SQL - El comando SELECT

El objetivo de este documento es guiar su aprendizaje del comando SELECT de SQL. La idea es que, estando conectado a sql*plus, lea este material mientras ejercita simultáneamente cada una de las distintas modalidades del comando SELECT descritas a continuación.

A lo largo de este material se hace referencia a las tablas de departamento (DEPT) y empleado (EMP) de una empresa, usadas en los tutoriales de los productos de ORACLE y pertenecienes al usuario con username: scott y password: tiger. Para conocer cómo están formadas estas tablas puede usar, conectado como el usuario scott, el comando *describe* (desc). Por ejemplo:

```
desc emp;
desc dept;
```

PARTE I. El comando básico.

Una instrucción SELECT es utilizada para recuperar o mostrar datos de una tabla de la base de datos. Debe contener una cláusula FROM. Por ejemplo para obtener todas las columnas de una tabla:

```
select * from dept;
```

Para ejecutar cualquier comando en el interpretador de sql*plus recuerde terminar sus instrucciones con punto y coma (;) o bien ingresar un slash (/) en una línea en blanco, si bien estos caracteres no son parte de la instrucción.

Para listar columnas específicas de una tabla, indique los nombres de esas columnas, separados por comas (,):

```
select deptno, dname
from   dept;
```

La cláusula DISTINCT es empleada para suprimir valores duplicados en una columna (o columnas) seleccionada(s). Por ejemplo:

```
select distinct job from emp;
```

Para constatar el efecto de la palabra DISTINCT ingrese el mismo comando sin esta cláusula (select job from emp).

Como se indicó, DISTINCT aplica para toda la fila seleccionada (con algunas limitaciones). Compare el efecto de las siguientes instrucciones:

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

¿Qué información obtiene al ejecutar este último comando ?

La cláusula WHERE es empleada para especificar qué filas deben ser seleccionadas. Funciona como una especie de filtro del conjunto de filas a ser retornadas. Por ejemplo:

```
select ename, sal from emp
where deptno = 20;
```

Es posible usar los operadores =, < , >, < =, =>. Recuerde que:

- Las columnas especificadas en la cláusula WHERE deben ser parte de la tabla especificada en la clausula FROM; (2) las columnas indicadas en la clausula WHERE no tienen que estar referenciadas en la cláusula SELECT.
- Las comparaciones de columnas tipo caracter requieren comillas simples alrededor de los datos, las comparaciones numéricas no. Ejemplo:

```
...where ename = 'SMITH'
...where sal = 2000
```

- El lenguaje es *case sensitive* para los valores de los atributos.

Algunos ejercicios:

1. Listar la información de todos los empleados de la tabla emp.
2. Listar el número, nombre, nombre del cargo y fecha de contratación de los empleados del departamento 10.
3. Seleccionar el nombre y sueldo de los empleados que son oficinistas (clerks).
4. Seleccionar el nombre, cargo y sueldo de los empleados contratados el 17 de Diciembre de 1980. ¿ Cómo preguntar por campos del tipo DATE ?

Las expresiones aritméticas pueden aparecer tanto en la cláusula FROM como en la cláusula WHERE. Por ejemplo:

```
select ename, job, sal* 0.1
from emp;
```

selecciona el nombre y el trabajo de cada empleado y además devuelve el 10% de su salario.

```
select deptno, ename, job
from emp
where comm > 0.25*sal;
```

selecciona aquellos empleados cuya comisión es mayor al 25% de su sueldo (salario).

Es posible usar los operadores lógicos AND y OR, así como los paréntesis para construir expresiones lógicas más elaboradas. Las reglas convencionales de precedencia de operadores aplican también en SQL.

Los operadores IN y NOT IN extienden la potencialidad para formular *queries* más elaborados. Estos operadores son especialmente importantes al momento de formular consultas anidadas. Ejemplo:

```
where loc not in ('CHICAGO','DALLAS','BOSTON')
where loc in ( select distinct loc
               from location
               where country = 'USA')
```

El operador BETWEEN es empleado para seleccionar valores que están dentro o fuera de un rango dado de valores. Ejemplo:

```
select ename, sal from emp
where sal between 1250 and 1600;
```

o bien,

```
...where sal not between 1250 and 1600;
```

Note que BETWEEN es un operador inclusivo mientras que NOT BETWEEN es exclusivo.

