ADMINISTRACIÓN DE BASE DE DATOS

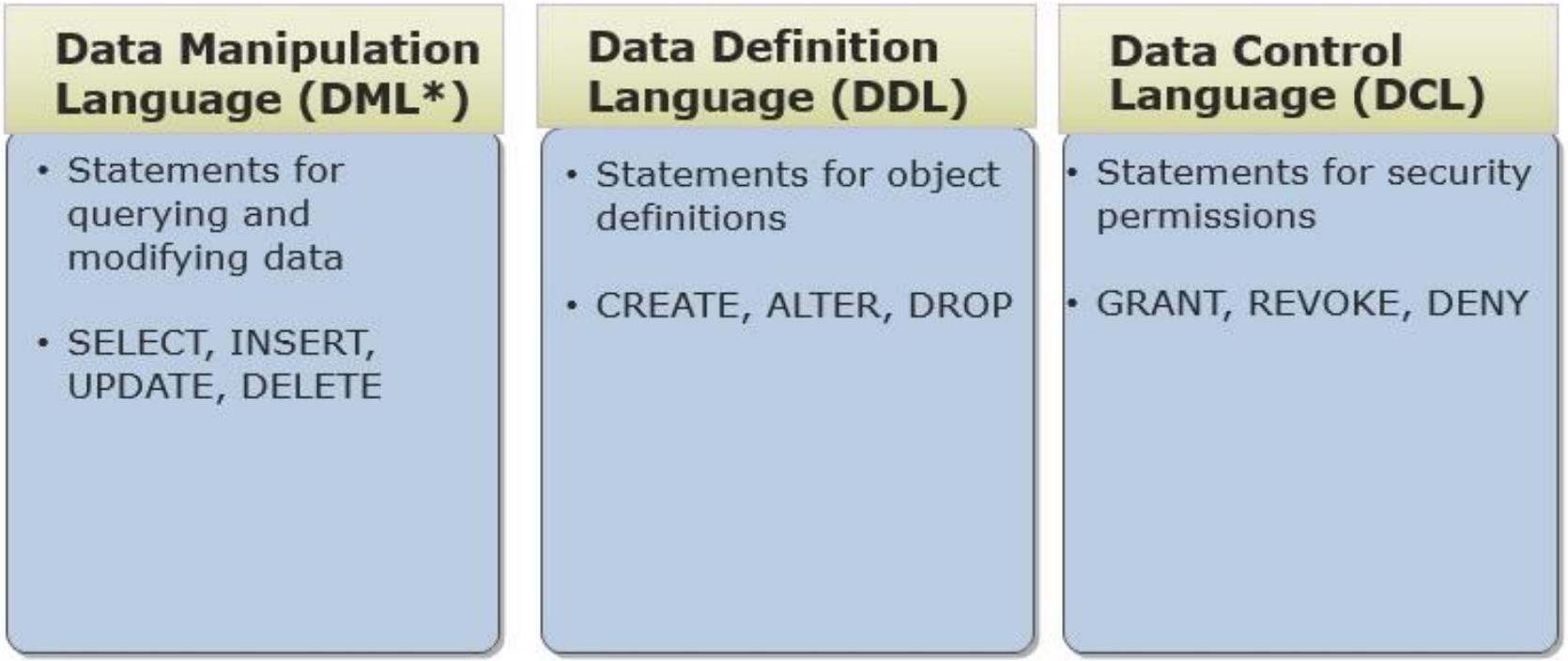
Unidad I

RECUPERACIÓN AVANZADA DE DATOS

Ana E. Congacha
acongacha@unach.edu.ec

LENGUAJE DE CONSULTA ESTRUCTURADO

Tipos de instrucciones de SQL



ELIMINACIÓN DE UNA TABLA

La eliminación de una tabla elimina la definición de la tabla y todos sus datos, así como las especificaciones de permisos sobre dicha tabla.

