Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Querétaro

Laboratorio 20
DBMS y consultas en SQL

Juan Jose Goyeneche Sánchez - A01712547

Equipo Big Caesars

01 de Mayo del 2025

Construcción de Software y toma de Decisiones

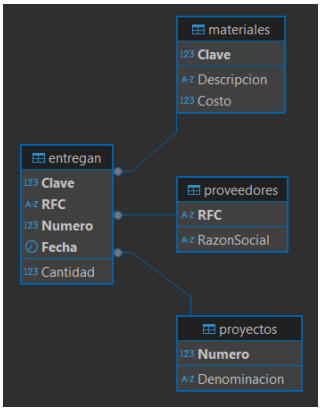
Grupo TC2005B

Consultas básicas en SQL

Como recordarás, todas las consultas que pueden plantearse con álgebra relacional, pueden expresarse con SQL. En esta lectura se ilustra la equivalencia entre la notación del álgebra relacional y la de SQL, por medio de ejemplos basados en un esquema de referencia.
Al diseñar consultas en SQL es importante considerar los siguientes puntos:

La lista de columnas de la cláusula SELECT es la lista de la proyección final (más externa) La lista de tablas de la cláusula FROM incluye a todas las tablas participantes. Las condiciones se expresan en la cláusula WHERE, combinándolas con AND (o con OR según el significado específico).

A modo de referencia, incluimos los esquemas de las tablas que creaste en la práctica anterior y que serán con las que trabajaremos en esta práctica:



Materiales(Clave, Descripción, Costo)
Proveedores(RFC, RazonSocial)
Proyectos(Numero,Denominacion)
Entregan(Clave, RFC, Numero, Fecha, Cantidad)

Convenio: para evitar las letras griegas originales del álgebra relacional, en esta lectura se utiliza la siguiente notación:

SL{condición} : selección con el criterio condición.
PR{lista de columnas}: proyección de lista de columnas.
JN: reunión natural (natural join).
JN{condición}: reunión con el criterio condición (teta join).
UN: unión.
IN: intersección.
- : diferencia
X: producto cartesiano.

Abre una sesión de Analizador de Consultas y ejecuta cada una de las sentencias SQL. 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.

A continuación se presenta la equivalencia entre los operadores y SQL:

Consulta de un tabla completa

Algebra relacional.
materiales

SQL
select * from materiales

	Clave	Descripcion	Costo
1	1	Cemento gris	120,5
2	2	Varilla de acero	75
3	3	Tabique rojo	6,25

Selección

Algebra relacional.
SL{clave=3}(materiales)

SQL
select * from materiales
where clave=3

	Clave	Descripcion	Costo
1	3	Tabique rojo	6,25

Proyección

Algebra relacional.
PR{clave,rfc,fecha} (entregan)

SQL
select clave,rfc,fecha from entregan

	clave	rfc	fecha
1	1	ABC1234567890	2025-04-25
2	1	ABC1234567890	2025-04-27
3	2	XYZ9876543210	2025-04-26

Reunión Natural

Algebra relacional.
entregan JN materiales

SQL
select * from materiales,entregan

where materiales.clave = entregan.clave

Clave	Descripcion	Costo	Clave	RFC	Numero	Fecha	Can
1	Cemento gris	120,5	1	ABC1234567890	101	2025-04-25 GM	
1	Cemento gris	120,5	1	ABC1234567890	101	2025-04-27 GM	
2	Varilla de acero	75	2	XYZ9876543210	102	2025-04-26 GM	

Si algún material no ha se ha entregado ¿Aparecería en el resultado de esta consulta?

No, solo aparecen materiales que coinciden tanto la clave del material como la del producto. Y en ese caso solo tendría la clave materiales.

Reunión con criterio específico

Algebra relacional.
entregan JN{entregan.numero <= proyectos.numero} proyectos

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

Clave	RFC	Numero	Fecha	Cantidad	Numero	Denominacion
1	ABC1234567890	101	2025-04-25 GM	100	101	Construcción de Puente
1	ABC1234567890	101	2025-04-27 GM	150	101	Construcción de Puente
1	ABC1234567890	101	2025-04-25 GM	100	102	Edificio de Oficinas
1	ABC1234567890	101	2025-04-27 GM	150	102	Edificio de Oficinas
2	XYZ9876543210	102	2025-04-26 GM	200	102	Edificio de Oficinas

Nota: falto un ‘;’ al final de la linea del sql.

