FUNDAMENTOS DE PROGRAMACION CON SQL SERVER 2019

EDUCACIÓN IT

UNIDAD 1

- Arquitectura de un Servidor de Base de Datos
- Diseño de Base de Datos - Beneficios
- Tipos de Sentencias SQL: DML, DDL y DCL
- Tipos de Datos
- Operadores Lógicos
- Criterios de Seleccion - (Sentencias: Between / In)
- Ordenamiento

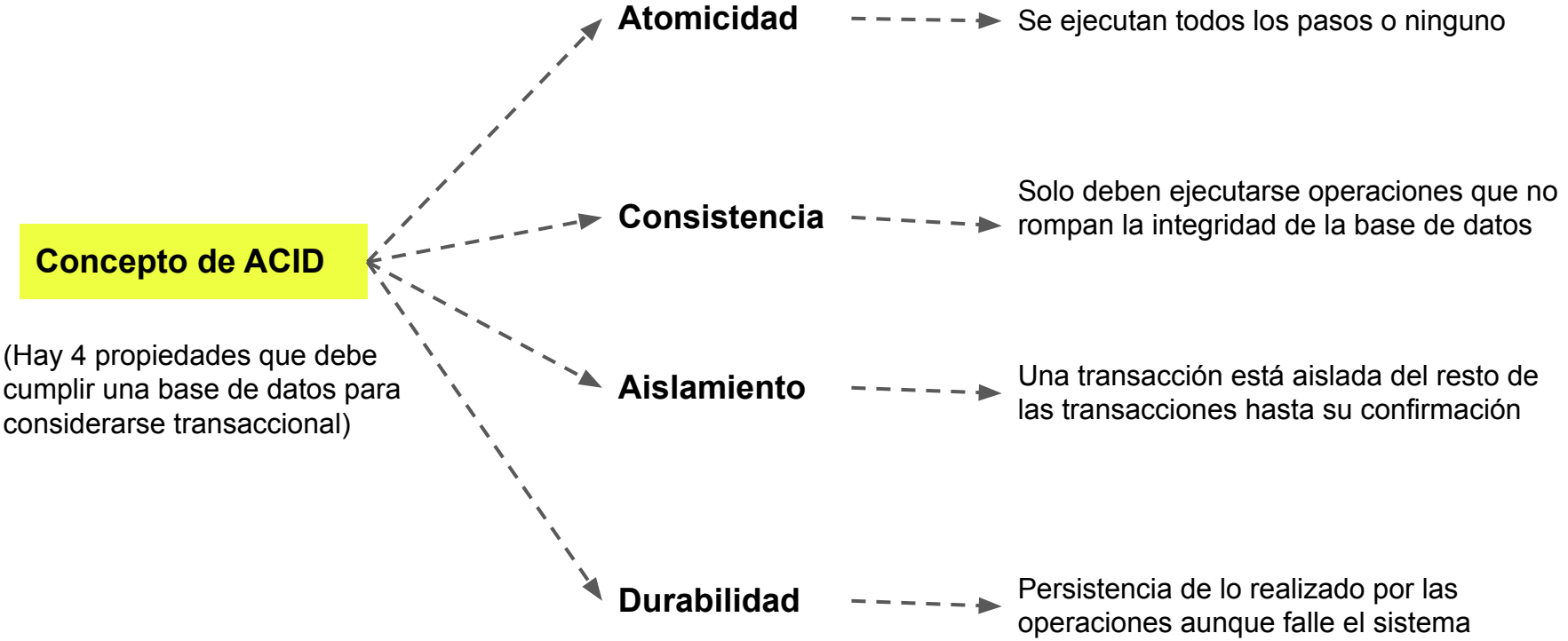
Arquitectura - ¿Qué es una Base de Datos?

Una base de datos es una herramienta para recopilar y organizar información.

Los datos que almacenan deben pertenecer a un mismo contexto y tener relaciones en común.

Ej: Una lista de alumnos no tiene nada que ver con una lista de libros. Pero en el contexto de una “biblioteca escolar”, tiene sentido que exista una relación entre ambas

BD RELACIONALES	BD NO RELACIONALES
Características: -Los registros se organizan en filas y columnas -Para trabajar con los datos utiliza SQL -Cumple con el estándar ACID Ejemplos -SQL Server, Oracle, MariaDB, Postgresql, mysql	Características: -No trabajan con estructuras definidas en tablas (datos no estructurados o semi estructurados) -No utiliza lenguaje SQL para consultas -Son muy utilizadas en aplicaciones para celulares, redes sociales, big data y juegos Tipos: -Clave-Valor, Documentos y Grafos Ejemplos: -Cassandra, mongodb, Neo4j

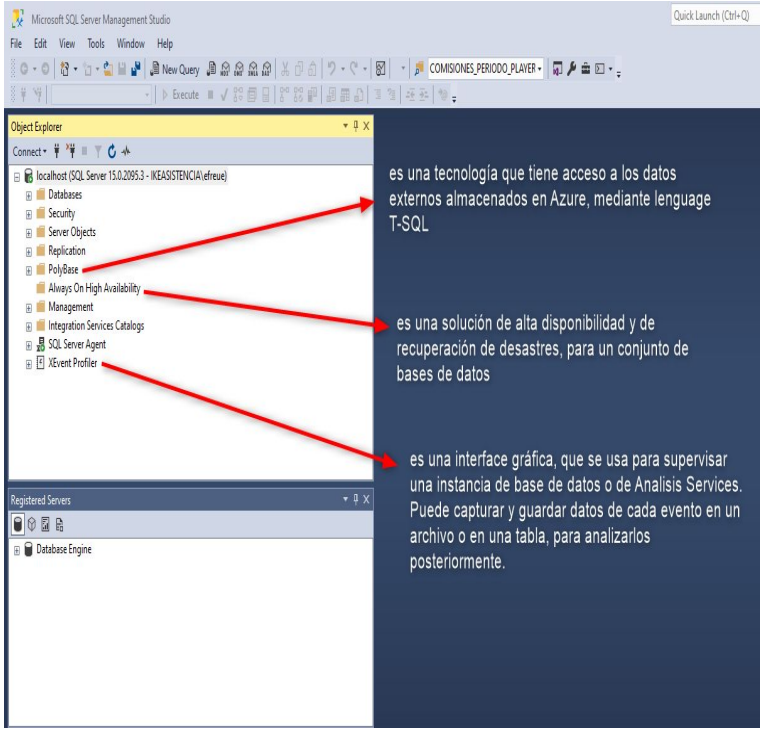

Concepto de ACID
(Hay 4 propiedades que debe cumplir una base de datos para considerarse transaccional)
Atomicidad
Se ejecutan todos los pasos o ninguno
Consistencia
Solo deben ejecutarse operaciones que no rompan la integridad de la base de datos
Aislamiento
Una transacción está aislada del resto de las transacciones hasta su confirmación
Durabilidad
Persistencia de lo realizado por las operaciones aunque falle el sistema

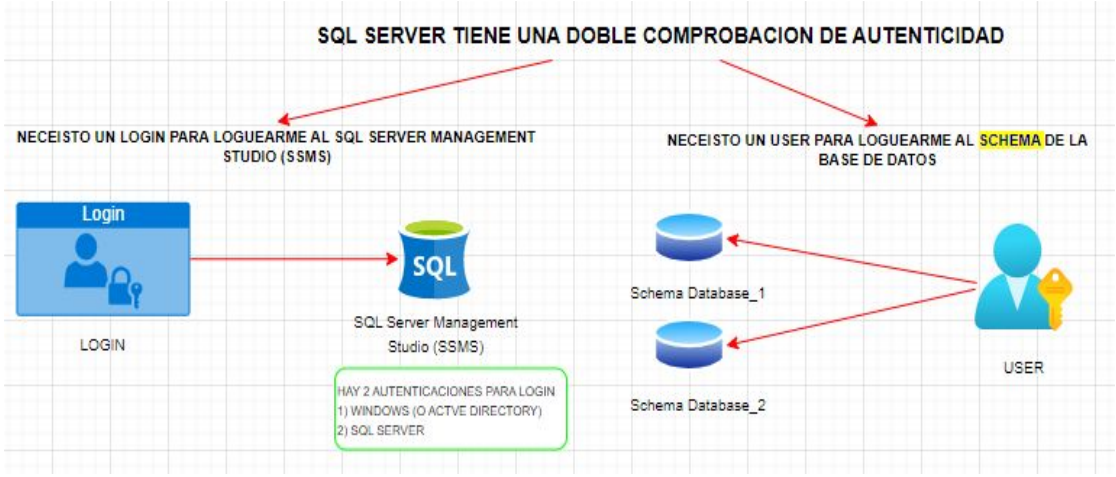
EDICIONES SQL SERVER	
Enterprise	Sirve para aplicaciones de misión crítica y data warehouse a gran escala, tanto locales como en la nube Para grandes empresas
Standard	Proporciona administración básica de base de datos con funcionalidades de BI, para pequeña y mediana empresa (tanto locales como en la nube)
Web	Edición de bajo costo para utilizar en hosts webs (azure, amazon, etc), que proporciona administración y escalabilidad, tanto para pequeña como gran escala
Developer	Incluye todas las funcionalidades de la versión Enterprise, pero el licenciamiento incluído no nos permite usarlo en entornos de producción. Por ello, es perfecto para testear características avanzadas de SQL
Express	Es una versión gratuita, pero viene con bastantes limitaciones. Se utiliza para desarrollos muy pequeños