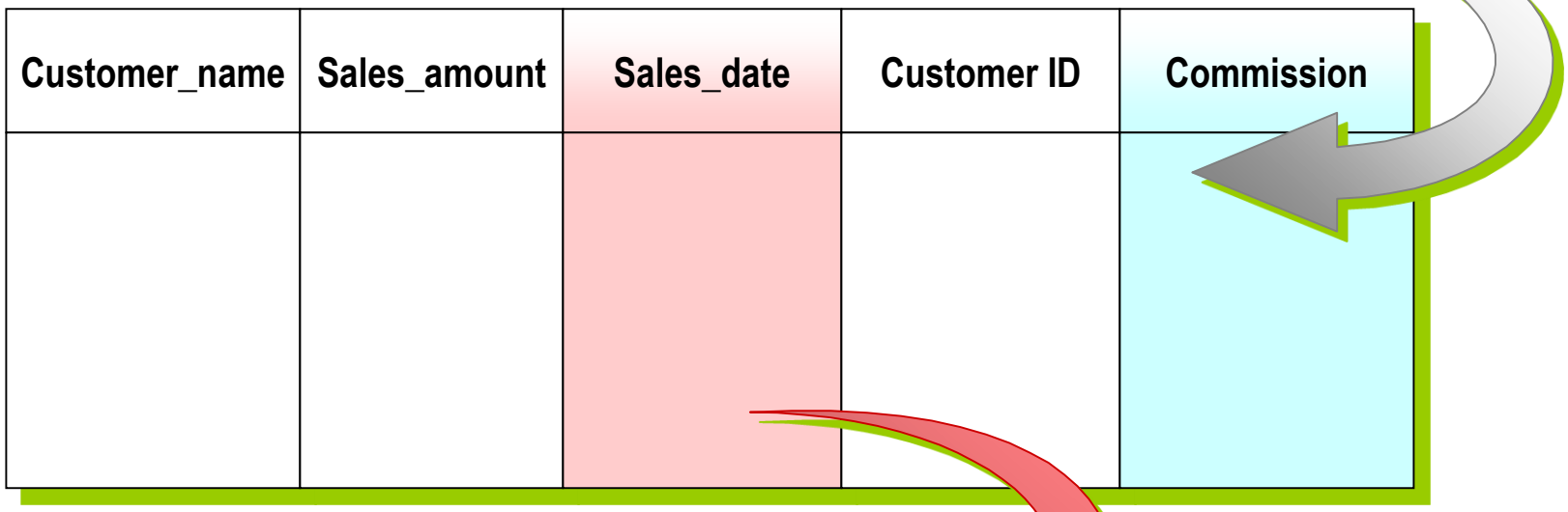
Sintaxis

DROP TABLE *nombreTabla [,…n]*

AGREGAR Y QUITAR COLUMNAS DE UNA TABLA

AGREGAR

```
ALTER TABLE CategoriesNew
    ADD Commission money null
```



QUITAR

```
ALTER TABLE CategoriesNew
    DROP COLUMN Sales_Date
```

DML

INSERCIÓN DE UNA FILA DE DATOS MEDIANTE VALORES

- Debe atenerse a las restricciones de destino o la transacción INSERT fallará.
- Use una lista de columnas para especificar las columnas de destino.
- Especifique una lista de valores correspondiente (‘’ caracter y fechas)

```
USE northwind
INSERT clientes
       (customerid, companyname, contactname, contacttitle
      ,address, city, region, postalcode, country, phone
      ,fax)

VALUES ('PECOF', 'Pecos Coffee Company', 'Michael Dunn'
          ,'Owner', '1900 Oak Street', 'Vancouver', 'BC'
          ,'V3F 2K1', 'Canada', '(604) 555-3392'
          ,'(604) 555-7293')
GO
```

USO DE LA INSTRUCCIÓN INSERT…SELECT

- Todas las filas que cumplan la instrucción SELECT se insertan.
- Compruebe que existe la tabla que recibe las nuevas filas.
- Asegúrese de que son compatibles los tipos de datos.
- Determine si existe un valor predeterminado o si se permiten valores Null

```
USE northwind
INSERT customers
 SELECT substring (firstname, 1, 3)
         + substring (lastname, 1, 2)
        ,lastname, firstname, title, address, city
        ,region, postalcode, country, homephone, NULL
  FROM employees
GO
```

Uso de la instrucción DELETE

```
USE northwind
DELETE orders
WHERE shipcountry=’spain’
GO
```

Uso de la instrucción TRUNCATE TABLE

```
USE northwind
TRUNCATE TABLE orders
GO
```

Actualización de filas basada en datos de la tabla

```
UPDATE products
SET unitprice = (unitprice * 1.1)
where unitprice > 1.5
GO
```

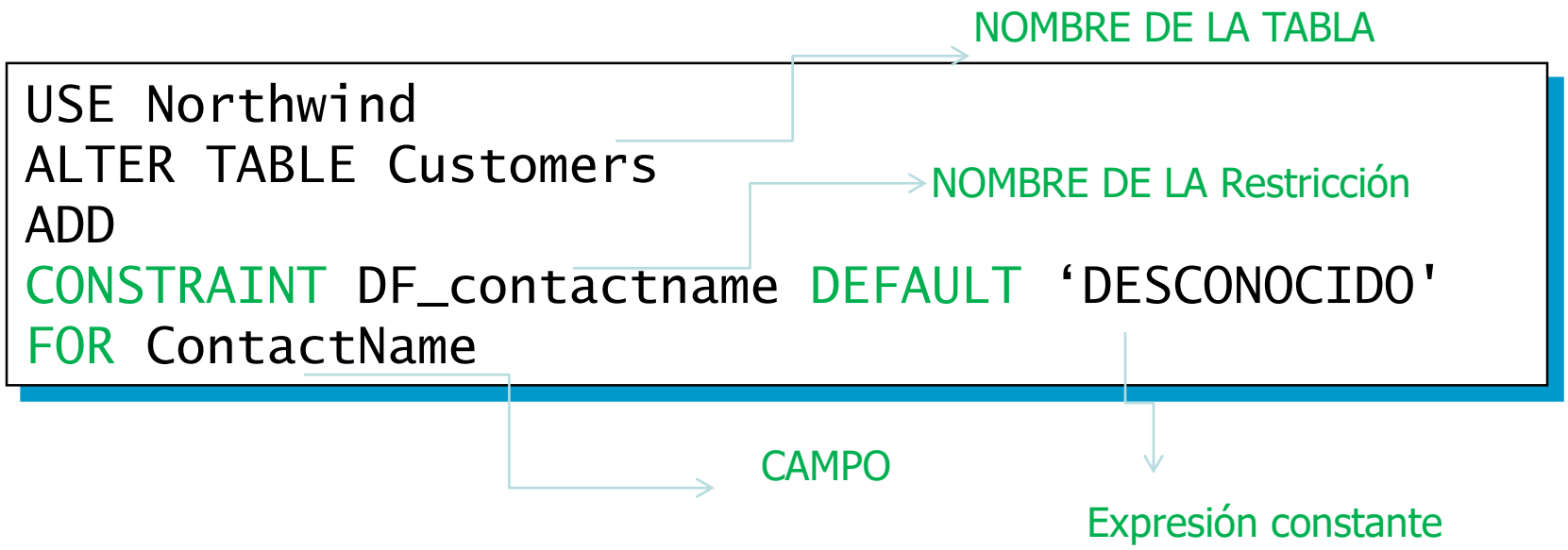
```
UPDATE DimEmployee
SET FirstName = 'Gail'
WHERE EmployeeKey = 500;
```

