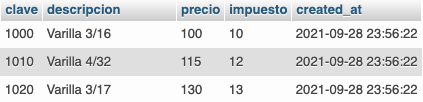
**Consulta de una Tabla Completa**

**Algebra relacional**  
materiales  
  
**SQL**  
SELECT \*

FROM materiales



Renglones: 45

**Selección**

**Algebra relacional**  
SL{clave=1000}(materiales)  
  
**SQL**  
SELECT \*

FROM materiales

WHERE clave=1000

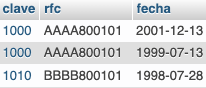


Renglones: 1

**Proyección**

**Algebra relacional**  
PR{clave,rfc,fecha} (entregan)  
  
**SQL**  
SELECT clave, rfc, fecha

FROM entregan



Renglones: 87

**Reunión Natural**

**Algebra relacional**  
entregan JN materiales  
  
**SQL**  
SELECT \*

FROM materiales, entregan

WHERE materiales.clave = entregan.clave



Renglones: 45

**Reunión con criterio específico**

**Algebra relacional**  
entregan JN{entregan.numero <= proyectos.numero} proyectos  
  
**SQL**  
SELECT \* from entregan, proyectos

WHERE entregan.numero <= proyectos.numero



Renglones: 45

**Unión (se ilustra junto con selección)**

**Algebra relacional**  
SL{clave=1450}(entregan) UN SL{clave=1300}(entregan)  
  
**SQL**  
(SELECT \* FROM entregan WHERE clave=1450)

union

(SELECT \* FROM entregan WHERE clave=1300)



Renglones: 2

**Intersección (se ilustra junto con selección y proyección)**

**Algebra relacional**  
PR{clave}(SL{numero=5001}(entregan)) IN PR{clave}(SL{numero=5018}(entregan))  
  
**SQL**

**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)

**Con Subconsulta**

SELECT clave

FROM entregan

WHERE numero = 5001 IN (SELECT clave FROM entregan WHERE numero = 5018)

**Diferencia (se ilustra con selección)**

**Algebra relacional**  
entregan - SL{clave=1000}(entregan)  
  
**SQL**  
(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.

SELECT \*

FROM entregan

WHERE clave != 1000



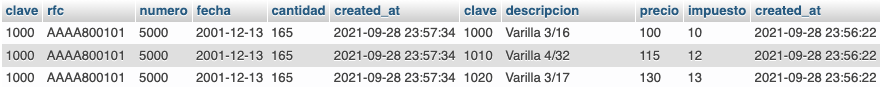
Renglones: 85

**Producto Cartesiano**

**Algebra relacional**

entregan X materiales  
  
**SQL**  
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? tuplas de materiales \* tuplas de entregan

Renglones: 3915

**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'.

Importante: Recuerda que cuando vayas a trabajar con fechas, antes de que realices tus consultas debes ejecutar la instrucción "set dateformat dmy". Basta con que la ejecutes una sola vez para que el manejador sepa que vas a trabajar con ese formato de fechas.

**SQL**  
SELECT M.descripcion, DATE\_FORMAT(fecha, '%e/%c/%Y') 'Fecha'

-- FORMATO DD/MM/YYYY

FROM entregan E, materiales M

WHERE fecha >= '2000-01-01' AND fecha <= '2000-12-31' AND (M.clave = E.clave)



¿Por qué aparecen varias veces algunas descripciones de material?

Por que el mismo material fue entregado en fechas distintas

Renglones: 12

**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.

¿Qué resultado obtienes en esta ocasión?

Todos los materiales entregados entre esas fechas sin repeticiones  
  
**SQL**  
SELECT DISTINCT M.descripcion

-- FORMATO DD/MM/YYYY

FROM entregan E, materiales M

WHERE fecha >= '2000-01-01' AND fecha <= '2000-12-31' AND (M.clave = E.clave)



Renglones: 10

**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.  
  
**SQL**  
SELECT P.numero, denominacion, fecha, cantidad

FROM proyectos P, entregan E

WHERE (P.numero = E.numero)

ORDER BY P.numero, fecha DESC



Renglones: 87

**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.  
  
**SQL**  
SELECT \*

FROM materiales

WHERE descripcion LIKE 'Si%'



Renglones: 2

¿Qué resultado obtienes?

Todas las descripciones de materiales que comienzan con *Si*

Explica que hace el símbolo '%'.

Permite que con que coincida la parte indicada te arroje el resultado, sin importar que haya o no más caracteres después.

¿Qué sucede si la consulta fuera : LIKE 'Si' ?

Únicamente arrojaría los resultados cuya descripción sea *Si,* si comienzan de esa manera, pero tienen más letras no aparecen en el resultado.