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)

Clave	RFC	Numero	Fecha	Cantidad
1	ABC1234567	101	2025-04-25	100
1	ABC1234567	101	2025-04-27	150
2	XYZ9876543	102	2025-04-26	200

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

WHERE clave = 1450 **OR** clave = 1300;

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)

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 <> 1;

Clave	RFC	Numero	Fecha	Cantidad
2	XYZ9876543210	102	2025-04-26 GM	200

Producto cartesiano

Algebra relacional.
entregan X materiales

SQL
select * from entregan,materiales

	Clave	RFC	Numero	Fecha	Cantidad	Clave	Descripcion
1	1	ABC1234567890	101	2025-04-25 GM	100	1	Cemento gris
2	1	ABC1234567890	101	2025-04-27 GM	150	1	Cemento gris
3	2	XYZ9876543210	102	2025-04-26 GM	200	1	Cemento gris
4	1	ABC1234567890	101	2025-04-25 GM	100	2	Varilla de acero
5	1	ABC1234567890	101	2025-04-27 GM	150	2	Varilla de acero

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

|Entregan| × |Materiales| = |Tabla entregan| × |Tabla materiales|

Construcción de consultas a partir de una especificación

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

SELECT descripcion FROM materiales – Selecciona solo la descripción de la tabla
JOIN entregan ON materiales.clave = entregan.clave – Que hayan sido entregados
WHERE YEAR(fecha) = 2025; – Se eslablece la fecha a buscar la condicion

descripcion
Cemento gris
Cemento gris
Varilla de acero

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.

¿Por qué aparecen varias veces algunas descripciones de material?
Porque el año, date no es exclusivo y un material puede ser entregado varias veces en un año.

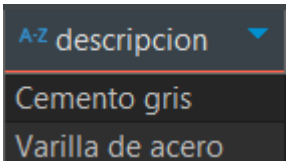
Uso del calificador distinct

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.

```
SELECT DISTINCT descripcion FROM materiales
JOIN entregan ON materiales.clave = entregan.clave
WHERE YEAR(fecha) = 2025;
```

¿Qué resultado obtienes en esta ocasión?



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.

```
SELECT proyectos.numero, proyectos.denominacion, entregan.fecha, entregan.cantidad
FROM proyectos
JOIN entregan ON proyectos.numero = entregan.numero
ORDER BY proyectos.numero, entregan.fecha DESC;
```

Uso de expresiones.

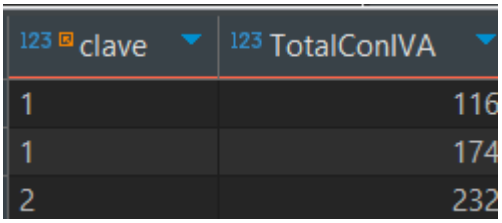
En álgebra relacional los argumentos de una proyección deben ser columnas. Sin embargo en una sentencia SELECT es posible incluir expresiones aritméticas o funciones que usen como argumentos de las columnas de las tablas involucradas o bien constantes. Los operadores son:

+ 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.

SELECT clave, cantidad * 1.16 AS TotalConIVA
FROM entregan;



clave	TotalConIVA
1	116
1	174
2	232

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?

En este caso ninguno, por que no existe tabla productos.

Explica que hace el símbolo '%'.

Representa cualquier número de caracteres.

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

Solo encuentra coincidencias exactas, es decir, cadenas que sean exactamente "Si".

¿Qué resultado obtienes?
Explica a qué se debe este comportamiento.

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?
¿Que resultado obtienes? ¿¿¿???

¿Para qué sirve DECLARE?
Para crear variables locales

¿Cuál es la función de @foo?
Una variable tipo cadena donde se van insertando estas, es una variable como podría ser X.

¿Que realiza el operador SET?
Asigna 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 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:

SELECT RFC FROM Entregan WHERE RFC LIKE '[A-D]%';
Selecciona dentro de la tabla entregan, dentro de la columna RFC cuando, inicia con la larrea, ABCD.

