UD 04 - REALIZACIÓN DE CONSULTAS

1. Introducción

SQL nació a partir de la publicación "A relational model of data for large shared data banks" de Edgar Frank.
IBM aprovechó el modelo que planteó Codd para desarrollar un lenguaje acorde con el modelo relacional (SEQUEL). Este lenguaje con el tiempo se convirtió en SQL. En 1979 la empresa Relational Software sacó al mercado la primera implementación comercial de SQL. Esa empresa es hoy día Oracle.
En 1992 ANSI e ISO completaron la estandarización de SQL y se definieron las sentencias básicas que se debía contemplar para que SQL fuera estándar. (ANSI-SQL o SQL92)
Hoy día todas las bases de datos cumplen este estándar, aunque cada fabricante puede añadir sus propias mejoras.

Después de la DDL (Data Definition Language) viene la DML (Data Manipularion Language): El conjunto de sentencias orientadas a consultas y al manejo de los datos creados.

En Oracle: Las sentencias pueden ser ejecutadas desde el Application Express de oracle usando SQL Workshops y eligiendo SQL Commands. También se puede desde el entorno de SQL *Plus. Run SQL Command Line. El primer paso es conectarse usando un nombre de usuario que tenga permisos necesarios para la tabla deseada (Orden CONNECT seguida del nombre de usuario)

En MySQL: Las sentencias pueden ser ejecutadas desde cualquier cliente gráfico (Workbench de MySQL por ejemplo) o desde la línea de comandos (Command Line Client), pedirá la contraseña del usuario root.

En ambos casos debe iniciarse o arrancarse el servidor de BBDD y conectar o abrir conexión con el servidor con un usuario con permisos necesarios
`mysql -u root -p` // Usuario root y pide la contraseña

2. La sentencia SELECT

La sentencia SELECT consta de 4 cláusulas:

- Cláusula **SELECT** (obligatoria) seguida de los nombres de las columnas separados por comillas simples
- Cláusula **FROM** (obligatoria) seguida del nombre de las tablas del que proceden las columnas de arriba
- Cláusula **WHERE** (opcional) seguida de criterio de selección o condición
- Cláusula **ORDER BY** (opcional) seguida de criterio de ordenación

(Después de SELECT se puede poner `ALL` (por defecto) o `DISTINCT` para que suprima filas con resultados iguales)

2.1. Cláusula SELECT

- Puede anteponerse el nombre de las tablas de la que proceden
- Pueden incluirse todas las columnas (con un *)
- Pueden ponerse alias a los nombres de las columnas (recomendado ya que estas son las cabeceras de presentación de los datos obtenidos). Debe ponerse entre comillas dobles a continuación del nombre de la columna
- Puede sustituirse el nombre de las columnas por constantes, expresiones o funciones SQL.

```
SELECT nombre "Nombre de la sala" FROM SALA;
SELECT 4*3/100 "MiExpresion", capacidad FROM SALA;
```

2.2. Cláusula FROM

Se definen los nombres de la tabla.

- Si es más de una deben aparecer separadas por comas (se le llama consulta combinada o join: habrá que aplicar una condición de combinación a través del WHERE).
- Puede asociarse un alias a las tablas para abreviar
- En Oracle se puede añadir el nombre del usuario que es propietario de las tablas `USUARIO.TABLA` así se distinguirá entre las tablas de un usuario y de otro (pueden tener el mismo nombre)

2.3. Cláusula WHERE

Se pondrá la condición que han de cumplir las filas.
El criterio de búsqueda o condición puede tener complejidad variable conjugándose operadores de diversos tipos, funciones, expresiones...

2.4. Cláusula HAVING

Si existen grupos de registros o si se quieren seleccionar datos de la tabla restringiéndolos a cumplir una condición dada por operadores aritméticos de agrupación (SUM, MAX, COUNT, MIN, AVG...) ya no puede usarse WHERE sino que debe usarse HAVING.

```
SELECT  nombre, apellido1, apellido2
FROM ASISTENTE
GROUP BY NOMBRE;

SELECT nombre
FROM empleado
GROUP BY nombre
HAVING AVG(salario) < 200;
```

Normalmente `HAVING` va acompañado de `GROUP BY` porque opera sobre los grupos que esta devuelve.

