



SQL

# TALLER INTERNACIONAL SQL SERVER BÁSICO

Instructor: Ing. Gerardo Valdez Medina

## TABLA CONTENIDO

### MODULO 1: INTRODUCCIÓN A SQL SERVER

- Que es SQL Server
- Historia
- Transact SQL
- Características de SQL
- Definición del lenguaje (DDL – DML)
- Donde descargar SQL Server (2012,2014)

### MODULO 2: DISEÑO DE LA BASE DE DATOS

- Modelo E-R ○ Entidad ○ Relación ○ Atributos
- Consideraciones de Diseño
- Modelado de elementos de datos
- Tablas ○ Restricciones tablas ○ Restricciones de Columnas ○ Clave primaria
  - Clave foránea

### MODULO 3: CONSULTAS BÁSICAS

- Recuperando datos con SELECT
- Expresiones y operadores aritméticos
- Cambiando orden de columnas
- Clausula DISTINTC

### MODULO 4: CONDICIONES DE BÚSQUEDA

- Filtrando Datos ○ WHERE ○ BETWEEN ○ IN ○ LIKE ○ VALORES NULL ○ Operadores AND OR
- Ordenamiento De Datos

### MODULO5: AGRUPANDO Y RESUMIENDO DATOS

- Funciones de Agregado
- COUNT
- MIN
- MAX
- SUM
- AVG
- GROUP BY descripción y características
- HAVING descripción y características

# MÓDULO 1: INTRODUCCIÓN A SQL SERVER

## HISTORIA SQL

Empieza con la primera versión, que fue desarrollada en los laboratorios de IBM por Andrew Richardson, Donald C. Messerly y Raymond F. Boyce, a comienzos de los años 70, viendo la luz en 1974. Se basaron en el “modelo relacional de datos para grandes bancos de datos compartidos” de Edgar Frank Codd. A esta versión le llamaron SEQUEL, Structured English QUery Language, y fue diseñada para acceder y manejar los datos de la base de datos System R de IBM. Las experimentaciones con este prototipo condujeron, entre 1976 y 1977, a una revisión del lenguaje (SEQUEL/2), que a partir de ese momento cambió de nombre por motivos legales, convirtiéndose en SQL.

El prototipo (System R), basado en este lenguaje, se utilizó internamente en IBM y lo adoptaron algunos de sus clientes elegidos. Gracias al éxito de este sistema, que no estaba todavía comercializado, también otras compañías empezaron a desarrollar sus productos relacionales basados en SQL.

Fue en 1986, cuando el ANSI (American National Estándar Institute) adoptó SQL como estándar para los lenguajes relacionales, que en 1987 se transformó en estándar ISO.

## ¿QUÉ ES UNA BASE DE DATOS?

Una base de datos está constituida por un conjunto de información relevante para una empresa o entidad incluyendo los procedimientos para almacenar, controlar, gestionar y administrar esa información. Además, la información contenida en una base de datos cumple una serie de requisitos o características:

- Los datos están interrelacionados, sin redundancias innecesarias.
- Los datos son independientes de los programas que los usan.
- Se emplean métodos determinados para incluir datos nuevos y para borrar, modificar o recuperar los datos almacenados.

## ¿QUÉ ES UN SGBD O SISTEMA DE GESTIÓN DE BASE DE DATOS?

Un Sistema de Gestión de Bases de Datos (SGBD) es una aplicación comercial que permite construir y gestionar bases de datos, proporcionando al usuario de la Base de Datos las herramientas necesarias para realizar, al menos, las siguientes tareas:

- Definir las estructuras de los datos.
- Manipular los datos. Es decir, insertar nuevos datos, así como modificar, borrar y consultar los datos existentes.
- Mantener la integridad de la información.
- Proporcionar control de la privacidad y seguridad de los datos en la Base de Datos, permitiendo sólo el acceso a los mismos a los usuarios autorizados.

La herramienta más difundida para realizar todas estas tareas es el lenguaje SQL.

Algunos de los productos comerciales más difundidos son:

- ORACLE de Oracle Corporation.
- DB2 de I.B.M. Corporation
- SYBASE de Sybase Inc.
- Informix de Informix Software Inc.
- SQL Server de Microsoft Corporation.

## USO DEL LENGUAJE

En este manual se dará a conocer el uso del lenguaje DML (Sentencias de manipulación de datos), las cuales se utilizan para: □ Recuperar información. (SELECT) □ Actualizar la información: □

Añadir filas (INSERT)

- Eliminar filas (DELETE)
- Modificar filas (UPDATE)

Así como también se mostrara las sentencias DDL (Sentencias de definición de datos, las cuales se utilizan para:

- Crear objetos de base de datos (CREATE)
- Eliminar objetos de base de datos (DROP)
- Modificar objetos de base de datos (ALTER)

## MÓDULO 2: DISEÑO DE LA BASE DE DATOS

El carácter formal del modelo relacional hace relativamente sencilla su representación y gestión por medio de herramientas informáticas. No es casual, pues, que haya sido elegido como referencia para la construcción de la gran mayoría de los Sistemas de Gestión de Bases de Datos comerciales disponibles en el mercado.

El modelo relacional se basa en el concepto matemático de relación. En este modelo, la información se representa en forma de “tablas” o relaciones, donde cada fila de la tabla se interpreta como una relación ordenada de valores (un conjunto de valores relacionados entre sí).

El hecho de que los SGBDR (Sistemas gestores de base de datos) respeten el modelo relacional permite trabajar con una estructura lógica de la organización de los datos (tablas, vistas, índices...), independiente de la estructura física (archivos, etc.). Cada SGBDR debe proporcionar una vista lógica al usuario, que garantice un almacenamiento físico de la información.

Esta restricción es, al mismo tiempo, la fortaleza de los SGBDR, ya que la gestión de los datos desde un punto de vista lógico, simplifica mucho su uso. De esta manera, los usuarios poco o nada acostumbrados a desarrollar aplicaciones se pueden iniciar sin problemas en SQL.

## COMPONENTES.

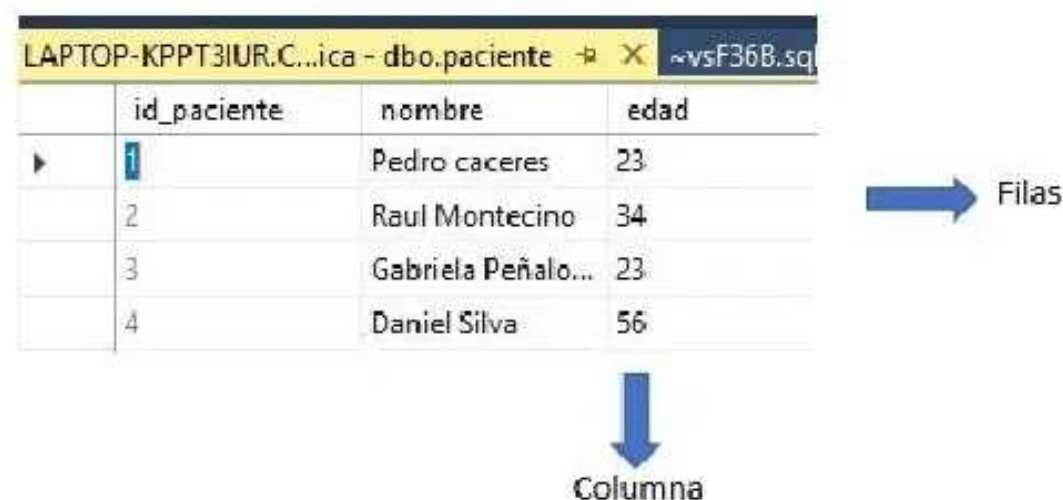
Como dijimos anteriormente, el modelo relacional de base de datos organiza y representa los datos mediante entidades. A continuación se muestra una tabla que compara la terminología utilizada en las bases de datos relacionales y sus alternativas, que permiten entender fácilmente sus componentes.

Relación	Tabla	Fichero
Tupla	Fila	Registro
Atributo	Columna	Campo
Grado	No. de columnas	No. de Campos
Cardinalidad	No. de filas	No. de registros

**Tablas:** Las tablas, también llamada relaciones, son los pilares esenciales de cualquier base de datos, ya que almacenan los datos. Una base de datos debería tener una tabla distinta para cada asunto principal, como registros de empleados, pedidos de clientes, métodos de entrega, proveedores, etc.