Sistema Gestor de Base de Datos (SGBD)

Son herramientas que vamos a utilizar para poder administrar nuestras bases de datos

SQL Server Management Studio (SSMS)





SCHEMA

- Es una división lógica (o un subconjunto de objetos), que se crea dentro de la base de datos, para agruparlos de acuerdo a una categoría
- Contiene tablas, vistas, store procedures, funciones, triggers
- Dentro de la misma base de datos pueden crearse distintos schemas (RRHH, Departamento de Informática, etc), pero existe uno por default: “DBO”
- Si no indicamos un esquema, todas las tablas, vistas, store procedures, etc que creemos, quedarán bajo el schema DBO
- Un usuario siempre debe estar asociado a un schema, ya que **el schema es el propietario de las tablas, vistas, etc**. Al usuario se le dan permisos sobre ese schema

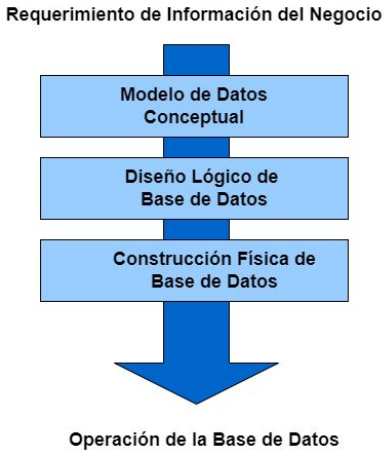
¿Por qué preocuparse por el diseño de la base de datos?

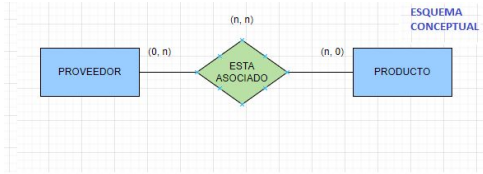
Es fundamental para la **integridad de los datos** (es decir, que la información almacenada sea precisa, completa, consistente y confiable)

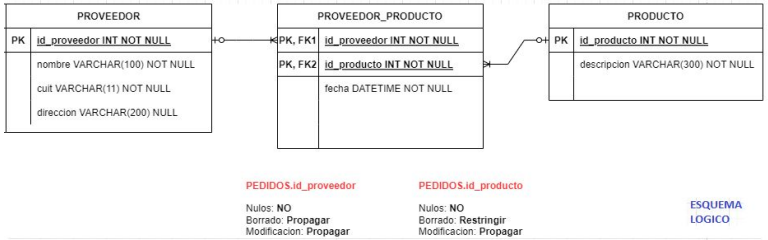
¿Qué pasa si una base de datos está mal diseñada?

- Habrá dificultades para acceder a los datos
- Las búsquedas podrán producir información errónea o duplicada
- Podrán perderse datos o modificarse de manera incorrecta

¿Cuáles son las etapas para realizar el diseño de base de datos?







Diseño de Base de Datos - Etapas

DELETE FROM PROVEEDOR WHERE id_proveedor = 1

PROVEEDOR

id_proveedor	nombre	cuit	direccion
1	Prov 1	30201562228	Calle 46
2	Prov 2	30333882376	NULL
3	Prov 3	30449968532	NULL
4	Prov 4	30998512244	NULL

PROVEEDOR_PRODUCTO

id_proveedor	id_producto	fecha
1	1	2022-01-12 13:05:53.992
1	2	2022-01-12 13:05:53.992
1	3	2022-01-14 16:44:51.990
2	1	2022-01-15 18:23:43.991
2	3	2022-01-15 18:23:44.991
3	2	2022-01-16 11:42:55.997

PRODUCTO

id_producto	descripcion
1	Coca Cola
2	Sprite
3	Arroz Gallo Oro x 1
4	Fideos Matarazo x 500

PROVEEDOR_PRODUCTO.id_proveedor → PROVEEDOR.id_proveedor: Propaga borrado

Diseño de Base de Datos - Etapas

UPDATE PROVEEDOR SET id_proveedor = 10 WHERE id_proveedor = 2

PROVEEDOR

id_proveedor	nombre	cuit	direccion
1	Prov 1	30201562228	Calle 46
2	Prov 2	30333882376	NULL
3	Prov 3	30449968532	NULL
4	Prov 4	30998512244	NULL

PROVEEDOR_PRODUCTO

id_proveedor	id_producto	fecha
1	1	2022-01-12 13:05:53.992
1	2	2022-01-12 13:05:53.992
1	3	2022-01-14 16:44:51.990
2	1	2022-01-15 18:23:43.991
2	3	2022-01-15 18:23:44.991
3	2	2022-01-16 11:42:55.997

PRODUCTO

id_producto	descripcion
1	Coca Cola
2	Sprite
3	Arroz Gallo Oro x 1
4	Fideos Matarazo x 500

PROVEEDOR_PRODUCTO.id_proveedor → PROVEEDOR.id_proveedor: Propaga update

```
DELETE FROM PRODUCTO WHERE id_producto = 2
DELETE FROM PRODUCTO WHERE id_producto = 4
```

PROVEEDOR

id_proveedor	nombre	cuit	direccion
1	Prov 1	30201562228	Calle 46
2	Prov 2	30333882376	NULL
3	Prov 3	30449968532	NULL
4	Prov 4	30998512244	NULL

ERROR: delete on table "PROVEEDOR_PRODUCTO" violates foreign key constraint

PROVEEDOR_PRODUCTO

id_proveedor	id_producto	fecha
1	1	2022-01-12 13:05:53.992
1	2	2022-01-12 13:05:53.992
1	3	2022-01-14 16:44:51.990
2	1	2022-01-15 18:23:43.991
2	3	2022-01-15 18:23:44.991
3	2	2022-01-16 11:42:55.997

PRODUCTO

id_producto	descripcion
1	Coca Cola
2	Sprite
3	Arroz Gallo Oro x 1
4	Fideos Matarazo x 500

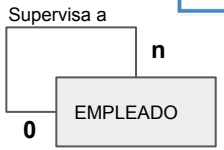
PROVEEDOR_PRODUCTO.id_producto → PRODUCTO.id_producto: Restringir borrado

Diseño de Base de Datos - Etapas

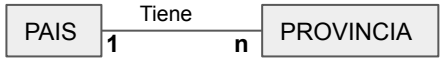
Propiedades de una Relación

Grado

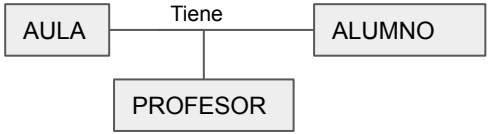
1) Unario: Relaciona una entidad consigo misma



2) Binario: Dos entidades forman parte de la relación



3) Ternario: Tres entidades forman parte de la relación



Cardinalidad

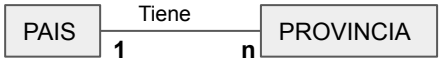
1) Relación 1:1
Un país solo puede tener un presidente (y viceversa)

2) Relación 1:N
Un cliente tiene muchas facturas. Pero cada factura solo tiene un cliente

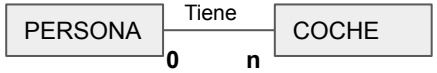
3) Relación N:M
En una película pueden trabajar muchos actores. Y un actor puede trabajar en muchas películas