2.5. Ordenación de registros. Cláusula ORDER BY.

Para especificar criterio de ordenación. Puede ordenarse por tipo carácter, número o fecha.
Es posible ordenar por más de una columna o a través de una expresión creada por columnas, una constante (aunque no tenga sentido) o funciones SQL.
Se sigue el primer criterio y en caso de empate el segundo, el tercero...

```
SELECT nombre, apellido1, apellido2
FROM ASISTENTE
ORDER BY apellido1, apellido2, nombre;
```

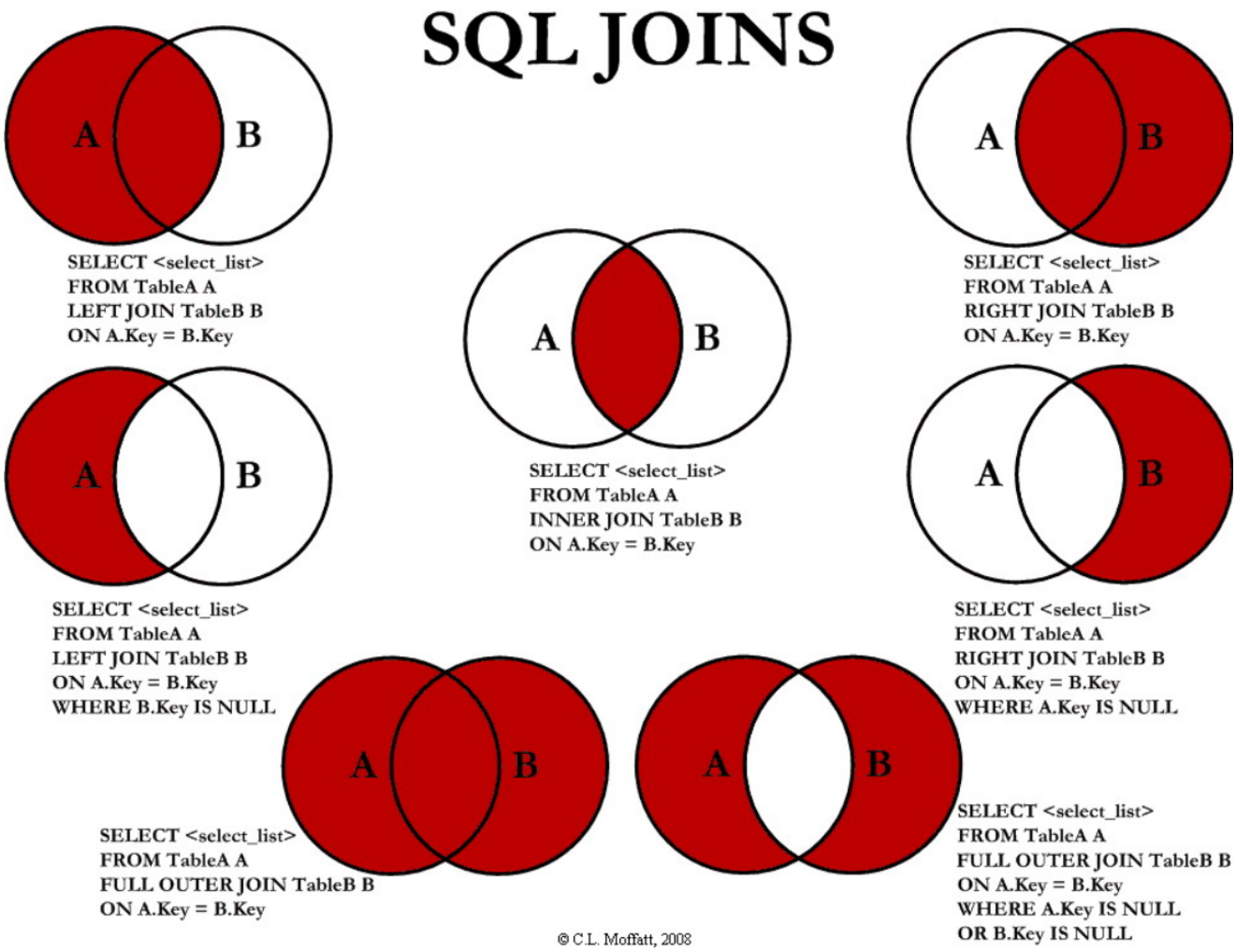
Se pueden referenciar los campos por su posición en la lista de selección.
Si se pone un número mayor a la cantidad de campos aparece un mensaje de error y la sentencia no se ejecuta.

```
FROM ASISTENTE
ORDER BY 4; -- Ordena por la empresa
```