No deben duplicarse los datos en varias tablas. Esto es un error común fácil de evitar si se estructuran bien las tablas.

Todas las tablas están organizadas en Filas, y columnas, tal como se muestra en la siguiente imagen, en la cual se crea una tabla vacía, que permite guardar los datos de un paciente.



	id_paciente	nombre	edad
1		Pedro caceres	23
2		Raul Montecino	34
3		Gabriela Peñalo...	23
4		Daniel Silva	56

**Registros (tuplas):** Una tupla o registro es un conjunto de hechos acerca de una persona, de un evento o de cualquier otro elemento de interés. Por ejemplo el Cliente Daniel Berrios y su dirección y fecha de nacimiento. Cada tupla o registro contiene los valores que toma cada uno de los campos de un elemento

de la tabla. En una base de datos bien estructurada, cada tupla o registro debe ser único. Es decir, no deben existir dos o más registros que contengan exactamente la misma información.

En la siguiente imagen, se muestra enmarcado el registro de un paciente.

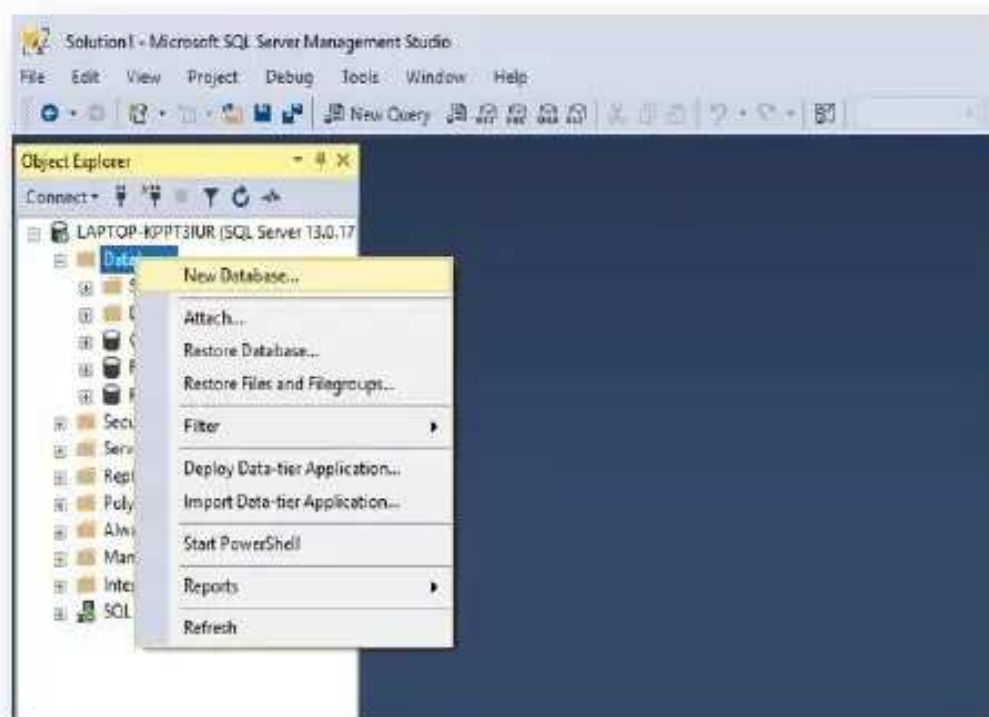
	id_paciente	nombre	edad
	1	Pedro caceres	23
▶	2	Raul Montecino	34
	3	Gabriela Peñaloza	23
	4	Daniel Silva	56

**Campos (atributos):** Los campos de la base de datos tienen valores que determinan el tipo de datos que pueden almacenar, cómo se muestran los datos y qué se puede hacer con ellos. Un valor importante para los campos es el tipo de datos, que puede ser número, texto, moneda (dinero), fecha, hora, etc. El tipo de datos limita y describe la clase de información del campo. También determina las acciones que se pueden realizar en el campo y la cantidad de memoria que utilizan los datos.

## CREAR TABLAS.

En el menú del costado, se encuentran las carpetas que permiten que el sistema de base de dato funcione. En la carpeta databases (Base de datos), se encuentran aquellas creadas por defecto y las que procederemos a generar.

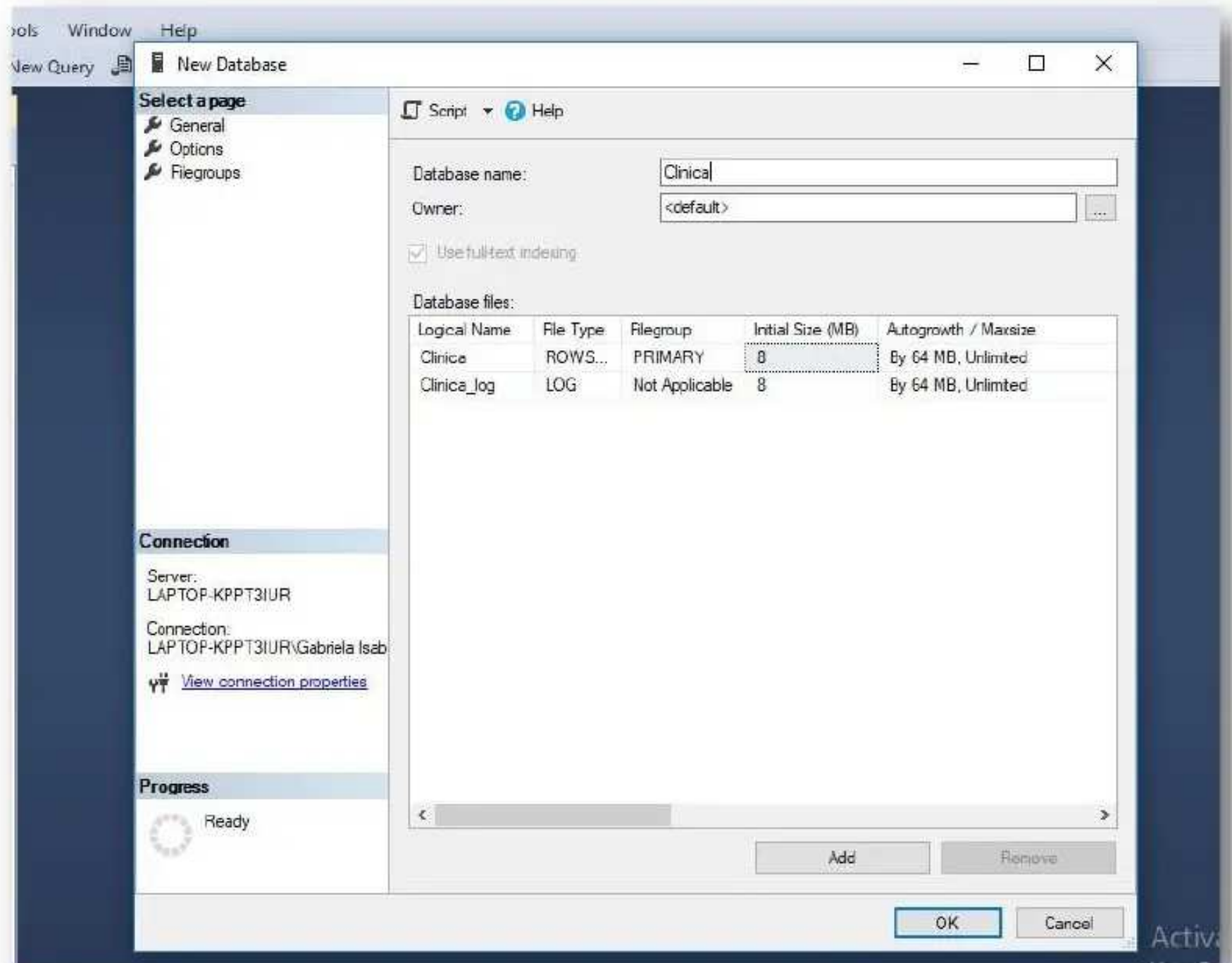
Para eso daremos click derecho y seleccionaremos New databases.

