

Módulo 4 / Bases de datos

AUTOR: GIRIBALDI, Pablo Germán

Índice

Módulo 4 / Bases de datos	1
Introducción	2
Conceptos fundamentales.	2
Sistemas de información	2
Almacenamiento de datos	3
Diferencia entre datos e información.	3
Bases de datos	4
Tipos de bases de datos	4
Según su variabilidad	4
Bases de datos estáticas	4
Bases de datos dinámicas	4
Según el contenido	4
Sistemas gestores de bases de datos.	7
Diseño de bases de datos	7
Diseño conceptual	8
Diagrama Entidad Relación	8
Diseño lógico	9
Tablas	11
Estructura de almacenamiento de datos	11
Las 12 reglas de Codd	11
Normalización	13
Primera Forma Normal (1FN)	13
Segunda Forma Normal (2FN)	14
Tercera Forma Normal (3FN)	15
Cuarta Forma Normal (4FN)	16
Quinta Forma Normal (5FN o forma normal de Proyección-Unión)	16
Diseño físico	17
¿Qué es SQL?	18
Instalación SQL Server	18
Instalación SQL Server Management Studio	22
Conexión al motor y creación de bases de datos	29
Scripts	37

Cláusulas	42
Funciones de agregación	43
Subconsultas	43
Procedimientos almacenados	44

Introducción

El siguiente texto contiene conceptos teóricos y prácticos sobre las actividades de quienes crean y administran bases de datos, ya que sin ellas, dependiendo del contexto, un sistema informático no tendría mucho sentido.

Durante el siguiente trayecto, nos enfrentaremos a la tarea de diseño e implementación de bases de datos, introduciéndonos en el lenguaje SQL (Lenguaje de consulta estructurado) para realizar consultas desde las más sencillas a las más complejas, incluyendo funciones de agrupamiento y cálculos, como así también la creación física de nuestras bases de datos utilizando los sistemas de gestión de base de datos (SGDB) apoyándonos en la herramienta Microsoft SQL Server 2019 Developer Edition.

Conceptos fundamentales.

Antes de introducirnos en el diseño y construcción de bases de datos, es importante definir algunos conceptos básicos.

Sistemas de información

Un sistema de información (SI) es un conjunto ordenado de mecanismos para poder administrar y procesar datos de manera fácil y rápida.

Los SI están compuestos por una serie de recursos interconectados entre sí, manteniendo una disposición conveniente en base a su propósito. Estos recursos pueden ser humanos, de hardware, de software, de datos, de actividades.

Es importante destacar la diferencia entre *sistemas de información* y *sistemas informáticos*. Aunque estos últimos, por lo general, constituyen la mayor parte de un SI, no todos los SI van de la mano de la informática.

Aunque todos los SI tienen un objetivo en común que es el de organizar y optimizar la información de una organización, hay distintos tipos de SI dependiendo de su principal uso, algunos de ellos son:

- Sistemas de procesamiento de transacciones (**TPS**): Son aquellos que gestionan la información referente a las transacciones llevadas a cabo en una empresa. También son conocidos como sistemas de gestión operativa;

- Sistemas de información ejecutiva (**EIS**): Son aquellos que monitorean las variables gerenciales de un área en particular de una organización, teniendo en cuenta la información externa e interna;
- Sistemas de información gerencial (**MIS**) Son aquellos que contemplan la información general de una empresa u organización;
- Sistemas de soporte a decisiones (**DSS**): Son aquellos que tienen en cuenta la información interna y externa para la toma de decisiones de la empresa;

Almacenamiento de datos

El almacenamiento de datos es el proceso que utiliza distintos medios de grabación para guardar los datos, utilizando los recursos de una PC u otros dispositivos. Las formas más utilizadas son el almacenamiento de objetos, de archivos y el almacenamiento en bloques, utilizando cada uno de ellos con distintos objetivos.

- Almacenamiento de archivos: Los datos son almacenados en carpetas y archivos, guardados en discos duros.
- Almacenamiento en bloques: Los datos son almacenados en bloques de tamaño uniforme. Este tipo de almacenamiento es más complejo y a la vez, ideal para los datos de acceso.
- Almacenamiento de objetos: Los datos son almacenados en forma de objetos con identificadores. Este tipo de almacenamiento es el que más se adapta para almacenar datos que no son necesarios editarlos.

Diferencia entre datos e información.

Todo sistema informático tiene como objetivo principal administrar y controlar los datos de manera eficiente para la toma de decisiones, por ello, dependiendo lo que consideremos como *dato* deberíamos tenerlo en cuenta para nuestro sistema. Esto nos lleva a definir *datos* como todo aquello que se puede registrar en nuestra base de datos, que siendo procesado nos devolverá información relevante.

Podemos decir que si nosotros tenemos un **dato** de temperatura (por ejemplo **32°C**) no nos dice nada por sí sólo, pero si obtenemos ese dato luego de consultar la temperatura de una ciudad determinada en determinado momento se convierte en información para conocer el clima de esa ciudad, a eso procesamiento de datos le llamamos **información**.

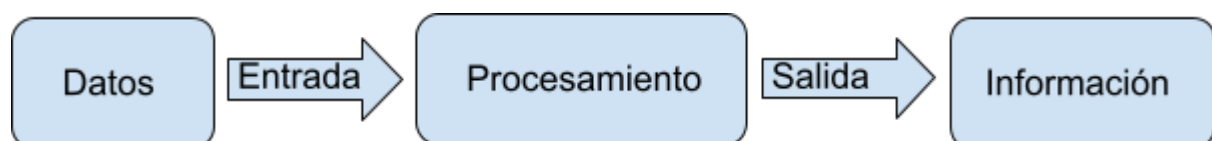


Figura 1: Procesamiento de datos

Bases de datos

Las bases de datos son los repositorios en los que se almacenarán los datos relacionados entre sí, almacenados sistemáticamente para su posterior procesamiento para entregar información al usuario. Las operaciones que se pueden realizar sobre las bases de datos las ejecuta el *motor de la base de datos* el cual es el encargado de recopilar e interpretar los comandos SQL teniendo como objetivo principal la creación, lectura, actualización y eliminación de datos.

Los motores de bases de datos están conformados por dos componentes principales que son un motor de almacenamiento y un procesador de consultas.

Tipos de bases de datos

Existen distintos tipos de bases de datos, las cuales pueden clasificarse de distintas formas, las cuales detallaremos a continuación.

Según su variabilidad

Bases de datos estáticas

Son aquellas que sus datos no pueden modificarse, están diseñadas especialmente para la lectura de sus datos. Por lo general se suelen utilizar para almacenar datos históricos, que no cambiarán, para realizar proyecciones estadísticas y ayudar a la toma de decisiones.

Bases de datos dinámicas

A diferencia de las bases de datos estáticas, éstas son aquellas que sus datos se pueden actualizar, ya sea agregando, modificando o eliminando los mismos durante el transcurso del tiempo.

Según el contenido

Éstas se dividen según la prioridad del contenido a analizar. Entre estas encontramos, las bases de datos **jerárquicas**, **las de red**, **las deductivas**, **las multidimensionales**, **las bibliográficas**, **las de texto completo** y **las relacionales** entre otras. Las últimas mencionadas son aquellas en las que nos basaremos en este trayecto.

Las **bases de datos relacionales** son aquellas que están compuestas por distintas tablas relacionadas entre sí mediante **claves primarias** y **claves foráneas**, en donde no pueden existir dos tablas con el mismo nombre y las mismas se componen de un conjunto de **campos** o **columnas** y **registros** o **filas**.

Las **claves primarias** son la clave principal de una tabla, en donde delimita que un valor **no se puede repetir**, ya que es único. Las **claves foráneas**, en cambio, se establecen en las

tablas hijas y sus valores se pueden repetir, pero **deben existir en la tabla padre y deben ser del mismo tipo**.

Como vemos en el siguiente ejemplo, la tabla **Cientes** está compuesta por las columnas *Id_Cliente* - clave principal- *Nombre*, *Apellido*, *Id_TipoDocumento* -clave foránea de la tabla TiposDocumentos- y *NroDocumento*. Y la tabla **TiposDocumentos** compuesta por las columnas *Id_TipoDocumento* -clave principal- y *Nombre*.

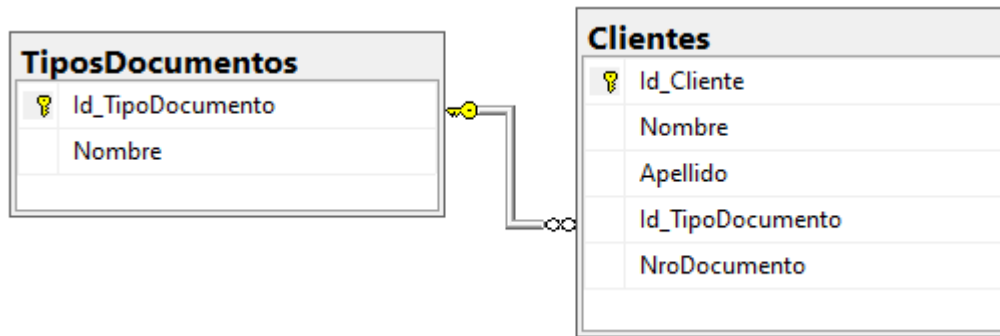


Figura 2 : Ejemplo relación entre tablas

Las empresas proporcionan los motores de bases de datos bajo un nombre comercial, algunos de ellos son:



SQLite: Es una biblioteca que implementa un Sistema gestor de bases de datos (SGBD), la cual permite transacciones sin necesidad de un servidor dedicado o de realizar configuraciones previas;



MySQL: De características multiusuario y multi hilo, es utilizado en la mayoría de los sistemas web en la actualidad y el más popular a la hora de realizar aplicaciones como software libre;



Oracle: Diseñado para gestionar grandes volúmenes de contenidos en un repositorio único para reducir los riesgos de pérdida de información.

