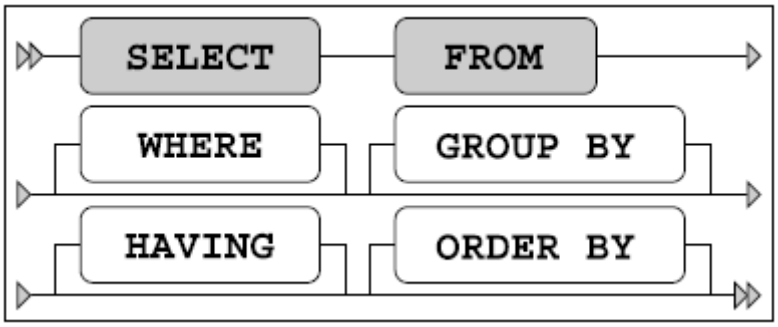
UNIDAD DE TRABAJO 2: BASES DE DATOS RELACIONALES

TEMA 6: CONSULTAS (parte 1)

6.1. La orden SELECT



Sabemos que:

FROM	¿Qué tablas necesitamos para recuperar datos?
WHERE	¿Cuál es la condición para filtrar las filas?
GROUP BY	¿Cómo debemos agrupar las filas?
HAVING	¿Cuál es la condición para filtrar los grupos?
SELECT	¿Qué columnas queremos que se vean en el resultado?
ORDER BY	¿En qué orden?

En el **WHERE** podemos usar varios operadores (como BETWEEN, LIKE, IN, CASE, NOT, AND, OR), y hacer la condición tan complicada como nosotros queremos.

1. Realizar una consulta de todos los departamentos

```
select * from departments;
```

2. Seleccionar el nombre, las iniciales, trabajo y salario de los empleados del departamento 30.

```
select ename, init, job, msal
from employees
where deptno = 30;
```

Alias de columna:
Cuando no queremos que en el resultado se muestre el nombre de la columna que tenemos en la tabla (porque por ejemplo no describa bien el resultado en el

contexto de la consulta) podemos usar alias de columnas. Simplemente escribiremos el nombre de la columna seguido por el alias sin separarlo por comas.

3. Seleccionar el nombre, las iniciales, trabajo y salario de los empleados del departamento 30 pero que en la columna del salario en vez de usar msal en la salida, usar: salary

```
select ename, init, msal salary
from employees
where deptno = 30;
```

La claúsula DISTINCT:

A veces, los resultados contienen filas duplicadas. Podemos eliminar dichas filas usando DISTINCT después del select:

4. Seleccionar los trabajos y los departamentos de los empleados:
(Comprobar los resultados si lo hacemos con el DISTINCT y si lo hacemos sin él).

```
select DISTINCT job, deptno
from employees;
```

Expresiones en las columnas:

Podemos especificar expresiones en el SELECT. Por ejemplo:

5. Mostrar el rango salarial de los empleados (para ello haremos la diferencia entre el mayor salario y el menor):

```
select grade, upperlimit - lowerlimit
from salgrades;
```

6. Concatenar los nombres de los empleados con sus iniciales en una sola columna y calcular el sueldo anual multiplicando el sueldo mensual por 12.

```
select init||' '||ename name, 12 * msal yearsal
from employees
where deptno = 10;
```

La tabla DUAL

Es una tabla ficticia con una sola fila y una sola columna. Sirve para hacer llamada a funciones que simplemente evalúan expresiones que hemos puesto en la lista de columnas del select.

Por ejemplo:

```
select 123 * 456 from dual;
select sysdate from dual;
```

6.2 La cláusula WHERE

Sabemos que en le where podemos especificar el filtro para seleccionar filas. Podemos tener condiciones simples o compuestas.

OPERADOR	DESCRIPCIÓN
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
=	Igual
<>	Distinto

7. Seleccionar el nombre, iniciales y salario mensual de los empleados que ganan mas de 3000 euros al mes.

```
select ename, init, msal
from employees
where msal >= 3000;
```

8. Seleccionar el nombre de los departamentos y su dirección que no pertenezcan a Chicago.

```
select dname, location
from departments
where location <> 'CHICAGO';
```

6.3 La cláusula ORDER BY

Ordenar por. Además podemos poner descendente o ascendente. Si no especificamos nada lo muestra ordenado ascendentemente.

