Martin Antonio Vivanco Palacios A01701167

Laboratorio 15: Creación de consultas utilizando SQL Básico

Consulta de una tabla completa

```
select * from materiales
```

Ejemplos:

1000	Varilla 3/16	100.00
1010	Varilla 4/32	115.00
1020	Varilla 3/17	130.00

Selección

```
select * from materiales
where clave=1000
```

Ejemplos:

1000	Varilla 3/16	100.00

Proyección

```
select clave,rfc,fecha from entregan
```

Ejemplos:

1000	AAAA800101	1998-07-08 00:00:00.000
1000	AAAA800101	1999-08-08 00:00:00.000
1000	AAAA800101	2000-04-06 00:00:00.000

Reunión Natural

```
select * from materiales,entregan
where materiales.clave = entregan.clave
```

Ejemplos:

1000	Varilla 3/16	100.00	1000	AAAA800101	5000	1998-07-08 00:00:00.000 165.00
1000	Varilla 3/16	100.00	1000	AAAA800101	5019	1999-08-08 00:00:00.000 254.00

1000	Varilla 3/16 7.00	100.00	1000	AAAA800101	5019	2000-04-06 00:00:00.000	

Si algún material no ha se ha entregado ¿Aparecería en el resultado de esta consulta? No aparecería ya que su clave no estaría en la tabla entregan

Reunión con criterio específico

```
select * from entregan,proyectos
where entregan.numero < = proyectos.numero
```

Ejemplos:

1000	AAAA800101	5000	1998-07-08 00:00:00.000	165.00	5000	Vamos Mexico
1200	EEEE800101	5000	2000-03-05 00:00:00.000	177.00	5000	Vamos Mexico
1400	AAAA800101	5000	2002-03-12 00:00:00.000	382.00	5000	Vamos Mexico

Unión (se ilustra junto con selección)

```
(select * from entregan where clave=1450)
union
(select * from entregan where clave=1300)
```

Ejemplos:

1300	GGGG800101	5005	2002-06-10 00:00:00.000	521.00
1300	GGGG800101	5005	2003-02-02 00:00:00.000	457.00
1300	GGGG800101	5010	2003-01-08 00:00:00.000	119.00

¿Cuál sería una consulta que obtuviera el mismo resultado sin usar el operador Unión? Compruébalo.

```
select * from entregan

where clave=1450 or clave=1300
```

Intersección (se ilustra junto con selección y proyección)

Nota: Debido a que en SQL server no tiene definida alguna palabra reservada que nos permita hacer esto de una manera entendible, veremos esta sección en el siguiente laboratorio con el uso de Subconsultas. Un ejemplo de un DBMS que si tiene la implementación de una palabra reservada para esta función es Oracle, en él si se podría generar la consulta con una sintaxis como la siguiente:

```
(select clave from entregan where numero=5001)
intersect
(select clave from entregan where numero=5018)
```

Ejemplos:

1010

Diferencia (se ilustra con selección)
(select * from entregan)
minus
(select * from entregan where clave=1000)

Nuevamente, "minus" es una palabra reservada que no está definida en SQL Server, define una consulta que regrese el mismo resultado.

Consulta alternative: (select * from entregan where clave<>1000)

1010	BBBB800101	5001	2000-05-03 00:00:00.000	528.00
1010	BBBB800101	5018	2000-11-10 00:00:00.000	667.00
1010	BBBB800101	5018	2002-03-29 00:00:00.000	523.00

Producto cartesiano
select * from entregan,materiales

¿Cómo está definido el número de tuplas de este resultado en términos del número de tuplas de entregan y de materiales?

La multiplicación de materiales con entregan, eso significa que son todas las combinaciones.

Construcción de consultas a partir de una especificación

Plantea ahora una consulta para obtener las descripciones de los materiales entregados en el año 2000.