DETERMINACIÓN DEL TIPO DE RESTRICCIÓN QUE SE VA A UTILIZAR

Tipo de integridad	Tipo de restricción
Dominio	DEFAULT
	CHECK
	REFERENTIAL
Entidad	PRIMARY KEY
	UNIQUE
Referencial	FOREIGN KEY
	CHECK

Restricciones DEFAULT

- Sólo una restricción DEFAULT por columna
- No se puede utilizar con la propiedad IDENTITY



Restricciones CHECK

- Restringe los datos que los usuarios pueden escribir en una columna particular a unos valores específicos. Las restricciones CHECK son similares a las cláusulas WHERE donde se pueden especificar las condiciones bajo las que se aceptan los datos.
- Se utilizan con las instrucciones INSERT y UPDATE
- Pueden hacer referencia a otras columnas en la misma tabla

```
USE Northwind
ALTER TABLE dbo.Employees
ADD
CONSTRAINT CK_birthdate
CHECK (BirthDate > '01-01-1900' AND BirthDate < getdate())
```

NOMBRE DE LA Restricción

Expresión lógica

Ejemplo

```
ALTER TABLE DocExc
ADD ColumnD int NULL
CONSTRAINT CHK_ColumnD_DocExc
CHECK (ColumnD > 10 AND ColumnD < 50);
GO

-- Adding values that will pass the check constraint
INSERT INTO DocExc (ColumnD) VALUES (49);
GO

-- Adding values that will fail the check constraint
INSERT INTO DocExc (ColumnD) VALUES (55);
GO
```

Restricciones PRIMARY KEY

- Sólo una restricción PRIMARY KEY por tabla
- No se permiten valores nulos

```
USE Northwind
ALTER TABLE Customers
ADD
CONSTRAINT PK_Customers PRIMARY KEY (CustomerID)
```

Restricciones UNIQUE

- Permite varias restricciones UNIQUE en una tabla

```
USE Northwind
ALTER TABLE Suppliers
ADD
CONSTRAINT U_CompanyName
  UNIQUE (CompanyName)
```

Restricciones FOREIGN KEY

- Deben hacer referencia a una restricción PRIMARY KEY
- Proporcionan integridad referencial de una o de varias columnas

```
USE Northwind
ALTER TABLE Orders
ADD

CONSTRAINT FK_Orders_Customers
  FOREIGN KEY (CustomerID)
  REFERENCES Customers(CustomerID)
```

SELECT

- La instrucción SELECT permite recuperar datos.
- La sintaxis básica de una consulta de selección es la siguiente:

SELECT [ALL | DISTINCT] <listaSelección>

FROM {<tablaOrigen>}

WHERE <condiciónBúsqueda>

Ejemplos

ESTUDIANTE (CódigoE, Nombre, Especialidad, Grado)

CLASE (CodigoC, Nombre, Horario, Aula)

INSCRIPCION (CodigoI, NombreAspirante)

```
SELECT CódigoE, Nombre, Especiliadad
FROM ESTUDIANTE
```

```
SELECT Especialidad
FROM ESTUDIANTE
```

```
USE northwind
SELECT employeeid, lastname, firstname, title
FROM employees
GO
```

Eliminación de filas duplicadas DISTINCT

Si necesita obtener una lista de valores únicos, puede utilizar la cláusula DISTINCT para eliminar las filas duplicadas del conjunto de resultados.

```
SELECT DISTINCT Especialidad
FROM ESTUDIANTE
```

Elimina renglones duplicado

El uso de la cláusula **WHERE** permite obtener filas en función de las condiciones de búsqueda que se especifiquen.

Estructura fundamental SQL

```
SELECT CódigoE, Nombre, Especiliadad, Grado
FROM ESTUDIANTE
WHERE Especialidad = "Matemáticas"
```

Proporciona la (s) condición (es) para la selección

Forma equivalente de la consulta anterior

* Todos los campos de la tabla

```
SELECT *
FROM ESTUDIANTE
WHERE Especialidad = "Matemáticas"
```

FILTROS DE DATOS

A veces se desea limitar el conjunto de resultados que devuelve una consulta. Para limitar los resultados se puede especificar condiciones de búsqueda en una cláusula WHERE con las que filtrar los datos. No hay límite en el número de condiciones de búsqueda que se pueden incluir en una instrucción SELECT. La tabla siguiente describe el tipo de filtro y la correspondiente condición de búsqueda que se puede usar para filtrar los datos.