SELECT RFC FROM Entregan WHERE RFC LIKE '[^A]%';
Selecciona dentro de la tabla entregan, dentro de la columna RFC cuando, no empieza con la letra A.

SELECT Numero FROM Entregan WHERE Numero LIKE '___6';
encuentra numeros de 4 cifras donde la última sea 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?
WHERE Fecha BETWEEN '2000-01-01' AND '2000-12-31'

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] )
```

¿Qué hace la consulta?
Devuelve entregas cuyo número esté entre el 5000 y el 5010 y que el proveedor tenga una razón social que empieza con “La”.

¿Qué función tiene el paréntesis ( ) después de EXISTS?
Encierra una subconsulta, la cual determina si existe al menos una fila que cumpla la condición.

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

```
SELECT RFC, Cantidad, Fecha, Numero
FROM Entregan
WHERE Numero BETWEEN 5000 AND 5010 AND
RFC IN (
  SELECT RFC
  FROM Proveedores
  WHERE RazonSocial LIKE 'La%'
);
```

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.

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
```

Devuelve las primeras dos filas de la tabla proyectos.

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

SELECT TOP Numero FROM Proyectos

Error, seria TOP 1.

Modificando la estructura de un 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 e.Clave, m.Precio, e.Cantidad, m.PorcentajeImpuesto,
      (e.Cantidad * m.Precio * (1 + m.PorcentajeImpuesto / 100)) AS ImporteTotal
FROM Entregan e
JOIN materiales m ON 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
JOIN entregan e ON m.Clave = e.Clave
JOIN proyectos p ON p.IdProyecto = e.IdProyecto
WHERE p.Denominacion = '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
JOIN entregan e ON m.Clave = e.Clave
JOIN proveedores p ON p.RFC = e.RFC
WHERE p.RazonSocial = 'Acme tools';
```

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

```
SELECT RFC
FROM entregan
WHERE YEAR(Fecha) = 2000
GROUP BY RFC
HAVING AVG(Cantidad) >= 300;
```

El Total entregado por cada material en el año 2000.

```
SELECT Clave, SUM(Cantidad) AS TotalEntregado
FROM entregan
WHERE YEAR(Fecha) = 2000
GROUP BY Clave;
```

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

```
CREATE VIEW vistaVentas2001 AS
SELECT Clave, SUM(Cantidad) AS Total
FROM entregan
WHERE YEAR(Fecha) = 2001
GROUP BY Clave;

SELECT TOP 1 Clave
FROM vistaVentas2001
ORDER BY Total DESC;
```

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

```
SELECT * FROM materiales
WHERE Descripcion LIKE '%ub%';
```

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

```
SELECT p.Denominacion, SUM(e.Cantidad * m.Precio * (1 +
m.PorcentajeImpuesto / 100)) AS TotalPagar
FROM entregan e
JOIN materiales m ON e.Clave = m.Clave
JOIN proyectos p ON e.IdProyecto = p.IdProyecto
GROUP BY p.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 ProveedoresA AS
SELECT DISTINCT p.Denominacion, e.RFC, pr.RazonSocial
FROM entregan e
JOIN proyectos p ON e.IdProyecto = p.IdProyecto
JOIN proveedores pr ON pr.RFC = e.RFC
WHERE p.Denominacion = 'Televisa en acción';

CREATE VIEW ProveedoresB AS
SELECT DISTINCT e.RFC
FROM entregan e
JOIN proyectos p ON e.IdProyecto = p.IdProyecto
WHERE p.Denominacion = 'Educando en Coahuila';

SELECT * FROM ProveedoresA
WHERE RFC NOT IN (SELECT RFC FROM ProveedoresB);
```

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 DISTINCT p.Denominacion, e.RFC, pr.RazonSocial
FROM entregan e
JOIN proyectos p ON e.IdProyecto = p.IdProyecto
JOIN proveedores pr ON pr.RFC = e.RFC
WHERE p.Denominacion = 'Televisa en acción'
AND e.RFC NOT IN (
  SELECT e2.RFC
  FROM entregan e2
  JOIN proyectos p2 ON e2.IdProyecto = p2.IdProyecto
  WHERE p2.Denominacion = 'Educando en Coahuila'
);
```

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