Una desventaja que presenta es su coste, aunque cuenta con una versión gratuita, sus funcionalidades no son las mismas que las ediciones de pago;



SQL Server: Multiusuarios capaz de suministrar grandes cantidades de datos a muchos usuarios de manera simultánea. Está basado en el lenguaje Transact-SQL.

Al igual que Oracle, es pago y también cuenta con versiones gratuitas con funcionalidades acotadas.



MariaDB: Siendo una derivación de MySQL, MariaDB cuenta con la mayoría de características de este, incluyendo diversas extensiones además de mantener la filosofía *open source*.

Las bases de datos **no relacionales (NoSQL)** son aquellas que nos permiten almacenar información en donde las bases de datos relacionales generan ciertas problemáticas en cuanto a escalabilidad y rendimiento, además que no requieren estructuras de datos fijas (tablas). Se utilizan en entornos distribuidos, los cuales deben tener disponibilidad y operatividad constantes, ya que gestionan grandes volúmenes de datos.

Dentro de los motores de bases de datos no relacionales encontramos algunos como:



MongoDB: Es orientado a ficheros que almacena la información en estructuras BSON (basado en el término Binary JSON, es una representación binaria de mapas y datos) que permite la escalabilidad y su integración con facilidad;



Redis: Está basado en la filosofía de almacenamiento clave-valor el cual se podría ver como un vector que almacena datos de distintos tipos como enteros, decimales, cadenas, listas, etc, siendo utilizado principalmente para el almacenamiento en memoria caché y administración de sesiones.

Sistemas gestores de bases de datos.

Un sistema gestor de base de datos (SGBD o DBMS por sus siglas en inglés) es un programa o conjunto de programas que posibilitan la creación y administración de las bases de datos, actuando como intermediario entre los usuarios, las aplicaciones y la base de datos.

Los **DBMS** proporcionan los servicios necesarios para:

Definir la base de datos;

La inserción, actualización, eliminación y consulta de los datos;

El acceso controlado a la base de datos;

Creación de copias de seguridad;

Exportación e importación de datos;

Creación de procedimientos almacenados;

Creación de vistas; Etc.

Diseño de bases de datos

El proceso de diseño de base de datos es un proceso complejo que se puede descomponer en distintas etapas, resolviendo cada nivel independientemente del resto. Esto nos permite obtener una base de datos capaz de satisfacer los requisitos informacionales de un sistema informático.

Dichas etapas se podrían nombrar como **diseño conceptual, diseño lógico y diseño físico**.

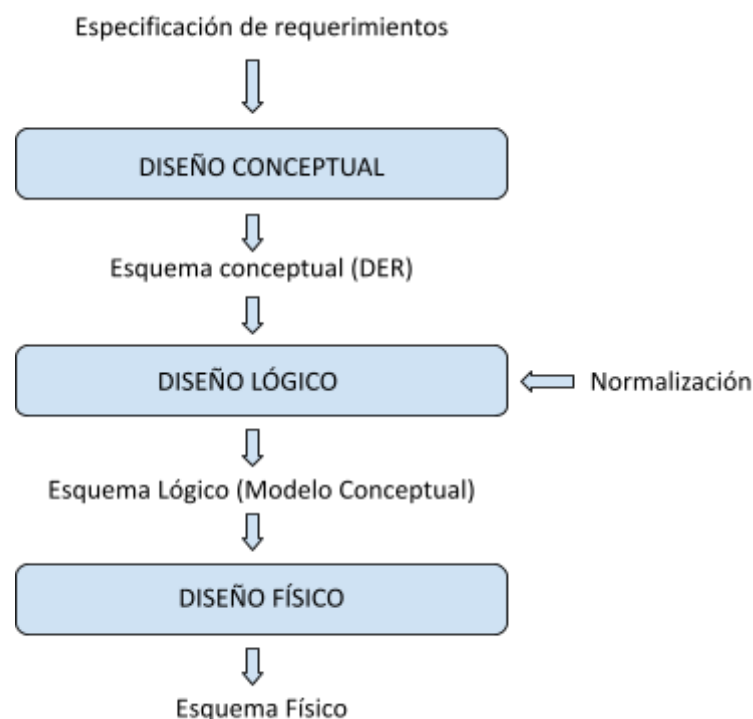


Figura 3: Diseño de bases de datos

Diseño conceptual

El **diseño** conceptual surge a partir de las especificaciones de requisitos del usuario, de allí se obtiene el **esquema** conceptual de la base de datos. Este esquema define la estructura de la base de datos, independientemente de la tecnología o el sistema de gestión de base de datos que se utilice. Distinto al esquema, el **modelo** conceptual es el lenguaje que se utilizará para describir el esquema conceptual.

El objetivo de este diseño es describir la información de la base de datos en sí y no de cómo se almacenará la información en ésta.

Diagrama Entidad Relación

El DER representa la relación que tienen las entidades entre sí. Una entidad se puede decir que es una unidad que tiene atributos y contiene información que conforma una base de datos, siendo una representación de objeto, persona, cosa, etc.

En el DER, la entidad se representa con un rectángulo al que conectamos a sus atributos y a las relaciones mediante una línea.

Cada relación se representa con un rombo y así, dos o más entidades se pueden conectar mediante una misma relación en común indicando la *cardinalidad*. En cambio los atributos se representan mediante un círculo.

La *cardinalidad* o *multiplicidad* indica la cantidad de elementos de una entidad que se relacionan con una instancia de otra entidad.

Estas pueden ser:

- **Uno a Uno** (1:1) se da cuando un elemento de una entidad se puede relacionar solamente con un solo registro de otra entidad y viceversa.
- **Uno a Muchos** (1:M) se da cuando un registro de una entidad A se puede relacionar con cero o muchos registros de otra entidad B y cada registro de la entidad B se relaciona con un sólo registro de la entidad A.
- **Muchos a Muchos** (N:M) se da cuando un registro de una entidad se relaciona con cero o varios registros de otra entidad

Si continuamos con el desarrollo de nuestra base de datos, podemos decir que antes de llegar a tenerla, tuvimos que crear nuestro DER, el cual se ve de la siguiente manera:

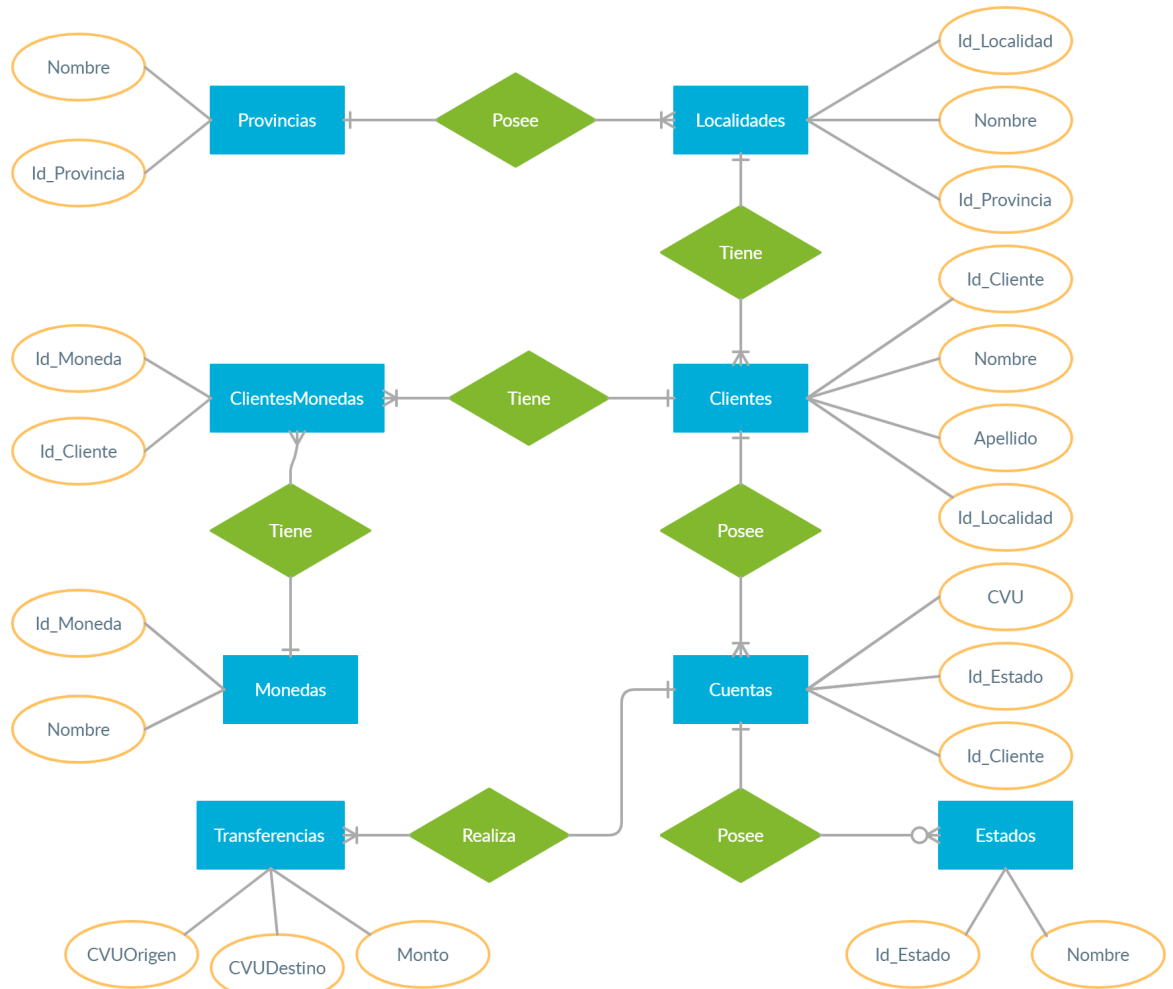


Figura 4: Diagrama Entidad Relación

Diseño lógico

Este diseño es parte del esquema conceptual y da como resultado la estructura de la base de datos, definiendo las estructuras de datos que pueden soportar los sistemas gestores de bases de datos. A este esquema se lo llama **esquema lógico**.