Se puede añadir:

- `DESC` De mayor a menor
- `ASC` De menor a mayor

2.6. Tipos de JOIN



INNER JOIN: Devuelve las filas que tengan columnas cuya condición coincida en ambas tablas.

```
SELECT d.nombre, e.nombre
FROM departamento d
JOIN empleado e
ON d.id = e.departamento;
```

LEFT JOIN: Devuelve todas las filas de la tabla de la izquierda y las coincidentes de la tabla derecha. En los campos no coincidentes de la izquierda aparece NULL.

Todos los proyectos, apareciendo los empleados que están asignados a algún proyecto.

```
SELECT p.nombre, e.nombre
FROM proyecto p
LEFT JOIN empleado e
ON p.id = e.proyecto;
```

Proyectos sin empleado

```
SELECT p.nombre
FROM proyecto p
LEFT JOIN empleado e
ON p.id = e.proyecto
AND e.proyecto IS NULL;
```

RIGHT JOIN: Devuelve todas las filas de la tabla de la derecha más las coincidentes con la de la tabla izquierda. En los campos no coincidentes de la derecha aparece NULL.

Todos los empleados, apareciendo los proyectos a los que está asignado algún empleado.

```
SELECT p.nombre, e.nombre
FROM proyecto p
RIGTH JOIN empleado e
ON p.id = e.proyecto;
```

Empleados sin proyecto.

```
SELECT p.nombre
FROM proyecto p
RIGTH JOIN empleado e
ON p.id = e.proyecto
AND p.id IS NULL;
```

OUTER JOIN o **FULL OUTER JOIN**: Todas las filas de la tabla de la derecha y todas las filas de la tabla de la izquierda. En los campos no coincidentes aparece NYLL.
El FULL OUTER JOIN no existe en MySQL, ahí debe hacerse `UNION` entre LEFT JOIN y RIGTH JOIN.

Listas todos los proyectos y todos los empleados

```
SELECT p.nombre, e.nombre
FROM proyecto p
OUTER JOIN empleado e
ON p.id = e.proyecto;

-- En MySQL
SELECT nombre, e.nombre
FROM proyecto p
LEFT JOIN empleado e
ON p.id = e.proyecto
UNION
SELECT p.nombre, e.nombre
FROM proyecto p
RIGTH JOIN empleado e
ON p.id = e.proyecto.
```

3. Operadores

3.1. Operadores de comparación o relacionales

Para comparar dos variables.

```
= != < > <= >= IN, NOT IN, BETWEEN, NOT BETWEEN, LIKE '%ddfs_' (% sustituye a un conjunto de caracteres _ a un caracter), IS NULL, IS NOT NULL.
```

Cuando se hace un ORDER BY los valores NULL aparecen en primer lugar con ASC y al final con DESC.

Se pueden usar expresiones regulares con `REGEXP`

3.2. Operadores aritméticos y de concatenación

Para realizar cálculos con valores numéricos `+ - * /`
La concatenación se hace con `CONCAT` en MySQL o con `||` en Oracle

```
-- MySQL
SELECT CONCAT(cdemp," ", nombre)
FROM empleado;
-- Oracle
SELECT nombre, apellido1 || ' ' || apellido2
FROM ponente;
```

3.3. Operadores lógicos

Para evaluar más de una expresión lógica. `AND OR NOT`

3.4. Precedencia

El orden de precedencia es:

- Multiplicación y división
- Suma y resta
- Concatenación (En el caso de Oracle)
- Todas las comparaciones
- Operadores (IS NULL, IS NOT NULL, LIKE, BETWEEN...)
- NOT
- AND
- ORD

4. Consultas calculadas

Los campos calculados se obtienen a través de la sentencia `SELECT` poniéndose la expresión deseada.
No modifica los valores originales delas columnas, ni de la tabla de resultados, solo muestra una columna nueva con los valores calculados.

```
SELECT tema, precio, precio + 10 AS nuevoPrecio
FROM CONFERENCIA;

SELECT nombre, salario, salario + 25
FROM empleado;
```

(También pueden ir en WHERE)

5. Funciones

Hay funciones ya creadas que facilitan la creación de consultas más complejas.
Estas varían según el SGBD. Realmenteon operaciones que se realizan sobre los datos y realizan un cálculo determinado.