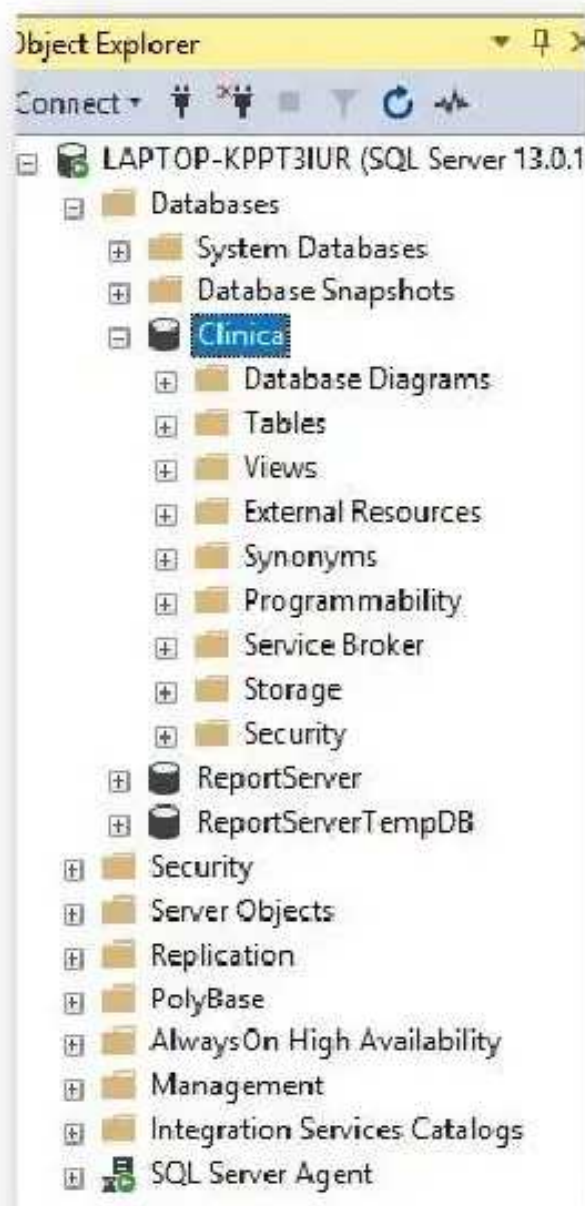


Luego, le daremos un nombre relacionado al contenido de nuestra base de datos.



Para el ejemplo, crearemos una base de datos que permita el almacenado de datos de una clínica.





Como veremos, se han creado todas las carpetas necesarias para el buen funcionamiento de la BD, las cuales van a ser vistas, a medida que sean requeridas para este manual.

Para comenzar a realizar las tablas, se debe ir a la carpeta tables (tablas) ubicada en la lista del costado, y daremos click derecho y seleccionaremos new, y luego table.

Al abrir la tabla se nos desplegará una ventana que permite el ingreso de las columnas de una tabla.

Siguiendo el ejemplo de la creación de la base de datos para una clínica, procederemos a crear una tabla para el registro de pacientes, en la cual lo dejaremos de la siguiente manera:



LAPTOP-KPPT3IUR.Clinica - dbo.paciente* X LAPTOP-KPPT3IUR.Clinica -			
	Column Name	Data Type	Allow Nulls
id_paciente		int	<input type="checkbox"/>
Rut		varchar(12)	<input checked="" type="checkbox"/>
nombre		varchar(50)	<input checked="" type="checkbox"/>
fecha_nacimiento		date	<input checked="" type="checkbox"/>
dirección		varchar(50)	<input checked="" type="checkbox"/>
telefono		varchar(10)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

\*Para este caso, es posible utilizar el rut como PK, sin embargo, por un tema de orden en la creación de tablas, dejaremos la PK como id\_nombreDeTabla.

## TIPOS DE CLAVES

Se puede considerar de una clave:

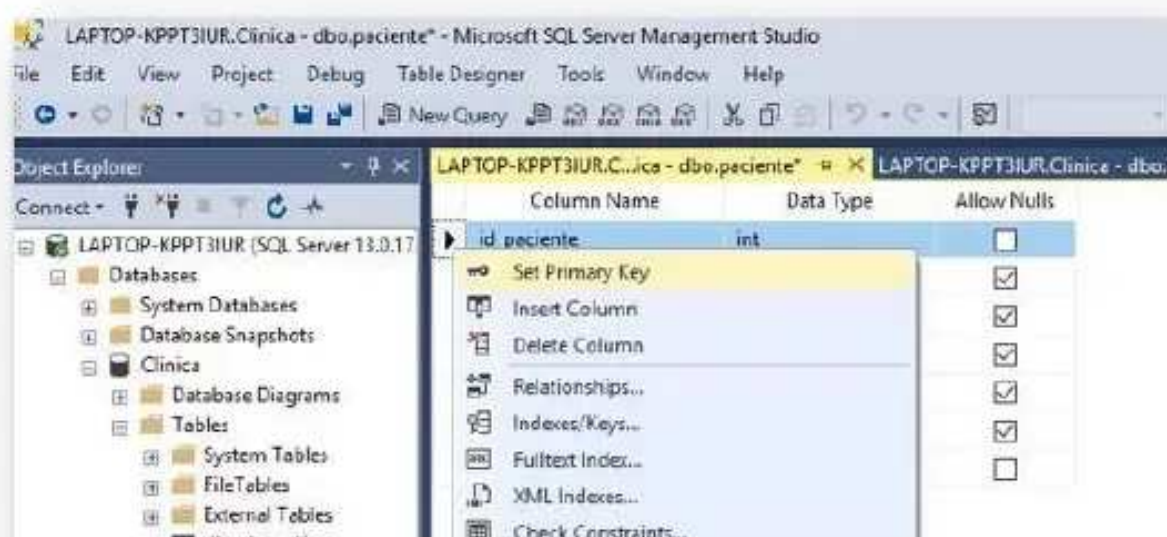
- 1.- Es un atributo o conjunto de atributos cuyos valores distinguen a un registro en una tabla.
- 2.- Una clave es el conjunto mínimo de atributos cuyos valores le dan una identificación única a la tupla en la relación.
- 3.- Una clave es una referencia que se utiliza para identificar los registros de forma única y está formada por uno o más campos de los registros.

- Clave primaria o principal.- Es un identificador único para cada registro. No puede contener entradas nulas. Para cada registro de una relación se utiliza un identificador único, denominado clave primaria o clave principal. Para elegir un campo de una tabla como clave primaria debe cumplir con las siguientes características:
  - Deberá seleccionarse la que ocupe un menor espacio de almacenamiento.
  - Tener una codificación sencilla.
  - El contenido de sus valores deben ser conocido y no variar.
  - No debe tener valores nulos.    ○ Podrá utilizarse en otras tablas para construir interrelaciones.
  - Deben ser fácilmente recordables por el usuario. Por ejemplo, si se tiene una tabla con 2500 procedimientos de una clínica, podríamos poner un id\_procedimiento como clave principal.

Esto significa que si proporcionamos la clave principal la base de datos puede encontrar al

registro que contenga la clave proporcionada. Cada procedimiento tiene un identificador único, así como cada ciudadano tiene un RUT único, diferente del de todos los demás. La clave principal debe ser una información que no cambie con frecuencia.

En la imagen anterior, se puede apreciar que se identificó la Primary key como id\_paciente, la cual se agrega haciendo click derecho en la columna en vista diseño, y luego dando click en set primary key, (establecer clave principal).



Clave externa: Define una columna o combinación de columnas cuyos valores coinciden con la clave principal de la misma u otra tabla. Permite la declaración de la llave foránea.

## TIPOS DE DATOS

En SQL Server, cada columna, variable local, expresión y parámetro tiene un tipo de datos relacionado. Un tipo de datos es un atributo que especifica el tipo de datos que el objeto puede contener: datos de enteros, datos de caracteres, datos de moneda, datos de fecha y hora, cadenas binarias, etc.