9. Seleccionar el número del departamento, el nombre del empleado, sus iniciales y el salario mensual de los empleados que ganen menos de 1500 ordenados por número de departamento y luego por nombre de empleado.

```
select deptno, ename, init, msal
from employees
where msal < 1500
```

```
order by deptno, ename;
```

10. Seleccionar el nombre del empleado, salario anual más comisión llamando a la columna 'yearsal' de los empleados cuyo trabajo sea 'SALESREP' ordenado por salarios anuales descendentemente.

```
select ename, 12*msal+comm as yearsal
from employees
where job = 'SALESREP'
order by yearsal desc;
```

11. Queremos saber las notas del empleado 7788 ordenadas.

```
select evaluation
from registrations
where attendee = 7788
order by evaluation;
```

6.4 AND, OR, and NOT

OR

Se pueden combinar condiciones simples y compuestas mediante el uso de los operadores lógicos AND y OR. Si utiliza Y, se indica que cada fila debe ser VERDADERO para ambas condiciones. Si utilizamos O, sólo una de las condiciones tiene que evaluar en TRUE. Suena bastante fácil, ¿no?

Lo cierto es que usamos las palabras and y or de una manera bastante descuidada en idioma hablado. El oyente entiende fácilmente nuestras intenciones basándose en el contexto, la entonación o el lenguaje corporal. Pero en un lenguaje formal (como SQL) hay un riesgo de cometer errores en la traducción de las preguntas del lenguaje natural.

12. Seleccionar el código, la categoría y la duración de los cursos cuya categoría sea 'BLD' o duren 2 días.

```
select code, category, duration
from courses
where category = 'BLD'
or duration = 2;
```

AND, OR y la precedencia

Hay un posible problema si las condiciones compuestas contienen una mezcla de operadores AND y OR.
¿Qué creéis que sacará esta consulta?

```
select 'is true ' as condition
from dual
where 1=1 or 1=0 and 0=1;
```

Vemos que sí que es verdad.

Para que no haya errores, debemos siempre utilizar los paréntesis:

```
select 'is true ' as condition
from dual
where (1=1 or 1=0) and 0=1;

no rows selected

select 'is true ' as condition
from dual
where 1=1 or (1=0 and 0=1);

CONDITION
---------
is true
```

NOT

Sirve, como su propio nombre indica, para negar una condición. El NOT se coloca antes de la condición: where NOT ……

13. Seleccionar nombre, trabajo y departamento de aquellos empleados que trabajen en un departamento cuyo número de dpto. sea menor o igual que 10.

```
select ename, job, deptno
from employees
where NOT deptno > 10;
```

Lógicamente nos damos cuenta de que esta consulta se podría haber hecho sin el NOT:

```
select ename, job, deptno
from employees
where deptno <= 10;
```

14. Seleccionar código, categoría y duración de los cursos cuya categoría sea "BLD" o duración 2 pero que no sea dicha categoría con duración 2.

```
select code, category, duration
from courses
where (category = 'BLD' or duration = 2)
and not (category = 'BLD' and duration = 2);
```

15. Realizar la siguiente consulta de dos maneras distintas:
Seleccionar los empleados cuyo nombre sea BLAKE o inicial R

```
select * from employees where NOT (ename = 'BLAKE' AND init = 'R')
select * from employees where ename <> 'BLAKE' OR init <> 'R'
```

6.5 BETWEEN, IN and LIKE

A veces con el uso del AND, OR y NOT nos pueden aparecer condiciones realmente complicadas; en esta sección, veremos el que con los operadores BETWEEN, IN y LIKE podemos simplificar algunas de las expresiones.

BETWEEN

Este operador no nos da nuevas posibilidades, sólo una forma de formular condiciones más fácil.

16. Seleccionar el nombre, iniciales y salario de aquellos empleados cuyo salario sea mayor que 1300 y menor que 1600.

```
select ename, init, msal
from employees
where msal between 1300 and 1600;
```

Se puede combinar con el NOT:

```
where msal NOT between 1000 and 2000
where NOT msal between 1000 and 2000
```

17. Realizar la misma consulta de otra manera.

```
select ename, init, msal
from employees
where msal > 1300 AND msal < 1600;
```

IN

Con el operador IN podemos comparar una columna o el resultado de una expresión de columna con una lista de valores. Es una forma más sencilla de escribir una serie de condiciones OR. En vez de escribir: empno=7499 OR empno=7566 OR empno=7788 sólo tendremos que usar una lista IN.

Cuando comparemos cadenas de caracteres, cuidado con las mayúsculas o minúsculas, debe coincidir con lo que hay en la BD. Además hay que ponerlo entre comillas simples.