Oracle proporciona la tabla DUAL con un único campo llamado DUMMY y una única FILA.
MySQL proporciona para pruebas las bases de datos de nombre Test. Esta base de datos no contiene ninguna tabla.

Veamos, sobre todo, las de Oracle.

5.1. Funciones numéricas

- Valor absoluto - **ABS(n)**:
  Oracle: `SELECT ABS(-17) FROM DUAL;` -- Resultado 17
  MySQL: `SELECT ABS(-17);` -- Resultado 17 (No se necesita consultar ninguna tabla)
- e^n - **EXP(n)**:
  `SELECT EXP(2) FROM DUAL;` -- Resultado 7,38
- CEIL(n) Entero inmediatamente superior o igual a n
- FLOOR(n) Entero inmediatamente inferior o igual a n
- MOD(m,n) Resto dividir m y n
- POWER(valor,exponente) Potencia valor ^ exponente
- ROUND(n, decimales) Redondea n al siguiente con los decimales indicados
- SQRT(n) Raíz cuadrada
- TRUNC(m,n) Trunca número a la cantidad especificada. Si no se pone segundo argumento se truncan todos los decimales.
- SIGN(n) Si es positivo retorna 1. Si es negativo -1. Si es 0 retorna 0.

5.2. Funciones de cadena de caracteres

Oracle

- CHR(n) Carácter cuyo valor codificado es n. `SELECT CHR(81) FROM DUAL;` -- Resultado: Q
- ASCII(n) Valor ascii. `SELECT ASCII(\`Q`) FROM DUAL;` -- Resultado 81
- CONCAT(cad1, cad2) Concatena
- LOWER(cad) Pasa a minúscula
- UPPER(cad) Pasa a mayúscula
- INITCAP(cad) Devuelve con primer carácter en mayúscula
- LPAD(cad1, n, cad2) Devuelve cad1 con longitud n, ajustada a la derecha, rellenando por la izquierda con cad2. `****M`
- RPAD(cad1, n, cad2) Devuelve cad1 con longitud n, ajustada a la derecha, rellenando por la derecha con cad1. `M****`
- REPLACE(cad,ant,nue) Replaza en la cadena, la ant por la nue
- SUBSTR(cad, m,n) Divide la cadena cad compuesta por n caracteres a partir de la posición m
- LENGTH(cad): Longitud de la cadena. Ojo. Van entre comillas simples!!!!!!!!!!
- TRIM(cad): Suprime espacios en blanco a izquierda y a derecha y dobles del interior
- LTRIM(cad): Solo a izquierda
- RTRIM(cad): Solo a derecha
- INSTR(cad, cadBuscada `,[posInicial, nAparicion]`): Posición en la que se encuentra la cadena buscada en la cadena inicial cad. Se puede empezar a buscar en una posición concreta e indicar un número de aparición. Si no encuentra nada devuelve 0.

MySQL

- ASCII(n)
- CONCAT(cad1, cad2)
- LOWER(cad)
- UPPER(cad)
- INSTR(cadena,subcadena): Posición de la primera ocurrencia
- RPAD(cad1, n, cad2)
- REPLACE(cad, ant, nue)
- SUBSTR(cad, m, n)
- LENGTH(cad)
- TRIM(cad)
- LTRIM(cad)
- RTRIM(cad)

5.3. Funciones de manejo de fechas

En Oracle los dos datos para manejar fechas son `DATE` y `TIMESTAMP`

Pueden realizarse operaciones como suma o diferencia.

- SYSDATE. Fecha y hora actual
- SYSTIMESTAMP. Fecha y hora actual en formato TIMESTAMP
- ADD_MONTHS(fecha,n)
- MONTHS_BETWEEN(fecha1, fecha2)
- LAST_DAY(fecha) Último día del mes al que pertenece la fecha (DATE)
- NEXT_DAY(fecha, d) Día que corresponde al añadir a la fecha el día d.
  NEXT_DAY('29/04/2024','MARTES')
- EXTRACT(valor FROM fecha) Extrae valor concreto: Ejemplo `SELECT EXTRACT(MONTH FROM SYSDATE) FROM DUAL`
  Con + y - se pueden sumar y restar días, dandola fecha correspondiente.