Recuerda que la fecha puede indicarse como '01-JAN-2000' o '01/01/00'.

```
SET DATEFORMAT dmy

select m.descripcion, e.Fecha

from entregan e, Materiales m

where m.Clave = e.Clave

and e.Fecha between '01/01/00' and '31/12/00'
```

¿Por qué aparecen varias veces algunas descripciones de material?

Porque tienen diferentes fechas.

Uso del calificador distinct

En el resultado anterior, observamos que una misma descripción de material aparece varias veces.

Agrega la palabra distinct inmediatamente después de la palabra select a la consulta que planteaste antes.

```
SET DATEFORMAT dmy
select distinct m.descripcion
from entregan e, Materiales m
where m.Clave = e.Clave
and e.Fecha between '01/01/00' and '31/12/00'
```

¿Qué resultado obtienes en esta ocasión?

Los mismos materiales, pero sin repetición

Ordenamientos.

Si al final de una sentencia select se agrega la cláusula

order by campo [desc] [,campo [desc] ...]

donde las partes encerradas entre corchetes son opcionales (los corchetes no forman parte de la sintaxis), los puntos suspensivos indican que pueden incluirse varios campos y la palabra desc se refiere a descendente. Esta cláusula permite presentar los resultados en un orden específico.

Obtén los números y denominaciones de los proyectos con las fechas y cantidades de sus entregas, ordenadas por número de proyecto, presentando las fechas de la más reciente a la más antigua.

```
set dateformat dmy
SELECT p.Numero, p.Denominacion, e.Fecha, e.Cantidad
FROM Proyecto p, Entregan e
WHERE  p.Numero = e.Numero
ORDER BY p.Numero, e.Fecha DESC
```

5000	Vamos Mexico	2002-03-12 00:00:00.000	382.00
5000	Vamos Mexico	2000-03-05 00:00:00.000	177.00

5000 Vamos Mexico 1998-07-08 00:00:00.000 165.00

Uso de expresiones.

En álgebra relacional los argumentos de una proyección deben ser columnas. Sin embargo, en una sentencia SELECT es posible incluir expresiones aritméticas o funciones que usen como argumentos de las columnas de las tablas involucradas o bien constantes. Los operadores son:

+ Suma
- Resta
* Producto
/ División

Las columnas con expresiones pueden renombrarse escribiendo después de la expresión un alias que puede ser un nombre arbitrario; si el alias contiene caracteres que no sean números o letras (espacios, puntos etc.) debe encerrarse entre comillas dobles (" nuevo nombre"). Para SQL Server también pueden utilizarse comillas simples.

Operadores de cadena

El operador LIKE se aplica a datos de tipo cadena y se usa para buscar registros, es capaz de hallar coincidencias dentro de una cadena bajo un patrón dado.

También contamos con el operador comodín (%), que coincide con cualquier cadena que tenga cero o más caracteres. Este puede usarse tanto de prefijo como sufijo.

SELECT * FROM productos where Descripcion LIKE 'Si%'

¿Qué resultado obtienes?

1120 Sillar rosa 100.00

1130 Sillar gris 110.00

Explica que hace el símbolo '%' toma el valor de cualquier cadena, por lo que cualquier palabra que empieza con si es seleccionada
¿Qué sucede si la consulta fuera : LIKE 'Si' ? solo serían seleccionadas las palabras que fueran estrictamente iguales a ‘si’.
¿Qué resultado obtienes? Nada debido a que no hay palabras iguales a sí.

Otro operador de cadenas es el de concatenación, (+, +=) este operador concatena dos o más cadenas de caracteres.
Su sintaxis es: Expresión + Expresión.
Un ejemplo de su uso, puede ser:
SELECT (Apellido + ', ' + Nombre) as Nombre FROM Personas;

```
DECLARE @foo varchar(40);
DECLARE @bar varchar(40);
SET @foo = '¿Que resultado';
SET @bar = ' ¿¿¿??? '
SET @foo += ' obtienes?';
PRINT @foo + @bar;
```

¿Qué resultado obtienes de ejecutar el siguiente código?
¿Para qué sirve DECLARE? Para declarar una variable
¿Cuál es la función de @foo? es el nombre de la variable.
¿Que realiza el operador SET? Para poner valor a una variable.