Ejemplo: Seleccionar los departamentos que estén en Dallas o Boston. (Hacerlo de dos formas, con el IN o con operador lógico).

```
select *
from departments
where location IN ('DALLAS','BOSTON');

select *
from departments
where location='DALLAS' OR location='BOSTON';
```

18. Seleccionar el número de empleado, el nombre y las iniciales de aquellos empleados cuyo número de empleado sea 7499 o 7566 o 7788.

```
select empno, ename, init
from employees
where empno in (7499,7566,7788);
```

Como en el caso del BETWEEN el IN también se puede combinar con el NOT.

19. Seleccionar los registros cuya evaluación no sea ni 3, ni 4 ni 5.

```
select * from registrations
where evaluation NOT IN (3,4,5);
```

20. Realizar la misma consulta con expresiones equivalentes (utilizando el OR y el AND):

```
where NOT evaluation in (3,4,5)
where NOT (evaluation=3 OR evaluation=4 OR evaluation=5)
where evaluation<>3 AND evaluation<>4 AND evaluation<>5
```

LIKE

Normalmente utilizaremos el operador LIKE en la cláusula WHERE de las consultas en combinación con un patrón de búsqueda.

21. Devolver todos los curso que tienen algo que ver con SQL (utilizad el patrón de búsqueda %SQL%).

```
select * from courses
where description LIKE '%SQL%';
```

Hay dos caracteres especialmente importantes cuando utilizamos un patrón de búsqueda en una cadena después del operador LIKE.

- **%:** significa cero, uno o más caracteres
- _: significa exactamente un carácter.

22. Devolver todos los empelados que tengan una A mayúscula en el segundo carácter de su nombre.

```
select empno, init, ename
from employees
where ename like '_A%';
```

También podremos combinarlo con el NOT.

23. Seleccionar los empleados cuyo nombre sea BLAKE

```
select * from employees where ename like 'BLAKE'
```

24. Seleccionar los empleados cuyo nombre comience por BL

```
select * from employees where ename = 'BL%'
```

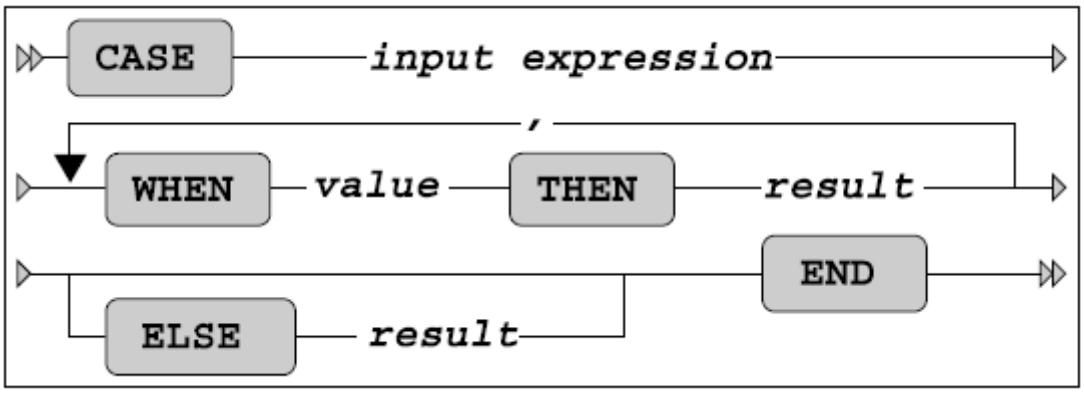
Cuando queremos buscar el % o el _ con el operador LIKE, tenemos que suprimir el significado especial de estos caracteres. Se usará el ESCAPE tal y como se ve en el siguiente ejemplo (la barra de escpa antes del % lo que hace es anular el significado especial de ese carácter).

```
select empno, begindate, comments
from history
where comments like '%0\%%' escape '\';
```

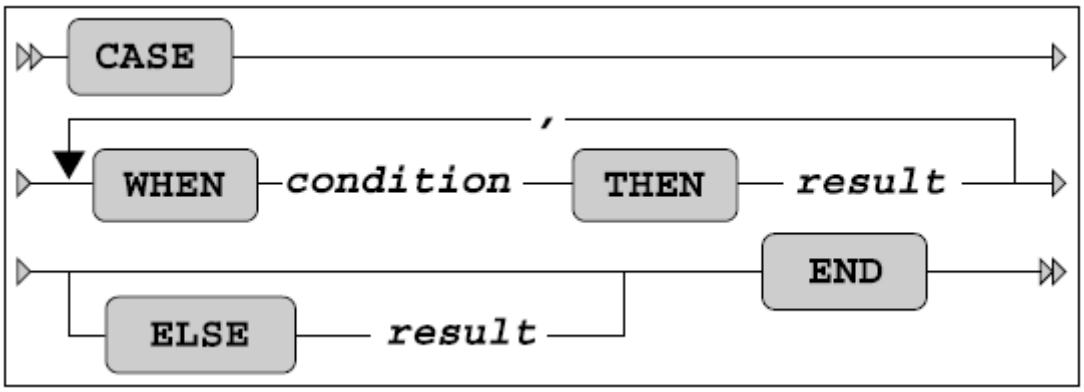
EMPNO BEGINDATE COMMENTS
-------- ---------- --
7566 01-JUN-1989 From accounting to human resources; 0% salary change
7788 15-APR-1985 Transfer to human resources; 0% salary raise