Condicionalidad

1) Relación obligatoria entre entidades



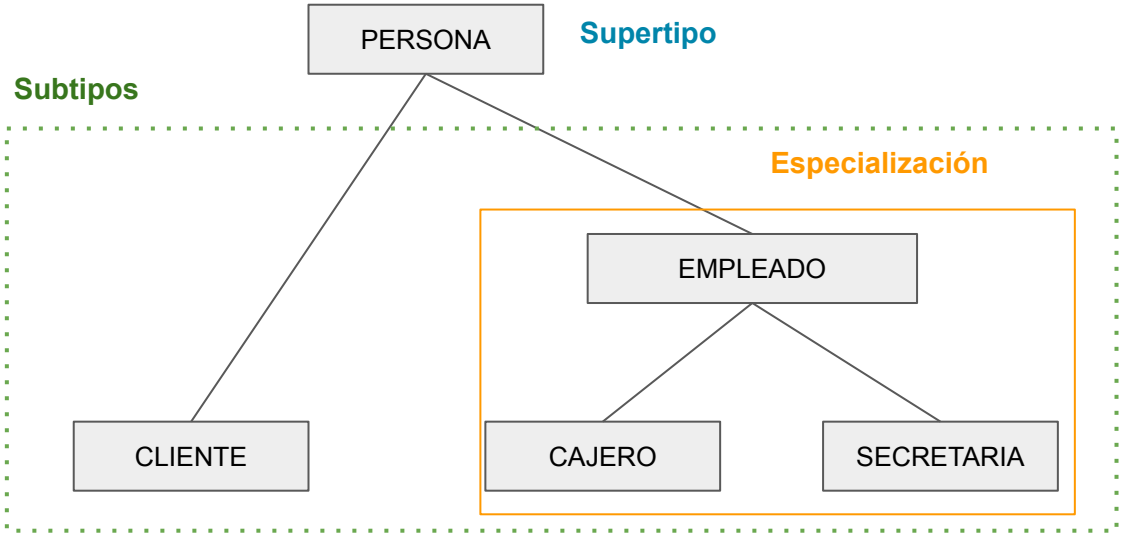
2) Relación opcional entre entidades

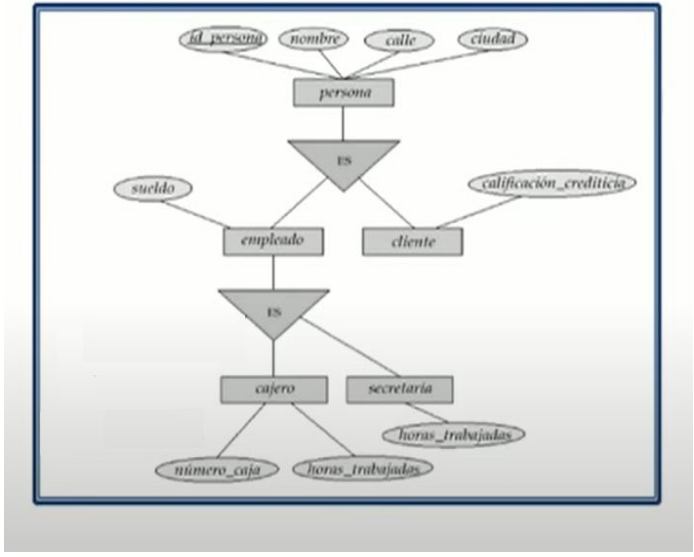


Diseño de Base de Datos - Etapas

Agrupaciones de Entidades

- **Supertipo** — Es un tipo de entidad que se relaciona con uno o más subtipos
- **Subtipo** — Es un tipo de entidad, que hereda todos los atributos del supertipo
- **Especialización** — Es el hecho de agrupar un conjunto de entidades subtipo, que pertenecen a un supertipo





Tipos de Sentencias SQL

SENTENCIAS		DESCRIPCIÓN
DDL	**Definición de datos**	
	CREATE TABLE	Crea una nueva tabla a la base de datos
	DROP TABLE	Elimina una tabla de la base de datos
	ALTER TABLE	Modifica la estructura de una tabla existente
	CREATE VIEW	Crea una nueva vista a la base de datos
	DROP VIEW	Elimina una vista de la base de datos
	CREATE INDEX	Construye un índice para una/s columna/s
	DROP INDEX	Elimina un índice para una/s columna/s

Tipos de Sentencias SQL

SENTENCIAS	DESCRIPCIÓN
DCL **Control de Acceso** GRANT REVOKE COMMIT	 Le da permiso a los usuarios sobre las tablas Elimina los permisos de los usuarios sobre las tablas Confirma una transaccion
DML **Manipulación de datos** SELECT INSERT UPDATE DELETE MERGE	 Recupera datos de la base de datos Añade nuevas filas de datos a una tabla de la base de datos Modifica datos existentes en una tabla de la base de datos Elimina filas de datos a una tabla de la base de datos Inserta y Actualiza datos en la misma consulta

Operadores de Comparacion SQL

Los operadores de comparación comprueban si dos expresiones son iguales o distintas. Se pueden usar en todas las expresiones, excepto en las de los tipos de datos *text, ntext, image.*

En la siguiente tabla se presentan los operadores de comparación Transact-SQL.

Operador	Significado
=	Igual a
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
<>	No es igual a
!=	No es igual a (no es del estándar ISO)
!<	No es menor que (no es del estándar ISO)
!>	No es mayor que (no es del estándar ISO)

Operadores de Logicos SQL

Los operadores lógicos comprueban la veracidad de alguna condición. Al igual que los operadores de comparación, devuelven el tipo de datos Boolean con el valor TRUE, FALSE o UNKNOWN.

Operador	Significado
AND	TRUE si ambas expresiones booleanas son TRUE.
OR	TRUE si cualquiera de las dos expresiones booleanas es TRUE.

En el ejemplo siguiente hace uso del operador 'OR'.

Sintaxis

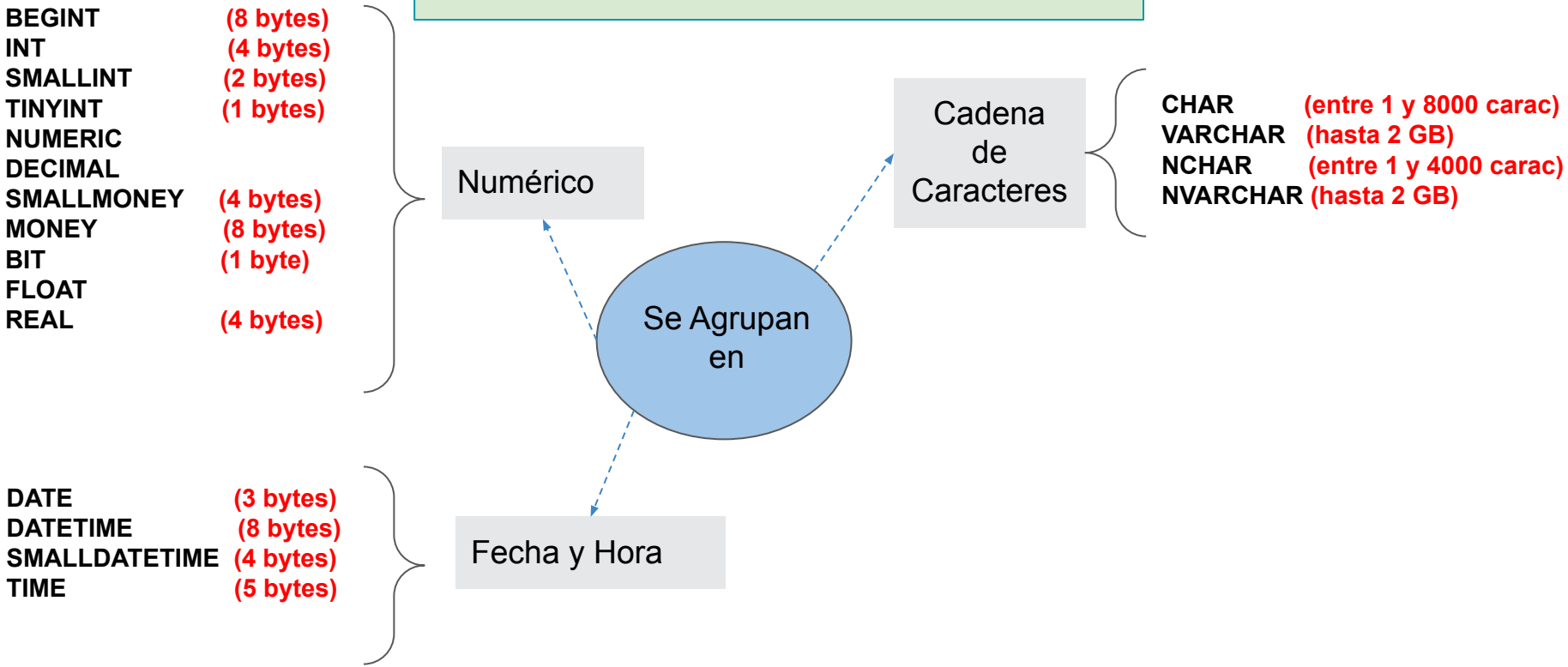
```
SELECT WeightUnitMeasureCode
FROM   Production.Product
WHERE  (WeightUnitMeasureCode = 'G' OR WeightUnitMeasureCode = 'LB')
       AND (ProductID = 513 OR ProductID = 739);
--------------------------------
WeightUnitMeasureCode ProductID
G                     513
LB                    739
```

UNIDAD 2

- Tipos de Datos
- Función DISTINCT
- Operador UNION
- Expresion CASE
- Funciones de Agregado
- Agrupamiento

Tipos de Datos en SQL

SIEMPRE: “Usar el **tipo de dato más pequeño** para ahorrar espacio en la base de datos”



Operadores en SQL

Función DISTINCT

- No Elimina registros físicamente, sino que elimina de la respuesta, las filas duplicadas
- Se usa en la cláusula SELECT

Estructura

```
SELECT DISTINCT
Columna1,
Columna2
FROM TABLA
```

Operador UNION

-Es una instrucción que permite concatenar los resultados de dos o más consultas SELECT, en un único conjunto de resultados
-Las consultas SELECT a concatenar, deben tener las mismas columnas (respetando el orden), y ser del mismo tipo de dato

UNION ALL: incluye duplicados
UNION: se excluyen los duplicados

Estructura

```
SELECT
Columna1,
Columna2
FROM TABLA1
UNION
SELECT
Columna1,
Columna2
FROM TABLA2
```

```
SELECT
Columna1,
Columna2
FROM TABLA1
UNION ALL
SELECT
Columna1,
Columna2
FROM TABLA2
```

Expresión CASE en SQL

Características

-Evalúa una lista de condiciones y devuelve el primer resultado que cumple con la condición