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

<b>Numéricos exactos</b>	Cadenas de caracteres Unicode
<b>Numéricos aproximados</b>	Cadenas binarias
<b>Fecha y hora</b>	Otros tipos de datos
<b>Cadenas de caracteres</b>	

## TIPO DE DATO DE CARÁCTER

Tipo de datos		
<b>Char(n)</b>	Datos de caracteres de longitud fija hasta un máximo de 8,000 caracteres.	Longitud definida de 1 byte
<b>Nchar(n)</b>	Datos de caracteres Unicode de longitud fija.	Longitud definida 2 bytes
<b>VarChar(n)</b>	Dato de caracteres de longitud variable hasta un máximo de 8,000 caracteres.	1 byte por caracter
<b>VarChar(max)</b>	Datos de caracteres de longitud variable hasta un máximo de 2gb	1 byte por caracter
<b>nVarChar(n)</b>	Datos de caracteres Unicode de longitud variable hasta un máximo de 8,000 caracteres	2 byte por caracter
<b>nVarChar(max)</b>	Datos de caracteres Unicode de longitud variable hasta un máximo de 2GB.	2 byte por caracter
<b>Text</b>	Datos de caracteres de longitud variable	1 byte por caracter
<b>nText</b>	Datos de caracteres Unicode de longitud variable	2 byte por caracter
<b>Sysname</b>	Tipo de dato usado para nombre de tablas o columnas equivale a nvarchar(128)	2 byte por caracter

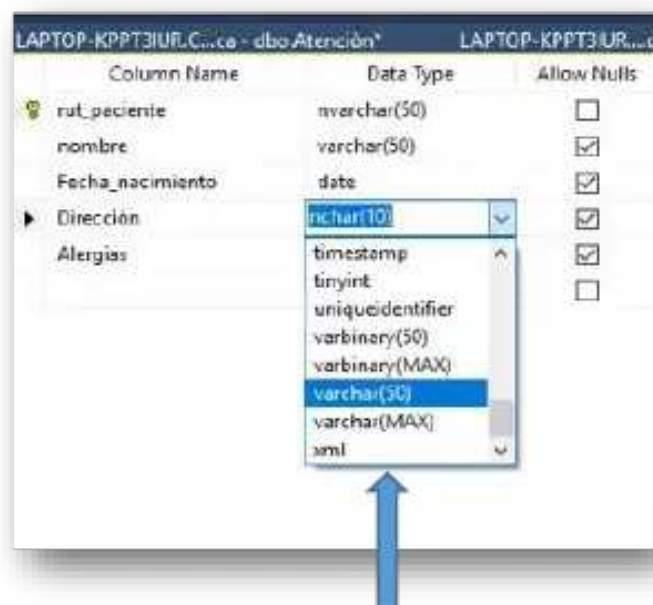
## TIPO DE DATO NUMERICO

Tipo de datos		
<b>Bit</b>	Datos enteros con valor 1 ó 0	1 bit
<b>Tinyint</b>	Enteros desde 0 a 255	1 byte
<b>Smallint</b>	Enteros desde -32,768 a 32,767	2 byte
<b>Int</b>	Enteros desde -2,147,483,648 a 2,147,483,647	4 byte
<b>Bigint</b>	Enteros desde $-2^{63}$ a $2^{63}-1$	8 byte
<b>Decimal or Numeric</b>	Números de precisión exacta hasta $-10^{38} + 1$	Varia acorde con el tamaño
<b>Money</b>	Números desde $-2^{63}$ a $2^{63}$ ,	8 byte
<b>SmallMoney</b>	Números desde -214,748.3648 a +214,748.3647	4 byte
<b>Float</b>	Números con precisión de coma flotante desde $-1.79E + 308$ a $1.79E + 308$ , dependiendo del bit de precisión .	4 o 8 bytes
<b>Real</b>	Flotante con 24-bit de precisión	4 byte

## TIPO DE DATO FECHA

Tipo de datos	Formato	Intervalo	Tamaño de almacenamiento (bytes)	Ajuste de zona horaria
<b>Time</b>	hh:mm:ss[.nnnnnnnn]	De 00:00:00.0000000 a 23:59:59.9999999	De 3 a 5	No
<b>Date</b>	AAAA-MM-DD	De 0001-01-01 a 9999-12-31	3	No
<b>Smalldatetime</b>	AAAA-MM-DD hh:mm:ss	De 1900-01-01 a 2079-06-06	4	No
<b>Datetime</b>	AAAA-MM-DD hh:mm:ss[. nnn]	De 1753-01-01 a 9999-12-31	8	No
<b>datetime2</b>	AAAA-MM-DD hh:mm:ss[.nnnnnnnn]	De 0001-01-01 00:00:00.0000000 a 9999-12-31 23:59:59.9999999	De 6 a 8	No
<b>Datetimeoffset</b>	AAAA-MM-DD hh:mm:ss[.nnnnnnnn] [+  ]hh:mm	De 0001-01-01 00:00:00.0000000 a 9999-12-31 23:59:59.9999999 (en UTC)	De 8 a 10	Si

Los tipos de datos son definidos al momento de crear la tabla y agregar las columnas, tal como se muestra en la imagen.

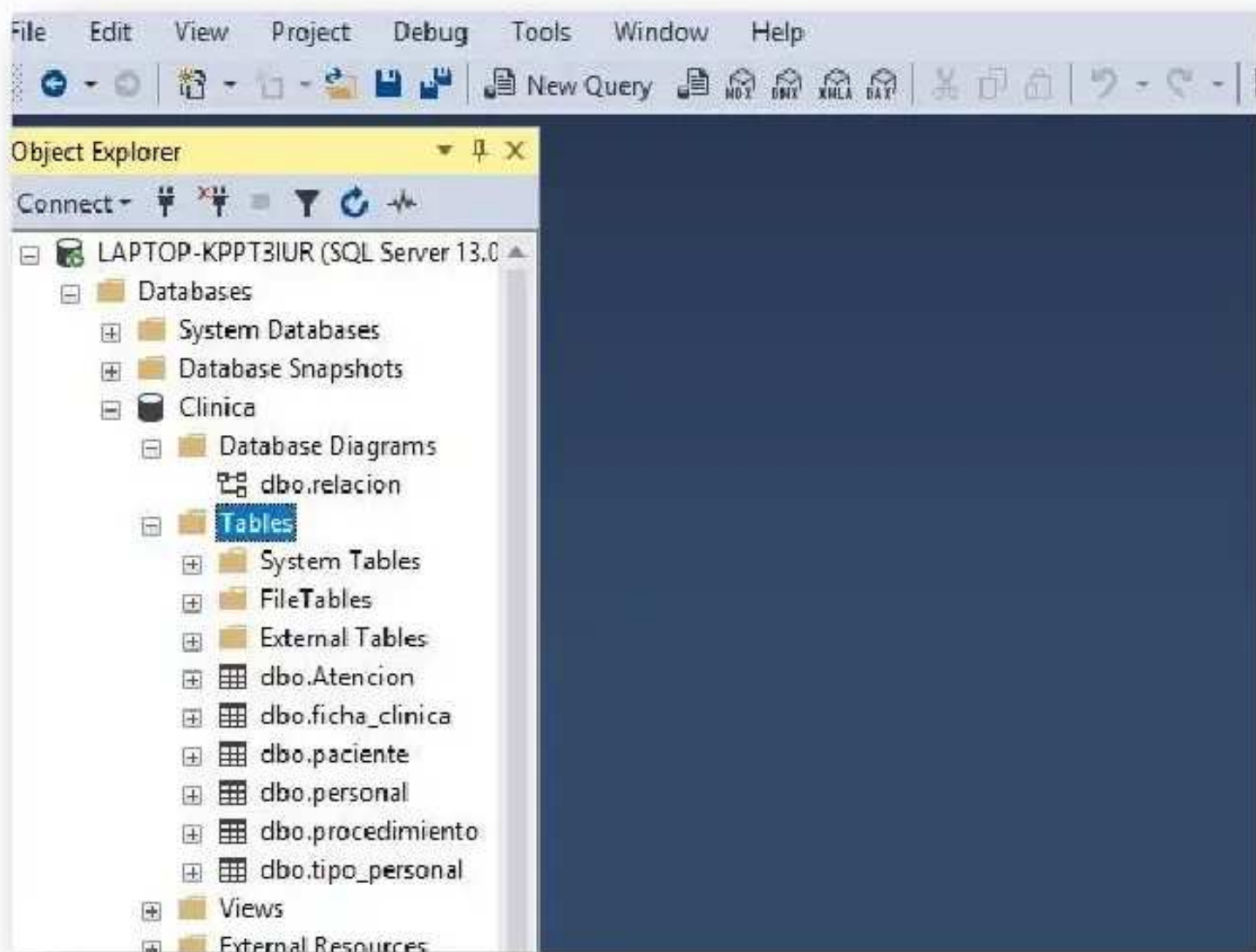


**Tipo de datos**

## EJERCICIOS MODULO 2

Para la base de datos creada en los ejemplos, genere los siguientes ejercicios que permitirán almacenar más información importante para la clínica. Se muestran imágenes para que usted sepa cómo debería quedar su BD.

