Lenguaje Estructurado de Consultas
Structured Query Language

SQL: El SQL es un Lenguaje de Manejo de Datos (LMD) basado en los lenguajes formales álgebra relacional y cálculo de predicados de tuplas. Está considerado el lenguaje estándar para consulta de datos en los SGBD relacionales.

La estructura básica de una expresión en SQL se compone de tres cláusulas:

SQL

Cláusula *Select:*

Corresponde a la operación de proyección del álgebra relacional. Sirve para listar todos los atributos que se deseen en el resultado de la consulta.

Cláusula *From:*

Corresponde a la operación de producto cartesiano del álgebra relacional. Su argumento consta de una lista de relaciones que se examinan durante la ejecución de la expresión.

Cláusula *Where:*

Corresponde al predicado de la selección (elegir) del álgebra relacional. Este predicado incluye atributos que aparecen en las relaciones indicadas en la cláusula *from*.

SQL

Sintaxis

```
Select a1, a2, … , an
from R1, R2, …, Rm
where P;
```

Equivalente en álgebra relacional:

$$\pi_{a1,a2,\ldots,an}(\sigma_{(P)}(((R1 X (R2 X (\ldots X (Rm\text{-}1 X Rm))))))$$

SQL

Varientes en las Cláusulas:

1. Si en la expresión de SQL se omite la cláusula where se asume que P es verdadero.
2. Si se sustituye a1,a2,...,an por "∗" (asterisco) se eligen todos los atributos de las relaciones que aparecen en la cláusula *from*.

Así como en el álgebra relacional, todo resultado de la ejecución de una expresión en SQL es una relación.

Ejemplos en SQL

Pieza(p#,nombre-p,color-p,peso-p,ciudad-alm)
Precio(p#,precio-p,precio-p-sug)
Préstamo(numPréstamo,ci-ch,nombre-suc,importe)
Depósito(numDepósito,ci-ch,nombre-suc,saldo)

Consulta 1: Listar todos nombres de las sucursales donde se han solicitado préstamos.

```
Select nombre-suc
from Préstamo;
```

La relación resultante está compuesta por los nombres de todas las sucursales que aparecen en la relación Prestamo, si hay nombres de sucursales repetidos en varias tuplas, estos aparecen repetidos en el resultado.

Ejemplos en SQL

Pieza(p#,nombre-p,color-p,peso-p,ciudad-alm)
Precio(p#,precio-p,precio-p-sug)
Préstamo(numPréstamo,ci-ch,nombre-suc,importe)
Depósito(numDepósito,ci-ch,nombre-suc,saldo)

Si se desea eliminar los duplicados se debe indicar explícitamente:

```
Select distinct nombre-suc
 from Préstamo;
```

Pieza(p#,nombre-p,color-p,peso-p,ciudad-alm)
Precio(p#,precio-p,precio-p-sug)
Préstamo(numPréstamo,ci-ch,nombre-suc,importe)
Depósito(numDepósito,ci-ch,nombre-suc,saldo)

Consulta 2: Hallar los cuentahabientes que tienen tanto préstamos como depósitos.

```
(Select  ci-ch
 from Deposito )
intersect
(Select ci-ch
 from Prestamo);
```

Pieza(p#,nombre-p,color-p,peso-p,ciudad-alm)
Precio(p#,precio-p,precio-p-sug)
Préstamo(numPréstamo,ci-ch,nombre-suc,importe)
Depósito(numDepósito,ci-ch,nombre-suc,saldo)

Consulta 3: Listar los cuentahabientes que tengan o bien un préstamo o un depósito o ambos en la sucursal "Centro".

```
(Select  ci-ch
 from Deposito
 where nombre-suc = "Centro")
 union
(Select ci-ch
 from Prestamo
 where nombre-suc = "Centro");
```

La unión elimina los duplicados.

Pieza(p#,nombre-p,color-p,peso-p,ciudad-alm)
Precio(p#,precio-p,precio-p-sug)
Préstamo(numPréstamo,ci-ch,nombre-suc,importe)
Depósito(numDepósito,ci-ch,nombre-suc,saldo)

Consulta 4: Hallar a todos los cuentahabientes que han pedido préstamos y no han hecho ningún depósito.

```
(Select  ci-ch
 from Prestamo)
 minus
(Select ci-ch
 from Deposito);
```

Union y Minus son conjuntistas

Pieza(p#,nombre-p,color-p,peso-p,ciudad-alm)
Precio(p#,precio-p,precio-p-sug)
Préstamo(numPréstamo,ci-ch,nombre-suc,importe)
Depósito(numDepósito,ci-ch,nombre-suc,saldo)

