2.4.1. Consultas en SQL (continuación)

Algunas consultas requieren que ciertos valores de la base de datos sean antes recuperados y luego usados en las condiciones de comparación. Estas consultas pueden ser implementadas a través de *consultas anidadas*, donde hay consultas completas dentro de las cláusulas WHERE de otras consultas. Por ejemplo, la consulta anterior (Q10) podría ser implementada así:

```
Q10A:  SELECT DISTINCT PNUMERO
       FROM   PROYECTO
       WHERE  PNUMERO IN   ( SELECT PNUMERO
                             FROM PROYECTO, DEPARTAMENTO, EMPLEADO
                             WHERE DNUM = DNUMERO AND RUTGERENTE = RUT AND APPAT = "Pérez" )
              OR
              PNUMERO IN   ( SELECT PNO
                             FROM TRABAJA_EN, EMPLEADO
                             WHERE ERUT = RUT AND APPAT = "Pérez" )
```

La primera consulta anidada selecciona los números de proyecto de los proyectos que tienen a "Pérez" como gerente, mientras que la segunda selecciona los números de proyecto que tienen a "Pérez" como trabajador. El operador de comparación IN compara un valor v con un conjunto de valores V, y regresa TRUE si v es uno de los elementos en V.

Además del operador IN, pueden ser usados otros operadores para comparar un valor v (típicamente un nombre de atributo) con un conjunto V (típicamente una consulta anidada). El operador "= ANY" (o "= SOME") regresa TRUE si el valor v es igual a algún valor en el conjunto V. ANY y SOME tienen el mismo significado, y en este caso son equivalentes a IN. Otros operadores que pueden ser combinados con ANY (o con SOME) incluyen >, >=, <, <= y <>. Por ejemplo, la condición "v > ALL V" regresa TRUE si el valor v es mayor que *todos* los valores del conjunto V.

Consulta 11

Recuperar los nombres de los empleados cuyo salario es mayor que el salario de todos los empleados del departamento 5.

```
Q11:  SELECT NPILA, APPAT
      FROM   EMPLEADO
      WHERE  SUELDO > ALL ( SELECT SUELDO
                            FROM EMPLEADO
                            WHERE NDEPTO = 5 )
```

En las consultas anidadas pueden usarse atributos de las consultas exteriores. Por ejemplo:

Consulta 12

Recuperar el nombre de cada empleado que tenga un dependiente con el mismo nombre de pila y sexo que el empleado.

```
Q12:  SELECT E.NPILA,
            E.APPAT
      FROM   EMPLEADO
      WHERE  E.RUT IN (  SELECT ERUT
                         FROM CARGA
                         WHERE ERUT = E.RUT AND E.NPILA =
                         NOMBRE_CARGA AND CARGA.SEXO =
                         E.SEXO )
```

Las consultas anidadas *siempre* pueden ser escritas como consultas de un sólo bloque. Por ejemplo, la consulta anterior puede ser escrita así:

```
Q12A: SELECT E.NPILA, E.APPAT
      FROM   EMPLEADO E, CARGA C
      WHERE  E.RUT = C.ERUT AND E.SEXO = C.SEXO AND E.NPILA =
             C.NOMBRE_CARGA
```

La cláusula **EXISTS** es usada para chequear cuándo el resultado de una consulta anidada está vacío (no contiene tuplas). Por ejemplo:

Consulta 13

Recuperar los nombre de los empleados que no tienen dependientes.

```
Q13:  SELECT NPILA,
             APPAT
      FROM   EMPLEADO
      WHERE  NO EXISTS (  SELECT *
                          FROM CARGA
                          WHERE ERUT = RUT )
```

Es posible renombrar los atributos que aparecen en el resultado de una consulta, agregando el calificador AS seguido del nuevo nombre. Por ejemplo, la consulta 3 podría ser modificada para distinguir el nombre del empleado del nombre de su supervisor, de la siguiente manera:

```
Q3A:  SELECT E.NPILA AS NOMBRE_EMP, E.APPAT AS APELLIDO_EMP,
             S.NPILA AS NOMBRE_SUP, S.APPAT AS APELLIDO_SUP
      FROM   EMPLEADO AS E, EMPLEADO AS S
      WHERE  E.RUT = S.RUTSUPERV
```

SQL permite usar algunas funciones como COUNT (cuenta el número de tuplas en una consulta), SUM (regresa la suma de algún atributo numérico), MIN (regresa el mínimo), MAX (regresa el máximo) y AVG (regresa el promedio). Por ejemplo:

Consulta 14

Regresar la suma de los salarios de todos los empleados, el salario máximo, el salario mínimo y el promedio de los salarios.

```
Q14:  SELECT  SUM(SUELDO), MAX(SUELDO), MIN(SUELDO), AVG(SUELDO)
      FROM    EMPLEADO
```

Consulta 15

Regresar el número de empleados del departamento de "Investigación".

```
Q15:  SELECT COUNT(*)
      FROM   EMPLEADO, DEPARTAMENTO
      WHERE  DNUMERO = NDEPTO AND DNOMBRE = "Investigación"
```