En SQL, el valor NULL representa la ausencia de datos. Por ejemplo, es posible conocer los empleados que no ganan comisiones de la siguiente manera:

```
select ename, job, comm
from emp
where comm is NULL;
```

Es posible ordenar el resultado de las filas retornadas empleando la cláusula ORDER BY. Por ejemplo:

```
select ename, job, sal
from emp
where deptno = 30
order by sal;
```

El ordenamiento ascendente (ASC) es el que se ejecuta por *default*. Si se quiere un orden descendente se debe agregar la palabra DESC. Es posible ordenar por varias columnas simultáneamente, como en el siguiente ejemplo:

```
select ename, job, sal, hiredate
from emp
order by job asc, hiredate desc;
```

Es posible, para propósitos de claridad, sustituir el nombre de una columna al efectuar una consulta (crearle un alias a la columna). Por ejemplo, usando el mismo *query* anterior:

```
select  ename nombre, job cargo, sal sueldo,  hiredate,
     fecha_contratacion
from emp
order by cargo asc, fecha_contratacion desc;
```

Más ejercicios:

1. Mostrar el nombre, salario mensual, salario diario y salario por hora de todos los empleados. Asumir que la columna SAL es el salario mensual del trabajador. Renombrar las columnas con nombres adecuados.
2. Mostrar los nombres y número de los gerentes (MANAGERS) que ganan más de 2600. Mostrar en orden alfabético por nombre.
3. Seleccionar aquellos empleados que son o gerentes o presidentes. Construir la consulta empleando, y sin emplear, la cláusula IN.

En SQL es posible conocer, por ejemplo, el promedio de sueldo de todos los empleados de la empresa, o bien, el promedio de sueldo en cada departamento de la empresa. De la misma manera es posible conocer el sueldo más alto, el más bajo, contar el número de empleados totales o el número de empleados de cada departamento.

El conjunto de facilidades que permiten tales operaciones se conocen como funciones de agregación. Estas funciones actúan sobre un grupo o conjunto de filas y retornan una fila de información resumida por conjunto. Por ejemplo, obtener la suma de todos los sueldos de la empresa sería:

```
select sum(sal) from emp;
```

Note que el grupo que se toma por *default* es toda la tabla.

Las funciones de agregación son: MIN, MAX, SUM, AVG, COUNT, STDDEV, VARIANCE. Cualquiera de estas funciones aplican a valores numéricos. Los valores nulos se ignoran al calcular SUM, AVG, STDDEV y VARIANCE.

La función COUNT cuenta el número de filas retornadas por un *query*. Recordar que campos o filas con valores igual a NULL son ignoradas por el COUNT. Algunos ejemplos:

```
select count(*) from   emp;
```

cuenta el número de empleados.

```
select count(*) from   emp where deptno = 30;
```

cuenta el número de empleados del departamento 30.

```
select count( job ) from emp;
```

cuenta el número de empleados que tienen cargos, o dicho de otra manera, el número de filas donde job no es NULL.

```
select count (distinct job) from emp;
```

cuenta el número de cargos distintos de la tabla emp.

Se pueden ver algunos ejemplos del uso de los operadores MIN y MAX a continuación:

```
select min(sal), max(sal) from emp;
```

obtiene el menor y mayor sueldo.

```
select min(hiredate), max(hiredate) from emp;
```

obtiene la fecha más antigua y más reciente en que alguien fue empleado.

Hasta ahora se han demostrado funciones de agregación donde el grupo es toda la tabla (recuerde que las funciones de agregación retornan una fila por grupo o conjunto). ¿Qué sucede si en lugar de contar todos los empleados se desea conocer cuántos empleados hay en cada departamento?. En este caso existen tantos grupos distintos como departamentos en la empresa (que tengan empleados):

```
select deptno, count(*) from emp
group by deptno;
```

(se lee: agrupado por departamentos).