Tipo de filtro	Condición de búsqueda
Operadores de comparación	=, >, <, >=, <= y <>
Comparaciones de cadenas	LIKE y NOT LIKE
Operadores lógicos: combinación de condiciones	AND, OR
Operador lógico: negación	NOT
Intervalo de valores	BETWEEN y NOT BETWEEN
Listas de valores	IN y NOT IN
Valores desconocidos	IS NULL e IS NOT NULL

```
USE northwind
SELECT employeeid, lastname, firstname, title
FROM employees
WHERE employeeid = 5

USE northwind
SELECT lastname, city
FROM employees
WHERE country = 'USA'

USE northwind
SELECT orderid, customerid
FROM orders
WHERE orderdate < '1/8/96'
```

```
SELECT  Nombre, Grado
FROM ESTUDIANTE
WHERE Especialidad = "Matemáticas" and Grado = "Primero"
```

```
SELECT  Nombre
FROM ESTUDIANTE
WHERE Especialidad IN ( "Matemáticas", "Contabilidad ")
```

Equivalente al operador OR

Desplegar los nombres de los estudiantes que tienen una especialidad ya sea en Matemáticas o en contabilidad.

```
SELECT  Nombre
FROM ESTUDIANTE
WHERE Especialidad  NOT IN ( "Matemáticas", "Contabilidad")
```

*Desplegar los nombres de los estudiantes que **no** tienen especialidad en Matemáticas o en contabilidad.*

BETWEEN

- Las cláusulas WHERE también se pueden referir a rangos.

```
SELECT  Nombre, Especialidad
FROM  ESTUDIANTE
WHERE EID BETWEEN 200 AND 300
```

Expresión equivalente

```
SELECT  Nombre, Especialidad
FROM  ESTUDIANTE
WHERE EID >= 200 AND EID <= 300
```

LIKE

- Esta palabra reservada se utiliza para seleccionar valores parciales.

Tipos de caracteres comodín

Utilice los siguientes caracteres comodín para formar el criterio de búsqueda de la cadena de caracteres:

Comodín	Descripción
%	Cualquier cadena de cero o más caracteres.
_	Cualquier carácter individual.
[]	Cualquier carácter individual contenido en el intervalo o conjunto especificado.
[^]	Cualquier carácter individual *no* contenido en el intervalo o conjunto especificado.

Expresión	Devuelve
`LIKE 'BR%'`	Todos los nombres que comiencen por las letras BR.
`LIKE 'Br%'`	Todos los nombres que comiencen por las letras Br.
`LIKE '%een'`	Todos los nombres que terminen con las letras een.
`LIKE '%en%'`	Todos los nombres que contengan las letras en.
`LIKE '_en'`	Todos los nombres de tres letras que terminen con las letras en.
`LIKE '[CK]%'`	Todos los nombres que comiencen por C o por K.
`LIKE '[S-V]ing'`	Todos los nombres de cuatro letras que terminen con las letras ing y comiencen por cualquier letra comprendida entre S y V.
`LIKE 'M[^c]%'`	Todos los nombres que comiencen por la letra M y cuya segunda letra no sea c.

Ejemplo

```
USE northwind
SELECT companyname
FROM customers
WHERE companyname LIKE 'A%'
```

IS NULL

- Una columna tiene un valor NULL cuando no se ha especificado ningún valor para ella durante la entrada de datos y no tiene definido un valor predeterminado. Un valor NULL no es lo mismo que cero (que es un valor numérico) o blanco (que es un valor de carácter).
- Esta palabra reservada se utiliza para buscar valores nulos (o faltantes) de una o más columnas.

```
SELECT  Nombre
FROM  ESTUDIANTE
WHERE Grado IS NULL
```

```
USE northwind
SELECT companyname, fax
FROM suppliers
WHERE fax IS NULL
GO
```

FUNCIONES SQL INTERCONSTRUIDAS

(O FUNCIONES AGREGADAS)

NO INCLUYEN NINGUN CAMPO NULL EN SU CALCULO

- COUNT CONTAR (NÚMERO DE RENGLONES DE UNA TABLA)
- SUM SUMAR (TOTALIZA COLUMNAS NUMERICAS)
- AVG PROMEDIAR
- MAX VALOR MAXIMO
- MIN VALOR MINIMO

```
SELECT  COUNT (*) as TotalEmpleados
FROM  Employees
```

```
SELECT  COUNT (Region)
FROM  Employees
```

```
SELECT  COUNT (DISTINCT [Region])
FROM  Employees
```

```
SELECT  SUM  (Precio*Cantidad)
FROM DetallePedido
```

```
SELECT  AVG  (Gastos)
FROM Pedido
WHERE Gastos >100
```

```
SELECT  MIN  (Gastos)
FROM Pedido
WHERE Pais = "España"
```

```
SELECT  MAX  (Gastos)
FROM Pedido
WHERE Pais = "España"
```

Identificadores

- **Identificadores estándar**
  - El primer carácter debe ser un carácter alfabético
  - Otros caracteres pueden incluir letras, números o símbolos
  - Los identificadores que comienzan con un símbolo tienen usos especiales
- **Identificadores delimitados**
  - Se utilizan cuando los nombres contienen espacios incrustados
  - Se utilizan cuando partes de los nombres incluyen palabras reservadas
  - Deben encerrarse entre corchetes ([ ]) o dobles comillas (" ")