¿Qué resultado obtienes?



Una tabla vacía

Explica a qué se debe este comportamiento.

Ya que la descripción de ningún material es como tal ‘Si’, regresa una tabla vacía.

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: 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?

¿Cuál es la función de @foo?

¿Que realiza el operador SET?

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 caracter que no se encuentre dentro del intervalo o del conjunto especificado.

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

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

**SQL**

SELECT rfc

FROM entregan

WHERE rfc LIKE '[A-D]%';

-- No funciona :c

SELECT rfc

FROM entregan

WHERE rfc LIKE 'A%' OR rfc LIKE 'B%' OR rfc LIKE 'C%' OR rfc LIKE 'D%';



Renglones: 47

**SQL**  
SELECT rfc

FROM entregan

WHERE rfc LIKE '[^A]%';  
-- No funciona :c

SELECT rfc

FROM entregan

WHERE rfc NOT LIKE 'A%' AND rfc NOT LIKE 'B%' AND rfc NOT LIKE 'C%' AND rfc NOT LIKE 'D%';



Renglones: 40

**SQL**

SELECT Numero

FROM entregan

WHERE Numero LIKE '\_\_\_6';

Regresa todos los números de materiales entregados que contengan un 6



Renglones: 8

**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.

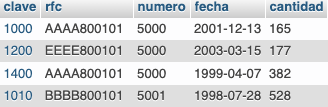
**ANY** o **SOME** Es un operador que compara un valor numérico con un conjunto de valores. La condición es verdadera cuando al menos un dato del conjunto cumple la condición.

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.  
  
**SQL**  
SELECT clave, rfc, numero, fecha, cantidad

FROM entregan

WHERE numero BETWEEN 5000 AND 5010;



Renglones: 43

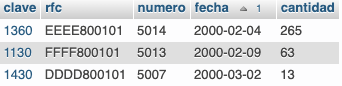
¿Cómo filtrarías rangos de fechas?

**SQL**  
SELECT clave, rfc, numero, fecha, cantidad

FROM entregan

WHERE fecha BETWEEN '2000-01-01' AND '2002-01-01'

ORDER BY fecha ASC;



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

**SQL**  
-- Muestra el rfc, cantidad, fecha y número de las entregas en las que el rfc de los proveedores se ecuentra entre los que sus razón social coincide con 'La'

SELECT rfc, cantidad, fecha, numero

FROM entregan

WHERE numero BETWEEN 5000 AND 5010 AND

EXISTS (

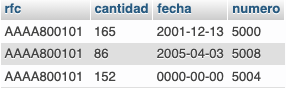
-- Muestra el rfc de los proveedores cuya razón social comience con 'La'

SELECT rfc

FROM proveedores

WHERE razonsocial LIKE 'La%' AND entregan.rfc = proveedores.rfc

)



Renglones: 12

¿Qué hace la consulta?

Muestra el rfc, cantidad, fecha y número de las entregas , el elemento existe en la siguiente subconsulta:

Selecciona el rfc de los proveedores cuya razón social comienza con ‘La’

Y si su número se encuentra entre 5000 y 5010

¿Qué función tiene el paréntesis ( ) después de EXISTS?

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 parametro. 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

**SQL**  
SELECT rfc, cantidad, fecha, numero

FROM entregan

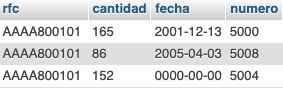
WHERE numero BETWEEN 5000 AND 5010 AND rfc IN (

SELECT rfc

FROM proveedores

WHERE razonsocial LIKE 'La%' AND entregan.rfc = proveedores.rfc

)



Renglones: 12

**NOT** Simplemente niega la entrada de un valor booleano.

**SQL**  
SELECT rfc, cantidad, fecha, numero

FROM entregan

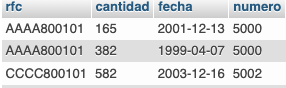
WHERE numero BETWEEN 5000 AND 5010 AND rfc NOT IN (

SELECT rfc

FROM proveedores

WHERE razonsocial NOT LIKE 'La%' AND entregan.rfc = proveedores.rfc

)



Renglones: 12

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.

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? Explica por qué.

SELECT TOP 2 \* FROM Proyectos

-- No funciona

SELECT \*

FROM proyectos LIMIT 2



Renglones: 2

Regresa todas las columnas de las primeras dos filas de la tabla materiales

¿Qué sucede con la siguiente consulta? Explica por qué.

SELECT TOP Numero FROM Proyectos

-- No Funciona

SELECT numero

FROM proyectos LIMIT

Regresa un error ya que no se le esta indicando el número de filas que deseamos que muestre.