-Se utiliza en las instrucciones: SELECT, UPDATE, DELETE Y SET

-Se utiliza en cláusulas: WITH, IN, WHERE, ORDER BY y HAVING

Estructura

```
CASE
WHEN condicion1 THEN resultado1
WHEN condicion2 THEN resultado2
ELSE resultado3
END
```

Funciones de Agregado en SQL

Características

-Una función de agregado realiza un cálculo sobre un conjunto de valores y devuelve un solo valor

-Las funciones de agregado ignoran los valores NULL

-Se utiliza en cláusulas: SELECT y GROUP BY

AVG	Devuelve el promedio
MAX	Devuelve el valor máximo
MIN	Devuelve el valor mínimo
SUM	Devuelve la sumatoria
COUNT	Devuelve la cantidad. El resultado es de tipo int
COUNT_BIG	Devuelve la cantidad. El resultado es de tipo bigint

Agrupamiento en SQL

GROUP BY ROLLUP

- Genera diferentes combinaciones de agrupamiento de forma jerárquica (totales y subtotales).
- El resultado de **GROUP BY ROLLUP** siempre está contenido en la respuesta de GROUP BY CUBE
- Será reemplazado en futuras versiones

GROUP BY GROUPING SETS

- Nos permite poder personalizar las diferentes combinaciones de agrupamiento que deseemos.
- Ej:**GROUP BY GROUPING SETS((Pais, Color), (Pais), ())** además de generar las agrupaciones por Pais y Color (como una sentencia group by), generar también:
  a) Subtotales por Pais
  b) Una fila con el Total General

GROUP BY CUBE

- Genera todas las posibles combinaciones de agrupamiento
- Ej: **GROUP BY CUBE(Pais, Color)** ademas agrupar por Pais y Color (como una sentencia group by), genera tambien:
  a) Una fila con el Total General
  b) Subtotales por país
  c) Subtotales por Color

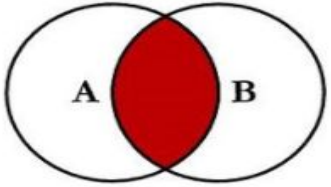
GROUPING GROUPING_ID

- Podemos utilizar las funciones GROUPING y/o GROUPING_ID, para identificar los distintos tipos de agrupaciones (CUBE, ROLLUP o GROUPING SETS)

UNIDAD 3

- Tipos de Joins - Cross Apply
- Tipos de Datos
- Expresión de Tabla Común (CTE)
- Bulk Insert - Select Into

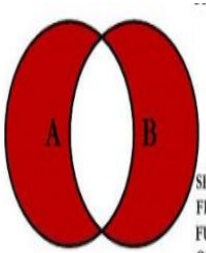
Tipos de Joins



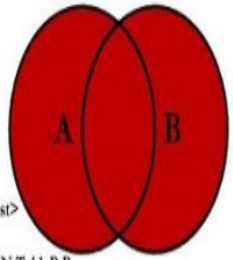
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key

1 - Inner Join

4- Full Join



SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL

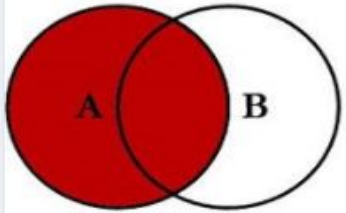


SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key

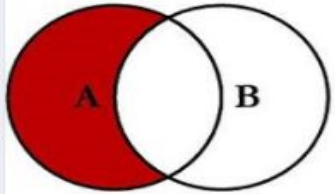
Tipos de Joins

2- Left Join

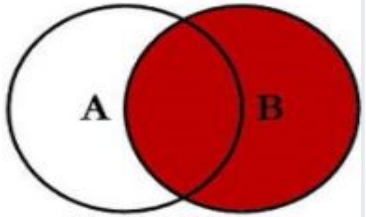
3- Right Join



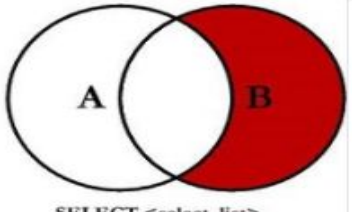
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key



SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL



SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key



SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL

Cross Apply

El operador APPLY permite unir dos tablas (A y B) con la siguiente lógica:

Retorna todos los registros de la tabla A, si al relacionarlos con la tabla B, existe una coincidencia. Esto es similar al INNER JOIN, la diferencia está en lo siguiente:

INNER JOIN	CROSS APPLY
SELECT A.[key], A.[name], X.[key], X.[Name] FROM TABLEA A INNER JOIN (SELECT B.[key], B.[Name] FROM TABLEB B) X ON A.[Key] = X.[Key]	SELECT A.[key], A.[name], X.[key], X.[Name] FROM TABLEA A CROSS APPLY (SELECT B.[key], B.[Name] FROM TABLEB B WHERE B.[key] = A.[Key]) X

En el inner join, uno debe cargar en memoria todo el contenido de la tabla B, y luego buscar la relacion con la tabla A, a partir de la columna “Key”
En el cross apply, uno debe cargar en memoria todo el contenido de la tabla A, si este existe en la tabla B, a partir de la columna “Key”

Tipos de Tablas en SQL Server

Existen 4 tipos de Tablas:

1 - Tablas Temporales Locales

- Tablas que se mantienen durante la sesión en la cual son creadas, y **no pueden ser accedidas desde otra sesión**
- Desaparecen cuando finaliza la sesión que la está utilizando o se elimina explícitamente la tabla
- Se crean en la base tempdb
- Acepta instrucciones DDL y DML

2 - Tablas Temporales Globales

- Tablas muy similares a las tablas temporales locales, pero se diferencian en que son **accesibles desde todas las sesiones**
- Desaparecen una vez que la sesión que creo la tabla, finaliza. O si alguna sesión, elimina la tabla global
- Se crean en la base tempdb
- Acepta instrucciones DDL y DML

3 - Variable de tipo Tabla

- Se trata de una variable que se comporta como una tabla
- No se crea físicamente en la base de datos. Estará disponible en memoria mientras se ejecute el proceso
- No permite hacer DROP TABLE ni TRUNCATE

4 - Tablas Físicas

- Tablas similares a las tablas temporales globales, pero se diferencian porque son creadas como objetos sobre el motor de base de datos (están siempre disponibles)
- Solo desaparecen cuando hacemos un drop table
- Acepta instrucciones DDL y DML

Tabla CTE y Bulk Insert en SQL Server

Expresión de tabla común (CTE)

- Es un conjunto de resultados con nombre temporal, al que puede hacerse referencia dentro de una instrucción; SELECT, INSERT, UPDATE o DELETE.
- El CTE también se puede usar en una vista
- Podemos usar solo una cláusula (SELECT, INSERT, UPDATE O DELETE) con un CTE

Estructura

```
;WITH NombreCTE [(campos devueltos)]
AS
(
        SUBCONSULTA
)
SELECT * FROM NombreCTE
```

BULK INSERT (SELECT INTO)

- La instrucción SELECT INTO crea una nueva tabla, y la llena con el conjunto de resultados de la instrucción SELECT
- SELECT INTO se puede emplear para combinar datos de varias tablas o vistas en una nueva tabla
- Los índices y triggers definidos en la tabla origen no se transfieren a la tabla nueva, cuando usamos la instrucción SELECT INTO

Estructura

```
SELECT C.nombre, C.apellido
INTO dbo.TMP_PERSONAS_MAYORES
FROM dbo.Personas C
WHERE C.edad > 60
```

UNIDAD 4

- Subconsultas SQL
- Operadores IN, EXISTS, ANY, ALL en Subconsultas
- Distintas formas de Insertar, Modificar y Eliminar Registros
- Diferencia entre DELETE y TRUNCATE

Subconsultas en SQL Server

Definición:

Una subconsulta es una consulta anidada en una instrucción SELECT, INSERT, UPDATE o DELETE o bien, en otra subconsulta

USO DE SUBCONSULTAS

SELECT

```
USE AdventureWorks2019
go
--SE LISTA EL PRECIO DE CADA UNO DE LOS PRODUCTOS Y EL PRECIO PROMEDIO GLOBAL
SELECT
Id_Prod = ProductID,
Precio_Prod = ListPrice,
Precio_Prom_Global = (SELECT AVG(ListPrice) FROM Production.Product)
FROM Production.Product;
```

SUBCONSULTAS CORRELACIONADAS

(Son aquellas que se evalúan por cada registro de la tabla

```
--MUESTRA EL PRODUCTO MAS BARATO DE CADA SUBCATEGORIA
SELECT
p1.ProductSubcategoryID,
p1.ProductID,
p1.ListPrice
FROM Production.Product p1
WHERE ListPrice =   (
                        SELECT ListPrice = MIN(ListPrice)
                        FROM Production.Product p2
                        WHERE p2.ProductSubcategoryID = p1.ProductSubcategoryID
                    )
ORDER BY p1.ProductSubcategoryID;
```

FROM

```
--SE LISTA EL PRECIO DE CADA UNO DE LOS PRODUCTOS Y EL PRECIO PROMEDIO DE VENTA PARA CADA UNO DE LOS MISMOS
SELECT
ProductID = pp.ProductID,
ListPrice = pp.ListPrice,
PROMEDIO_VENTA = x.promedio
FROM Production.Product pp
INNER JOIN
(
    SELECT
    ProductID,
    promedio = AVG(LineTotal)
    FROM Sales.SalesOrderDetail
    GROUP BY ProductID
) x
ON pp.ProductID=x.ProductID;
```