Consulta 5: Encuentre los códigos y las ciudades donde se almacenan, las piezas tales que el precio de la pieza y el precio sugerido para la pieza sean iguales.

```
Select Precio.p#, ciudad-alm
        from Pieza, Precio
        where precio-p = precio-p-sug
              AND Precio.p# = Pieza.p#;
```

En SQL se pueden utilizar los operadores AND, OR y NOT

Muchas formas de hacer una misma consulta en SQL

En SQL se da una cantidad importante de redundancia en el sentido de que una misma consulta puede tener varias expresiones diferentes con las cuales se puede satisfacer. Por ejemplo, la Consulta 2 se puede satisfacer con la intersección de dos relaciones (como se vió anteriormente) y también utilizando el conector de pertenencia a un conjunto en cuya sintaxis se usa la palabra *in*, como sigue:

Consulta 2: Hallar los cuentahabientes que tienen tanto préstamos como depósitos.

```
(Select  ci-ch
 from Deposito
 where ci-ch    in
                    Select ci-ch
                    from Prestamo);
```

*El conector **in** pertenece al cálculo relacional de predicados.*

Muchas formas de hacer una misma consulta en SQL

Consulta 4: Hallar a todos los cuentahabientes que han pedido préstamos y no han hecho ningún depósito.

```
(Select  ci-ch
from Prestamo
where ci-ch    not in
                        Select ci-ch
                        from Deposito);
```

*El conector not **in** pertenece al cálculo relacional de predicados.*

Variables de Tupla en SQL

En el cálculo de predicados de tuplas se utilizan variables atadas (no libres) a relaciones. De igual forma, en SQL podemos establecer variables atadas a relaciones. Estas variables se indican explícitamente en la cláusula *from* de la expresión.

Por ejemplo:

Consulta 6: Hallar todos los cuentahabientes que tengan préstamos en la misma sucursal que el cuentahabiente de cédula "V-21.114.828".

```
Select  P.ci-ch
from Prestamo P, Prestamo S
where P.nombre-suc = S.nombre-suc
        AND S.ci-ch = "V-21.114.828“;
```

Variables de Tupla en SQL

En la consulta anterior, las variables P y S están atadas a la relación Prestamo. Esa forma de escribir una expresión es útil cuando se desea consultar datos en una misma relación. Sin embargo, las variables pueden ser usadas en la cláusula from para relaciones diferentes, por ejemplo dada adicionalmente:

Cuentahabiente(ci-ch,nombre-ch,direccion,telefono)

Consulta 6: Hallar la dirección de todos los cuentahabientes hayan pedido préstamos.

```
Select  C.direccion
from Cuentahabiente C, Prestamo P
where C.ci-ch = P.ci-ch;
```

En SQL se pueden utilizar otros conectores de relaciones como:
any, all, and, como sigue:

Conector “a alguno” o “a cualquiera”

Consulta 7: Encontrar todas las piezas que tienen un peso mayor que alguna (por lo menos una) pieza almacenada en Porlamar.

```
Select  p#
from Pieza
where peso-p > any
                    (Select peso-p
                     from Pieza
                     where ciudad-alm = "Porlamar");
```

Esta misma consulta usando variables de tupla:

Consulta 7: Encontrar todas las piezas que tienen un peso mayor que alguna (por lo menos una) pieza almacenada en Porlamar.

```
Select  P.p#
from Pieza P, Pieza S
where P.peso-p > S.peso-p
AND S.ciudad-alm = "Porlamar“;
```

Conector “a todos”

Consulta 8: Encontrar todas las piezas cuyos pesos sean mayores a todas las piezas almacenadas en Valencia.

```
Select  p#
from Pieza
where peso-p > all
                    (Select peso-p
                     from Pieza
                     where ciudad-alm = "Valencia");
```

De la misma forma son válidas las expresiones que contengan:
< any, <= any, < > any, = any

Consulta 9: Hallar las piezas que pesen lo mismo que cualquier pieza (por lo menos una) que esté almacenada en Caracas.

```
Select  p#
from Pieza
where peso-p = any
                    (Select peso-p
                     from Pieza
                     where ciudad-alm = "Caracas");
```

Consulta 9: Hallar las piezas que pesen lo mismo que cualquier pieza (por lo menos una) que esté almacenada en Caracas.

```
Select  p#
from Pieza
where peso-p = any
                    (Select peso-p
                      from Pieza
                      where ciudad-alm = "Caracas");
```

o bien:

```
Select  p#
from Pieza
where peso-p in
                  (Select peso-p
                    from Pieza
                    where ciudad-alm = "Caracas");
```

Fuentes consultadas:

[1] Silberchatz, Korth. ,
"Fundamentos de Bases de Datos".

[2] Prof. Elsa Liliana Tovar.
Notas de clase compiladas entre 1997-2016.