- 1) Genere tablas que permitan guardar los datos de ficha\_clinica, personal, tipos de personal (considerando que un paciente puede ser atendido por médicos, enfermeras, tec, en enfermería, auxiliares, etc.).
- 2) Verifique su clave primaria para cada tabla, así como también las secundarias. Cada columna debe contar con el tipo de datos correspondiente, y así permitir el correcto ingreso de los datos.



3) Inserte datos para cada tabla (al menos 5 filas) por tabla.

LAPTOP-KPPT3IUR.C...ica - dbo.paciente						
LAPTOP-KPPT3IUR.C...ca - dbo.Atención						
	id_paciente	nombre	edad	Alergias	Enfermedades ...	Tipo_sangre
	1	Pedro caceres	23	Amoxicilina	Ninguna	O positivo
	2	Raul Montecino	34	Ninguna	cefaleas	AB positivo
	3	Gabriela Peñalo...	23	Ninguna	Presion Alta	AB positivo
	4	Daniel Silva	56	Amoxicilina	Ninguna	O positivo
	5	Gerardo Soto	85	Ninguna	Hipotirodismo	AB positivo
	6	Gabriel Alvarez	83	Paracetamol	Ninguna	B positivo

## MÓDULO 3: CONSULTAS BÁSICAS

Las peticiones para actuar sobre los datos se expresan mediante sentencias. Éstas deben escribirse de acuerdo con las reglas sintácticas y semánticas del lenguaje.

Permiten enunciar operaciones sobre tablas. Existen varios tipos según el tipo de operación que expresan, por lo que a continuación se aprecia un cuadro con las categorías de consultas.

<b>DML</b> Lenguaje de Manipulación de Datos (Data Manipulation Language)	Sentencias que permiten realizar consultas y actualizaciones de datos (inserción, borrado y modificación de filas): SELECT, INSERT, UPDATE, DELETE.
<b>DDL</b> Lenguaje de Definición de Datos (Data Definition Language)	Sentencias que permiten definir nuevos objetos (tablas, índices, claves, etc) o destruir los ya existentes: CREATE, DROP, ALTER. o DCL.
Sentencias de Control de Datos (Data Control Language)	Sentencias que permiten controlar aspectos varios, como, por ejemplo, la autorización de acceso a los datos: GRANT, REVOKE.

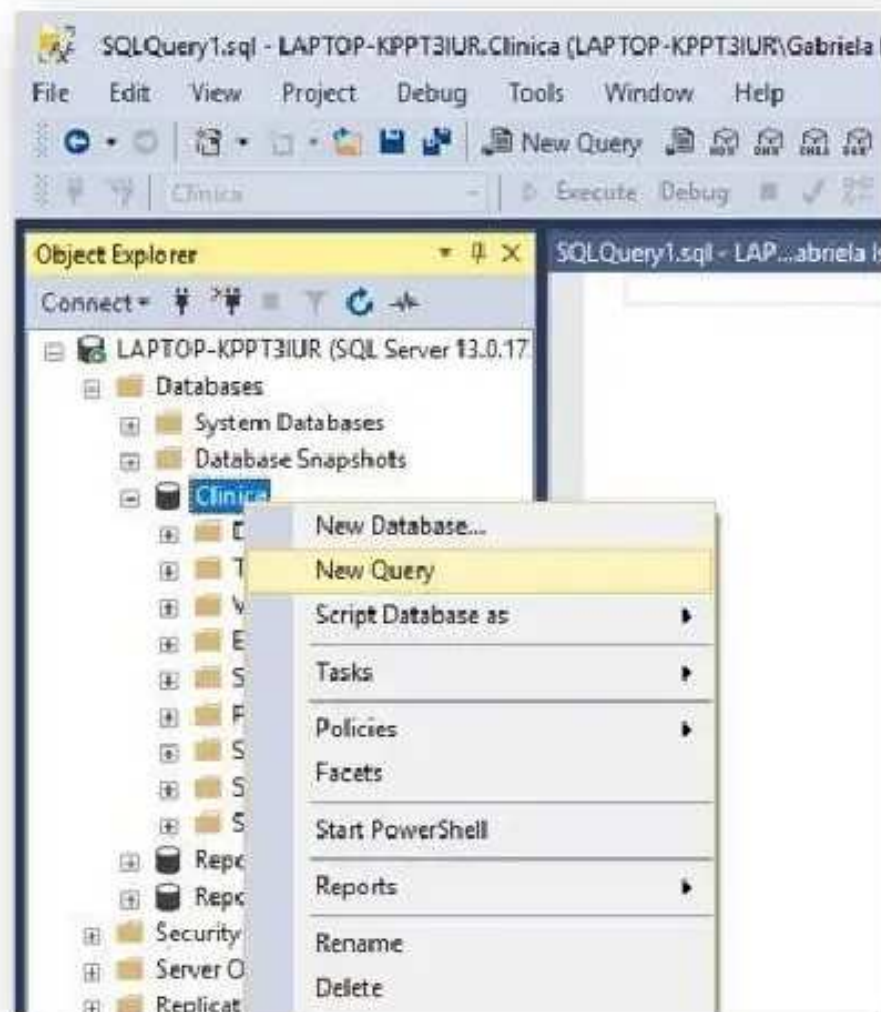


## ACCEDER AL EDITOR DE CONSULTAS

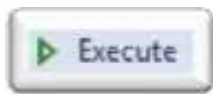
Para generar consultas en nuestra Base de datos, se debe abrir un editor de consultas, el cual permitirá ejecutar las acciones que se requieran. Para ello, en el menú superior, se selecciona new query (nueva consulta).



También es posible abrirlo desde las carpetas del costado, dando click derecho sobre la BD creada y seleccionando new query (nueva consulta).



Otros botones:



Ejecuta la consulta realizada.



Muestra la base de datos sobre la cual se realizan las consultas.



Permite guardar la consulta



Permite abrir otras consultas

## SELECT

Las consultas de selección se utilizan para indicar al motor de datos que devuelva información de las bases de datos. Esta información es devuelta en forma de conjunto de registros que se pueden almacenar en una nueva tabla.

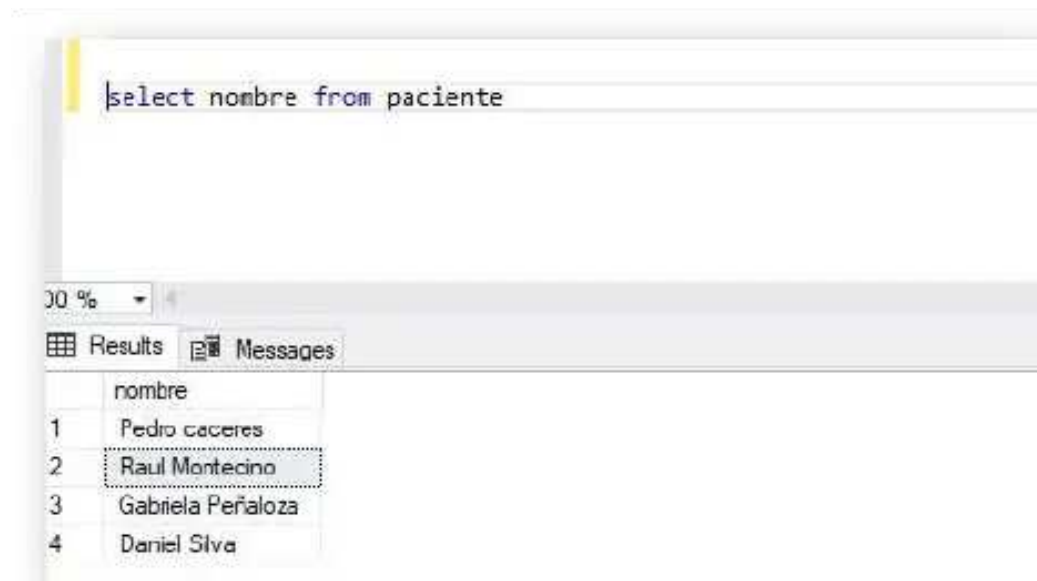
La sintaxis más sencilla de una consulta de selección es la siguiente:

**SELECT columna1, columna2, columna3 FROM nombre de tabla**

En este caso extrae la información de la columna 1,2 y 3 de la tabla.