EJEMPLOS

```
SELECT * FROM [Blanks In Table Name]
```

```
SELECT *
FROM [TableX]          --Delimiter is optional.
WHERE [KeyCol] = 124   --Delimiter is optional.
```

```
SELECT *
FROM [My Table]      --Identifier contains a space and uses a reserved keyword.
WHERE [order] = 10   --Identifier is a reserved keyword.
```

Directrices de denominación para los identificadores

- **Poner nombres cortos**
- **Utilizar nombres significativos cuando sea posible**
- **Utilizar una convención de denominación clara y sencilla**
- **Utilizar un identificador que distinga el tipo de objeto**
  - Vistas
  - Procedimientos almacenados

Cambio de nombre de las columnas

Si desea crear nombres de columnas más legibles, puede utilizar la palabra clave **AS** para reemplazar los nombres predeterminados por ***alias*** en la lista de selección.

```
USE northwind
SELECT  firstname AS First, lastname AS Last
       ,employeeid AS 'Employee ID:'
FROM employees
GO
```

First	Last	Employee ID:
Nancy	Davolio	1
Andrew	Fuller	2
Janet	Leverling	3
Margaret	Peacock	4
Steven	Buchanan	5
Michael	Suyama	6
Robert	King	7
Laura	Callahan	8
Anne	Dodsworth	9

Ejemplos

```
SELECT [CompanyName] AS Customer, ContactName AS [Contact Person]
FROM Customers
ORDER BY  [CompanyName] ASC

SELECT [CompanyName], Address + ', ' + PostalCode + ' ' + City + ', ' + Country AS Address
FROM Customers
ORDER BY  [CompanyName] DESC

SELECT [EmployeeID],  [FirstName] + ' '+ [LastName] AS NAME
FROM employees
WHERE [FirstName] = 'Laura'

SELECT o.OrderID, o.OrderDate, c.[CompanyName]
FROM Customers AS c, Orders AS o
WHERE c.CustomerID=o.CustomerID and c.[CompanyName]='Around the Horn'
```

Tipos de Datos

Categorías de tipos de datos

Los tipos de datos en SQL Server están organizados en las siguientes categorías:

Números exactos	Cadenas de caracteres Unicode
Números aproximados	Cadenas binarias
Fecha y hora	Otros tipos de datos

Debe elegir los tipos de datos adecuados que le permitan optimizar el rendimiento y conservar espacio en el disco.

Tipos de datos numéricos exactos

Los tipos de datos numéricos exactos permiten especificar *exactamente* la escala y la precisión que se va a utilizar. Por ejemplo, puede especificar tres cifras a la derecha de la coma decimal y cuatro a la izquierda. Las consultas devuelven siempre los datos introducidos. SQL Server acepta dos tipos de datos numéricos exactos por compatibilidad con ANSI: **decimal** y **numeric**.

En general, los tipos de datos numéricos exactos se deben utilizar en aplicaciones financieras en las que se quiere que los datos se reflejen de forma coherente (siempre con dos decimales) y en las consultas sobre dichas columnas (por ejemplo, para buscar todos los préstamos con un interés del 8,75 por ciento).

Tipos de datos numéricos aproximados

Los tipos de datos numéricos aproximados almacenan datos de la forma más precisa posible. Por ejemplo, la fracción 1/3 se representa en un sistema decimal como 0,33333 (periódico). El número no se puede almacenar con precisión, de modo que se almacena una aproximación. SQL Server acepta dos tipos de datos aproximados: **float** y **real**.

Si redondea números o realiza comprobaciones de calidad entre valores, debe evitar el uso de tipos de datos numéricos aproximados.

Creación y eliminación de tipos de datos definidos por el usuario

Creación

```
EXEC sp_addtype  city, 'nvarchar(15)', NULL
EXEC sp_addtype  region, 'nvarchar(15)', NULL
EXEC sp_addtype  country, 'nvarchar(15)', NULL
```

Eliminación

```
 EXEC  sp_droptype city
```

VARIABLES

- **Variable definida por el usuario en una instrucción DECLARE @** **Ejemplo:** DECLARE @NOMBREVARIABLE TIPO
- **Valores asignados con una instrucción SET o SELECT**
- **Las variables tienen el ámbito Local (@)**

```
USE northwind

DECLARE  @EmpID  varchar(11),@vlName char(20)
SET @vlname = 'Dodsworth'
SELECT @EmpID = employeeid
 FROM  employees
 WHERE LastName = @vlname
SELECT @EmpID AS EmployeeID
GO
```

Nota

- Los nombres de algunas funciones del sistema Transact-SQL comienzan por dos *arrobas* (@@). A pesar de que en versiones anteriores de SQL Server se hacía referencia a las funciones que empiezan por @@ como variables globales, estas funciones que empiezan por @@ no son variables y no tienen el mismo comportamiento. Las funciones que empiezan por @@ son funciones del sistema, y el uso de su sintaxis sigue las reglas de las funciones.