Sin embargo, tenemos otros operadores como [ ] , [^] y _.

[ ] - Busca coincidencia dentro de un intervalo o conjunto dado. Estos caracteres se pueden utilizar para buscar coincidencias de patrones como sucede con LIKE.

[^] - En contra parte, este operador coincide con cualquier carácter que no se encuentre dentro del intervalo o del conjunto especificado.

_ - El operador _ o guion bajo, se utiliza para coincidir con un carácter de una comparación de cadenas.

Ahora explica el comportamiento, función y resultado de cada una de las siguientes consultas:

```
SELECT RFC FROM Entregan WHERE RFC LIKE '[A-D]%';
SELECT RFC FROM Entregan WHERE RFC LIKE '[^A]%';
SELECT Numero FROM Entregan WHERE Numero LIKE '___6';
```

Operadores compuestos.

Los operadores compuestos ejecutan una operación y establecen un valor.
+ = (Suma igual)
- = (Restar igual)
* = (Multiplicar igual)
/ = (Dividir igual)
% = (Módulo igual)

Operadores Lógicos.

Los operadores lógicos comprueban la verdad de una condición, al igual que los operadores de comparación, devuelven un tipo de dato booleano (True, false o unknown).

ALL Es un operador que compara un valor numérico con un conjunto de valores representados por

un subquery. La condición es verdadera cuando todo el conjunto cumple la condición.

ANY o SOME Es un operador que compara un valor numérico con un conjunto de valores. La condición es verdadera cuando al menos un dato del conjunto cumple la condición.

La sintaxis para ambos es: valor_numerico {operador de comparación} subquery

BETWEEN Es un operador para especificar intervalos. Una aplicación muy común de dicho operador son intervalos de fechas.

```
SELECT Clave,RFC,Numero,Fecha,Cantidad
FROM Entregan
WHERE Numero Between 5000 and 5010;
```

¿Cómo filtrarías rangos de fechas? Con un where y betwen aplicando la función set dateformat antes.

EXISTS Se utiliza para especificar dentro de una subconsulta la existencia de ciertas filas.

```
SELECT RFC,Cantidad, Fecha,Numero
FROM [Entregan]
WHERE [Numero] Between 5000 and 5010 AND
Exists ( SELECT [RFC]
FROM [Proveedores]
WHERE RazonSocial LIKE 'La%' and [Entregan].[RFC] = [Proveedores].[RFC] )
```

AAAA800101	165.00	1998-07-08 00:00:00.000	5000
CCCC800101	582.00	2001-07-29 00:00:00.000	5002
AAAA800101	86.00	1999-01-12 00:00:00.000	5008

¿Qué hace la consulta? regresa los valores donde numero esta entre 5000 y 5010, aparte que los proveedores hayan entregado algo y su razón social inicie con la.
¿Qué función tiene el paréntesis ( ) después de EXISTS? Todo lo que le sigue ya que es una subconsulta.

IN Especifica si un valor dado tiene coincidencias con algún valor de una subconsulta. NOTA: Se utiliza dentro del WHERE pero debe contener un parámetro. Ejemplo: Where proyecto.id IN Lista_de_Proyectos_Subquery

Tomando de base la consulta anterior del EXISTS, realiza el query que devuelva el mismo resultado, pero usando el operador IN

```
SELECT RFC,Cantidad, Fecha,Numero
```

FROM [Entregan]

WHERE [Numero] Between 5000 and 5010 AND

RFC in (SELECT [RFC]

FROM [Proveedores]

WHERE RazonSocial LIKE 'La%' and [Entregan].[RFC] = [Proveedores].[RFC])

AAAA800101	165.00	1998-07-08 00:00:00.000	5000
CCCC800101	582.00	2001-07-29 00:00:00.000	5002
AAAA800101	86.00	1999-01-12 00:00:00.000	5008

NOT Simplemente niega la entrada de un valor booleano.