Por ejemplo, la consulta SELECT Nombre, Teléfono FROM Clientes; devuelve una tabla temporal con los campos nombre y teléfono de la tabla clientes.

Utilizando nuestra tabla de paciente de la BD clínica creada, extraeremos el nombre mediante la consulta select.



Si lo que se requiere es mostrar todas las columnas existentes en una tabla, entonces, reemplazaremos el nombre de las columnas por el símbolo asterisco, como se muestra en la sintaxis:

```
SELECT * FROM nombre de
tabla
```

Utilizando el ejemplo anterior, nuestra consulta quedaría `SELECT * FROM PACIENTE`.

## ORDER BY

Adicionalmente se puede especificar el orden en que se desean recuperar los registros de las tablas mediante la cláusula `ORDER BY`. Su sintaxis sería:

```
ORDER BY
lista_de_columnas
```

Donde Lista-campos representa los campos a ordenar. Por ejemplo, la consulta:

```
SELECT Nombre, edad FROM paciente ORDER BY Nombre.
```

Devuelve los campos Nombre y edad de la tabla paciente ordenados por el campo Nombre. Se pueden ordenar los registros por más de un campo.

Por ejemplo:

```
SELECT Nombre, edad FROM Clientes ORDER BY Nombre, edad;
```

Incluso se puede especificar el orden de los registros: ascendente mediante la cláusula `ASC` (valor por defecto) o descendente `DESC`. Por ejemplo:

```
SELECT Nombre, edad FROM paciente ORDER BY nombre DESC;
```

## CONSULTAS CON PREDICADO (ALL, TOP, DISTINCT)

En cuanto al conjunto de registros seleccionados, estos modificadores, que se incluyen entre `SELECT` y el primer nombre del campo a recuperar, provocan las siguientes acciones:

- **ALL:** Devuelve todos los campos de la tabla (valor por defecto).
- **TOP** Devuelve un determinado número de registros de la tabla.
- **DISTINCT** Omite repeticiones de registros cuyos campos seleccionados coincidan totalmente.
- **DISTINCTROW** Omite repeticiones de registros basándose en la totalidad del registro y no sólo en los campos seleccionados.

**ALL** Es el valor por defecto. El Motor de base de datos selecciona todos los registros que cumplen las condiciones de la instrucción SQL. No es conveniente abusar de este predicado ya que obligamos al motor de la base de datos a analizar la estructura de la tabla para averiguar los campos que contiene, es mucho más rápido indicar el listado de campos deseados. Se suele sustituir por el símbolo `*`.

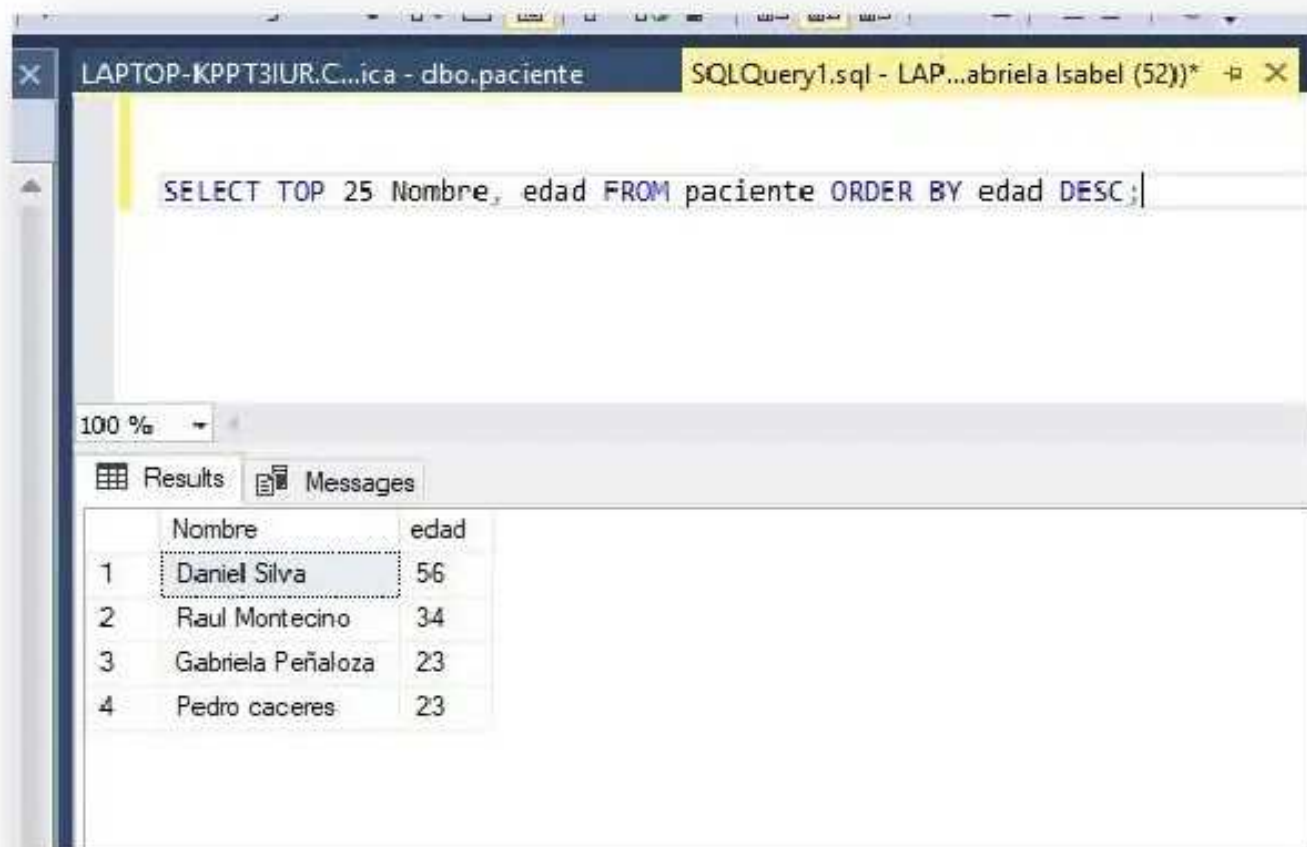
Por ejemplo:

`SELECT ALL FROM paciente;`

**TOP** Devuelve un cierto número de registros que corresponden al principio o al final de un rango especificado por una cláusula `ORDER BY`. Por ejemplo, supongamos que queremos recuperar los nombres de los 25 paciente con menor edad.

Para ello, emplearemos la consulta:

`SELECT TOP 25 Nombre, edad FROM paciente ORDER BY edad DESC;`



El predicado `TOP` no elige entre valores iguales. En el ejemplo anterior, si la edad de Gabriela y Pedro son iguales, la consulta devolverá 26 registros en lugar de 25. (Nótese que para la BD dato ejemplo sólo de cuentan con 5 registros, por lo tanto esa es la cantidad máxima de registros a mostrar).

## **DISTINCT**

Omite los registros que contienen datos duplicados en los campos seleccionados. Por ejemplo, varios pacientes listados en la tabla `paciente` pueden tener la misma edad. Si dos registros contienen 23 en el campo `edad`, la siguiente instrucción SQL devuelve un único registro:

`SELECT DISTINCT edad FROM paciente;`

LAPTOP-KPPT3IUR.C...ica - dbo.paciente SQLQ

```
SELECT DISTINCT edad FROM paciente
```

100 %

Results Messages

	edad
1	23
2	34
3	56

## AS

En determinadas circunstancias es necesario asignar un nombre nuevo a alguna columna determinada de un conjunto de registros devuelto por una consulta. Para ello, usaremos la palabra reservada AS, que se encarga de asignar el nombre que deseamos a la columna deseada.