Consulta 16

Cuente el número de salarios distintos en la base de datos.

```
Q16:  SELECT COUNT(DISTINCT SUELDO)
      FROM   EMPLEADO
```

Consulta 17

Regrese los nombres de todos los empleados que tienen más de dos dependientes.

```
Q17:  SELECT NPILA, APPAT
      FROM   EMPLEADO
      WHERE  ( SELECT COUNT (*)
               FROM CARGA
               WHERE ERUT = RUT ) >= 2
```

En muchos casos se quiere aplicar una función a *grupos de tuplas*en una relación. Por ejemplo, si queremos conocer el promedio de salarios de empleados por cada departamento. En estos casos necesitamos agrupar por cierto(s) atributo(s). Para hacer esto, SQL provee la cláusula GROUP BY.

Consulta 18

Para cada departamento regrese: el número de departamento, el número de empleados en ese departamento y el salario promedio.

```
Q18:  SELECT    NDEPTO, COUNT(*), AVG(SUELDO)
      FROM      EMPLEADO
      GROUP BY NDEPTO
```

Consulta 19

Para cada proyecto regrese su número, nombre, y número de empleados que trabajan en él.

```
Q19:  SELECT    PNUMERO, PNOMBRE, COUNT(*)
      FROM      PROYECTO, TRABAJA_EN
      WHERE     PNUMERO = PNO
      GROUP BY  PNUMERO, PNOMBRE
```

En el ejemplo anterior, el agrupamiento y las funciones son aplicados después de la unión de las dos relaciones. Sin embargo, algunas veces se desea recuperar los valores de estas funciones solamente para grupos que satisfacen ciertas condiciones. Por ejemplo, supongamos que queremos modificar la consulta anterior para que solamente aparezcan los proyectos con más de dos empleados. Para hacer esto, SQL provee la cláusula HAVING, la cual puede aparecer junto con la cláusula GROUP BY. HAVING provee una condición sobre el grupo de tuplas asociadas con cada valor de los atributos agrupados, y en el resultado aparecen solamente los grupos que satisfacen esta condición. Por ejemplo:

Consulta 20

Para cada proyecto en que trabajan más de dos empleados, recuperar el número de proyecto, el nombre del proyecto, y el número de empleados que trabajan en el proyecto.

```
Q20:  SELECT    PNUMERO, PNOMBRE, COUNT(*)
      FROM      PROYECTO, TRABAJA_EN
      WHERE     PNUMERO = PNO
      GROUP BY  PNUMERO, PNOMBRE
      HAVING    COUNT(*) > 2
```

Consulta 21

Para cada departamento que tenga más de cinco empleados, recuperar el número de departamento y el número de empleados que ganan más de $350.000.

```
Q21:  SELECT DNOMBRE, COUNT(*)
      FROM   DEPARTAMENTO, EMPLEADO
      WHERE  DNUMERO = NDEPTO AND SUELDO >
             350.000 AND NDEPTO IN
             ( SELECT NDEPTO
             FROM EMPLEADO
             GROUP BY NDEPTO
             HAVING COUNT(*) > 5 )
```

Finalmente, SQL permite ordenar las tuplas que resultan de las consultas, por los valores de uno o más atributos, usando la cláusula ORDER BY. Por ejemplo:

Consulta 22

Regrese una lista de empleados y los proyectos en los que trabajan, ordenada por departamento, y dentro de cada departamento, ordenada alfabéticamente por nombre y apellido.

```
Q22:  SELECT    DNOMBRE, NPILA, APPAT, PNOMBRE
      FROM      DEPARTAMENTO, EMPLEADO, TRABAJA_EN,
                PROYECTO
      WHERE     DNUMERO = NDEPTO AND RUT = ERUT AND PNO =
                PNUMERO
      ORDER
      BY        DNOMBRE, APPAT, NPILA
```

Por defecto, el ordenamiento es en orden ascendente (ASC). También se puede especificar un orden descendiente (DESC). Por ejemplo, si en la consulta anterior se desea tener ordenado descendentemente por nombre de departamento, y ordenado ascendentemente por nombre de empleado, se puede especificar así:

ORDER BY DNOMBRE **DESC**, APPAT **ASC**, NPILA **ASC**

Otras operaciones

Para insertar una tupla en una relación se usa INSERT. Por ejemplo:

```
INSERT INTO   EMPLEADO
VALUES        ( "Ricardo", "Cruz", "Pérez", "99887766", "1-9-72", "Blanco 1999",
              "M", 400, "33344555", 5 )
```

Para borrar tuplas de una relación se usa DELETE. Por ejemplo:

```
DELETE FROM EMPLEADO
WHERE       APPAT = "Pérez"
```

```
DELETE FROM EMPLEAD
            O
WHERE       NDEPTO IN ( SELECT DNUMERO
                        FROM DEPARTAMENTO
                        WHERE DNOMBRE = "Investigaciones" )
```

También se pueden modificar los valores de los atributos usando UPDATE. Por ejemplo:

```
UPDATE PROYECTO
SET    PUBICACION = "Macul", DNUM = 3
WHERE  PNUMERO = 10
```