WHERE

```
--SE LISTAN LOS PRECIOS QUE SON INFERIORES AL PRECIO PROMEDIO DE VENTA DE TODOS LOS PRODUCTOS
SELECT
ProductID,
ListPrice
FROM Production.Product
WHERE ListPrice < ( SELECT AVG(LineTotal) FROM Sales.SalesOrderDetail);
```

Operadores en SQL Server

OPERADORES

IN

- Determina si un valor especificado coincide o no con algún valor de una subconsulta o una lista

```
--LISTA LAS PERSONAS QUE SON VENDEDORES
SELECT FirstName, MiddleName, LastName
FROM Person.Person
WHERE BusinessEntityID IN (
                            SELECT BusinessEntityID
                            FROM Sales.SalesPerson
                          );
```

EXISTS

- La cláusula WHERE EXISTS sirve para validar si una subconsulta posee registros o no
- Está cláusula se usa junto con una subconsulta
- La subconsulta se evalúa primero, si no devuelve ningún resultado, la cláusula WHERE EXISTS no se ejecutará, y no se mostrarán los resultados de la consulta SQL principal.

```
--LISTA LAS PERSONA QUE NO SON VENDEDORES
SELECT FirstName, LastName
FROM Person.Person P
WHERE NOT EXISTS
(
    SELECT BusinessEntityID
    FROM Sales.SalesPerson S
    WHERE P.BusinessEntityID= S.BusinessEntityID
);
```

ANY

- El operador ANY devuelve verdadero **<u>si alguno</u> de los valores de subconsulta cumple la condición**
- Se usa con la cláusula WHERE o HAVING

```
--LISTA LOS PRODUCTOS Y PRECIOS QUE SUPERAN EL PRECIO PROMEDIO TOTAL
SELECT Name, ListPrice
FROM Production.Product
WHERE ListPrice > ANY   (
                            SELECT AVG(ListPrice) Promedio
                            FROM Production.Product
                        );
```

ALL

- El operador ALL retorna los valores **<u>si todos</u> los registros de la subconsulta cumplen la condición**
- Se usa con la cláusula WHERE o HAVING

```
--LISTA LOS PRODUCTOS CUYO PRECIO SEA DISTINTO AL PROMEDIO TOTAL
SELECT Name, ListPrice
FROM Production.Product
WHERE ListPrice <> ALL  (
                            SELECT AVG(ListPrice) Promedio
                            FROM Production.Product
                        );
```

Operaciones DML Masivas en SQL Server

OPERACIONES DML MASIVAS

INSERT

- Insertar datos en una tabla con el resultado de una consulta

```
INSERT INTO TABLA_NEW (col1, col2)
SELECT columna1, columna2
FROM TABLA
```

DELETE

- Borra datos en una tabla a partir de un joineo con otra tabla
- Se puede usar DELETE TOP para limitar el número de filas que se van a eliminar en forma aleatoria

```
DELETE S
FROM dbo.Sectores S
INNER JOIN dbo.SectoresNuevo SN ON S.Sector = SN.Sector
```

```
DELETE TOP (2) S
FROM dbo.Sectores S
INNER JOIN dbo.SectoresNuevo SN ON S.Sector = SN.Sector
```

UPDATE

- Actualizar datos en una tabla con valores de otra tabla
- Se puede usar UPDATE TOP para limitar el número de filas que se van actualizar en forma aleatoria

```
UPDATE S
SET
Sector = SN.SectorNuevo
FROM Sectores S
INNER JOIN SectoresNuevo SN ON S.Sector = SN.Sector
```

```
--SE ACTUALIZA UN AUMENTO DEL 25% PARA LAS HORAS DE VACACIONES, EN 10 REGISTROS ALEATORIAS
--DE LA TABLA Employee
UPDATE TOP (10) HumanResources.Employee
SET VacationHours = VacationHours * 1.25;
```

TRUNCATE

- Truncate elimina los registros de la tabla (al igual que DELETE), pero reinicia el valor de la columna IDENTITY
- Truncate elimina las particiones
- Truncate es mas rápido que la instrucción DELETE

Restricciones:

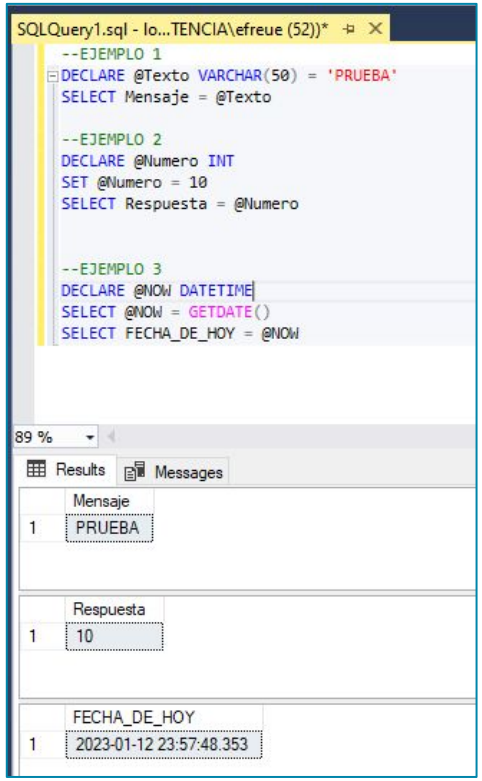
- No pueden truncarse tablas que tengan foreign key
- No puede truncarse tablas que participan en una vista
- No pueden truncarse tablas con triggers
- No puede ejecutarse un Truncate dentro de una transacción

UNIDAD 5

- Variables
- Batches vs Scripts
- Procedimientos Almacenados
- Estructuras de control
- Implementar Bloques Interactivos

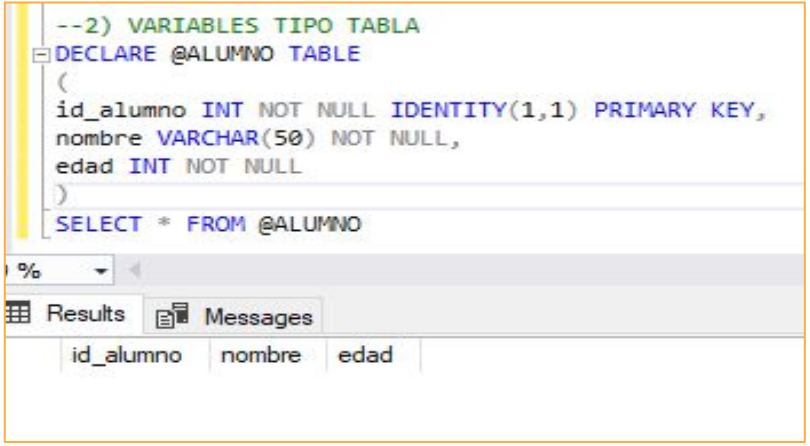
VARIABLES

VARIABLES ESCALARES
(Declarar y asginar un valor)



VARIABLE TIPO TABLA
(Declarar y asginar un valor)

- Se trata de una variable que se comporta como una tabla
- No se crea físicamente en la base de datos. Estará disponible en memoria mientras se ejecute el proceso
- No permite hacer DROP TABLE ni TRUNCATE



VARIABLES DE SISTEMA
(Solo pueden consultarser)

@@ROWCOUNT
(Retorna la cantidad de filas leídas)

@@IDENTITY
(Devuelve el último ID insertado en una columna identity de una tabla)

@@SERVERNAME
(Retorna el nombre del servidor local

@@SPID
(Se puede usar para identificar el número de proceso del servidor en la respuesta de **sp_who2 active**

CONTROL DE FLUJO

Encierra y ejecuta un conjunto de instrucciones Transact-SQL, entre los bloques BEGIN y END

IF … ELSE

```
DECLARE @NOMBRE VARCHAR(50) = 'Juan'
DECLARE @EDAD INT = 42
DECLARE @MENSAJE VARCHAR(200) = 'Datos Incorrectos. Ingresar un nombre y una edad mayor a 0'
IF (
        ISNULL(@NOMBRE,'') <> '' AND ISNULL(@EDAD,0) > 0
    )
BEGIN
    SET @MENSAJE = 'Su nombre es ' + @NOMBRE + ' y tiene ' + CAST(@EDAD AS VARCHAR(20)) + ' años'
    SELECT @MENSAJE
END
ELSE
BEGIN
    SELECT @MENSAJE
END
```