Repita la misma instrucción anterior eliminando la columna deptno de la cláusula SELECT. El resultado es que no es posible reconocer a qué departamento corresponde el conteo de empleados. Por lo tanto, es una práctica de sentido común colocar en el SELECT la columna o grupo de columnas por la cual se está agregando (es decir las columnas que aparecen en el GROUP BY).

Algunas notas en relación a la cláusula GROUP BY:

- La cláusula GROUP BY es empleada para definir los grupos de filas dentro de una instrucción SELECT.
- Definidos los grupos, es posible calcular información resumida para cada grupo presente en la tabla. Se retorna una fila por cada grupo.
- Recordar siempre que todas las columnas que aparecen en la cláusula SELECT deben estar al MISMO NIVEL DE AGREGACION. Las columnas en la cláusula SELECT deben ser columnas que aparezcan en el GROUP BY o ser funciones de agregación. En otras palabras, columnas que no son referenciadas en la cláusula GROUP BY no deben aparecer en la cláusula SELECT, excepto como argumentos de alguna función de agregación.
- La cláusula HAVING especifica los grupos que deben ser retornados. Es similar a la funcionalidad de la cláusula WHERE pero aplica directamente sobre operaciones de agregación. Las cláusulas WHERE y HAVING no son excluyentes. Algunos ejemplos:

```
select deptno, job, count(*) from emp
group by deptno, job
having count(*)  > 2;
```

  permite determinar que departamentos tienen más de dos empleados que hacen el mismo trabajo. Repita la misma instrucción eliminando la cláusula HAVING y observe la diferencia.

```
select deptno from emp where job = 'CLERK'
group by deptno
having count(*) > 2;
```

  permite conocer los departamentos que tienen al menos dos oficinistas.

Algunos ejercicios:

1. Obtener el salario promedio anual de cada cargo en cada departamento.
2. En un solo *query*, contar el número de personas del departamento 30 que reciben un salario y el número de personas que reciben comisión.
3. Calcular el sueldo promedio, mínimo y máximo de los oficinistas y de los gerentes.
4. Mostrar aquellos departamentos que tienen más de un oficinista.

PARTE II. Versiones más complejas del SELECT.

En esta segunda parte se detalla el uso de *queries* anidados y de los *joins*. Los ejemplos usados están basados en las tablas de departamento y empleados, del usuario scott/tiger utilizado anteriormente.

Los resultados de un *query* pueden ser sustituidos dinámicamente en la cláusula WHERE de otro *query*. La sintaxis general es la siguiente:

```
select  columna, columna, columna from   tabla
where columna = ( select columna from  tabla
                    [ where condición ] );
```

Por ejemplo, para mostrar los nombres de las personas que hacen el mismo trabajo de JONES, es necesario conocer primero cúal es el trabajo que Jones hace (*query* más interno), y luego, conocido este dato, recuperar las personas que hacen este mismo trabajo (*query* externo).

```
select ename, job from   emp
where job = (  select job from   emp
          where ename = 'JONES'  );
```

Los *queries* anidados permiten realizar con un solo *query*, operaciones que de otra forma requerirían más de un paso. Por ejemplo la consulta: obtener el nombre y la fecha de contratación de la última persona contratada, sin *queries* anidados se resuelve de la siguiente manera:

Paso 1: Obtener la fecha de contratación más reciente:

```
select max(hiredate) from   emp;
```

Paso 2: Obtener el nombre de la persona contratada en esa fecha:

```
select ename from   emp
where  hiredate  = 'fecha que se obtuvo  en  el  query
      anterior';
```

mientras que si se emplean *queries* anidados queda así:

```
select ename from  emp
where hiredate = ( select max(hiredate) from emp);
```

Los *queries* anidados pueden retornar una sola fila, varias filas, e incluso, retornar varias columnas.