6.6 CASE

Podemos resolver complicados problemas con la expresión CASE. Oracle soporta dos tipos de expresiones CASE: el CASE simple y el CASE de búsqueda.
La siguiente sintaxis nos muestra el CASE simple, donde nosotros especificamos una expresión de entrada que será comparada con los valores que están entre el WHEN y el THEN. El operador implícito es el igual. Los operandos de la izquierda son siempre la expresión de entrada y el operando de la derecha es el valor de la claúsula WHEN.



La siguiente sintáxis muestra la expresión CASE de búsqueda. No podemos especificar una expresión de entrada, pero en cambio, podemos especificar condiciones completas en la claúsula WHEN. De esta forma, tenemos la libertad de utilizar cualquier operador lógico en cada claúsula WHEN individual.



Las expresiones CASE son evaluadas de la siguiente manera:

- Oracle evalúa las expresiones WHEN en el orden en el cual las hayamos especificado y devuelve el resultado del THEN. Ya no evalúa las siguientes claúsulas WHEN, por lo que el orden del WHEN es importante.
- Si no hay ninguna expresión en el WHEN verdadera, Oracle devuelve la expresión del ELSE.
- Si no especificamos nada en el ELSE, Oracle devuelve un valor nulo.

Hay que tener cuidado en la expresión CASE simple con los tipos que pongamos, el tipo del dato que pongamos en la expresión CASE debe ser igual al tipo de dato que especifiquemos en las claúsulas WHEN.

Ejemplo case simple: seleccionar los números de empleados y fecha comienzo del curso S02 y la evaluación de dicho curso, pero en vez de 1, 2, 3, etc. se pondrá: 1→ malo, 2→ mediocre, 3→ ok, 4→ bueno, 5→ excelente, en otro caso, no calificado.

```
select attendee, begindate
        , case evaluation
                when 1 then 'bad'
                when 2 then 'mediocre'
                when 3 then 'ok'
                when 4 then 'good'
                when 5 then 'excellent'
                else 'not filled in'
        end
from registrations
where course = 'S02';
```

Ejemplo case búsqueda: se quiere subir el sueldo a varios empleados. Seleccionar el nombre y trabajo de los empleados ordenado descendentemente por el aumento que se les va a dar y nombre con las siguientes condiciones: si trabajo es "Trainer" 10%, si es "Manager" 20%, si el nombre es "Smith" el 30%, si no cumple ninguna, entonces 0%. Al aumento, llamarle RAISE.

```
select ename, job
        , case when job = 'TRAINER' then ' 10%'
        when job = 'MANAGER' then ' 20%'
        when ename = 'SMITH' then ' 30%'
        else ' 0%'
        end as raise
from employees
order by raise desc, ename;
```

6.7 SUBQUERIES

Para explicar el concepto de subqueries vamos a comenzar con un ejemplo con el operador IN. Supongamos que queremos poner en marcha una campaña de correo electrónico ya que tenemos un curso nuevo que deseamos promover.

El público para el nuevo curso es la comunidad de desarrolladores, por lo que queremos saber quienes han asistido a uno o más cursos de construcción (BLD categoría) en el pasado. ¿Cuál sería la consulta para saber este dato?

Primero tendremos que mirar en la tabla de cursos cuales son los que pertenecen a la categoría BLD.

```
select code
from courses
where category='BLD'
```

```
select attendee
from registrations
where course in ('JAV','PLS','XML');
```

Esta solución tiene por lo menos dos problemas. Para empezar, hemos tenido que hacer una consulta en la tabla CURSOS para comprobar qué cursos pertenecen a la categoría del curso BLD. Sin embargo, la pregunta original no se refería a los cursos específicos, sino que se refiere a los cursos de BLD. Otro problema es que la solución es bastante rígida. Supongamos que deseamos repetir la promoción por correo electrónico un año después para otro nuevo curso. En ese caso, tendríamos que revisar la consulta para reflejar la serie actual de cursos BLD.
La solución óptima para este problema es utilizar una consulta. De esta forma, podríamos Oracle consulta la tabla CURSOS y nosotros no tendríamos que especificar la lista de códigos de los cursos. En una sola consulta lo tendríamos todo:

```
select attendee
from registrations
where course in (select code
                 from courses
                  where category = 'BLD');
```