89 %

Results | Messages

	(No column name)
1	Su nombre es Juan y tiene 42 años

WHILE

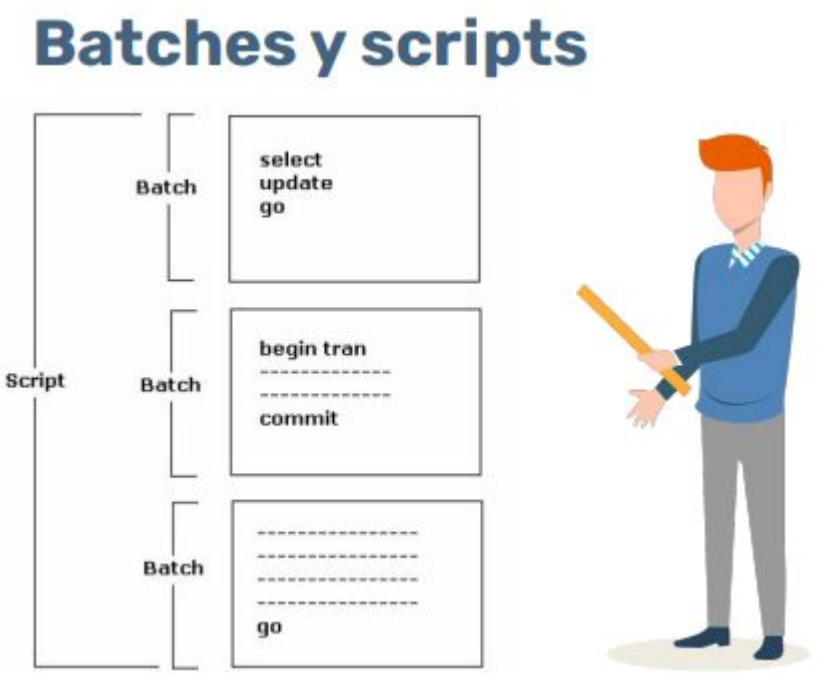
```
DECLARE @NUM INT;
SET @NUM=0 ;
WHILE (@NUM <= 10)
BEGIN
    PRINT @NUM
    IF @NUM=7
    BEGIN
        SET @NUM=@NUM+1;
        PRINT 'ESPERO 5 SEGUNDOS'
        WAITFOR DELAY '00:00:05'
        CONTINUE;
    END
    -- Dos formas de salir forzando el while
    IF @NUM=8 GOTO mensaje;
    IF @NUM=9 BREAK;
    --
    SET @NUM=@NUM+1;

END
mensaje:
PRINT 'SALI DEL WHILE POR EL GOTO'
```

89 %

Messages

```
0
1
2
3
4
5
6
7
ESPERO 5 SEGUNDOS
8
SALI DEL WHILE POR EL GOTO
```

SCRIPT VS BATCHES



Batch (o Lote)

- Es un lote de una o más sentencias Transact-SQL, que se ejecutan una única vez.
- La cláusula GO indica el fin de un Batch

SCRIPT

- Los Script son series de Batches ejecutados uno después de otro
- Indican el orden en que se ejecuta cada Batch

Ejemplo de mal uso de la Cláusula GO

Sintaxis

```
DECLARE @Variable int;
GO -- Su uso aquí genera un error
SET @Variable=4;
```

¿Que es un Store Procedure?

-Es un procedimiento almacenado físicamente en una base de datos, que SQL Server compila, en un único plan de ejecución
-Pueden ser ejecutados en cualquier momento

Tipos de SP

SISTEMA

Vienen con SQL Server
Empiezan con el prefijo “sp_”

Ejemplos:

sp_tables
sp_columns
sp_databases
sp_execute

EXTENDIDOS

Vienen con SQL Server
Empiezan con el prefijo “xp_”

Ejemplo:

xp_cmdshell
(Se utiliza para ejecutar sentencias de comando de windows. Ej: copiar archivos, crear carpetas, etc)

DEFINIDOS POR EL USUARIO

SIN PARAMETROS

```
CREATE PROCEDURE dbo.PA_Empleados
AS
    SELECT LastName, FirstName, Department
    FROM   HumanResources.vEmployeeDepartmentHistory;
GO

-- LLAMADO O INVOCACIÓN
EXEC dbo.PA_Empleados;
GO
```

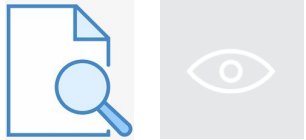
PARAMETROS DE INPUT

```
IF OBJECT_ID ('dbo.PA_Empleados', 'P') IS NOT NULL
    DROP PROCEDURE dbo.PA_Empleados;
GO

CREATE PROCEDURE dbo.PA_Empleados @LastName VARCHAR(50),
@FirstName VARCHAR(50)
AS
    SELECT LastName, FirstName, Department
    FROM HumanResources.vEmployeeDepartmentHistory
    WHERE FirstName = @FirstName AND LastName = @LastName;
GO
-- Invocación al SP pasando los parámetros teniendo en cuenta el orden
EXEC dbo.PA_Empleados 'Ackerman', 'Pilar';

-- Invocación al SP pasando los parámetros sin tener en cuenta el orden
EXEC dbo.PA_Empleados @FirstName = 'Pilar', @LastName = 'Ackerman';
```

PARAMETROS DE OUTPUT

```
IF OBJECT_ID ('dbo.PA_Empleados', 'P') IS NOT NULL
    DROP PROCEDURE dbo.PA_Empleados;
GO

CREATE PROCEDURE dbo.PA_Empleados @LastName VARCHAR(50)='Arifin',
@FirstName VARCHAR(50)='Zainal', @Department VARCHAR(50) OUTPUT
AS
   SELECT @Department=[Department]
   FROM HumanResources.vEmployeeDepartmentHistory
   WHERE FirstName = @FirstName AND LastName = @LastName;
GO

-- Declaro la variable donde se guardará el departamento del
empleado
DECLARE @Departamento VARCHAR(50);

-- Obtengo el departamento de Akerman
EXEC dbo.PA_Empleados 'Ackerman', 'Pilar', @Departamento OUTPUT;

-- Verifica el departamento obtenido
SELECT @Departamento AS [Valor obtenido desde un OUTPUT];
GO
```

Sentencia EXECUTE



Ejecuta uno de los siguientes módulos: procedimiento almacenado del sistema, procedimiento almacenado definido por el usuario, función con valores escalares definida por el usuario o procedimiento almacenado extendido.

Ejecuta una cadena de comandos o una cadena de caracteres dentro de un proceso por lotes de Transact-SQL.

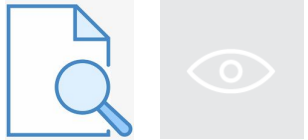
El siguiente ejemplo ejecuta una consulta sql simple

Sintaxis

```
EXEC ('SELECT * FROM Production.Product');
```

UNIDAD 6

- Tipos de Funciones
- Triggers
- Conversión de tipos de datos



¿Que es una Funcion?

-Es un conjunto de instrucciones SQL que realizan una tarea específica.

Consideraciones a la hora de utilizar funciones en sql server

-Las funciones no pueden ejecutar store procedures
-No pueden insertar datos en otra tabla con la instrucción INTO. EJ:
SELECT * INTO Tabla_2 FROM Tabla_1
-No se puede formatear el resultado en XML
-En las funciones tipo tablas, el resultado no se puede regresar con la instrucción ORDER BY en el return
-Solo las tablas temporales como variables están permitidas dentro de las funciones

Tipos de Funciones

Integradas
-Son funciones que ya existen de forma predeterminada en sql server

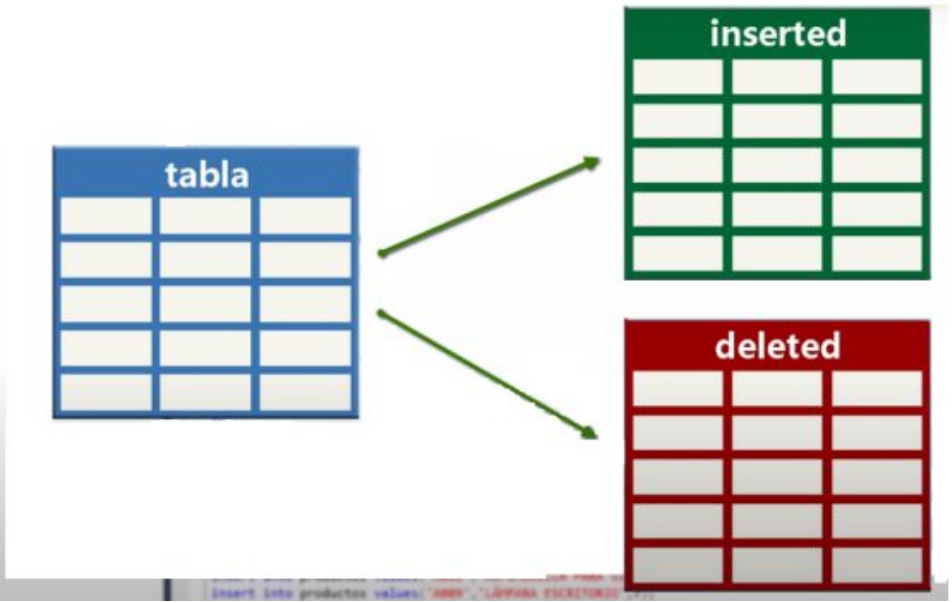
AVG(): Calcula el promedio.
SUM(): Realiza la suma.
COUNT(): Contabiliza el total de registros.
DATETIME(): Regresa la fecha y hora del sistema en SQL Server.
CONCAT(): Concatena cadenas de texto.
LOWER(): Regresa el texto en minúsculas.
UPPER(): Regresa el texto en mayúsculas.
MAX(): Regresa el valor máximo.