El *modelo* lógico es el lenguaje que determina el esquema lógico.

En cuanto al diseño lógico, se determina según el SGBD que se vaya a utilizar sin depender del producto concreto.

Luego de obtener nuestro esquema conceptual y teniendo en cuenta los puntos anteriormente mencionados, podremos obtener nuestro diseño lógico, el cual se verá de la siguiente manera:

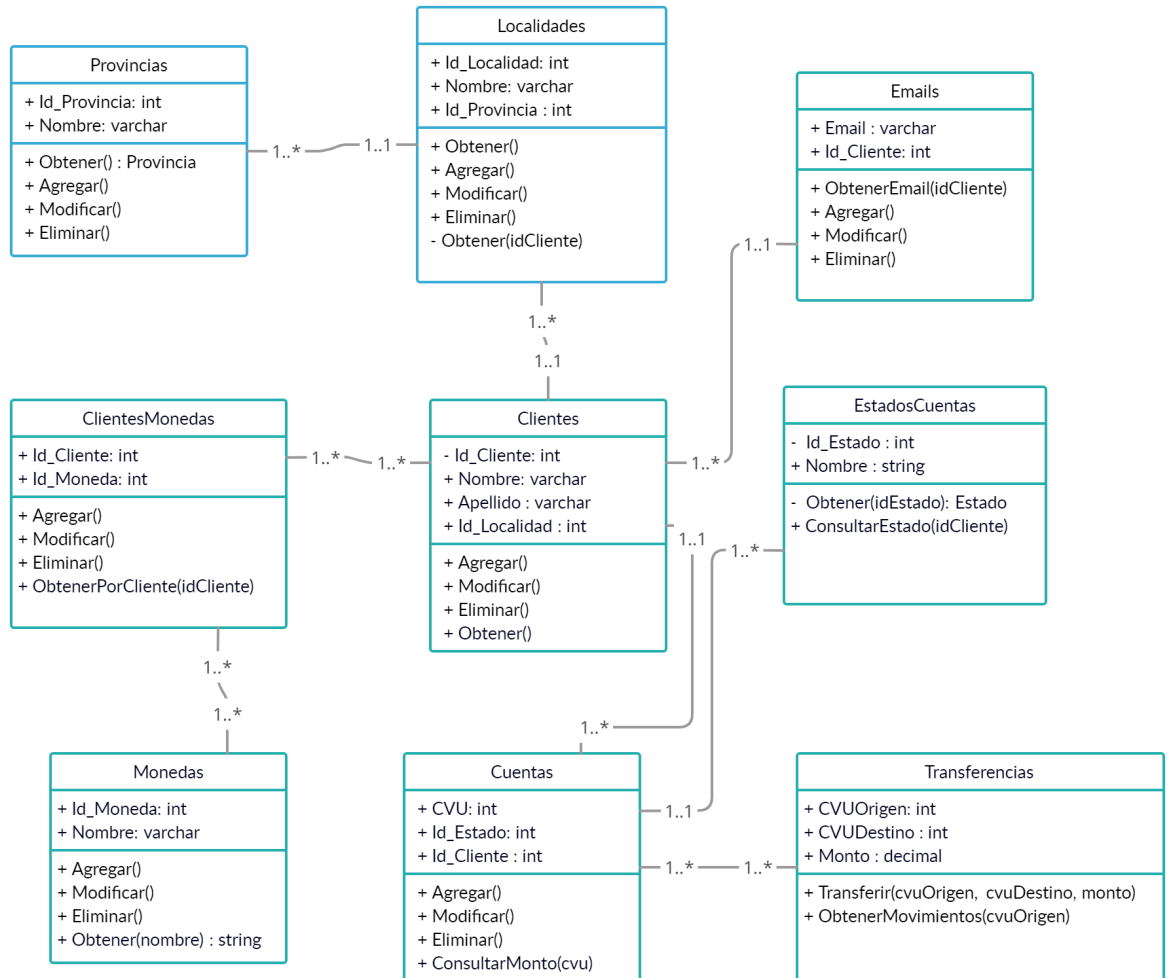


Figura 5: Diagrama de clases

Antes de transformar nuestro DER a un modelo relacional, debemos tener en cuenta algunos conceptos fundamentales como qué son y qué estructura tienen las tablas, las reglas de normalización y las 12 reglas de Codd, sin dejar de lado que:

- Todas las entidades se convierten en tablas;
- Los atributos de las entidades se convierten en las columnas de la tabla que corresponde;
- El identificador de la entidad se convierte en la clave primaria de la tabla;
- En todas las relaciones que sean N:M se creará una tabla en donde tendrá como clave primaria las claves primarias de las entidades asociadas;
- En las relaciones 1:N se crearán claves foráneas de la tabla que tiene como cardinalidad (tabla A) 1 en la tabla que tiene como cardinalidad (tabla B) N. Esto quiere decir que la clave primaria de la tabla A pasa a la tabla B relacionando ambas tablas;
- En las relaciones 1:1 las dos tablas deberán fusionarse en una sola y definir un identificador como clave primaria común.

Tablas

Una tabla de una base de datos es un objeto donde se guardan los datos organizados con un formato de filas y columnas. Cada fila representa un *registro* (el cual puede también almacenar valores nulos) y cada columna representa un *campo* (dentro del registro) el cual debe ser único y tener asociado un tipo de datos.

Estructura de almacenamiento de datos

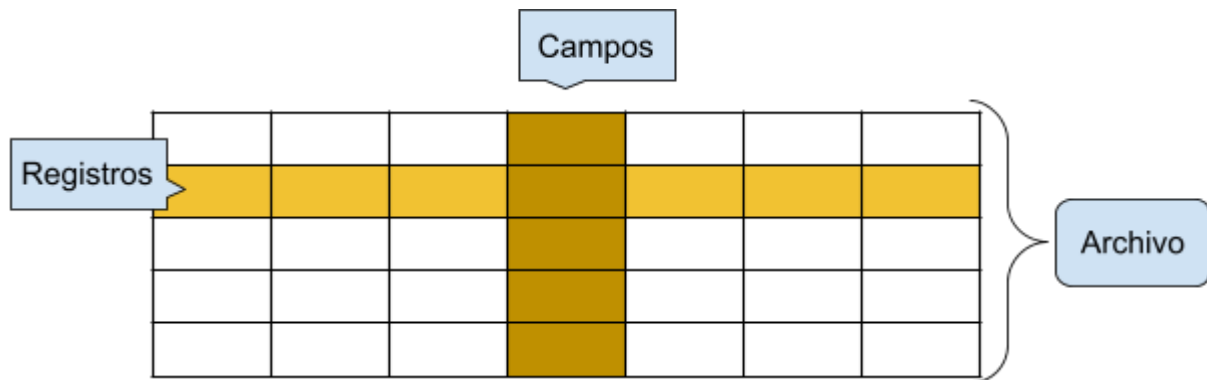


Figura 6: Estructura de una tabla

Las 12 reglas de Codd

Las 12 reglas de Codd (numeradas del 0 al 12) son un conjunto de normas establecidas por Edgar Codd, para que una base de datos del modelo relacional se considere verdaderamente relacional, teniendo en cuenta que, a mayor cantidad de reglas cumplidas, *más relacional* será la base de datos.

Regla 0 - Fundación

El sistema debe utilizar su estructura relacional solamente para gestionar bases de datos.

Regla 1 - Información

Toda la información debe estar representada unidireccionalmente por los valores en posiciones dentro de las filas, dentro de las tablas. Esto quiere decir que los valores solamente pueden estar representados solamente de una forma: Con valores en las tablas, es decir, a nivel lógico.

Regla 2 - Acceso garantizado

Todos los datos deben ser accesibles, debiendo ser cada valor escalar individual lógicamente direccionable especificando el nombre de la tabla, la columna que lo contiene y la clave primaria.

Regla 3 - Tratamiento sistemático de los valores nulos

El DBMS debe tener la capacidad de manejar valores nulos, reconociendo como distinto este valor a cualquier otro, independientemente del tipo de datos que sea la columna.

Regla 4 - Catálogo en línea relacional

El catálogo en línea es el diccionario de datos, el cual se debe poder consultar utilizando las mismas técnicas para los datos que para los metadatos. También da acceso a la estructura de la base de datos, debiendo ser accesible para los usuarios autorizados.

Regla 5 - Sublenguaje de datos completo

Debe existir, por lo menos un lenguaje capaz de realizar todas las funciones del DBMS, sin quedar excluida ninguna función. También pueden existir otros lenguajes para hacer ciertas tareas, pero también se deben poder llevar a cabo con el lenguaje principal.

Regla 6 - Actualización de vistas

Todas las vistas deben mostrar información actualizada y los datos no pueden ser distintos entre las vistas y las tablas base.

Regla 7 - Inserciones, modificaciones y eliminaciones de alto nivel

El sistema debe permitir la manipulación de datos de alto nivel, es decir sobre conjuntos de registros. Esto quiere decir que también se podrán realizar modificaciones, inserciones y eliminaciones sobre conjuntos de tuplas y/o tablas al mismo tiempo.

Regla 8 - Independencia física de los datos

Los cambios físicos que se realicen en la base de datos no afecta a las aplicaciones ni a los esquemas lógicos.

Regla 9 - Independencia lógica de los datos

Los cambios que se realicen sobre el esquema lógico de la base de datos no afectan al resto de los esquemas. Esto quiere decir que al realizar cambios sobre los nombres de las tablas o de las columnas o modificamos la información de los registros, las aplicaciones externas no se verán afectadas.