@@error
@@rowcount
@@trancount

- https://learn.microsoft.com/es-es/sql/t-sql/language-elements/variables-transact-sql?view=sql-server-ver16

ANALISIS

```
CREATE TABLE TestTable (cola int, colb char(3))

DECLARE @MyCounter int

-- Initialize the variable.
SET @MyCounter = 0

WHILE (@MyCounter < 26)
BEGIN;
   -- Insert a row into the table.
   INSERT INTO TestTable VALUES
   (@MyCounter, CHAR( ( @MyCounter + ASCII('a') ) ));
   SET @MyCounter = @MyCounter + 1
END

-- View the data.
SELECT cola, colb
FROM TestTable

DROP TABLE TestTable
```

ANALISIS

```
USE AdventureWorks2012;
GO
DECLARE @find varchar(30);
/* Also allowed:
DECLARE @find varchar(30) = 'Man%';
*/
SET @find = 'Man%';
SELECT p.LastName, p.FirstName, ph.PhoneNumber
FROM Person.Person AS p
JOIN Person.PersonPhone AS ph ON p.BusinessEntityID = ph.BusinessEntityID
WHERE LastName LIKE @find;
```

```
LastName            FirstName                Phone
------------------- ------------------------ --------------------------
Manchepalli         Ajay                     1 (11) 500 555-0174
Manek               Parul                    1 (11) 500 555-0146
Manzanares          Tomas                    1 (11) 500 555-0178

(3 row(s) affected)
```

ANALISIS

```
-- Uses AdventureWorks

DECLARE @lastName varchar(30), @firstName varchar(30);

SET @lastName = 'Walt%';
SET @firstName = 'Bryan';

SELECT LastName, FirstName, Phone
FROM DimEmployee
WHERE LastName LIKE @lastName AND FirstName LIKE @firstName;
```

FUNCIONES DE AGRUPAMIENTO

Presentación de los primeros *n* valores - TOP

- Presenta sólo las *n* primeras filas de un conjunto de resultados
- Especifica el intervalo de valores con la cláusula ORDER BY
- Devuelve las filas iguales si se utiliza WITH TIES

Ejemplo 1

```
USE northwind
SELECT TOP 5 orderid, productid, quantity
 FROM [order details]
 ORDER BY quantity DESC
GO
```

Ejemplo 2

```
USE northwind
SELECT TOP 5 WITH TIES orderid, productid, quantity
 FROM [order details]
 ORDER BY quantity DESC
GO
```

ANALISIS

```
CREATE TABLE dbo.Cars(Model varchar(15), Price money, Color varchar(10));

INSERT dbo.Cars VALUES
    ('sedan', 10000, 'red'), ('convertible', 15000, 'blue'),
    ('coupe', 20000, 'red'), ('van', 8000, 'blue');

select *
from [dbo].[Cars]

SELECT TOP(1) Model, Color, Price
FROM dbo.Cars
WHERE Color = 'red'
UNION ALL
SELECT TOP(1) Model, Color, Price
FROM dbo.Cars
WHERE Color = 'blue'
ORDER BY Price ASC
```

Nota: No todos los sistemas de bases de datos admiten la cláusula SELECT TOP.

MySQL admite la cláusula LIMIT para seleccionar un número limitado de registros, mientras que Oracle utiliza ROWNUM.

Ejemplo

```
SELECT * FROM Customers
LIMIT 3;
```

Uso de la cláusula GROUP BY

```
USE northwind
SELECT productid, orderid
      ,quantity
 FROM orderhist
GO
```

```
USE northwind
SELECT productid
   ,SUM(quantity) AS total_quantity
FROM orderhist
GROUP BY productid
GO
```

productid	orderid	quantity
1	1	5
1	1	10
2	1	10
2	2	25
3	1	15
3	2	30

Sólo se agrupan las filas que cumplan la cláusula WHERE

productid	total_quantity
1	15
2	35
3	45

productid	total_quantity
2	35

```
USE northwind
SELECT productid
   ,SUM(quantity) AS total_quantity
 FROM orderhist
 WHERE productid = 2
 GROUP BY productid
GO
```

Uso de la cláusula GROUP BY con la cláusula HAVING

```
USE northwind
SELECT productid, orderid
       ,quantity
 FROM orderhist
GO
```

```
USE northwind
SELECT productid, SUM(quantity)
   AS total_quantity
 FROM orderhist
 GROUP BY productid
 HAVING SUM(quantity)>=30
GO
```

productid	orderid	quantity
1	1	5
1	1	10
2	1	10
2	2	25
3	1	15
3	2	30

productid	total_quantity
1	15
2	35
3	45

productid	total_quantity
2	35
3	45

- **Where** para excluir filas que no desea agrupar

```
USE northwind
SELECT productid
    ,SUM(quantity) AS total_quantity
 FROM orderhist
 WHERE productid = 2
 GROUP BY productid
GO
```

- **Having** para filtrar los registros una vez agrupados

```
USE northwind
SELECT productid, SUM(quantity)
    AS total_quantity
 FROM orderhist
 GROUP BY productid
 HAVING SUM(quantity)>=30
GO
```

EJEMPLOS

```
SELECT *
FROM Employees

select Title, count(*)
from Employees
GROUP BY [Title]
ORDER BY TITLE

select Title, count(*)
from Employees
GROUP BY [Title]
HAVING count(*) >=6
```