LA CONDICIÓN DEL WHERE (JOIN)

Siempre tenemos que tener cuidado al realizar las subconsultas de forma que no comparemos cosas distintas. Por ejemplo, la siguiente consulta no se traduce en un mensaje de error, sin embargo, el resultado es extraño.

```
select attendee
from registrations
```

```
where EVALUATION in (select DURATION
                     from courses
                     where category = 'BLD');
```

Tanto evaluación como duración son numéricos, y por lo tanto no hay mensaje de error, pero lógicamente esta subconsulta no tiene sentido. Oracle sólo tiene dos restricciones para las subconsultas:

- Los tipos de datos deben coincidir, o la base de datos Oracle debe ser capaz de hacer que coincidan con la conversión de tipo de datos implícito.
- Los campos que seleccionamos en la subconsulta deben coincidir con los campos que estamos comparando.

MAL:

```
select attendee
from registrations
where course in
        (select course, begindate
        from offerings
        where location = 'CHICAGO');
```

BIEN:

```
select attendee
from registrations
where (course, begindate) in
        (select course, begindate
        from offerings
        where location = 'CHICAGO');
```

OPERADORES DE COMPARACIÓN EN LA CONDICIÓN JOIN

Hasta ahora hemos explorado las subqueries con el operador IN. Pero podemos establecer relaciones entre la querie y la subquerie usando alguno de los operadores de comparación (=, <, >, <=, >=, <>). Tenemos que tener en cuenta que cuando usamos uno de estos comparadores la subconsulta SOLO puede devolver un valor.

25. Seleccionar el nombre, iniciales y fecha de los empleados que sean mayores que el empleado cuyo número de empleado es 7876.

```
select ename, init, bdate
from employees
where bdate > (select bdate
```

```
        from employees
        where empno = 7876);
```

Como vemos, nunca podrá retornar más de un valor ya que el número de empleado es clave primaria.

26. Seleccionar el nombre, iniciales y fecha de los empleados que sean mayores que el empleado cuyo nombre es "Jones". Explica qué ocurre.

```
select ename, init, bdate
from employees
where bdate > (select bdate
        from employees
        where ename = 'JONES');
```

6.8 EL VALOR NULL

Cuando en una columna no se ha insertado ningún valor se dice que tiene valor NULL. El problema del valor NULL es que cuando comparamos un valor con una columna que puede tener valores nulos, el resultado será impredecible. Tendremos que tratar dicha columna para que en el caso de NULL ponga otro valor.

1. ¿Cómo sabemos si una columna tiene valor NULL?: hay una expresión que es "IS NULL" o "IS NOT NULL". Por ejemplo: sacar los empleados cuya comisión es mayor que 50.

   ```
   select *
   from employees
   where comm IS NOT NULL
   and comm > 50;
   ```

   Con esto nos aseguramos que como la primera condición del WHERE ya fuerza a comisiones que no sean nulas sólo opero con comisiones que tienen un valor.

Para ver la problemática del NULL realizar las siguientes consultas:

1. Seleccionar el número de empleado, nombre y comisión de aquellos que tengan una comisión mayor que 400.

```
select empno, ename, comm
from employees
where comm > 400;
```

2. Realizar la misma pero de aquellos que tengan menos que 400. Explica qué ocurre.

```
select empno, ename, comm
```

```
from employees
where comm <= 400;
```

3. Realizar la misma consulta (que la 2) utilizando el operador IS NULL para que nos de el resultado correcto.

```
select empno, ename, comm
from employees
where comm <= 400
or comm is null;
```

EL VALOR NULL Y EL OPERADOR IGUAL (=)

```
select * from registrations where evaluation IS null
select * from registrations where evaluation = null
```

Nunca se puede utilizar el = para comparar con valores nulos.

6.9 LAS TABLAS DE VERDAD

Op1	NOT (Op1)
TRUE	FALSE
FALSE	TRUE
UNK	UNK

Op1	Op2	Op1 AND Op2	Op1 OR Op2
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	UNK	UNK	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	UNK	FALSE	UNK
UNK	TRUE	UNK	TRUE
UNK	FALSE	FALSE	UNK
UNK	UNK	UNK	UNK