Regla 10 - Independencia de integridad

Las reglas de integridad deben ser gestionadas y almacenadas por el DBMS, posibilitando los cambios necesarios sin afectar a las aplicaciones existentes.

Regla 11 - Independencia de distribución

Las bases de datos pueden gestionarse o almacenarse de forma distribuida en distintos servidores, sin afectar al uso de la misma ni a la programación del usuario, debiendo mantenerse el esquema lógico de la base de datos ya sea distribuida o no.

Regla 12 - No subversión

No se permitirán lenguajes o formas de acceso que salteen las reglas anteriores.

Normalización

La normalización es la técnica de estandarización y validación de datos que se utiliza para diseñar las tablas y establecer las relaciones entre ellas. Esto nos permite una mayor organización para eliminar la redundancia de datos y proteger la integridad de los mismos.

Sin la normalización, tendríamos datos redundantes los cuales traen aparejados problemas de mantenimiento, además de ocupar espacio en el disco duro, como también, dependencias incoherentes.

Las formas normales se pueden clasificar en:

Primera Forma Normal (1FN)

Una tabla está en primera forma normal cuando todos los datos son atómicos (es decir, cada atributo tiene su propia celda) y todas las columnas contienen el mismo tipo de datos. Sirve para eliminar los grupos repetidos.

Como vemos en el siguiente ejemplo, la primera forma normal no se cumple por el atributo *Email*.

Id_Cliente	Nombre	Apellido	Email	Provincia	Localidad	CVU	EstadoCuenta
1	JORGE	GOMEZ	jorge@test.com; jgomez@test.com	SALTA	EL JARDIN	122334455	HABILITADA
2	LUISA	PEREZ	luisaperez@test.com	CÓRDOBA	CARLOS PAZ	212223334	HABILITADA
3	MARTHA	GONZALES	martha@test.com	TUCUMAN	TAFI VIEJO	312233445	HABILITADA
4	JUAN	PEREZ	juan@test.com	CÓRDOBA	LA FALDA	185274196	HABILITADA
5	PASCUAL	VARGAS	vargasp@test.com; pvargas@hotmail.com	CÓRDOBA	LA FALDA	321654789	HABILITADA

Lo podemos solucionar creando un registro por cada *Email* que queremos que sea indivisible o eliminar el atributo que no cumple dicha condición.

Para cumplir con la primera forma normal duplicando registros con valores repetidos convertimos el atributo *Email* en PK.

Clientes

Id_Cliente	Nombre	Apellido	Email	Provincia	Localidad	CVU	EstadoCuenta
1	JORGE	GOMEZ	jgomez@test.com	SALTA	EL JARDIN	122334455	HABILITADA
1	JORGE	GOMEZ	jorge@test.com	SALTA	EL JARDIN	122334455	HABILITADA
2	LUISA	PEREZ	luisaperez@test.com	CÓRDOBA	CARLOS PAZ	212223334	HABILITADA
3	MARTHA	GONZALES	martha@test.com	TUCUMAN	TAFI VIEJO	312233445	HABILITADA
4	JUAN	PEREZ	juan@test.com	CÓRDOBA	LA FALDA	185274196	HABILITADA
5	PASCUAL	VARGAS	pvargas@hotmail.com	CÓRDOBA	LA FALDA	321654789	HABILITADA
5	PASCUAL	VARGAS	vargasp@test.com	CÓRDOBA	LA FALDA	321654789	HABILITADA

Para cumplir con la primera forma normal eliminando el atributo *Email* tendremos que crear una tabla *Emails* que contenga las columnas *Email (PK)* e *Id_Cliente*.

Cientes

Id_Cliente	Nombre	Apellido	Provincia	Localidad	CVU	EstadoCuenta
1	JORGE	GOMEZ	SALTA	EL JARDIN	122334455	HABILITADA
2	LUISA	PEREZ	CÓRDOBA	CARLOS PAZ	212223334	HABILITADA
3	MARTHA	GONZALES	TUCUMAN	TAFI VIEJO	312233445	HABILITADA
4	JUAN	PEREZ	CÓRDOBA	LA FALDA	185274196	HABILITADA
5	PASCUAL	VARGAS	CÓRDOBA	LA FALDA	321654789	HABILITADA

Emails

Email	Id_Cliente
jorge@test.com	1
jgomez@test.com	1
luisaperez@test.com	2
martha@test.com	3
juan@test.com	4
vargasp@test.com	5
pvargas@hotmail.com	5

Segunda Forma Normal (2FN)

Para estar en la segunda forma normal, a las condiciones de la primera forma normal se le debe agregar que los atributos que no forman parte de ninguna clave, su funcionamiento depende de la clave primaria. Ésta sirve para eliminar los datos redundantes.

Como se puede ver a continuación, al crear las tablas *Provincias* y *Localidades* y eliminando el atributo *Provincia* de la tabla *Cientes*, obtendremos la segunda forma normal, ya que obtendremos la *Provincia* desde la tabla *Localidades* y la *Localidad* desde la tabla *Cientes*, Además, creamos la tabla *EstadosCuentas* con *Id_Estado (PK)* y *Nombre*.

Al aplicar la segunda forma normal nos quedarían las tablas *Cientes*, *Provincias*, *Localidades* y *EstadosCuentas*.

Provincias

Id_Provincia	Nombre
1	CÓRDOBA
2	SALTA
3	TUCUMAN

Localidades

Id_Localidad	Nombre	Id_Provincia
1	EL JARDÍN	2
2	CARLOS PAZ	1
3	TAFI VIEJO	3
4	LA FALDA	1

EstadosCuentas

Id_Estado	Nombre
1	HABILITADA
2	DESHABILITADA

Cientes

Id_Cliente	Nombre	Apellido	Id_Localidad	CVU	Id_EstadoCuenta
1	JORGE	GOMEZ	1	122334455	1
2	LUISA	PEREZ	2	212223334	1
3	MARTHA	GONZALES	3	312233445	1
4	JUAN	PEREZ	4	185274196	1
5	PASCUAL	VARGAS	4	321654789	1

Tercera Forma Normal (3FN)

Para cumplirse la tercera forma normal, a las dos condiciones anteriores hay que agregarle que los atributos que no son clave no pueden depender de manera transitiva de una clave candidata. Aquí se deben eliminar las columnas que no dependen de la clave principal. Ésta forma normal es considerada como el estándar a cumplir por los esquemas relacionales.

En el siguiente ejemplo, agregamos la tabla *Cuentas* la cual tiene los atributos *CVU* (*PK*), *Id_Estado* e *Id_Cliente* y en nuestra tabla *Cientes* eliminaremos los atributos *CVU* e *Id_EstadoCuenta*.

EstadosCuentas

Id_Estado	Nombre
1	HABILITADA
2	DESHABILITADA

Provincias

Id_Provincia	Nombre
1	CÓRDOBA
2	SALTA
3	TUCUMAN

Localidades

Id_Localidad	Nombre	Id_Provincia
1	EL JARDÍN	2
2	CARLOS PAZ	1
3	TAFI VIEJO	3
4	LA FALDA	1

Cuentas

CVU	Id_Estado	Id_Cliente
122334455	1	1
212223334	1	2
312233445	1	3
185274196	1	4
321654789	1	5

Cientes

Id_Cliente	Nombre	Apellido	Id_Localidad
1	JORGE	GOMEZ	1
2	LUISA	PEREZ	2
3	MARTHA	GONZALES	3
4	JUAN	PEREZ	4
5	PASCUAL	VARGAS	4

Cuarta Forma Normal (4FN)

Antes de cumplir la cuarta forma normal, nuestra tabla debe cumplir con la tercera forma normal y que no posea dependencias multivaluadas. Las dependencias multivaluadas son aquellas donde la existencia de dos o más relaciones N:M (Muchos a muchos) causa redundancia. Para evitar eso, utilizamos la 4FN.

Por ejemplo, si habilitamos los movimientos que tiene cada cliente con uno o más tipos de moneda, crearemos la tabla *Monedas*, la cual tendrá los atributos *Id_Moneda* y *Nombre* y la tabla *CientesMonedas* que tendrá las columnas *Id_Cliente* e *Id_Moneda*, en donde asignaremos con que tipos de monedas puede operar cada cliente.

Monedas

Id_Moneda	Nombre
1	PESO
2	DOLAR
3	BITCOIN

CientesMonedas

Id_Cliente	Id_Moneda
1	1
1	2
2	1
2	3
3	1
3	2
4	1
5	1
5	2
5	3

Quinta Forma Normal (5FN o forma normal de Proyección-Unión)

Una relación se encuentra en 5FN si se encuentra en 4FN y además, las dependencias existentes son las dependencias de *Join* con sus proyecciones (esto ocurre cuando podemos obtener la tabla original por medio de la unión de sus proyecciones), relacionándose entre sí a través de la clave primaria o alguna de sus claves alternativas.

Diseño físico

El diseño físico es parte del esquema lógico y su resultado es el esquema físico, el cual describe la implementación de las estructuras de almacenamiento y los métodos para tener un acceso eficiente a los datos. Este, además, depende del SGBD que se va a utilizar y el esquema físico se expresa mediante su LDD (lenguaje de definición de datos).

Se debe tener un conocimiento vasto del SGBD donde se implementará la base de datos teniendo en cuenta varios aspectos como los tipos de datos, las restricciones, los índices disponibles y el lenguaje entre los más importantes.

Los índices son estructuras asociadas a una tabla (o vista) que sirve para hacer más veloz la recuperación de las filas de dicha tabla. Además de contar con las claves que se generan a partir de una o muchas columnas de la tabla.

Las restricciones sirven para establecer las reglas de una tabla, especificando qué tipos de datos se pueden añadir, evitando que se inserten datos incorrectos.

