Fundamentos de Bases de Datos.
Práctica 8.

Profesora: Dra. Amparo López Gaona
alg@ciencias.unam.mx
Laboratorio: Lic. Carlos Augusto Escalona Navarro
caen@ciencias.unam.mx

4 de noviembre de 2021

Se dan a conocer especificaciones de entrega para la práctica 8.

1. Consultas avanzadas y uso de funciones

1.1. Subconsultas

El uso de subconsultas es una técnica que permite utilizar el resultado de una tabla SELECT en otra consulta SELECT. Permite solucionar consultas complejas mediante el uso de resultados previos conseguidos a través de otra consulta.
El SELECT que se coloca en el interior de otro SELECT se conoce con el término de SUBSELECT.
Ese SUBSELECT se puede colocar dentro de las cláusulas:
WHERE, HAVING, FROM o JOIN.

Las subconsultas simples son aquellas que devuelven una única fila. Si además devuelven una única columna, se las llama subconsultas escalares, ya que devuelven un único valor.

1.2. Uso de subconsultas múltiples filas

En la sección anterior se comentaba que las subconsultas sólo pueden devolver una fila. Pero a veces se necesitan consultas que deben mostrar el atributo A y B de la tabla X cuyo atributo A es mayor al de cualquier elemento del subconjunto de X. Para esto la subconsulta nececita ese resultado para todos los A del subconjunto de X. Pero no podremos utilizar un operador de comparación directamente ya que esa subconsulta devuelve más de una fila. La solución a esto es utilizar instrucciones especiales entre el operador y la consulta, que permiten el uso de subconsultas de varias filas.
Estos operadores son:

ANY o SOME.- Compara con cualquier registro de la subconsulta. La instrucción es válida si hay un registro en la subconsulta que permite que la comparación sea cierta. Se suele utilizar la palabra ANY (SOME es un sinónimo).
Ejemplo:

```
SELECT *
FROM DISCOGRAFICA.DISCO
WHERE GENERO >= ANY(
    SELECT  MAX(GENERO )
    FROM DISCOGRAFICA.DISCO
    INNER JOIN DISCOGRAFICA.DISQUERA USING(DISQUERA) );
```

ALL.- Compara con todos los registros de la consulta. La instrucción resulta cierta si es cierta toda comparación con los registros de la subconsulta.
Ejemplo:

```
    SELECT
    NOMBRE,
    LUGARORIGEN,
    NUMFANS
FROM
    DISCOGRAFICA.CLUB
WHERE
    NUMFANS > ALL (
            SELECT
                MIN (NUMFANS)
            FROM
                    DISCOGRAFICA.CLUB )
ORDER BY
    1;
```

IN.- No usa comparador, ya que sirve para comprobar si un valor se encuentra en el resultado de la subconsulta.

Ejemplo:

```
SELECT
    NOMBRE,
    LUGARORIGEN,
    NUMFANS
FROM
    DISCOGRAFICA.CLUB
WHERE
    LUGARORIGEN IN ('Italia' , 'Australia')
 ORDER BY   1;
```

NOT IN.- Comprueba si un valor no se encuentra en una subconsulta.
Ejemplo:

```
SELECT
    NOMBRE,
    LUGARORIGEN,
    NUMFANS
FROM
    DISCOGRAFICA.CLUB
WHERE
    LUGARORIGEN NOT IN ('Italia' , 'Australia')
 ORDER BY 1;
```

1.3. Consultas con exists

Este operador devuelve verdadero si la consulta que le sigue devuelve algún valor. Si no, devuelve falso. Se utiliza normalmente mediante consultas correlacionadas. Ejemplo:

```
SELECT * FROM DISCOGRAFICA.CLUB
WHERE EXISTS (SELECT NOMBRE, LUGARORIGEN, NUMFANS
FROM DISCOGRAFICA.CLUB WHERE LUGARORIGEN IN ('Italia' , 'Australia'));
```

1.4. Subconsultas en la instrucción update

La instrucción UPDATE permite modificar filas. Es muy habitual el uso de la cláusula WHERE para indicar las filas que se modificarán. Esta cláusula se puede utilizar con las mismas posibilidades que en el caso del SELECT, por lo que es posible utilizar subconsultas. Ejemplo:

```
UPDATE DISCOGRAFICA.CLUB SET NUMFANS=NUMFANS*1.10
WHERE NUMFANS IN (SELECT NUMFANS FROM DISCOGRAFICA.CLUB
WHERE NUMFANS > 50000);
```

También podemos utilizar subconsultas en la cláusula SET de la instrucción UPDATE.
Ejemplo:

```
UPDATE DISCOGRAFICA.CLUB  SET NOMBRE=(SELECT NOMBRE
FROM DISCOGRAFICA.CLUB  WHERE NOMBRE = 'Henu') WHERE NOMBRE = 'Floor';
```

1.5. Subconsultas en la instrucción delete

Al igual que en el caso de las instrucciones INSERT o SELECT, DELETE dispone de cláusula WHERE y en dichas cláusulas podemos utilizar subconsultas.