Escalares
-Recibe parámetros de entrada y se debe indicar el tipo de valor que regresa la función

Con Valor de Tabla (o Multi sentencia)
-Recibe parámetros de entrada y retorna una tabla

¿Que es un Trigger?

Es una especie de procedimiento almacenado el cual se dispara cuando ejecutamos acciones de INSERT, UPDATE o DELETE sobre una tabla o vista. O también sobre instrucciones CREATE, ALTER y DROP.
A diferencia de los procedimientos almacenados, los triggers no se pueden ejecutar de forma directa, sino que se disparan automáticamente en las tablas donde fueron definidos, cuando se cumple cierta acción



Cada vez que realizamos acciones de INSERT, UPDATE y DELETE sobre una tabla, por mas que no las veamos, se crean las siguientes tablas temporales con la misma estructura de la tabla, en la que estamos realizando las acciones:

1) **INSERTED:** se crean cuando insertamos registros en una tabla, una vez que los registros terminan de ser insertados, se eliminan automáticamente de INSERTED
2) **DELETED:** Se crean cuando eliminamos registros en una tabla. una vez que los registros son eliminados, estos se eliminan automáticamente de DELETED
3) Se crean dos tablas temporales (DELETED e INSERTED): cuando actualizamos registros en una tabla. La actualización se realiza así:
   a) Se guarda en la tabla DELETED el registro a modificar
   b) Se guarda en la tabla INSERTED el registro modificado

Conversion de Tipos de Datos

CAST VS CONVERT

CAST es una función estándar ANSI-SQL
CONVERT es una función específica de SQL Server

Ambas funciones convierten una expresión de un tipo de dato a otro

CONVERSIÓN IMPLÍCITA

- No son visibles para el usuario
- SQL convierte automáticamente los datos de un tipo a otro
- Ejemplo:Cuando a una variable de tipo varchar, le pasamos un valor numérico.

```
declare @texto varchar(50) = 10
select @texto
```

CONVERSIÓN EXPLÍCITA

- Si no realizamos la conversión, sql server fallara
- Nosotros debemos realizar la conversión de un valor a distintos tipos de datos
- Ejemplo:

```
SELECT CONVERT(VARCHAR, GETDATE(), 101),
CONVERT(VARCHAR, GETDATE(), 103), CONVERT(VARCHAR, GETDATE(), 105), CONVERT(VARCHAR, GETDATE(),
111), CONVERT(VARCHAR, GETDATE(), 112)

---------   ---------   ---------   ---------   ----------
12/05/2018  05/12/2018  05-12-2018  2018/12/05  20181205
```

UNIDAD 7

- Transacciones
- Captura de Errores
- Cursores
- Pivot

¿Que es una Transacción?

Es un conjunto de operaciones transact SQL que se ejecutan como un único bloque. (si no se ejecuta completo, fallará)

Si una transacción se ejecuta

Todas las operaciones realizadas durante la transacción, se confirman y quedan permanente en la base de datos

Todas las operaciones realizadas durante la transacción, deben cancelarse y revertirse a un estado anterior, a la ejecución de la transacción.

Transacciones

TRANSACCIONES IMPLÍCITAS

SET IMPLICIT_TRANSACTIONS ON
- Es el comportamiento predeterminado de SQL Server
- La instrucción BEGIN TRANSACTION siempre se ejecuta en forma oculta, con cada instrucción SQL (por eso, después de cada instrucción, hay que poner commit o rollback, para finalizar la transacción)
- Está forma de trabajar puede ser problemático en un ambiente productivo

TRANSACCIONES EXPLÍCITAS

SET IMPLICIT_TRANSACTIONS OFF
- En este caso, nosotros le indicamos al SQL, cuando inicia (BEGIN TRANSACTION) y cuando finaliza (COMMIT TRANSACTION o ROLLBACK) una transacción.
- Cada transacción puede contener muchas instrucciones sql
- Una vez creada la transacción, está permanecerá abierta y no finalizará hasta encontrar un ROLLBACK o COMMIT
- Es el modo de trabajo recomendado

Transacciones - SAVE TRANSACTION

¿SIRVEN PARA?

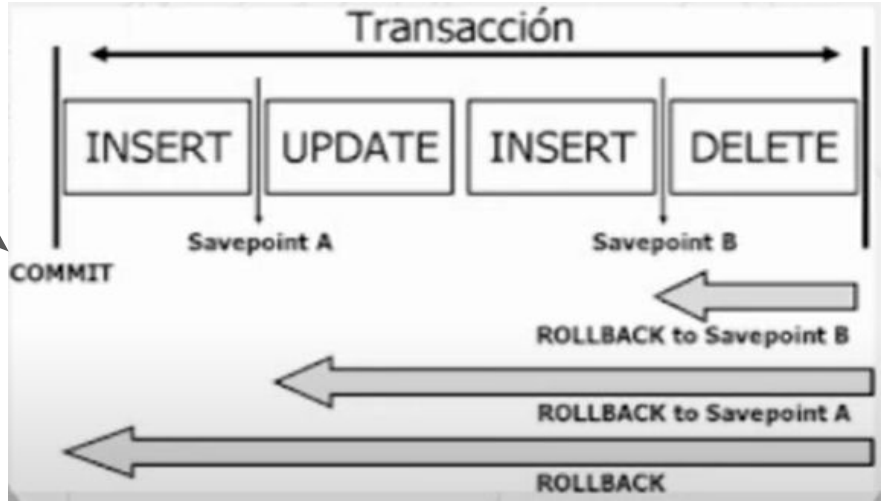
Manejar las transacciones por pasos, pudiendo hacer **rollbacks** hasta un punto marcado por el **savepoint** y no por toda la transacción

¿COMANDOS MÁS UTILIZADOS?

Se declara con la sentencia:
SAVEPOINT + Nombre

Revertir un punto de guardado:
ROLLBACK TO SAVEPOINT + Nombre

Sea descartado de la ejecución
RELEASE SAVEPOINT + Nombre



¿Que es un bloqueo?

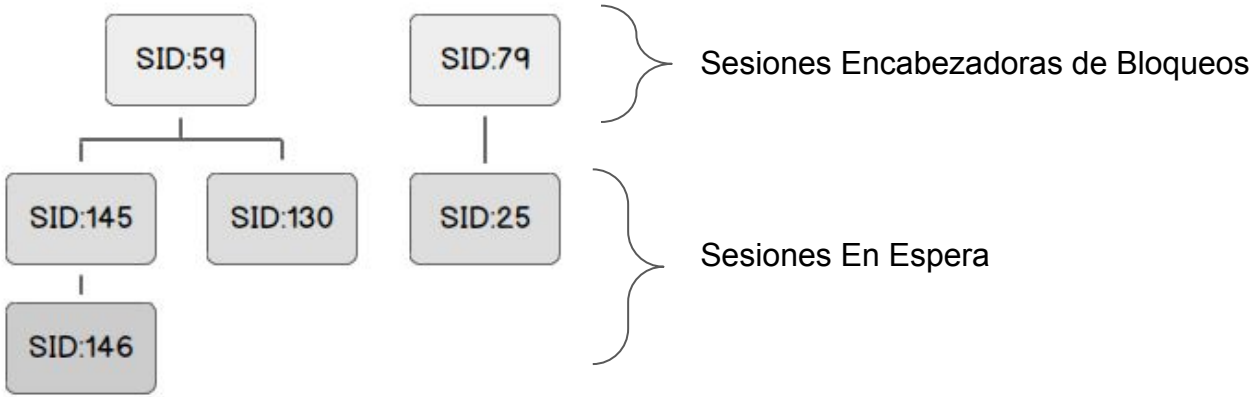
Un bloqueo es el proceso por el que la base de datos restringe el acceso a los datos en un entorno de varios usuarios

Básicamente hay dos formas de realizar bloqueos:

1) **Bloqueo Compartido (Shared):** Son bloqueos realizados para la lectura de datos
2) **Bloqueo Exclusivo (Exclusive):** Son bloqueos realizados para la escritura de datos

Las sesiones respetan una Jerarquía cuando ocurren bloqueos

Cada recuadro representa una sesión de SQL. Si ocurren el bloqueos de un objeto por una sesión, el resto de las sesiones debera esperar la liberación del bloqueo, para poder acceder nuevamente al objeto



Isolation Levels - Concepto:

El nivel de aislamiento especifica cómo se gestionan las transacciones (y los bloqueos) en una base de datos.

Lecturas Sucias: Esto ocurre cuando se leen datos no confirmados

Lecturas Fantasmas: Esto ocurre cuando los datos con los que está trabajando una transacción, han sido modificados por otra transacción, desde que los leyó por primera vez

TIPOS DE ISOLATION LEVELS

Read Uncommitted:
- Nivel de aislamiento mas bajo
- Permite lecturas sucias y lecturas fantasmas

Read Committed:
- Nivel de aislamiento predeterminado
- Permite leer solamente datos que han sido confirmados
- bloqueo del resto de las transacciones hasta que la transacción principal termine
- No permite lecturas sucias pero si lecturas fantasmas

Repeatable Read:
- Es similar al READ COMMITED, pero con la garantía adicional que no permitirá lecturas diferentes (cuando haya modificaciones desde otras transacciones)

Serializable Read:
- Es el nivel de aislamiento mas estricto, ya que no permite lecturas sucias ni lecturas fantasmas
- Tiene un impacto en el rendimiento de la base de datos

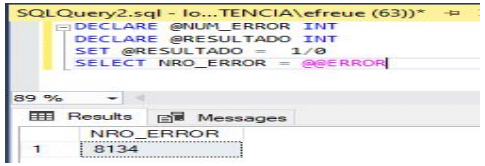
Snapshot:
- Similar a SERIALIZABLE READ
- Utiliza el control de versiones de filas, para evitar lecturas sucias y fantasmas en la transacción existente.