Para definir el diseño físico, deberíamos aclarar antes los tipos de datos de SQL. Los tipos de datos determinan que valor puede contener una columna, ya sea números enteros, decimales, cadenas de caracteres, fechas, etc.

Algunos de ellos son:

Tipo de Datos	Longitud	Descripción
BIT	1 byte	Valores Si/No - True/False - Verdadero/Falso
DATETIME	8 bytes	Fecha u hora entre los años 100 y 9999.
DOUBLE	8 bytes	Un valor en punto flotante de precisión doble con un rango de $-1.79769313486232 \times 10^{308}$ a $-4.94065645841247 \times 10^{-324}$ para valores negativos siendo para valores positivos entre $4.94065645841247 \times 10^{-324}$ a $1.79769313486232 \times 10^{308}$ y 0.
LONG	4 bytes	Valor entero largo entre -2,147,483,648 y 2,147,483,647.
LONGTEXT	1 byte por carácter	De cero a un máximo de 1.2 gigabytes.
TEXT	1 byte por carácter	De cero a 255 caracteres.

¿Qué es SQL?

SQL (Structured query language o lenguaje de consulta estructurada) es un lenguaje informático de alto nivel que sirve para gestionar y organizar los datos de una base de datos relacional.

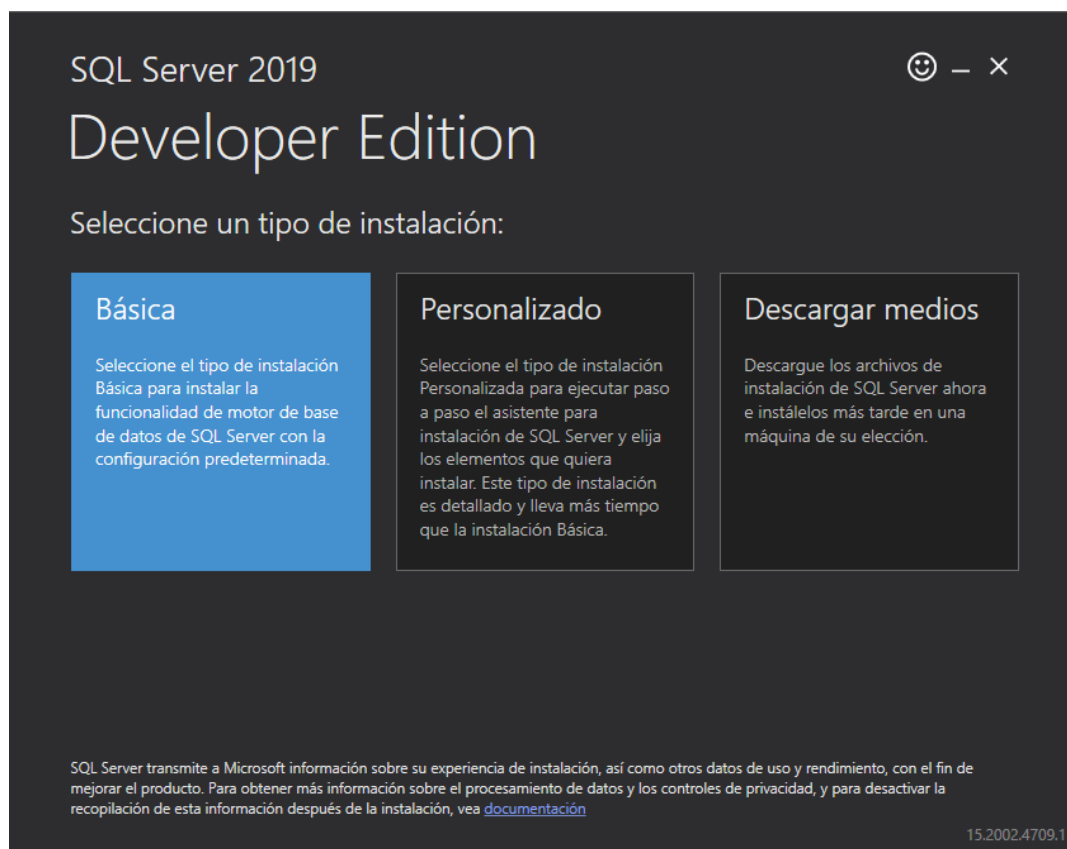
Debido a su base teórica y la orientación a la manipulación de registros en conjunto, permite un alto rendimiento en orientación a objeto, teniendo como principales características el **lenguaje de definición de datos (LDD)** el cual permite definir los esquemas de relación; el **lenguaje de manipulación de datos (LMD)** el cual se basa en álgebra relacional y en cálculo de tuplas; el **lenguaje de integridad** el cual especifica las restricciones de integridad de los datos almacenados; **creación de vistas** el cual permite la posibilidad de crear vistas; el **control de transacciones** para especificar cuándo comienzan y terminan las transacciones; **Seguridad** para especificar control de acceso; posibilidad de **incorporar SQL en lenguajes de programación**.

Instalación SQL Server

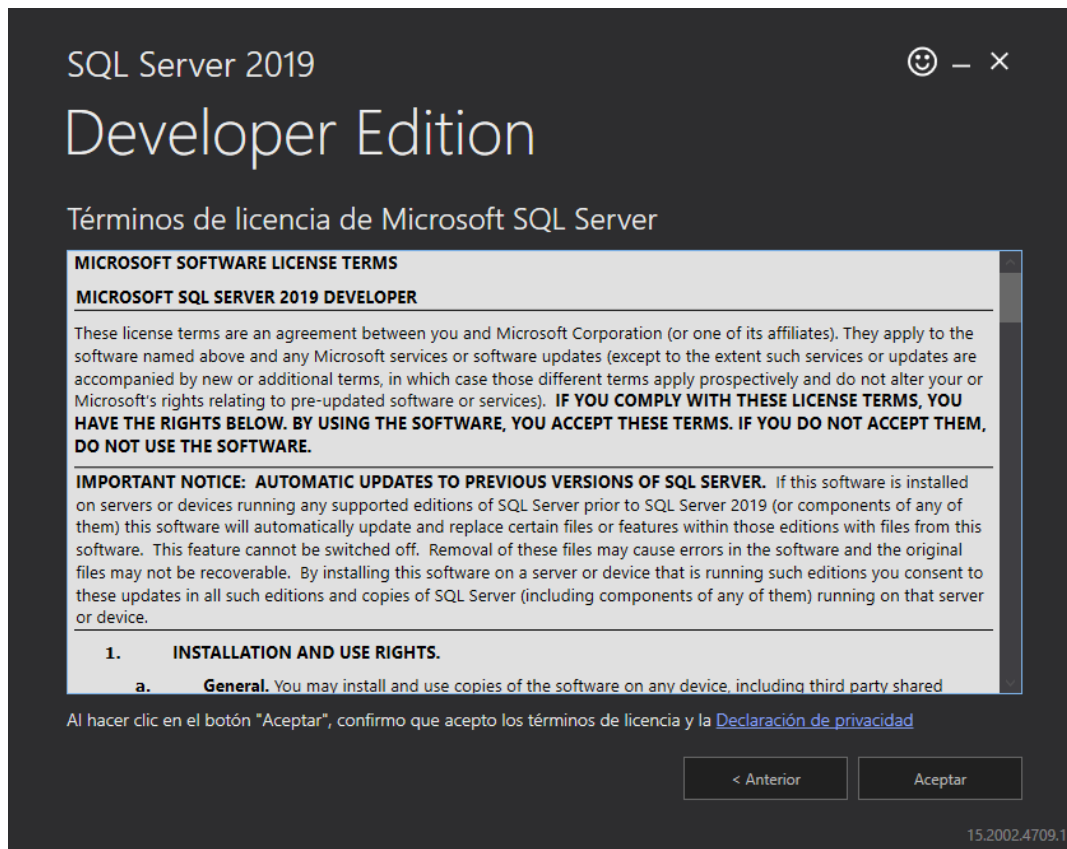
Una vez aclarados los conceptos anteriores, podremos poner manos a la obra.

La creación y gestión de la base de datos la llevaremos a cabo con el motor de bases de datos *Microsoft SQL Server 2019* edición *Developer* (Descarga de instalador <https://go.microsoft.com/fwlink/?linkid=866662>) y el SGBD será *Microsoft SQL Server Management Studio v 18.6* (Descarga de instalador <https://aka.ms/ssmsfullsetup>).