Tomando de base la consulta anterior del EXISTS, realiza el query que devuelva el mismo resultado, pero usando el operador NOT IN Realiza un ejemplo donde apliques algún operador : ALL, SOME o ANY

SELECT RFC,Cantidad, Fecha,Numero

FROM [Entregan]

WHERE [Numero] Between 5000 and 5010 AND

RFC not in (SELECT [RFC]

FROM [Proveedores]

WHERE RazonSocial not LIKE 'La%' and [Entregan].[RFC] = [Proveedores].[RFC])

El Operador TOP, es un operador que recorre la entrada, un query, y sólo devuelve el primer número o porcentaje especifico de filas basado en un criterio de ordenación si es posible.

¿Qué hace la siguiente sentencia? Regresa los dos primeros valores de la tabla proyectos

SELECT TOP 2 * FROM Proyectos

¿Qué sucede con la siguiente consulta? regresa los primeros dos números de la tabla número.

SELECT TOP 1 Numero FROM Proyectos

Modificando la estructura de una tabla existente.

Agrega a la tabla materiales la columna PorcentajeImpuesto con la instrucción:
ALTER TABLE materiales ADD PorcentajeImpuesto NUMERIC(6,2);

A fin de que los materiales tengan un impuesto, les asignaremos impuestos ficticios basados en sus claves con la instrucción:
UPDATE materiales SET PorcentajeImpuesto = 2*clave/1000;
esto es, a cada material se le asignará un impuesto igual al doble de su clave dividida entre diez.

Revisa la tabla de materiales para que compruebes lo que hicimos anteriormente.

¿Qué consulta usarías para obtener el importe de las entregas, es decir, el total en dinero de lo entregado, basado en la cantidad de la entrega y el precio del material y el impuesto asignado?

SELECT p.Denominacion, SUM(e.Cantidad*m.Costo*(1 + m.PorcentajeImpuesto/100)) as Importe

FROM Materiales m, Proyectos p, Entregan e

WHERE e.Clave=m.Clave

Creación de vistas

La sentencia

Create view nombrevista (nombrecolumna1 , nombrecolumna2 ,..., nombrecolumna3)
as select...

Permite definir una vista. Una vista puede pensarse como una consulta etiquetada con un nombre, ya que en realidad al referirnos a una vista el DBMS realmente ejecuta la consulta asociada a ella, pero por la cerradura del álgebra relacional, una consulta puede ser vista como una nueva relación o tabla, por lo que es perfectamente válido emitir la sentencia:

select * from nombrevista

¡Como si nombrevista fuera una tabla!

Comprueba lo anterior, creando vistas para cinco de las consultas que planteaste anteriormente en la práctica. Posteriormente revisa cada vista creada para comprobar que devuelve el mismo resultado.

La parte (nombrecolumna1,nombrecolumna2,.de la sentencia create view puede ser omitida si no hay ambigüedad en los nombres de las columnas de la sentencia select asociada.

Importante: Las vistas no pueden incluir la cláusula order by.

A continuación se te dan muchos enunciados de los cuales deberás generar su correspondiente consulta.

En el reporte incluye la sentencia, una muestra de la salida (dos o tres renglones) y el número de renglones que SQL Server reporta al final de la consulta.