**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;

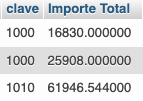
Esto es, a cada material se le asignará un impuesto igual al doble de su clave dividida entre diez.

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?  
  
**SQL**  
SELECT M.clave,(((PorcentajeImpuesto/100) \* precio) + precio) \* cantidad 'Importe Total'

FROM materiales M, entregan E

WHERE (M.clave = E.clave)



Renglones: 87

**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.

**SQL**  
CREATE VIEW View1 (clave, decripcion) AS SELECT clave, descripcion FROM materiales WHERE descripcion LIKE 'Si%';

SELECT\*

FROM View1



Renglones: 2

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.**

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

**SQL**  
SELECT M.clave, descripcion

FROM proyectos P, materiales M, entregan E

WHERE P.numero = E.numero AND M.clave = E.clave AND denominacion = 'México sin ti no estamos completos'



Renglones: 3

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

**SQL**  
SELECT M.clave, descripcion

FROM proyectos P, materiales M, entregan E

WHERE P.numero = E.numero AND M.clave = E.clave AND denominacion = 'México sin ti no estamos completos'



Renglones: 0

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

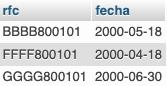
**SQL**  
SELECT rfc, fecha

FROM entregan

WHERE fecha BETWEEN '2000-01-01' AND '2000-12-31'

GROUP BY rfc

HAVING AVG(cantidad) >= 300



Renglones: 3

4) El Total entregado por cada material en el año 2000.

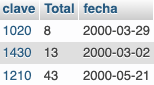
**SQL**  
SELECT clave, SUM(cantidad) 'Total', fecha

FROM entregan

WHERE fecha BETWEEN '2000-01-01' AND '2000-12-31'

GROUP BY clave

ORDER BY SUM(cantidad) ASC



Renglones: 10

5) La Clave del material más vendido durante el 2001

**SQL**  
SELECT clave

FROM entregan

WHERE fecha BETWEEN '2001-01-01' AND '2001-12-31'

GROUP BY clave

ORDER BY SUM(cantidad) DESC

LIMIT 1



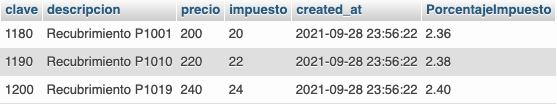
Renglones: 1

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

**SQL**  
SELECT \*

FROM materiales

WHERE descripcion LIKE "%ub%"



Renglones: 12

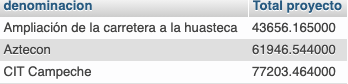
7) Denominación y suma del total a pagar para todos los proyectos.

**SQL**  
SELECT denominacion, (((PorcentajeImpuesto/100) \* precio) + precio) \* cantidad 'Total proyecto'

FROM entregan E, materiales M, proyectos P

WHERE E.clave = M.clave AND E.numero = P.numero

GROUP BY denominacion



Renglones: 20

8) 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).

**SQL**  
CREATE VIEW Televisa (denominacion, rfc, razonsocial) AS

SELECT denominacion, P.rfc, razonsocial

FROM proveedores P, proyectos Y, entregan E

WHERE P.rfc = E.rfc AND Y.numero = E.numero AND denominacion = 'Televisa en acción'

SELECT \*

FROM Televisa

WHERE rfc NOT IN (

SELECT E.rfc

FROM proyectos P, entregan E

WHERE P.numero = E.numero AND denominacion = 'Educando en Coahuila'

)



Renglones: 3

9) 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).

**SQL**  
SELECT denominacion, P.rfc, razonsocial

FROM proveedores P, proyectos Y, entregan E

WHERE P.rfc = E.rfc AND Y.numero = E.numero AND denominacion = 'Televisa en acción' AND P.rfc NOT IN (

SELECT E.rfc

FROM proyectos P, entregan E

WHERE P.numero = E.numero AND denominacion = 'Educando en Coahuila'

)



Renglones: 3

10) 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.

**SQL**  
SELECT descripcion, precio, denominacion

FROM entregan E, proyectos P, materiales M

WHERE E.numero = P.numero AND E.clave = M.clave AND denominacion = 'Televisa en acción' AND E.rfc IN(

SELECT E.rfc

FROM entregan E, proyectos P

WHERE E.numero = P.numero AND denominacion = 'Educando en Coahuila'

)



Renglones: 2

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

**SQL**  
SELECT M.descripcion, SUM(E.cantidad) 'Veces entregado', SUM(E.cantidad)\*(M.precio+(PorcentajeImpuesto/100)) 'costo'

FROM entregan E, materiales M, proyectos P

WHERE M.clave = E.clave AND P.numero = E.numero

GROUP BY M.descripcion



Renglones: 42