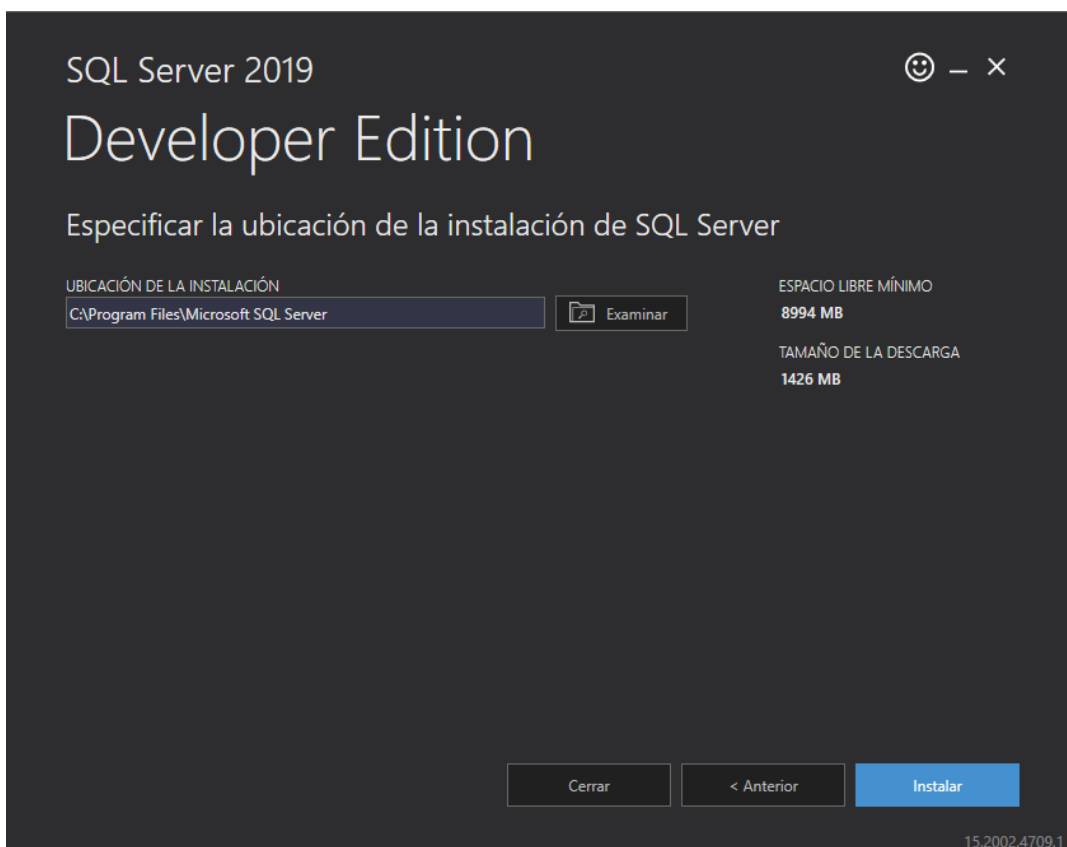
Primero llevaremos a cabo la instalación del motor de bases de datos siguiendo los pasos a continuación detallados:



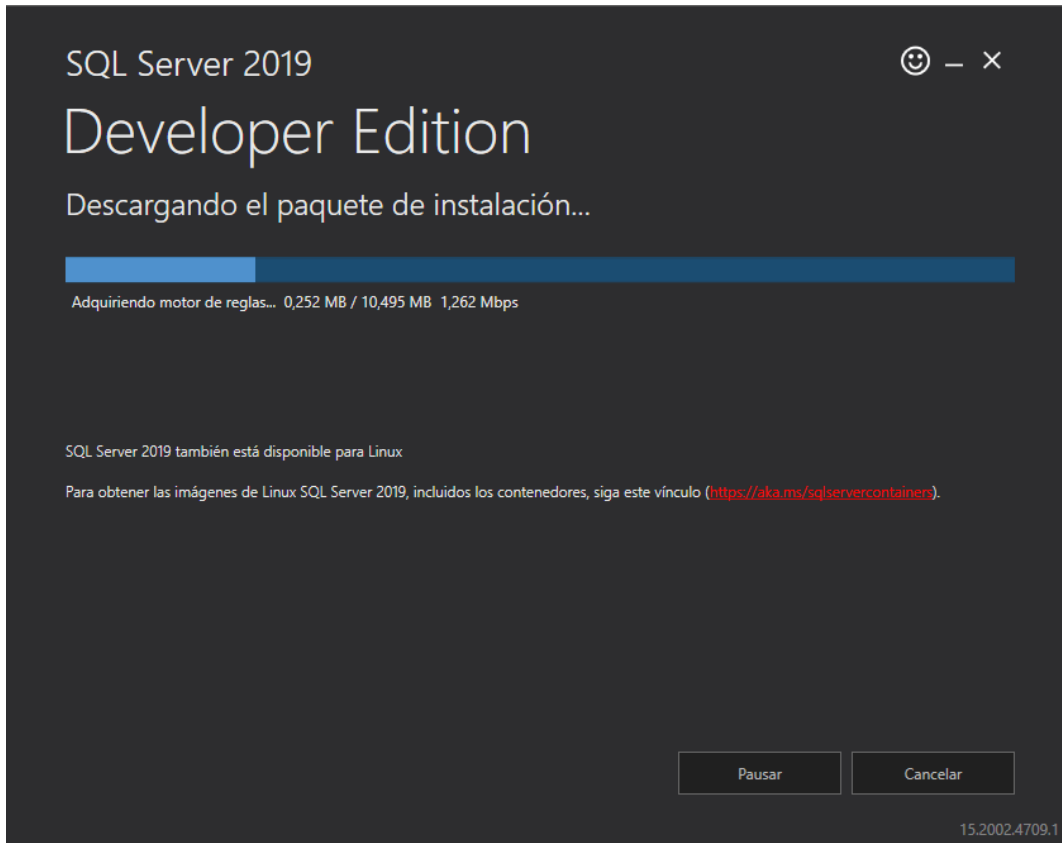
*Paso 1: Ejecutamos el instalador del motor de base de datos SQL Server.
Seleccionamos la opción “Básica”.*



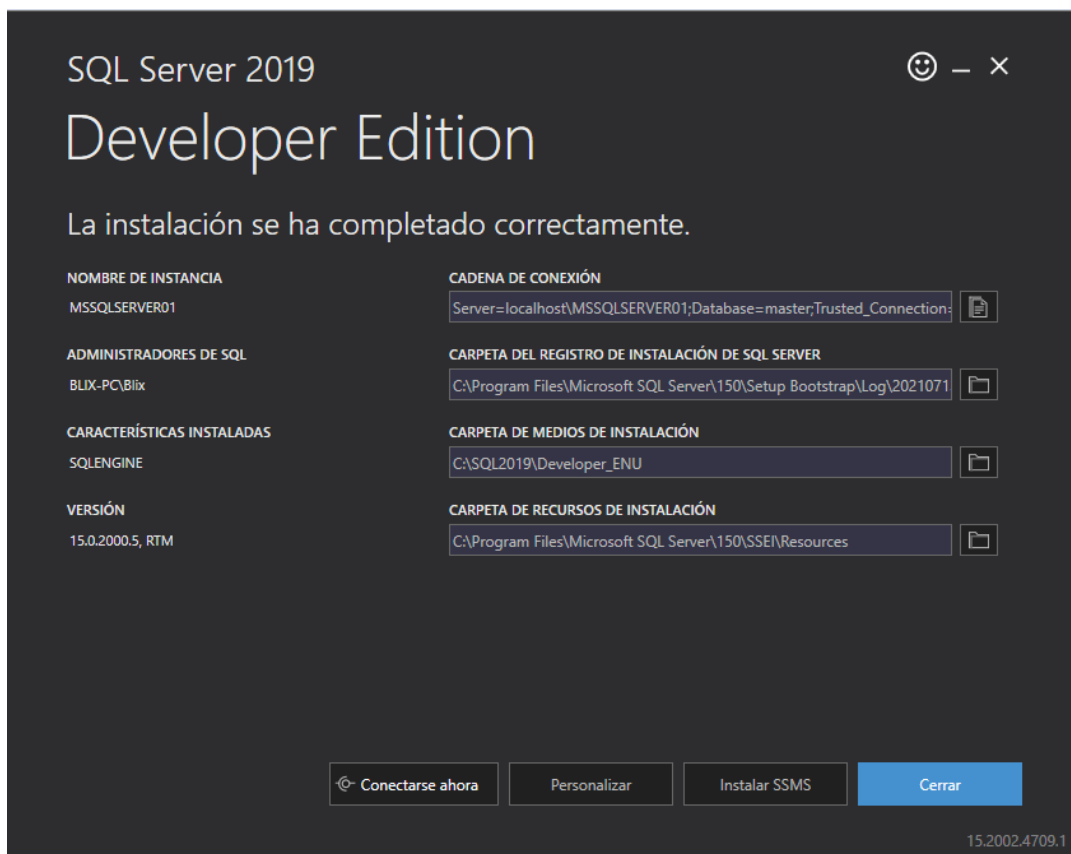
Paso 2: Hacemos click en “Aceptar”.



Paso 3: Seleccionamos la ubicación donde se instalará y hacemos click en “Instalar”.



Paso 4: Instalación de motor de bases de datos.



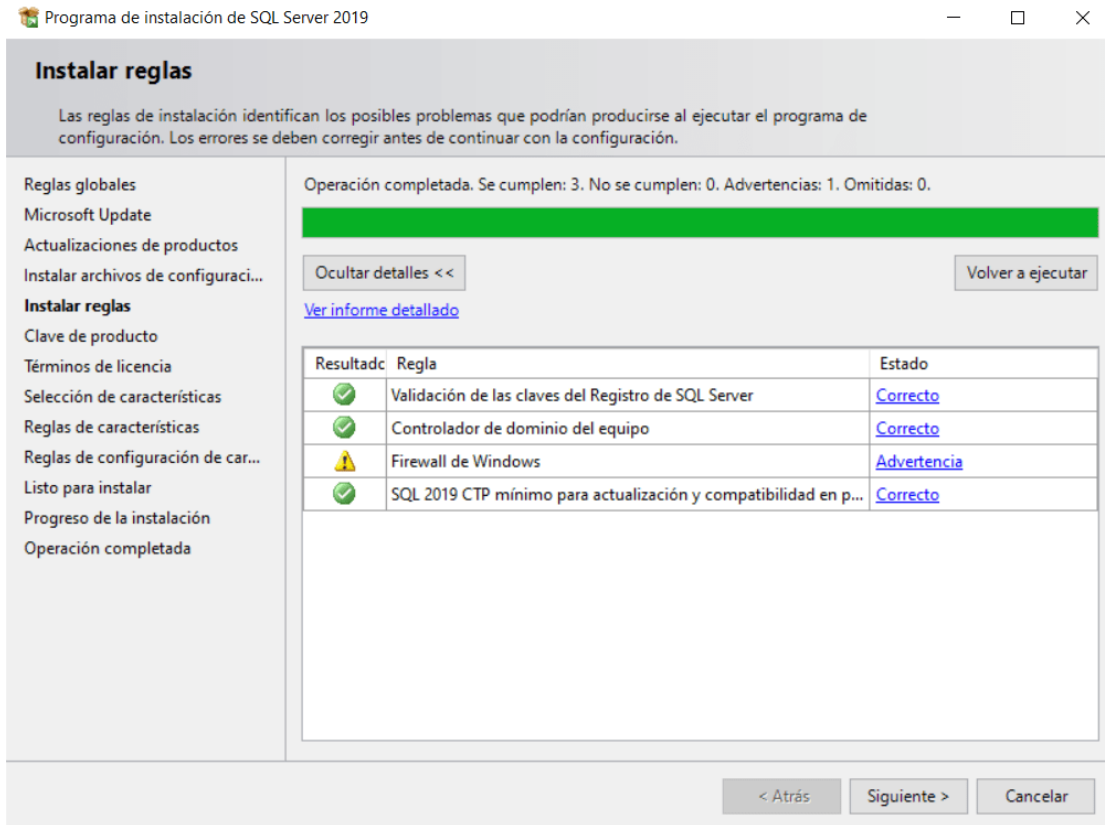
Paso 5: Una vez instalado el motor, obtendremos los detalles de nuestro motor luego hacemos click en “Cerrar” para finalizar la instalación.