En MySQL también hay DATE y TIMESTAMP (mismo formato que DATETIME)

Entre sus funciones más comunes:

- CURDATE() o NOW(). Fecha actual y fecha y hora actual, respectivamente
- CURRENT_TIMESTAMP() Fecha y hora actual en formato DATETIME
- ADDDATE('2024-11-01',4) añade a la fecha los días indicados
- DATETIFF('fecha1','fecha2') Días de diferencia entre fechas
- DATE_FORMAT (feccha,formato) Fecha formateada según formato especificado '%d/%m/%Y' (19/04/2024)
- DAY(fecha) Día del mes de una fecha
- MONTH(fecha) Mes de una fecha
- YEAR(fecha) Año de una fecha
- DAYNAME(fecha) Nombre del día de una fecha
- MONTHNAME(fecha) Nombre del mes de una fecha
  Las funciones pueden emplearse enviando como argumento el nombre de un campo o columna de tipo fecha.

5.4. Funciones de conversión

Oracle y MySQL convierte automáticamente datos de forma que el resultado de la expresión tenga sentido (texto -> número y al revés; texto -> fecha y viceversa), pero hay ocasiones en las que se querrá realizar esas conversiones de modo explícito.

Oracle

- `TO_NUMBER(cad, formato)`. Convierte textos en números. Se usa para dar un formato concreto.

Formatos para números y su significado.

Símbolo	Significado
9	Posiciones numéricas. Si el número que se quiere visualizar contiene menos dígitos de los que se especifican en el formato, se rellena con blancos.
0	Visualiza ceros por la izquierda hasta completar la longitud del formato especificado.
$	Antepone el signo de dólar al número.
L	Coloca en la posición donde se incluya, el símbolo de la moneda local (se puede configurar en la base de datos mediante el parámetro NSL_CURRENCY)
S	Aparecerá el símbolo del signo.
D	Posición del símbolo decimal, que en español es la coma.
G	Posición del separador de grupo, que en español es el punto.

- `TO_CHAR(d, formato)`. Convierte número o fecha d a cadena de caracteres. Se usa para fechas ya que de número a texto se hace de forma implícita.
- `TO_DATE(cad, formato)`. Convierte textos a fechas, indicando el formato.

Formatos para fechas y su significado.

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

MySQL

`CONVERT(expresion,tipo_dato)` Devuelve la expresión convertida al tipo de dato suministrado. Ej.: `SELECT CONVERT(21, DECIM(6,2));` --Devuelve 21.00
`CAST(expresion AS tipo_dato)`. Análoga a la anterior. `SELECT CAST(21 AS DECIMAL(6,2));` -- Devuelve 21.00.

5.5. Otras funciones: NVL y DECODE - IFNULL

Las **funciones con nulos** permiten permiten manejar los campos NULL

Oracle
`NVL(valor, expr1)`: Si el valor es nulo, devuelve expr1 (Debe ser del mismo tipo de valor)

`DECODE(expre1, cond1, valor1, [expr2, valor2, cond2], default)` Se evalúa la expr1, si se cumple devuelve el valor1. Si no, evalúa la siguiente y así hasta que una de las condiciones se cumpla. Si ninguna se cumple se devuelve el default.

MySQL
`IFNULL(expr1, expr2)`: Devuelve expr1 si es distinta de null, en caso contrario devuelve expr2.
`IF(expr1, expr2, exp3)` Devuelve expr2 o expr3 en función de si expr1 es verdadera o falsa.

6. Consultas de resumen

Las funciones de agrupamiento o agregado toman un grupo de datos y producen un único dato que resume el grupo. Todas tienen una estructura `FUNCION [ALL|DISTINCT] EXPRESION` y debe tenerse en cuenta que:

- ALL indica que deben tomarse todos los valores de la columna (valor por defecto)
- DISTINCT indica que se considerarán todas las repeticiones del mismo valor como uno solo (solo considera valores distintos).
- En la expresión nunca puede aparecer una función de agregado ni una subconsulta
- Todas las funciones se aplican a las filas del origen de datos una vez ejecutada la cláusula WHERE (si la tuviéramos)
- Todas las funciones (salvo COUNT) ignoran los valores NULL
- Podemos encontrar una función de agrupamiento en cualquier lugar en el que pueda aparecer el nombre de una columna. Puede formar parte de una expresión pero no se pueden anidar funciones.
- No se pueden mezclar funciones de columnas con nombres de columna ordinarios, salvo ciertas excepciones.