Los materiales (clave y descripción) entregados al proyecto "México sin ti no estamos completos".

```
select m.clave, m.Descripcion
from Materiales m, Proyectos p, Entregan e
where m.Clave = e.Clave
and p.Numero = e.Numero
and p.Denominacion like 'México sin ti no estamos completos'
```

Los materiales (clave y descripción) que han sido proporcionados por el proveedor "Acme tools".

```
select m.clave, m.Descripcion
from Materiales m, Proveedores p, Entregan e
where m.Clave = e.Clave
and e.RFC=p.RFC
and p.RazonSocial like 'Acme tools'
```

El RFC de los proveedores que durante el 2000 entregaron en promedio cuando menos 300 materiales.

```
set dateformat dmy
select p.RazonSocial
from Materiales m, Proveedores p, Entregan e
where p.RFC=e.RFC
and e.Fecha between '01/01/00' and '31/12/00'
HAVING AVG(cantidad)>299
GROUP BY RFC
```

El Total entregado por cada material en el año 2000.

```
set dateformat dmy
SELECT Descripcion, SUM(Cantidad) as ‘Total entregado’
FROM Materiales m, Entregan e
WHERE  e.Clave=m.Clave
```

AND Fecha BETWEEN ‘1/01/2000’ AND ‘31/12/2000’

GROUP BY Descripcion

La Clave del material más vendido durante el 2001. (se recomienda usar una vista intermedia para su solución)

set dateformat dmy

SELECT TOP 1 Clave

FROM Entregan

WHERE Fecha BETWEEN ‘1/01/2001’ AND ‘31/12/2001’

GROUP BY Clave

ORDER BY SUM(Cantidad) DESC

Productos que contienen el patrón 'ub' en su nombre.

SELECT Descripcion

FROM Materiales

WHERE Descripcion LIKE ‘%ub%’

Denominación y suma del total a pagar para todos los proyectos.

SELECT pro.Denominacion, SUM(m.Costo*e.Cantidad*(1+m.PorcentajeImpuestos/100)) as ‘Total a pagar’

FROM Materiales m, Proyectos pro, Entrega e

WHERE e.Clave=m.Clave AND e.Numero=pro.Numero

GROUP BY Denominacion

Denominación, RFC y RazonSocial de los proveedores que se suministran materiales al proyecto Televisa en acción que no se encuentran apoyando al proyecto Educando en Coahuila. (Solo usando vistas)

CREATE view vista1 as

SELECT Denominacion, p.RFC, p.RazonSocial

FROM Proyectos pro, Proveedores p, Entregan e

WHERE e.RFC=p.RFC AND e.Numero=pro.Numero

WHERE Denominacion LIKE ‘Televisa en acción’

AND Denominacion NOT IN (SELECT Denominacion

```
  FROM Proyectos pro
  WHERE Denominacion LIKE 'Educando en Coahuila'
  )
SELECT *
FROM vista1
```

Denominación, RFC y RazonSocial de los proveedores que se suministran materiales al proyecto Televisa en acción que no se encuentran apoyando al proyecto Educando en Coahuila. (Sin usar vistas, utiliza not in, in o exists)

```
SELECT Denominacion, e.RFC, p.RazonSocial
FROM Proyectos p, Proveedores pro, Entregan e
WHERE e.RFC=pr.RFC AND e.Numero=p.Numero
AND Denominacion  LIKE 'Televisa en acción'
```

Costo de los materiales y los Materiales que son entregados al proyecto Televisa en acción cuyos proveedores también suministran materiales al proyecto Educando en Coahuila.

```
SELECT Costo, Descripcion
FROM Materiales m, Proveedores pro, Proyectos p, Entregan e
WHERE e.Clave = m.Clave AND e.RFC = pro.RFC AND e.Numero = p.Numero
AND (p.Denominacion LIKE 'Televisa en acción' AND pro.RFC IN (
SELECT Costo, Descripcion
FROM Materiales m, Proveedores pro, Proyectos p, Entregan e
WHERE e.Clave = m.Clave AND e.RFC = pro.RFC AND e.Numero = p.Numero
AND p.Denominacion LIKE 'Educando en Coahuila))
```

Reto: Usa solo el operador NOT IN en la consulta anterior (No es parte de la entrega)

Nombre del material, cantidad de veces entregados y total del costo de dichas entregas por material de todos los proyectos.

Muchas de estas consultas requieren la utilización de funciones agregadas...

Se recomienda que revises nuevamente la lectura.