Instalación SQL Server Management Studio

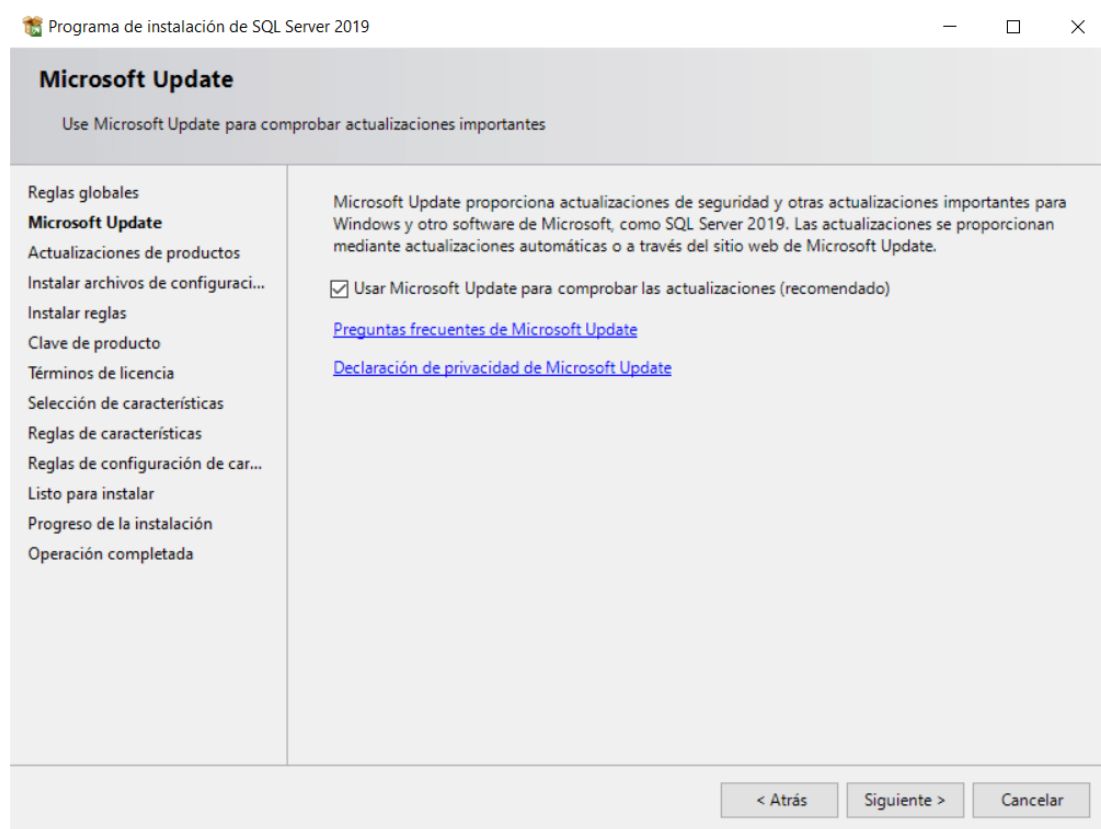
Una vez instalado el motor de base de datos, continuamos instalando el gestor siguiendo los pasos que se muestran a continuación:



*Paso 1: Seleccionamos **Nueva instalación independiente de Sql Server**.*



*Paso 2: Una vez realizada la prueba, hacemos click en **Siguiente**.*



Paso 3: Hacemos click en **Siguiente**.

Programa de instalación de SQL Server 2019

Clave de producto

Especifique la edición de SQL Server 2019 que se instalará.

Reglas globales

Microsoft Update

Actualizaciones de productos

Instalar archivos de configuraci...

Instalar reglas

Clave de producto

Términos de licencia

Selección de características

Reglas de características

Reglas de configuración de car...

Listo para instalar

Progreso de la instalación

Operación completada

Para validar esta instancia de SQL Server 2019, especifique la clave de 25 caracteres del certificado de autenticidad o del paquete del producto de Microsoft. También puede especificar una edición gratuita de SQL Server: Developer, Evaluation o Express. La edición Evaluation contiene el conjunto más completo de características de SQL Server, documentadas en los Libros en pantalla de SQL Server, y se activa con un período de expiración de 180 días. La edición Developer no tiene fecha de expiración y tiene el mismo conjunto de características que Evaluation, pero solo tiene licencia para el desarrollo de aplicaciones de bases de datos que no sean de producción. Para actualizar de una edición instalada a otra, ejecute el Asistente para actualizar la edición.

☒ Especifique una edición gratuita:

Developer

☐ Escriba la clave de producto:

- - - -

< Atrás Siguiente > Cancelar

Paso 4: Seleccionamos la edición **Developer** y hacemos click en **Siguiente**.

Términos de licencia

Para instalar SQL Server 2019, debe aceptar los Términos de licencia del software de Microsoft.

Reglas globales
Microsoft Update
Actualizaciones de productos
Instalar archivos de configuraci...
Instalar reglas
Clave de producto
Términos de licencia
Selección de características
Reglas de características
Reglas de configuración de car...
Listo para instalar
Progreso de la instalación
Operación completada

TÉRMINOS DE LICENCIA DEL SOFTWARE DE MICROSOFT

MICROSOFT SQL SERVER 2019 DEVELOPER

Los presentes términos de licencia constituyen un contrato entre usted y Microsoft Corporation (o una de las filiales de su grupo). Se aplican al software mencionado arriba y a cualquier servicio o actualización de software de Microsoft (excepto en la medida en que tales servicios o actualizaciones estén acompañados por términos nuevos o adicionales, en cuyo caso esos términos diferentes se aplican prospectivamente y no modifican sus derechos o los de Microsoft en relación con el software o los servicios previos a la actualización). **SI USTED CUMPLE LOS PRESENTES TÉRMINOS DE ESTA LICENCIA, DISPONDRÁ DE LOS DERECHOS QUE A CONTINUACIÓN SE DESCRIBEN. AL HACER USO DEL SOFTWARE, USTED ESTARÁ ACEPTANDO ESTOS TÉRMINOS. SI NO LOS ACEPTA, NO USE EL SOFTWARE.**

AVISO IMPORTANTE: ACTUALIZACIONES AUTOMÁTICAS DE LAS VERSIONES

☒ Acepto los términos de licencia y [Declaración de privacidad](#)

SQL Server transmite a Microsoft información sobre su experiencia de instalación, así como otros datos de uso y rendimiento, con el fin de mejorar el producto. Para obtener más información sobre el procesamiento de datos y los controles de privacidad, y también para desactivar la recopilación de esta información después de la instalación, vea la [documentación](#).

[Copiar](#) [Imprimir](#)

< Atrás Siguiente > Cancelar

Paso 5: Tildamos la casilla de **Términos y condiciones** y hacemos click en **Siguiente**.

Selección de características

Seleccione las características de Developer que desea instalar.

Reglas globales
Microsoft Update
Actualizaciones de productos
Instalar archivos de configuraci...
Instalar reglas
Clave de producto
Términos de licencia
Selección de características
Reglas de características
Configuración de instancia
Configuración del servidor
Configuración del Motor de ba...
Reglas de configuración de car...
Listo para instalar
Progreso de la instalación
Operación completada

¿Está buscando Reporting Services? [Descargar de la Web](#)

Características:	Descripción de la característica:
Características de instancia	
<input checked="" type="checkbox"/> Servicios de Motor de base de datos	La configuración y operación de cada característica de una instancia de SQL Server tiene lugar de forma...
<input type="checkbox"/> Replicación de SQL Server	
<input type="checkbox"/> Machine Learning Services y extensiones de lenguaje	
<input type="checkbox"/> R	
<input type="checkbox"/> Python	
<input type="checkbox"/> Java	
<input type="checkbox"/> Extracciones de texto completo y semánticas de búsqueda	
<input type="checkbox"/> Data Quality Services	
<input type="checkbox"/> Servicio de consultas de PolyBase para datos externos	
<input type="checkbox"/> Conector Java para orígenes de datos HDFS	

Requisitos previos de las características seleccionadas:

Ya instalado: Windows PowerShell 2.0 o posterior

Requisitos de espacio en disco

Unidad: C: 1003 MB requeridos, 135183 MB disponibles

[Seleccionar todo](#) [Anular la selección de todo](#)

Directorio raíz de instancia: C:\Program Files\Microsoft SQL Server\

Directorio de características compartidas: C:\Program Files\Microsoft SQL Server\

Directorio de características compartidas (x86): C:\Program Files (x86)\Microsoft SQL Server\

< Atrás Siguiente > Cancelar

*Paso 6: Dejamos las opciones seleccionadas por defecto y hacemos click en **Siguiente**.*

Programa de instalación de SQL Server 2019

Configuración de instancia

Especifique el nombre y el identificador de instancia de SQL Server. El identificador de instancia se convierte en parte de la ruta de acceso de instalación.