El tipo más sencillo de *query* anidado retorna sólo una columna de una sola fila. Por ejemplo, obtener los nombres de los empleados que trabajan en Chicago y el trabajo que realizan:

```
select ename, job from   emp
where deptno = ( select deptno from  dept
                 where loc = 'CHICAGO' );
```

Pueden existir varios *queries* anidados en la cláusula WHERE como se muestra en el ejemplo siguiente. Obtener el nombre, departamento y salario de todos aquellos empleados que hacen el mismo trabajo de Jones o que ganan tanto como Ford:

```
select ename, deptno, sal from   emp
where     job = (select job  from  emp
                    where ename = 'JONES') or
          sal = (select  sal from   emp
                    where ename = 'FORD' );
```

Si un *subquery* puede retornar más de un valor (varias filas) se deben usar los operadores IN y NOT IN. Por ejemplo, Obtener el nombre, trabajo, y el sueldo de las personas del departamento 20 que hacen el mismo trabajo que las personas en el departamento 30. Ordenar la lista por salario:

```
select ename, job, sal from   emp
where  deptno = 20 and
     job IN ( select job from  emp
          where deptno = 30 )
order by sal;
```

Si un *subquery* puede retornar más de una fila y se desea hacer alguna comparación distinta a la igualdad (<,<=,...), es necesario indicar cómo se deben comparar los valores retornados por el *subquery*. Es necesario emplear algunos de los operadores ANY o ALL. Por ejemplo, obtener el departamento, nombre, trabajo y sueldo de los empleados cuyo salario es mayor que sueldo de CUALQUIERA de los empleados del departamento 30:

```
select deptno, ename, job, sal
from   emp
where  sal   ANY (  select sal
                    from   emp
                    where deptno = 30 );
```

¿cómo se puede reformular este *query* sin usar el operador ANY ? Obtener el departamento, nombre, trabajo y sueldo de los empleados cuyo salario es mayor a los sueldos de TODOS los empleados del departamento 30.

```
select deptno, ename, job, sal
from   emp
where  sal   ALL  (select sal from   emp
                   where deptno = 30 );
```

¿cómo se puede reformular este *query* sin usar el operador ALL ?

Como se indicó, un sub*query* puede retornar más de una columna. Por ejemplo, para conocer quienes son los empleados mejores pagados de cada departamento se formularía el siguiente *query*:

```
select ename, deptno, job, sal
from   emp
where ( deptno, sal ) IN ( select deptno, max(sal)
                           from   emp
                           group by deptno );
```

Note que el orden y tipo de las columnas en el *query* más externo deben ser el mismo orden y tipo de las columnas del *query* más interno. También note el uso de paréntesis en las columnas en la cláusula WHERE.

Hasta ahora se han mostrado ejemplos donde primero se ejecuta el *query* interno (una sola vez) y, dado el resultado de éste, se ejecuta el *query* más externo. Existe otro tipo de *queries* anidados, denominados *queries* correlacionados, donde la ejecución del *query* más interno, depende de los valores que toma cada fila del *query* exterior. En otras palabras, para cada fila del *query* exterior, se ejecuta el *query* más interno y se retorna algún resultado al *query* exterior; este resultado es evaluado para conocer si se satisface alguna condición y se prosigue luego con la siguiente fila.

Por ejemplo, encontrar aquellos empleados que ganan más que el sueldo promedio de su propio departamento.

El *query* principal queda:

```
select deptno, ename, sal
from   emp
where  sal(sueldo promedio de los empleados  del
mismo departamento);
```

el *subquery* queda:

```
select avg(sal)
from  emp
where   deptno = (el departamento del empleado que  se
esta  evaluando);
```

y el *query* correlacionado sería:

```
select deptno, ename, sal
from   emp  E
where sal > (  select avg(sal )
               from  emp
               where deptno =  E.deptno );
```

Note que para distinguir la tabla EMP del *query* exterior se está usando un alias.