En este ejemplo, asignamos el nombre “edad\_paciente” a la columna de edad con AS.

```
SELECT DISTINCT edad AS edad_paciente FROM paciente
```

100 %

Results Messages

	edad_paciente
1	23
2	34
3	56

### EJERCICIOS MODULO 3

- 1) Listar todos los pacientes. `select * from paciente`
- 2) Mostrar sólo los nombres y enfermedades importantes de pacientes. `select nombre, [Enfermedades previas] from paciente`
- 3) Mostrar sólo los pacientes que tienen sangre de tipo AB positivo.  
`select nombre from paciente where Tipo_sangre= 'AB positivo'`
- 4) Mostrar los nombres y edades de pacientes que sean mayores de 18 años.  
`select nombre, edad from paciente where edad > 18`
- 5) Hallar el promedio de las edades de los pacientes. `select avg(edad) from paciente`
- 6) Mostrar el nombre y edad de pacientes, ordenados por edades.  
`select nombre, edad from paciente order by edad`

## MÓDULO 4: CONDICIONES DE BUSQUEDA

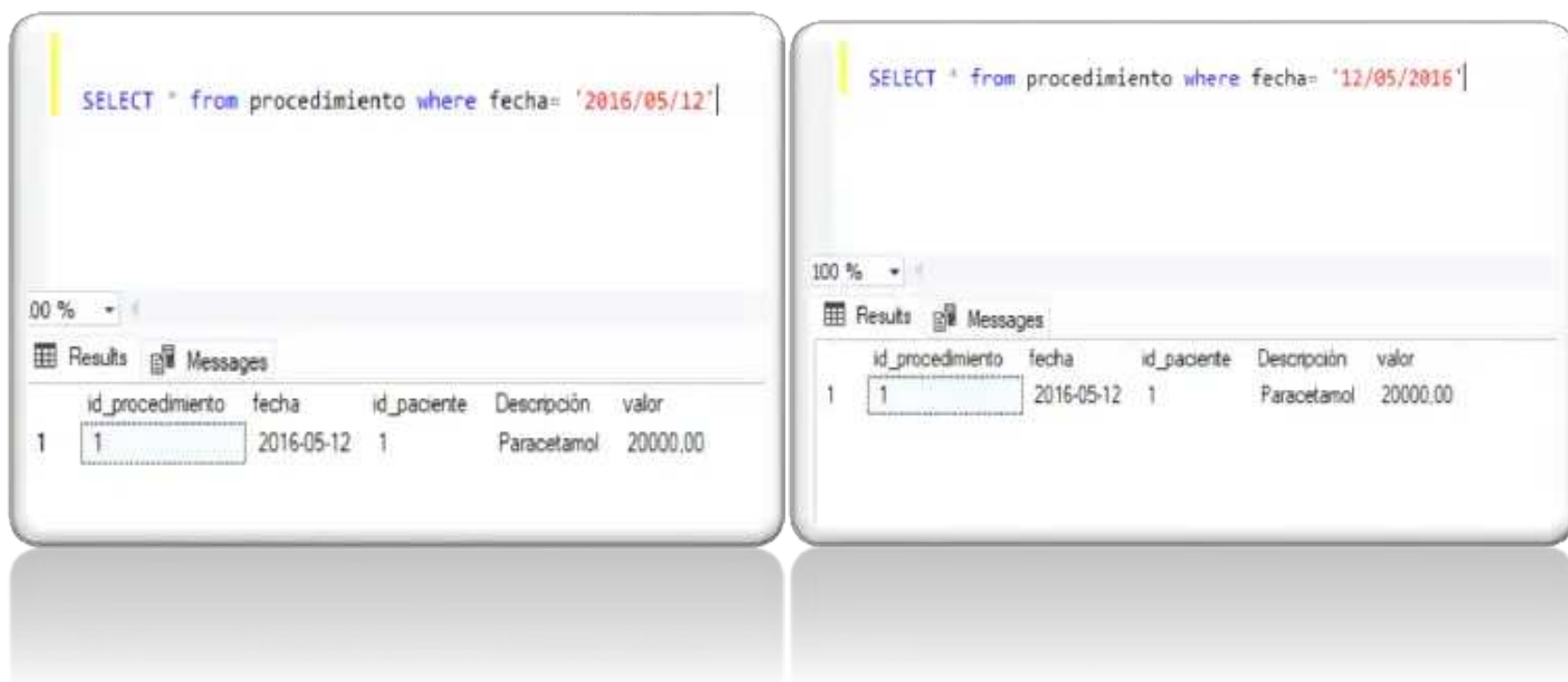
### CRITERIOS DE SELECCIÓN

Los métodos empleados hasta el momento para recuperar un conjunto de registros de una tabla devuelven todos los registros de la tabla. A continuación, se estudiarán las posibilidades de filtrar los registros con el fin de recuperar solamente aquellos que cumplan ciertas condiciones preestablecidas. Antes de comenzar debemos tener presente que:

Cada vez que se desee establecer una condición referida a un campo de texto la condición de búsqueda debe ir encerrada entre comillas simples.

Las fechas se deben escribir siempre encerrada entre almohadillas comilla simple. Por ejemplo, si deseamos referirnos al día 3 de septiembre de 1995 debemos escribir: '03/09/1995' ó '1995/09/03'. A continuación, un ejemplo con la fecha para ambas escrituras de fecha.





## FILTRAR DATOS

Algunos operadores lógicos soportados por SQL son: **AND**, **OR** y **NOT**. Los dos primeros poseen la sintaxis:

**expresión1 operador expresión2**

En donde expresión1 y expresión2 son las condiciones que evaluar, el resultado de la operación varía en función del operador lógico. Si a cualquiera de las anteriores condiciones le antepone el operador **NOT** el resultado de la operación será el contrario al devuelto sin el operador **NOT**.

Por ejemplo, considérense las consultas:

**SELECT \* FROM paciente WHERE Edad > 25 AND Edad < 50;**

**SELECT \* FROM paciente WHERE (Edad > 25 AND Edad < 50) OR nombre = 'Gabriela Peñaloza'**

El operador **BETWEEN** se utiliza para indicar que deseamos recuperar los registros según el intervalo de valores de un campo. Su sintaxis es:

**campo BETWEEN valor1 And valor2**

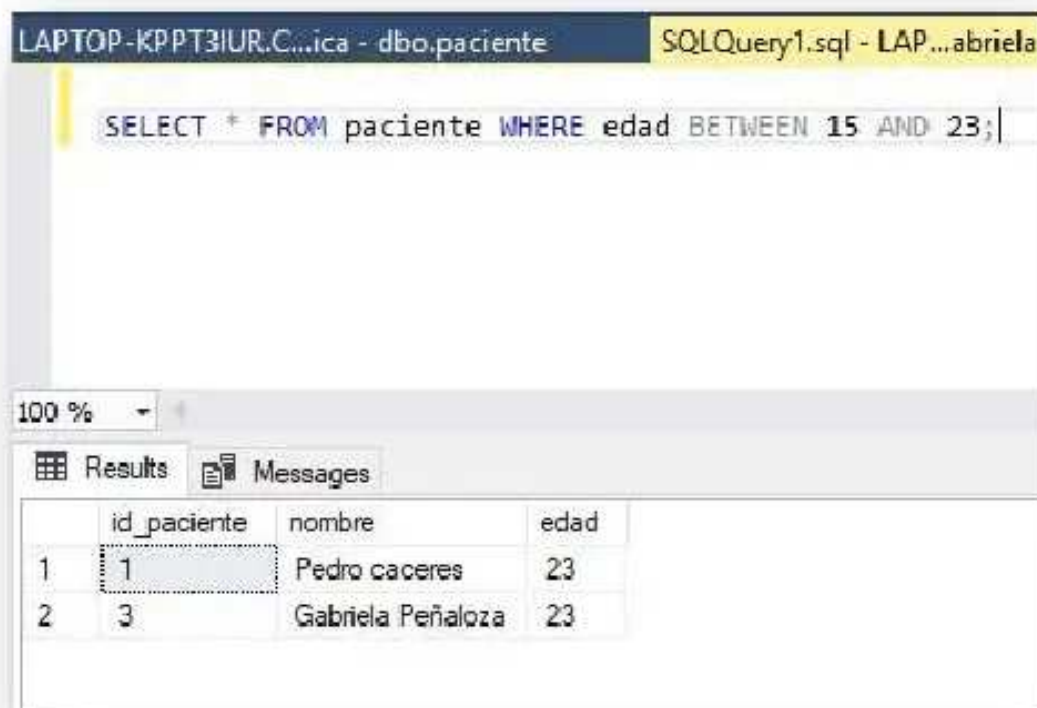
En este caso la consulta devolvería los registros que contengan en "campo" un valor incluido en el intervalo valor1, valor2 (ambos inclusive). Si antepone la condición **Not** devolverá aquellos valores no incluidos en el intervalo.

