLEPEDIDOS
     GROUP BY CODIGOPEDIDO
   ) AS TOTALPEDIDOS;
```

TotalPedidos es la tabla derivada formada por el resultado de la consulta entre parentesis

Pag. :131

Consultas Derivadas

- Ejemplo: **Seleccionar el departamento con mayor número de empleados.**
- La siguiente consulta nos devuelve el nº de empleados de cada departamento:

```
SELECT COUNT(*) FROM EMPLE GROUP BY DEPT_NO;
```

- Si le aplicamos el MAX nos devolverá el nº de empleados del departamento con mayor nº de empleados
- En MySQL:

```
SELECT MAX(NRO_EMPLE) FROM
        (SELECT COUNT(*) AS NRO_EMPLE FROM EMPLE GROUP BY DEPT_NO)  AS MAX_EMPLE
```

- Si lo aplicamos a la consulta principal, nos devolverá el departamento con mayor nº de empleados:

```
SELECT DEPT_NO
   FROM EMPLE
   GROUP BY DEPT_NO
   HAVING COUNT(*) =
    (SELECT MAX(NRO_EMPLE) FROM
                (SELECT COUNT(*) AS NRO_EMPLE FROM EMPLE GROUP BY DEPT_NO)
                AS MAX_EMPLE
    );
```

Pag. :132

Operadores: UNION, INTERSECT y MINUS

- Son operadores de conjuntos
- Los conjuntos son filas resultantes de cualquier SELECT.

SELECT COL1,COL2,... FROM TABLA1 WHERE condición
Operador_Conjunto
SELECT COL1,COL2,... FROM TABLA2 WHERE condición

Pag. :133

Reglas con operadores de conjuntos

- Las columnas de las dos consultas se relacionan en orden (izda a dcha)
- Los nombres de las columnas de las distintas consultas no tienen por qué ser iguales.
- Las consultas tienen el mismo número de columnas.
- Los tipos de datos deben coincidir, aunque su longitud no tiene por qué ser la misma.
- Los operadores de conjuntos se pueden concatenar.
- Los conjuntos se evalúan de izda. a dcha., se pueden utilizar paréntesis.

Pag. :134

UNION

- Combina los resultados de ambas consultas, convirtiendo las filas duplicadas en una sola fila.

```
SELECT col1,col2 FROM TABLA1 WHERE condicion
UNION
SELECT col1,col2 FROM TABLA2 WHERE condicion;
```

- El nº de columnas de las dos SELECT debe ser el mismo y del mismo tipo.
- Normalmente representan lo mismo pero en tablas distintas.

Pag. :135

Ejemplos de UNION

Disponemos de tres tablas: ALUM contiene los nombres de alumnos que hay actualmente en el centro, NUEVOS contiene los nombres de los futuros alumnos, y ANTIGUOS contiene los nombres de antiguos alumnos del centro. La descripción es la misma para las tres:

```
SQL> DESC ALUM;
                 Nombre                    ¿Nulo?   Tipo
                 --------------------------------   --------   -----
                 NOMBRE                               VARCHAR2(15)
                 EDAD                                 NUMBER(2)
                 LOCALIDAD                            VARCHAR2(15)

SQL> SELECT * FROM ALUM;
                   NOMBRE                 EDAD     LOCALIDAD
                   --------------------   ---------   ---------------
                   JUAN                        18     COSLADA
                   PEDRO                       19     COSLADA
                   ANA                         17     ALCALÁ
                   LUISA                       18     TORREJÓN
                   MARÍA                       20     MADRID
                   ERNESTO                     21     MADRID
                   RAQUEL                      19     TOLEDO
                   7 filas seleccionadas.
```

Pag. :136

Ejemplos de UNION

```
SQL> SELECT * FROM ANTIGUOS;

NOMBRE               EDAD LOCALIDAD
-------------------- ---------- ---------------
MARÍA                    20 MADRID
ERNESTO                  21 MADRID
ANDRES                   26 LAS ROZAS
IRENE                    24 LAS ROZAS

SQL> SELECT * FROM NUEVOS;