¡Recuerda que se puede poner DISTINCT si quieres que solo coja valores diferentes!

6.1. Funciones de agregado: SUM y COUNT

Función SUM(expresion): Suma de los valores. Solo para números.

```
SELECT SUM(precio) FROM CONFERENCIA;
SELECT SUM(DISTINCT precio) FROM CONFERENCIA;
```

Función COUNT(expresion): Cuenta elementos de un campo. Cuenta el total de filas pero no cuenta los registros con NULL, **salvo que se tenga en la expresión el comodin (*).** Puede contar cualquier tipo de dato incluido texto.

```
SELECT COUNT(nombre) FROM PONENTE;
SELECT COUNT(*) FROM PONENTE;
SELECT (DISTINCT nombre) FROM PONENTE;
```

6.2. Funciones de agregado: MIN y MAX

Función MIN(expresion): Valor mínimo de la expresión sin considerar nulos. Se puede incluir nombre de un campo, constante, función (no otras funciones agregadas)

Función MAX(expresion): Valor máximo de la expresión sin considerar NULL. Idem a MIN

6.3. Funciones de agregado: AVG, VAR, STDEV, STDEVP

Función AVG: Devuelve el promedio, omitiendo los NULL. Se aplica a campos de número, el tipo de dato del resultado varía según necesidades del sistema para representar el valor.

```
SELECT AVG(gratificacion) AS MEDIA_GRATIFICACIONES FROM PARTICIPAR;
SELECT AVG(nhoras) FROM trabaja;
```

Función VAR: Devuelve la varianza estadística

Función STDEV: Devuelve la desviación típica de una muestra

Función STDEVP: Devuelve la desviación típica de una población

Resumidamente las funciones de agregación: Toman un grupo de datos de una columna, producen un único dato que resume el grupo, usan una función de agregado en una consulta que la convierte en consulta de resumen.

7. Agrupamiento de registros `GROUP BY`

Para agrupaciones según un determinado campo se usa `GROUP BY`

- En `GROUP BY` se colocan las columnas por las que se va agrupar. En `HAVING` la condición que han de cumplir los grupos para que se realice la consulta.
- Es importante el orden: WHERE; GROUP BY; HAVING; ORDER BY
- Las columnas que aparecen en el SELECT y que no aparecen en GROUP BY deben tener una función de agrupamiento o producirá error.
- Otra opción es poner en la cláusula GROUP BY las mismas columnas que aparecen en el SELECT

```
SELECT cddep, count(cddep) FROM empleado GROUP BY cddep;

SELECT cddep, COUNT(cddep) FROM empleado WHERE salario > 1500 GROUP BY cddep HAVING
COUNT(cddep) >2;

SELECT empresa, COUNT(codigo) AS NUM_ASISTENTES FROM ASISTENTE WHERE empresa IS NOT NULL
GROUP BY empresa;

SELECT <code>empresa, COUNT(codigo) AS NUM_ASISTENTES FROM ASISTENTE WHERE empresa IS NOT
NULL GROUP BY empresa HAVING empresa = 'BigSoft' OR empresa= 'ProgConsulting';
```

8. Consultas multitabla

A la hora de coger datos de varias tablas en un único SELECT se podrán hacer operaciones de composición interna y de composición externa.

MySQL
En SQL 1999 se especifica sintaxis para consultar tablas que MySQL incorpora para separar las condiciones de asociación respecto a la selección de registros:

```
SELECT tabla1.columna1, tabla1.columna2, …, tabla2.columna1, tabla2.columna2, …
FROM tabla1
     [JOIN tabla2 USING (columna) |
     [JOIN tabla2 ON (tabla1.columna=tabla2.columna)] |
     [LEFT | RIGTH  OUTER JOIN tabla2 ON (tabla1.columna=tabla2.columna)]
```

Oracle

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

8.1. Composiciones internas

Combinar dos o más tablas sin ninguna restricción da lugar al producto cartesiano (todas las combinaciones de todas las filas de esas dos tablas).

Se indica poniendo en la cláusula FROM las tablas que se quieren poner separadas por comas. Se obtienen todas las posibles combinaciones de filas. A más tablas, mayor es el resultado final (operación costosa).