Tratamiento de Errores

Tratamiento de Errores

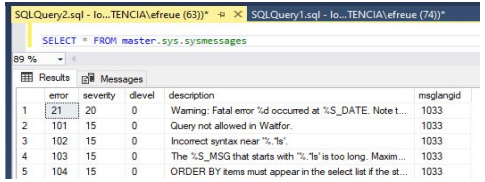
@@ERROR

Devuelve el número de error de la última instrucción SQL ejecutada



SYS.MESSAGES

Es una vista, que contiene una fila, para cada uno de los códigos de error y mensajes, definidos por el usuario y por el sistema



TRY.. CATCH

-Implementa un mecanismo de control de errores para Transact SQL. Si se produce un error en el bloque TRY, el control se transfiere a otro grupo de instrucciones, que está incluido en el bloque CATCH
-En el ámbito del bloque CATCH, se puede utilizar algunas de las siguientes funciones de sistema, para obtener información acerca del error:
ERROR_NUMBER(), ERROR_SEVERITY(), ERROR_STATE(), ERROR_PROCEDURE(), ERROR_LINE() Y ERROR_MESSAGE()
(Las cuales son utilizadas por: RAISERROR y THROW)

RAISERROR VS THROW

Características Comunes

-Producen una excepción y transfiere la ejecución a un bloque CATCH
-Puede hacer referencia a un mensaje definido en la vista sys.messages o puede generar un mensaje dinámicamente

Diferencias

En el bloque CATCH: “RAISERROR” continúa leyendo las sentencias. Y “THROW” no

RAISERROR

```
--EL RAISERROR CONTINUA LEYENDO DESPUES DE ENCONTRAR UN ERROR
BEGIN
    PRINT 'ANTES RAISERROR';
    RAISERROR('RAISERROR TEST', 16, 1);
    PRINT 'DESPUES RAISERROR';
END
```

```
89 %
Messages
ANTES RAISERROR
Msg 50000, Level 16, State 1, Line 30
RAISERROR TEST
DESPUES RAISERROR
```

THROW

```
--EL THROW NO CONTINUA LEYENDO DESPUES DE ENCONTRAR UN ERROR
BEGIN
    PRINT 'ANTES THROW';
    THROW 50000, 'THROW TEST', 1;
    PRINT 'DESPUES THROW';
END
```

```
89 %
Messages
ANTES THROW
Msg 50000, Level 16, State 1, Line 38
THROW TEST
```

Cursores

Tratamiento de Errores

DEFINICIÓN

- Un Cursor es una estructura que se carga en la memoria ram, con el resultado de una consulta sql
- Permite recorrer el resultado de esa consulta, fila por fila
- Con un cursor podemos leer la información y eventualmente, modificar dicho conjunto de datos

ESTRUCTURA SIMPLE

1) DECLARAMOS EL CURSOR Y LE INDICAMOS SOBRE QUE CONSULTA VA A TRABAJAR

```
DECLARE Nombre_Cursor CURSOR [LOCAL | GLOBAL]  [FORWARD_ONLY | SCROLL]
FOR select * from dbo.tmp_empleados
[FOR UPDATE]
```

2) ABRIMOS EL CURSOR EN MEMORIA

```
OPEN Nombre_Cursor
```

3) RECORREMOS EL CURSOR

```
FETCH [NEXT | PRIOR | FIRST | LAST] FROM Nombre_Cursor
```

4) CERRAMOS EL CURSOR (pero sigue estando en memoria)

```
OPEN Nombre_Cursor
```

5) SACO EL CURSOR DE LA MEMORIA

```
DEALLOCATE Nombre_Cursor
```

Cuando declaramos el cursor, podemos agregar las siguientes instrucciones:
1) LOCAL: El cursor solo funcionará dentro del procedimiento almacenado o trigger donde se haya creado
2) GLOBAL: El cursor podrá ser accedido y consultado por otros procedimientos o triggers
3) FOWARD_ONLY: Es el valor por default que toman los cursores. Significa que solo podemos leer los registros hacia adelante. (Si está opción es la elegida, cuando recorramos el cursor, solo podremos utilizar FETCH NEXT)
4) SCROLL: Si seleccionamos está opción, el cursor podrá ser recorrido en distintos sentidos (por lo tanto, cuando recorramos el cursor, podremos usar cualquiera de las opciones: NEXT, PRIOR, FIRST o LAST)
5) FOR UPDATE: Está opción nos permitirá realizar update sobre la tabla que está recorriendo el cursor

Cuando recorremos el cursor, podemos hacerlo de diferente forma (dependiendo si cuando declaramos el store, definimos la opción: “FOWARD_ONLY” o “SCROLL”)
a) NEXT: recorre el cursor fila por fila, pero siempre hacia adelante
b) PRIOR: recorre el cursor fila por fila, pero siempre hacia atras (solo funciona con SCROLL)
c) FIRST: Se posiciona en la primer fila del cursor (solo funciona con SCROLL)
d) LAST: Se posiciona en la última fila del cursor (solo funciona con SCROLL)

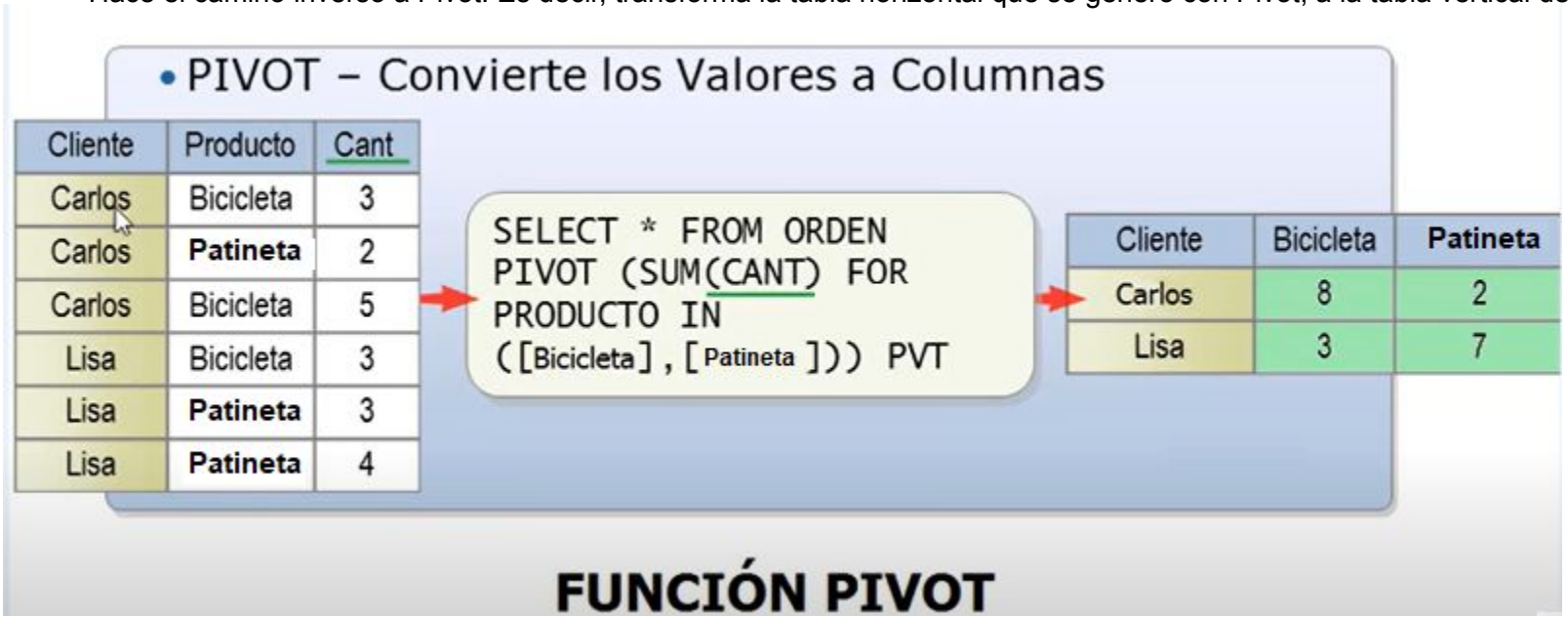
Pivot y Unpivot

Pivot:

- Transforma y agrupa los valores de una tabla vertical, en otra tabla horizontal
- Limitante: Cada uno de los valores de la columna “Producto” en la tabla vertical, serán columnas en la tabla horizontal

UnPivot:

- Hace el camino inverso a Pivot. Es decir, transforma la tabla horizontal que se generó con Pivot, a la tabla vertical de la que partió



MUCHAS GRACIAS