SUBCONSULTAS

- Son un conjunto de instrucciones SELECT.
- Es una consulta que está anidada dentro de una declaración SELECT.
- Utiliza los resultados de una consulta dentro de otra, que se considera la principal.
- A menudo, las subconsultas se pueden escribir como combinaciones.

```
--Utilizando Subconsultas
SELECT ProductName
FROM Products
WHERE UnitPrice=
    (SELECT UnitPrice
     FROM Products
     WHERE ProductName = 'Chai' )
```

```
--Utilizando JOIN que retorna el mismo resultado
SELECT Prd1.ProductName
FROM Products AS Prd1
     JOIN Products AS Prd2
       ON (Prd1.UnitPrice = Prd2.UnitPrice)
WHERE Prd2.ProductName = 'Chai'
```

```sql
SELECT ProductName
FROM Products
WHERE  UnitPrice=
    (SELECT UnitPrice
     FROM Products
     WHERE ProductName = 'Chai')
```

Visualizar los nombres de los productos cuyo precio unitario es igual al precio Unitario del producto 'Chai'

Visualizar el nombre del producto con el precio Unitario máximo.

```sql
select ProductName, UnitPrice
From Products
where UnitPrice =
      (Select max(UnitPrice)
      from Products)
```

- **Por qué utilizar subconsultas**
  - Para dividir una consulta compleja en varios pasos lógicos
  - Para responder una consulta que depende de los resultados de otra consulta
- **Por qué utilizar combinaciones en lugar de subconsultas**
  - SQL Server ejecuta combinaciones más rápidas que la subconsultas

Normalmente, utilizar combinaciones permite al optimizador de consultas recuperar datos de forma más eficiente.

Uso de una subconsulta como una expresión

Results | Messages

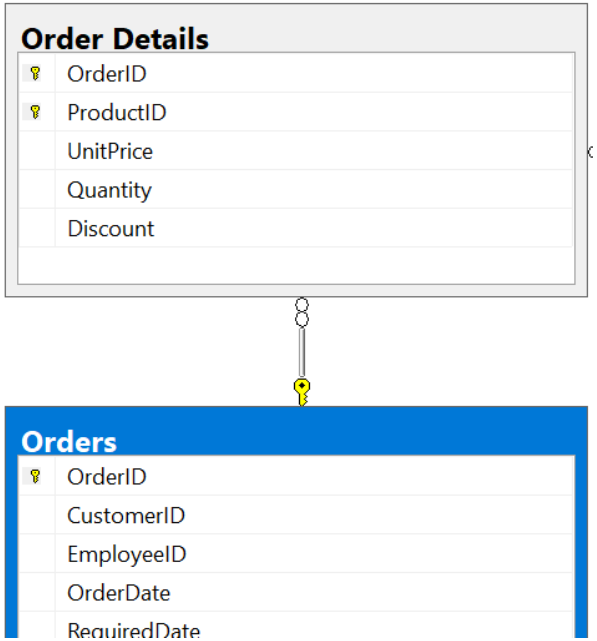
	ProductName	UnitPrice	average	difference
1	Chai	18,00	28,8663	-10,8663
2	Chang	19,00	28,8663	-9,8663
3	Aniseed Syrup	10,00	28,8663	-18,8663
4	Chef Anton's Cajun Seasoning	22,00	28,8663	-6,8663
5	Chef Anton's Gumbo Mix	21,35	28,8663	-7,5163
6	Grandma's Boysenberry Spread	25,00	28,8663	-3,8663
7	Uncle Bob's Organic Dried Pears	30,00	28,8663	1,1337
8	Northwoods Cranberry Sauce	40,00	28,8663	11,1337
9	Mishi Kobe Niku	97,00	28,8663	68,1337
10	Ikura	31,00	28,8663	2,1337
11	Queso Cabrales	21,00	28,8663	-7,8663
12	Queso Manchego La Pastora	38,00	28,8663	9,1337
13	Konbu	6,00	28,8663	-22,8663
14	Tofu	23,25	28,8663	-5,6163

```
SELECT
[ProductName],[UnitPrice],(SELECT
AVG([UnitPrice]) FROM[dbo].[Products]
) AS average,[UnitPrice]-(SELECT
AVG([UnitPrice]) FROM[dbo].[Products]
) AS difference
FROM [dbo].[Products]
```

1: UnitPrice
2: Promedio del UnitPrice
3: Diferencia de UnitPrice – Promedio

Consultas de tablas múltiples

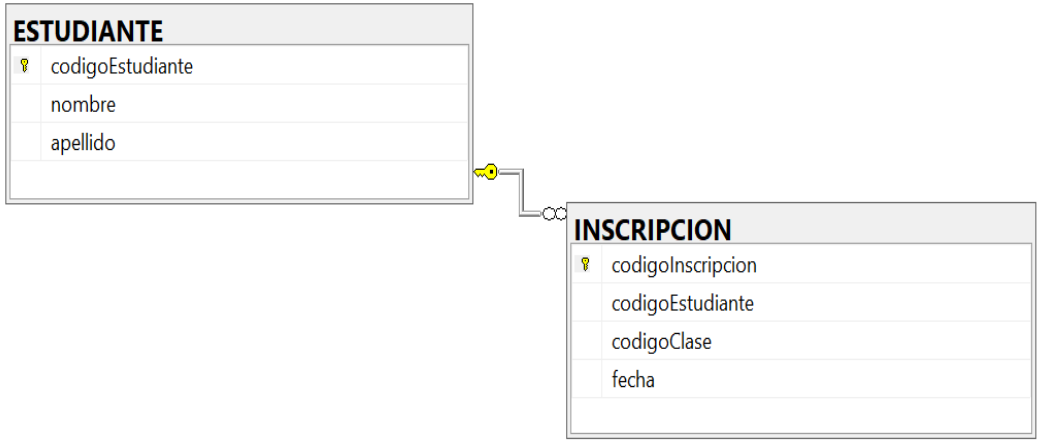
Devuelve una lista de clientes que han pedido más de 20 unidades del producto número 23.




```
SELECT orderid, customerid
FROM orders AS or1
WHERE orderid in (SELECT orderid
                  FROM [order details] AS od
                  WHERE od.ProductID = 23 and
                        Quantity>20)
```