Esta operación no es la más utilizada porque lo normal es que se quiera discriminar de alguna forma (asociar tablas - join)

Para hacer composición interna se parte del producto cartesiano y se eliminan las filas que no cumplen la condición de composición. Lo importante es emparejar campos que tengan valores iguales.

- Pueden combinarse tantas tablas como se quiera
- El criterio de combinación puede estar formado por más de una pareja de columnas
- En la SELECT pueden citarse columnas de ambas tablas condicionen o no la combinación
- Si las columnas se llaman igual, debe identificarse especificando la tabla de procedencia o un alias

Las columnas que aparecen en el WHERE se llaman **columnas de emparejamiento** porque emparejan las filas de las dos tablas. No tienen por qué estar en la lista de selección. Emprejamos tablas relacionadas entre sí y una de las columnas de emparejamiento será la clave principal en su tabla.

También puede combinarse una tabla consigo misma pero poniendo de forma obligatoria un alias a uno de los valores a repetir.

```
SELECT e.cdemp, e.nombre, e.fecha_ingreso, d.nombre, d.ciudad
FROM empleado e, departamento d
WHERE e.cddep= d.cddep;

SELECT e.cdemp, e.nombre, je.nombre
FROM empleado e, empleado je
WHERE e.cdjefe = je.cdemp;

SELECT referencia, tema, nombre, capacidad FROM CONFERENCIA, SALA<br />WHERE
CONFERENCIA.sala= SALA.Nombre;

SELECT DISTINCT A.codigo, A.nombre FROM ASISTENTE A,CONFERENCIA C, ASISTIR AT, WHERE
A.codigo= AT.codasistente AND C.referencia=AT.refconferencia AND C.Tema LIKE '%DATOS%';
```

8.2. Composiciones externas

A veces interesará seleccionar algunas filas de una tabla aunque estas no tengan correspondencia con las filas de otra tabla.

En MySQL
Composición externa LEFT OUTER JOIN o RIGHT OUTER JOIN

```
SELECT tabla1.columna1, tabla1.columna2, …, tabla2.columna1, tabla2.columna2, …
FROM tabla1
[LEFT | RIGTH  OUTER JOIN tabla2 ON (tabla1.columna=tabla2.columna)]
```

Obtener un listado con los códigos y nombres de los distintos departamentos y el código, nombre y salario de sus empleados. Ten en cuenta que deben aparecer todos los departamentos aunque no tengan asignados empleados.

```
SELECT d.cddep, d.nombre, e.cdemp, e.nombre, e.salario, e.cddep
FROM departamento d LEFT OUTER JOIN  empleado e ON e.cddep=d.cddep;
```

En Oracle
Se añade un signo entre paréntesis (+) en la igualdad entre campos.
El carácter (+) va detrás del nombre de la tabla en la que se desean aceptar valores nulos

Obtener un listado con los nombres de los distintos departamentos y sus jefes con sus datos personales. Ten en cuenta que deben aparecer todos los departamentos aunque no tengan asignado ningún jefe.

```
SELECT D.NOMBRE_DPTO, D.JEFE, E.NOMBRE, E.APELLIDO1, E.APELLIDO2
FROM DEPARTAMENTOS D, EMPLEADOS E
WHERE D.JEFE = E.DNI(+);
```

8.3. Composiciones en la versión SQL99

SQL99 tiene mejoras en la sintaxis para composiciones en consulta.