El operador EXISTS permite calificar la existencia de filas en el *query* más interno (que satisfagan alguna condición). La instrucción SELECT será cierta si se encuentra una o más filas. Por ejemplo, mostar los departamentos donde laboran empleados:

```
select deptno, dname, loc
from   dept
where exists  ( select deptno from emp
                where emp.deptno = dept.deptno );
```

Mostrar los datos de los empleados que tienen otros empleados a su cargo (empleados que les reportan):

```
select *
from   emp  E1
where  exists (select *
               from   emp E2
               where E2.mgr = E1.empno );
```

Algunos ejercicios:

1. ¿Qué empleados ganan menos que el 30% del salario del presidente?
2. ¿Quién fue el último empleado contratado de cada departamento?
3. Mostrar los departamentos donde no trabaja ningún empleado.
4. Mostar los datos de los departamentos donde sus empleados devengan comisiones. *Hint*: en un departamento se otorgan comisiones si alguno de sus empleados tiene comisión distinta de NULL.

En la práctica, casi siempre se requiere consultar más de una tabla para resolver un *query*. Para obtener columnas de dos o más tablas simultáneamente, es necesario realizar un JOIN de dos o más tablas. Por ejemplo, obtener para cada empleado, su nombre y el nombre del departamento para el cual trabaja:

```
select ename, dname
from  emp, dept
where emp.deptno = dept.deptno;
```

En este caso, en la cláusula FROM se indican las tablas a las que se les va a hacer JOIN y en la cláusula WHERE se indica la condición del JOIN.

La cláusula WHERE puede ser usada para indicar la condición de JOIN así como para especificar el criterio de selección. Por ejemplo, mostrar el nombre, trabajo, número de departamento, nombre del departamento y localidad de todos los oficinistas (CLERK):

```
select ename, job, dept.deptno, dname, loc
from  emp, dept
where emp.deptno = dept.deptno
     and  job = 'CLERK';
```

En el ejemplo anterior, es necesario calificar la columna deptno para indicar de cuál tabla se toma esta información, puesto que en ambas tablas el atributo tiene el mismo nombre.

El OUTER JOIN se denota con un signo más encerrado entre paréntesis (+). El OUTER JOIN fuerza que una fila con valores nulos sea generada para hacer *match* con cada fila de una segunda tabla con la cual no habría *match* en condiciones normales.

Por ejemplo, mostrar las localidades de todos los departamentos y los empleados que trabajan en estos. Incluir los departamentos sin empleados:

```
select loc, ename
from  emp,dept
where emp.deptno(+) = dept.deptno;
```

El OUTER JOIN une una fila de la tabla departamento con una fila de valores NULL de la tabla de empleados. Por ejemplo, mostrar sólo aquellos departamentos que no tienen empleados asignados en la actualidad:

```
select empno,  dept.deptno, loc
from  emp,  dept
where emp.deptno(+) = dept.deptno
and    empno is NULL;
```

Se puede hacer *join* de una tabla consigo misma, y recuperar filas que cumplen alguna condición, tal y como si fuesen dos tablas distintas. Por supuesto, es necesario utilizar alias para indicar de cuál instancia de cada tabla proceden las columnas retornadas por el *query*. Por ejemplo, listar el número del empleado, su nombre y trabajo y el número, nombre y trabajo de su jefe (a quien reporta):

```
select trabajador.empno,trabajador.ename,
     trabajador.job, jefe.empno, jefe.ename, jefe.job
from emp trabajador, emp jefe
where trabajador.mgr = jefe.empno;
```

Note que el presidente no aparece en el resultado de este *query* puesto que no reporta a nadie.

Algunos ejercicios.

Usando la funcionalidad del JOIN, formule las siguientes consultas:

1. ¿Cuántos empleados trabajan en New York?
2. ¿Cuáles empleados trabajan en New York?
3. Listar los nombres de los empleados y las ciudades donde estos trabajan. Ordenar la lista por ciudad.
4. Obtener el nombre y sueldo de los empleados que ganan más que sus jefes.
5. Obtener el número de departamento, localidad y número de empleados de cada localidad (incluyendo aquellas donde no existen trabajadores actualmente asignados).