Consultas de tablas múltiples

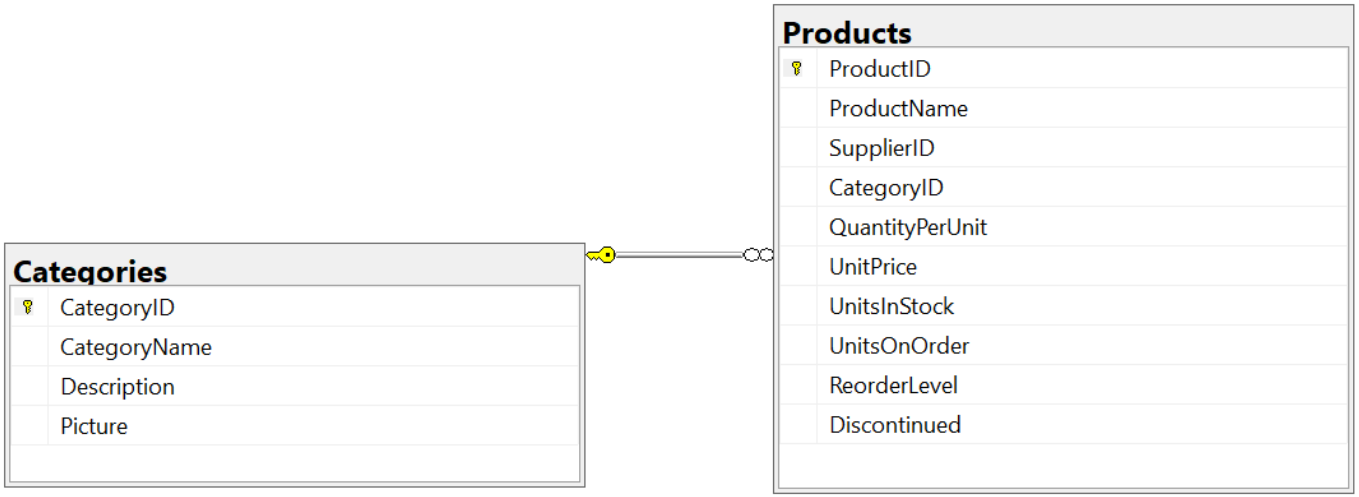
Listado de estudiantes inscritos en la clase DIBU1




```
SELECT  codigoEstudiante, nombre, apellido
FROM ESTUDIANTE
WHERE codigoEstudiante IN
        (SELECT codigoEstudiante
        FROM INSCRIPCION
        WHERE codigoClase = 'DIBU1')
```

subconsulta

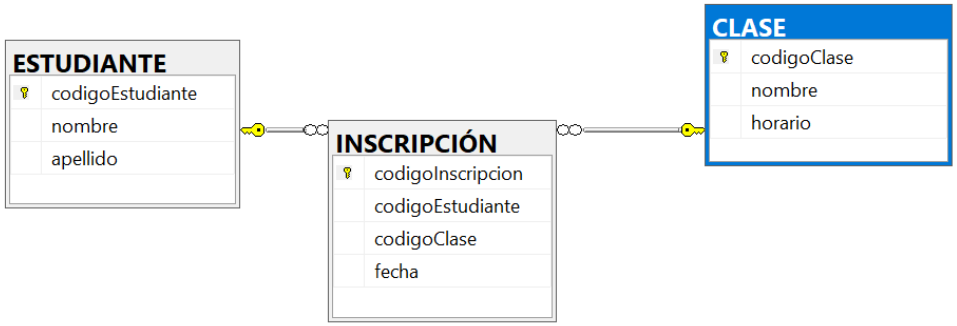
+ fácil comprender




```
SELECT  ProductName
FROM dbo.Products
WHERE CategoryID IN
        (SELECT  CategoryID
        FROM dbo.Categories
        WHERE CategoryName = 'SeaFood')
```

Las subconsultas pueden constar de tres o más tablas

Nombres de los estudiantes inscritos en la clase cuyo horario es los lunes, miércoles, viernes a las 3PM (LMV3)




```
SELECT  codigoEstudiante, nombre, apellido
FROM ESTUDIANTE
WHERE codigoEstudiante IN
        (SELECT codigoEstudiante
        FROM INSCRIPCION
        WHERE codigoClase IN
                (SELECT codigoClase
                FROM CLASE
                WHERE horario = 'LMV3'))
```

Funciona bien siempre que los atributos en la respuesta provenga de una sola tabla