- **CROSS JOIN**: Producto cartesiano de las filas de ambas tablas. No es necesario WHERE
- **NATURAL JOIN**: Detecta las claves de unión automáticamente basándose en el nombre. Funciona incluso si no están definidas claves primarias o ajenas.
- **JOIN USING:** Establecer relaciones indicando qué campos se quieren relacionar
- **JOIN ON:** Unir tablas en la que los nombres de columna no coinciden en ambas o se necesitan asociaciones más complicadas.
- **OUTER JOIN:** Eliminar el uso del signo (+) en composiciones externas usando OUTER JOIN (Oracle)
- **LEFT OUTER JOIN:** Composición externa izquierda. Todas las filas de la izquierda se devuelven aunque no haya columna en tablas combinadas.
- **RIGTH OUTER JOIN:** Composición externa derecha. Todas las filas de la derecha se devuelven aunque no haya columna en tablas combinadas.
  -FULL OUTER JOIN: Composición externa se devuelven todas las filas de los campos no relacionadas en ambas tablas (Oracle)

```
-- MySQL
SELECT e.cdemp, e.nombre, e.fecha_ingreso, d.nombre, d.ciudad
FROM empleado e
INNER JOIN departamento d ON e.cddep= d.cddep;

SELECT e.cdemp, e.nombre, je.nombre
FROM empleado e
INNER JOIN empleado je ON  e.cdjefe = je.cdemp;

-- Oracle
SELECT referencia, tema, nombre, capacidad FROM CONFERENCIA JOIN SALA ON <
CONFERENCIA.sala= SALA.nombre;

SELECT  D.NOMBRE_DPTO, D.JEFE, E.NOMBRE, E.APELLIDO1, E.APELLIDO2
FROM DEPARTAMENTOS D LEFT OUTER JOIN EMPLEADOS E ON ( D.JEFE = E.DNI);
```

Recomendación: Usar sintaxis de SQL99 usando JOIN tanto en composiciones internas como externas.

9. Otras consultas multitablas: Unión, intersección y diferencia de consultas

- **UNION:** Combina las filas del prime SELECT y del segundo SELECT desapareciendo duplicados.

```
SELECT NombreCia, Ciudad FROM  PROVEEDORES WHERE Pais = 'Alemania'
UNION
SELECT NombreCia, Ciudad  FROM CLIENTES WHERE Pais = 'Alemania';
```

- **INTERSECT:** Devuelve las que aparecen en ambos SELECT desapareciendo duplicados

```
SELECT nombre, domicilio FROM ingles INTERSECT
SELECT nombre, domicilio FROM frances INTERSECT
SELECT nombre, domicilio FROM portugues;
```

- **MINUS:** Devuelve filas que están en el prime SELECT pero no en el segundo SELECT. Las filas duplicadas del primer SELECT desaparecen.

```
SELECT nombre, domicilio FROM INGLES
MINUS
SELECT nombre,domicilio FROM PORTUGUES;
```

Es importante que se use en los dos SELECT el mismo número y tipo de columnas y en el mismo orden

10. Subconsultas

La subconsulta puede ir dentro de las claúsulas WHERE, HAVING o FROM.

El tipo de datos que devuelve la subconsulta y la columna con la que se compara.
Si se quiere que la subconsulta devuelva más de un valor y comparar el campo que se tiene condichos valores deben usarse instrucciones especiales entre el operador y la consulta:

- ANY: Compara con cualquier fila de la columna. Instrucción válida si hay un registro que permite que la comparación sea cierta.
- ALL: Compara con todas. Instrucción válida si es cierta en todas
- IN. Comprueba si el valor se encuentra en el resultado de la subconsulta.
- NOT IN: Compara si no s encuentra.

Obtener el asistente más joven.

```
SELECT nombre
FROM asistente
WHERE fechanac <= ALL (SELECT fechanac FROM asistente);
```

Obtener nombre, descripción y precio que superen o igualen el precio medio de las rutas

```
SELECT nombre, descripción, precio
FROM ruta
WHERE precio>= (SELECT AVG(precio) FROM ruta);
```

dni y nombre de empleados que no han ido a un restaurante

```
SELECT DNI, nombre
FROM empleado
WHERE DNI NOT IN (SELECT empleado FROM restaurante);
```

11. Preparación de Oracle y MySQL

Oracle
Se crea un espacio de trabajo (workspace) con el usuario y la contraseña. Este usuario será el administrador del espacio de trabajo y tendrá permisos para crear tablas.

Con la línea de comandos de SQL se puede ejecutar el archivo descargado
Inicio -> Todos los programas -> Oracle Database 11g Express Edition -> Run SQL Command Line.

```
connect TAREA/admin
@Ruta_del_archivo/archivo.sql
```

Se crean las tablas y se insertan los datos.

Después nos podemos desconectar con `disconnect`

MySQL
Con el cliente gráfico de Workbench podemos iniciar sesión o conectarnos al servidor de MySQL.
Puede cargarse el script. Pulsar en File / Run SQL o sobre el rayo amarillo y se ejecutará el script.
Se actualiza el cliente pulsando la flecha de recargar y se verá la nueva base de datos. O escribir `SHOW DATABASES`