Ejemplo:

```
DELETE FROM DISCOGRAFICA.CLUB  WHERE NOMBRE IN
(SELECT NOMBRE FROM DISCOGRAFICA.CLUB  WHERE NOMBRE = 'Henu');
```

1.6. Union

Combina los resultados de dos consultas juntas. En este sentido, UNION es parecido a Join, ya que los dos se utilizan para información relacionada en múltiples tablas. Una restricción de UNION es que todas las columnas correspondientes necesitan ser del mismo tipo de datos. También, cuando utilizamos UNION, sólo se seleccionan valores distintos (similar a SELECT DISTINCT). Ejemplo:

```
SELECT * FROM DISCOGRAFICA.DISCO  JOIN
DISCOGRAFICA.ESTA USING (NUMREF , ALBUM) JOIN DISCOGRAFICA.CANCION
USING (CODCAN) JOIN DISCOGRAFICA.COMPOSITOR USING (CURPC)
WHERE GENERO = 'Crossover'
UNION
SELECT * FROM DISCOGRAFICA.DISCO  JOIN
DISCOGRAFICA.ESTA USING (NUMREF , ALBUM) JOIN DISCOGRAFICA.CANCION
USING (CODCAN) JOIN DISCOGRAFICA.COMPOSITOR USING (CURPC)
WHERE GENERO = 'Pop';
```

1.7. Intersect

INTERSECT también opera en dos instrucciones SQL. La diferencia es que, mientras UNION actúa fundamentalmente como un operador OR (O) (el valor se selecciona si aparece en la primera o la segunda instrucción), el comando INTERSECT actúa como un operador AND (Y) (el valor se selecciona si aparece en ambas instrucciones).

Ejemplo:

```
SELECT DURACION FROM DISCOGRAFICA.DISCO  JOIN
DISCOGRAFICA.ESTA USING (NUMREF , ALBUM) JOIN DISCOGRAFICA.CANCION
USING (CODCAN) JOIN DISCOGRAFICA.COMPOSITOR USING (CURPC)
WHERE GENERO = 'Crossover'
INTERSECT
SELECT DURACION FROM DISCOGRAFICA.DISCO  JOIN
DISCOGRAFICA.ESTA USING (NUMREF , ALBUM) JOIN DISCOGRAFICA.CANCION
USING (CODCAN) JOIN DISCOGRAFICA.COMPOSITOR USING (CURPC)
WHERE GENERO = 'Pop';
```

1.8. Minus

MINUS opera en dos instrucciones SQL. Toma todos los resultados de la primera instrucción SQL, y luego sustrae aquellos que se encuentran presentes en la segunda instrucción SQL para obtener una respuesta final. Si la segunda instrucción SQL incluye resultados que no están presentes en la primera instrucción SQL, dichos resultados se ignoran.

Ejemplo:

```
SELECT duracion FROM DISCOGRAFICA.DISCO  JOIN
DISCOGRAFICA.ESTA USING (NUMREF , ALBUM) JOIN DISCOGRAFICA.CANCION
USING (CODCAN) JOIN DISCOGRAFICA.COMPOSITOR USING (CURPC)
WHERE GENERO = 'Crossover'
EXCEPT
SELECT duracion FROM DISCOGRAFICA.DISCO  JOIN
DISCOGRAFICA.ESTA USING (NUMREF , ALBUM) JOIN DISCOGRAFICA.CANCION
USING (CODCAN) JOIN DISCOGRAFICA.COMPOSITOR USING (CURPC)
WHERE GENERO = 'Pop';
```

1.9. Group by

Existen situaciones en las cuales sería deseable aplicar las funciones de agregación no sólo a un único conjunto de tuplas sino también a un grupo de conjuntos de tuplas; esto se especifica en SQL usando la cláusula group by. El atributo o atributos especificados en la cláusula group by se usan para formar grupos. Las tuplas con el mismo valor en todos los atributos especificados en la cláusula group by se colocan en un grupo.

```
SELECT expression1, expression2, ... expression_n,
       aggregate_function (aggregate_expression)
FROM tables [WHERE conditions]
GROUP BY expression1, expression2, ... expression_n;
```

1.10. Having

A veces es más útil establecer una condición que se aplique a los grupos que una que se aplique a las tuplas. Los predicados de la cláusula having se aplican después de la formación de grupos, de modo que se pueden usar las funciones de agregación.

Si en una misma consulta aparece una cláusula where y una cláusula having, se aplica primero el predicado de la cláusula where. Las tuplas que satisfagan el predicado de la cláusula where se colocan en grupos según la cláusula group by. La cláusula having, si existe, se aplica entonces a cada grupo; los grupos que no satisfagan el predicado de la cláusula having se eliminan. La cláusula select utiliza los grupos restantes para generar las tuplas resultado de la consulta

```
SELECT column_name(s)
FROM table_name WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

2. Actividad

Se debe crear un script **practica8.sql** el cual contendrá la solución de las siguientes consultas:

1. Conocer los datos de los empleados que tengan más de 15 años en servicio y ordenar de mayor a menor con respecto a los años de laborar.
2. Conocer el nombre, edad, puesto y la fecha en la que inició a trabajar cada uno de los empleados, ordenar ascendemente por nombre y descendemente por la fecha de ingreso.
3. Conocer el vivero con mayor cantidad de plantas a la venta.
4. Conocer los nombres de los empleados asociados con la mayor cantidad de ventas por día de cada uno de los viveros.
5. Conocer la cantidad de ventas y la suma total de estas ventas por día de cada uno de los viveros.
6. Conocer la forma de pago más recurrente.
7. Conocer la venta con mayor cantidad de plantas y en que vivero fue realizada.

3. Entregables

Se debe crear un archivo llamado **practica8.sql**, el cual contendrá las consultas pedidas anteriormente. Adicional a esto deberás incluir dentro de tus entregable un **backup** de tu base de datos para poder validar la solución de tus consultas.