NOMBRE               EDAD LOCALIDAD
-------------------- ---------- ---------------
JUAN                     18 COSLADA
MAITE                    15 ALCALÁ
SOFÍA                    14 ALCALÁ
ANA                      17 ALCALÁ
ERNESTO                  21 MADRID
```

Pag. :137

Ejemplos de UNION

- Ejemplo: visualizar los nombres de los alumnos actuales y de los futuros alumnos. Aparecerán los nombres que estén en ambas tablas:

SELECT NOMBRE FROM ALUM UNION SELECT NOMBRE FROM NUEVOS;

```
NOMBRE
--------------------
ANA                  MARÍA
ERNESTO              PEDRO
JUAN                 RAQUEL
LUISA                SOFÍA
MAITE                9 filas seleccionadas.
```

Pag. :138

UNION ALL

- Combina los resultados de ambas consultas, sin convertir a una sola fila las filas duplicadas.

```
SELECT col1,col2 FROM TABLA1 WHERE condicion
UNION ALL
SELECT col1,col2 FROM TABLA2 WHERE condicion;
```

Pag. :139

INTERSECT

- Devuelve las filas comunes a ambas Consultas. Convierte a una sola fila los duplicados.

```
SELECT col1,col2 FROM TABLA1 WHERE condicion
INTERSECT
SELECT col1,col2 FROM TABLA2 WHERE condicion;
```

Pag. :140

Ejemplo de INTERSECT

- Para obtener los alumnos que están actualmente en el centro y que estuvieron matriculados anteriormente. Aparecerán los nombre de los alumnos que están en ALUM y en ANTIGUOS.

```
SELECT NOMBRE FROM ALUM
    INTERSECT
SELECT NOMBRE FROM ANTIGUOS;
```

- Esta consulta se puede realizar también usando el operador **IN:**

```
SELECT NOMBRE FROM ALUM
    WHERE NOMBRE  IN (SELECT NOMBRE FROM ANTIGUOS);
```

```
NOMBRE
--------------------
ERNESTO
MARÍA
```

Pag. :141

MINUS

- Devuelve las filas que están en la primera consulta y que no están en la segunda.

```
SELECT .... FROM .... WHERE ....
MINUS
SELECT .... FROM .... WHERE ....
```

Pag. :142

Ejemplos de MINUS

- Para obtener el nombre y la localidad de los alumnos que están actualmente en el centro y que nunca hayan estado anteriormente. Necesitamos las filas que están en ALUM y que no aparezcan en ANTIGUOS.

```
SELECT NOMBRE,LOCALIDAD FROM ALUM
MINUS
 SELECT NOMBRE,LOCALIDAD FROM ANTIGUOS;
```

- Esta consulta se puede realizar con un **NOT IN:**

```
SELECT NOMBRE,LOCALIDAD FROM ALUM
WHERE  NOMBRE NOT IN
       (SELECT NOMBRE FROM ANTIGUOS)
ORDER BY LOCALIDAD;
```

NOMBRE	LOCALIDAD
ANA	ALCALÁ
JUAN	COSLADA
LUISA	TORREJÓN
PEDRO	COSLADA
RAQUEL	TOLEDO

Pag. :143

Mas ejemplos

- Seleccionamos los nombres de los alumnos de la tabla ALUM que están en NUEVOS o están en ANTIGUOS:

```
SELECT NOMBRE FROM ALUM WHERE NOMBRE IN
       (SELECT NOMBRE FROM NUEVOS UNION
        SELECT NOMBRE FROM ANTIGUOS);
```

- La consulta se puede realizar sin usar UNION, recurriendo al operador OR:

```
SELECT NOMBRE FROM ALUM WHERE
       NOMBRE IN (SELECT NOMBRE FROM NUEVOS)  OR
       NOMBRE IN (SELECT NOMBRE FROM ANTIGUOS);
```

Pag. :144

Mas ejemplos

- Para seleccionar los nombres de los alumnos de la tabla ALUM (actuales) que hayan estado matriculados anteriormente (ANTIGUOS) y vayan a estarlo en el futuro(NUEVOS) :

```
SELECT NOMBRE FROM ALUM INTERSECT
SELECT NOMBRE FROM NUEVOS INTERSECT
SELECT NOMBRE FROM ANTIGUOS;
```

- La consulta se puede realizar sin usar UNION, recurriendo al operador OR:

```
SELECT NOMBRE FROM ALUM WHERE
      NOMBRE IN (SELECT NOMBRE FROM NUEVOS)  AND
      NOMBRE IN (SELECT NOMBRE FROM ANTIGUOS);
```

Pag. :145

Actividades

- Realizar las actividades:

6.UD04.PRACTICA 6-CONSULTAS NBA

7.UD04.PRACTICA 7-CONSULTAS VARIADAS JARDINERÍA

8.UD04.PRACTICA 8-SUB-VEHICULOS

9.UD04.PRACTICA 9-SUB-VENTAS

Pag. :146

Vistas

- Una **vista** es una tabla sin contenido, totalmente virtual, que devuelve las filas que son el resultado de ejecutar una consulta SQL.
- La diferencia con una consulta ejecutada directamente es que, mientras cada sentencia SQL enviada al SGBD tiene que pasar por un proceso de compilación, la vista es una consulta cuya definición· ha sido almacenada previamente y que ya ha sido compilada, siendo por tanto el tiempo de ejecución bastante menor.
- También tiene una implicación importante en el hecho de que un usuario podría no tener acceso, a la información de varias tablas y, sin embargo, sí tener acceso a la vista que consulta esas tablas, proporcionando de esta manera un acceso controlado solo a determinadas filas y columnas de esas tablas.

Pag. :147

Vistas

- Por ejemplo, en una tabla de clientes, un usuario de una oficina de Madrid podría tener solo acceso a la información de los clientes de Madrid, y tan solo a ciertos campos.
- De esta manera, no tendría acceso a ningún campo de la tabla de clientes y, sin embargo, podría tener acceso a una vista que consulte aquellos clientes cuya provincia sea Madrid.

Pag. :148

Vistas: CREATE VIEW

- La sintaxis para crear una vista es la siguiente:

```
CREATE [OR REPLACE] VIEW
    [esquema.]nombre_vista [(lista_columnas)] AS
    sentencia_select;
```

- Para borrarla:

```
DROP VIEW nombre_vista;
```

Pag. :149

Ejemplo: Vistas

- Otro ejemplo:

```
create view jugadoresbulls as
   select nombre, posicion
     from   jugadores
     where  upper(nombre_equipo)='BULLS';
```

- Se utilizaría:

```
select * from jugadoresbulls ;
```

- Ejemplo:

```
create or replace view emple_loc as
      select apellido, salario, loc
      from emple e,depart d
      where e.dept_no=d.dept_no;
```

- Para utilizarla sería:

```
select * from emple_loc;
```

Pag. :150

Ejemplo: Vistas

- Ejemplo:

```
CREATE OR REPLACE VIEW PEDIDOS_CLIENTE AS
      SELECT C.NOMBRE_CLI , P.FECHA_PEDIDO,
                SUM (P.UNIDADES*A.PRECIO) TOTAL_PEDIDO
      FROM PEDIDOS P, CLIENTES C, ARTICULOS A
      WHERE P.IDCLIENTE=C.IDCLIENTE
            AND A.IDARTICULO=P.IDARTICULO
      GROUP BY NOMBRE_CLI, FECHA_PEDIDO;
```

- Se utilizará

```
SELECT NOMBRE_CLI,FECHA_PEDIDO, TOTAL_PEDIDO
      FROM PEDIDOS_CLIENTE
      WHERE  FECHA_PEDIDO='10/11/2013';
```

Pag. :151

Operaciones DML a través de vistas

- Para poder realizar sentencias DML(Insert, Delete, Update) a través de una vista: esta debe estar creada con filas de una sola tabla.
- **Delete:** No se puede **eliminar** una fila si la vista contiene:
  - Funciones de grupo (sum,avg,count..)
  - La cláusula GROUP BY
  - La cláusula DISTINCT.
- **Update:**Actualización de filas a través de una vista: Para actualizar filas en una tabla a través de una vista, esta ha de estar definida según las restricciones anteriores y , además, ninguna de las columnas que se va a actualizar se habrá definido como una expresión o una función.
- **Insert:** Inserción de filas a través de una vista: Para insertar filas en una tabla a través de una vista se han de tener en cuenta todas las restricciones anteriores y, además, todas las columnas obligatorias de la tabla asociada deben estar presentes en la vista.

Pag. :152

Práctica 10

- Realizar la siguiente práctica:

10.Ud4.PRACTICA 10-VISTAS

Pag. :153