Por ejemplo:

**SELECT \* FROM paciente WHERE edad BETWEEN 15 AND 23;**

(Devuelve los pacientes cuya edad esta entre 15 y 23 años.

SELECT \* FROM paciente WHERE edad NOT BETWEEN 15 AND 23;



## LIKE

El operador like Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL. Su sintaxis es:

**Expresión LIKE dato\_buscado**

Donde expresión es un campo, y dato\_buscado es un patrón contra el que se compara la expresión. Se puede utilizar el operador LIKE para encontrar valores en los campos que coincidan con el modelo especificado. Como dato\_buscado, se puede especificar un valor completo ('Gabriela Peñaloza'), o bien se pueden utilizar caracteres comodines como los reconocidos por el sistema operativo para encontrar un rango de valores (por ejemplo, LIKE Ga%).

Ejemplo:

SELECT \* FROM paciente where nombre LIKE Ga%



## IN

EL operador in devuelve los registros cuyo campo indicado coincide con alguno de los datos en una lista. Su sintaxis es:

**expresión [NOT] IN (valor1, valor2, . . .)**

Por ejemplo:

SELECT \* FROM paciente WHERE edad IN (23);

## MÓDULO 5: AGRUPANDO Y RESUMIENDO DATOS

Transact SQL pone a nuestra disposición múltiples funciones agregadas, las más comunes que usaremos son:

### COUNT

Calcula el número de registros devueltos por una consulta. Su sintaxis es la siguiente:

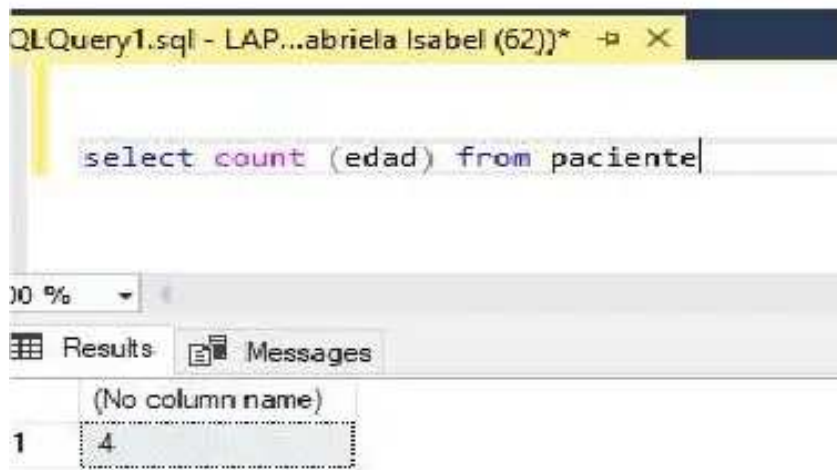
**COUNT(campos)**

En donde campos contiene el nombre del campo que desea contar. Los operandos de campos pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL). Puede contar cualquier tipo de datos incluso texto. Count simplemente cuenta el número de registros sin tener en cuenta qué valores se almacenan en los registros. La función Count no cuenta los registros que tienen campos null a menos que el campo sea el carácter comodín asterisco (\*).

Ejemplo:

SELECT COUNT(edad) FROM paciente.

SELECT COUNT(\*) FROM paciente.



## MAX Y MIN

Devuelven el mínimo o el máximo de un conjunto de valores contenidos en un campo específico de una consulta. Su sintaxis es:

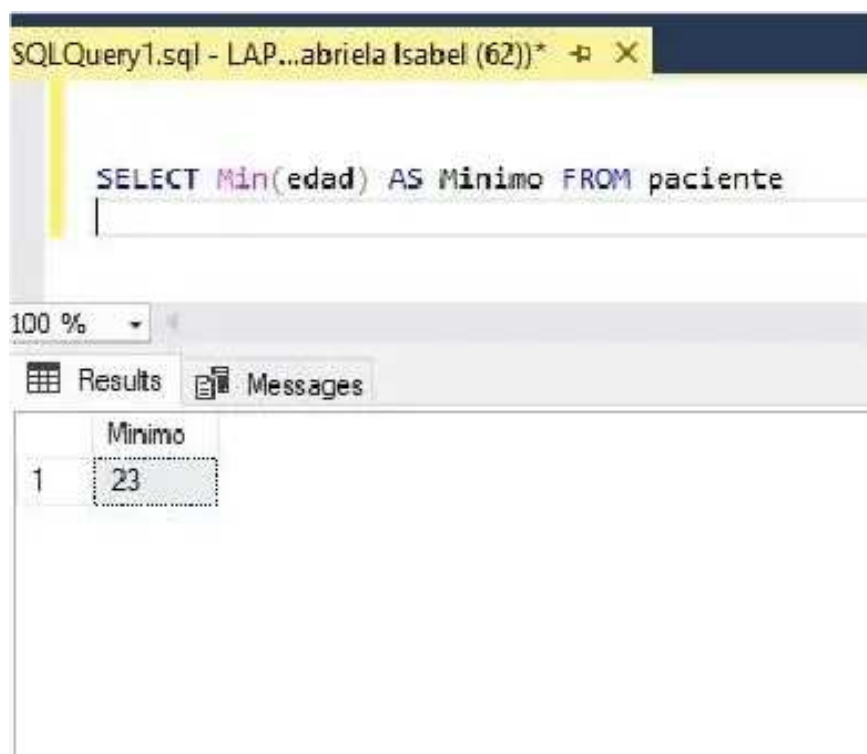
Min(expresión) Max(expresión)

En donde expresión es el campo sobre el que se desea realizar el cálculo. expresión pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL).

Ejemplo:

SELECT Min(edad) AS Mínimo FROM paciente

Devuelve cuál es la edad mínima de los pacientes de la clínica.



SELECT Max(edad) AS Maximo FROM paciente

Devuelve cuál es la edad mínima de los pacientes de la clínica.



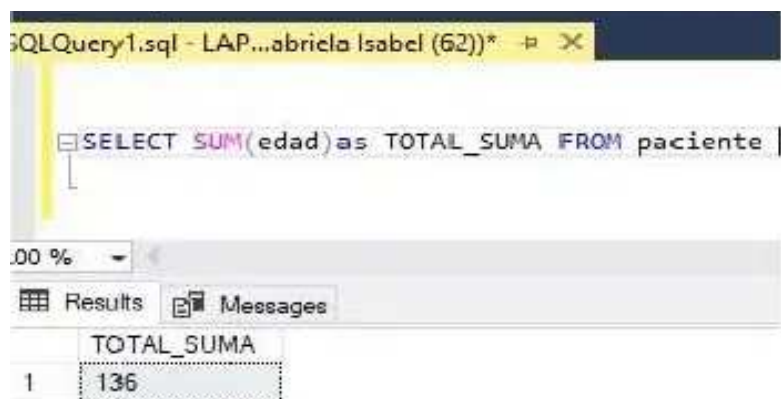
## SUM

Devuelve la suma del conjunto de valores contenido en un campo específico de una consulta.

Su sintaxis es:

**SUM(expresion)**

En donde expresión representa el nombre del campo que contiene los datos que desean sumarse o una expresión que realiza un cálculo utilizando los datos de dichos campos.



Ejemplo:

SELECT SUM(edad) FROM paciente.

Se realiza la suma de todas las edades ingresadas.

## AVG

Calcula el promedio de un conjunto de valores contenidos en un campo especificado de una consulta. Su sintaxis es:

**Avg(expresión)**

En donde expresión representa el campo que contiene los datos numéricos para los que se desea calcular el promedio. La función Avg no incluye ningún campo Null en el cálculo.

Ejemplo:

```
SELECT AVG(edad) FROM paciente
```

Esta consulta obtiene como resultado el promedio de todas las edades de la tabla paciente.