☒ Instancia predeterminada
☐ Instancia con nombre: MSSQLSERVER

Id. de instancia: MSSQLSERVER01

Directorio de SQL Server: C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER

Instancias instaladas:

Nombre de instancia	Id. de instancia	Características	Edición	Versión
---------------------	------------------	-----------------	---------	---------

< Atrás Siguiente > Cancelar

*Paso 7: Dejamos por defecto el nombre de la instancia y hacemos click en **Siguiente**.*

Configuración del servidor

Especifique las cuentas de servicio y la configuración de intercalación.

Reglas globales
Microsoft Update
Actualizaciones de productos
Instalar archivos de configuraci...
Instalar reglas
Clave de producto
Términos de licencia
Selección de características
Reglas de características
Configuración de instancia
Configuración del servidor
Configuración del Motor de ba...
Reglas de configuración de car...
Listo para instalar
Progreso de la instalación
Operación completada

Cuentas de servicio

Intercalación

Microsoft recomienda usar una cuenta diferente para cada servicio de SQL Server.

Servicio	Nombre de cuenta	Contraseña	Tipo de inicio
Agente SQL Server	NT Service\SQLAgent\$...		Manual
Motor de base de datos de SQL S...	NT Service\MSSQL\$INST...		Automático
SQL Server Browser	NT AUTHORITY\LOCAL ...		Deshabilitado

☐ Conceder el privilegio de realización de tareas de mantenimiento de volumen al servicio Motor de base de datos de SQL Server

Este privilegio habilita la inicialización instantánea de archivos, ya que evita la puesta a cero de las páginas de datos. Esto puede conllevar la divulgación de información al permitir el acceso a contenido eliminado.

[Haga clic aquí para obtener más detalles.](#)

< Atrás
Siguiente >
Cancelar

Paso 8: Dejamos por defecto las características y hacemos click en **Siguiente**.

Configuración del Motor de base de datos

Especifique el modo de seguridad de la autenticación, los administradores, los directorios de datos, el valor TempDB, el grado máximo de paralelismo, los límites de memoria y la configuración del flujo de archivos del Motor de base de datos.

Reglas globales
Microsoft Update
Actualizaciones de productos
Instalar archivos de configuraci...
Instalar reglas
Clave de producto
Términos de licencia
Selección de características
Reglas de características
Configuración de instancia
Configuración del servidor
Configuración del Motor de b...
Reglas de configuración de car...
Listo para instalar
Progreso de la instalación
Operación completada

Configuración del servidor

Directorios de datos

TempDB

MaxDOP

Memoria

FILESTREAM

Modo de autenticación

☒ Modo de autenticación de Windows

☐ Modo mixto (autenticación de SQL Server y de Windows)

Especifique la contraseña de la cuenta de administrador del sistema de SQL Server (sa).

Escribir contraseña:

Confirmar contraseña:

Especifique los administradores de SQL Server

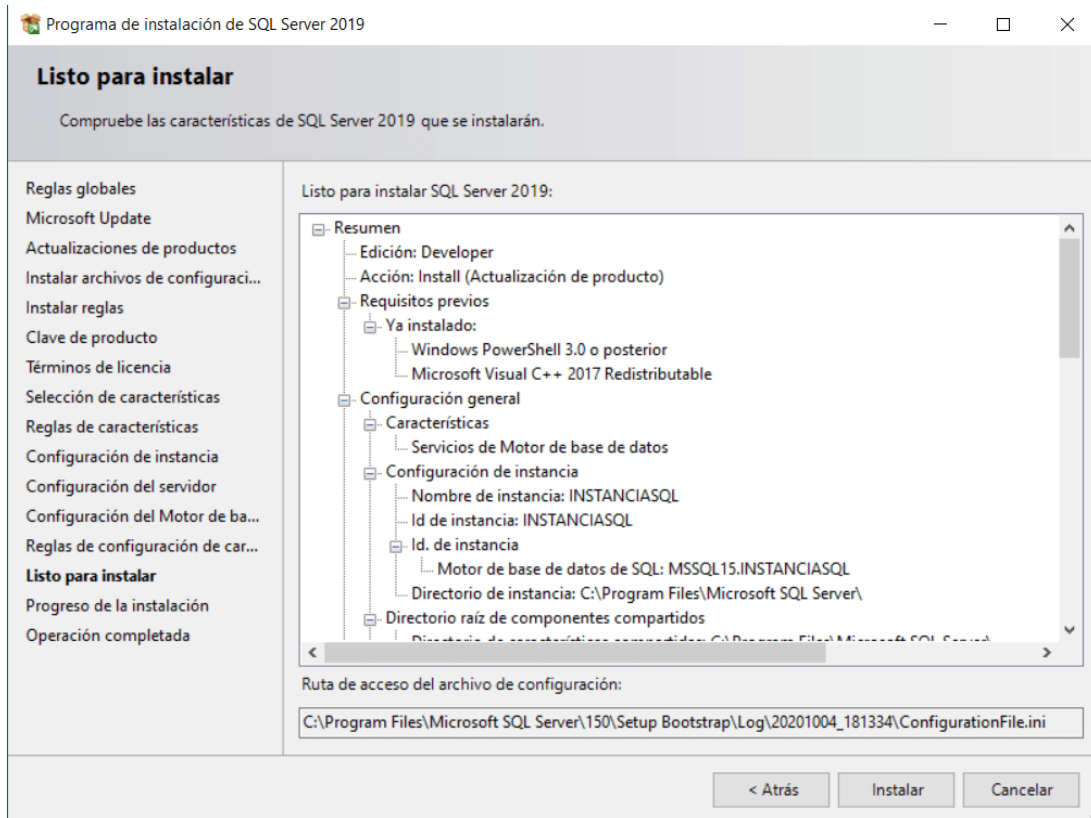
BLIX-PC\Blix (Blix)

Los administradores de SQL Server tienen acceso sin restricciones al Motor de base de datos.

Agregar usuario actual **Agregar...** Quitar

< Atrás
Siguiente >
Cancelar

Paso 9: Dejamos por defecto el nombre de la instancia y hacemos click en *Siguiente*.



Paso 10: Hacemos click en *Instalar*. Una vez terminado el proceso, ya tenemos instalado nuestro gestor de bases de datos.

Conexión al motor y creación de bases de datos

Comenzamos por la conexión a nuestro servidor. En **Tipo de servidor** dejamos seleccionada la opción por defecto **Motor de base de datos**, en **Nombre de servidor** ingresamos el nombre de nuestra PC (MI-EQUIPO). En el caso de nuestro ejemplo, utilizaremos la instancia **MSSQLSERVER01**, para poder conectar debemos agregar el nombre de la instancia al nombre del servidor (MI-EQUIPO\NOMBRE-INSTANCIA).

Y por último, el modo de **autenticación**. Si elegimos el modo *autenticación de Windows* el usuario conectado será el que está autenticado en el sistema operativo. Si elegimos el modo *autenticación de SQL Server* tendremos que especificar el *nombre de usuario* y la *contraseña* del motor de base de datos.

Teniendo todos estos conceptos esclarecidos, ya podremos *conectarnos* a nuestro motor y crear nuestra base de datos.

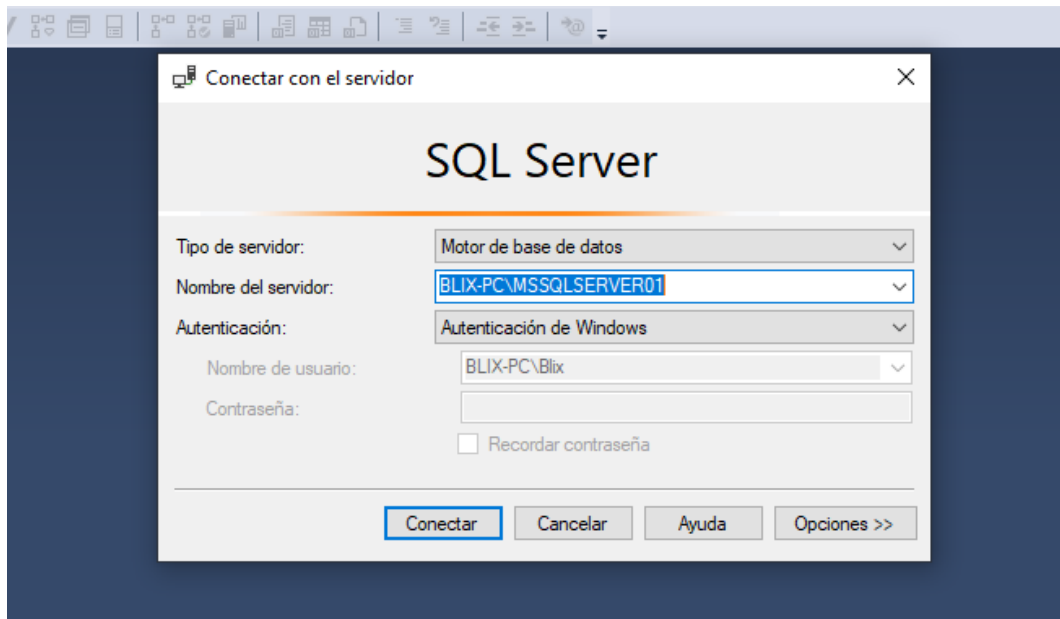


Figura 7: Conexión a motor de bases de datos.

Para crear la base de datos, hacemos click derecho en *Bases de datos* y en el menú emergente seleccionaremos *Nueva base de datos*.

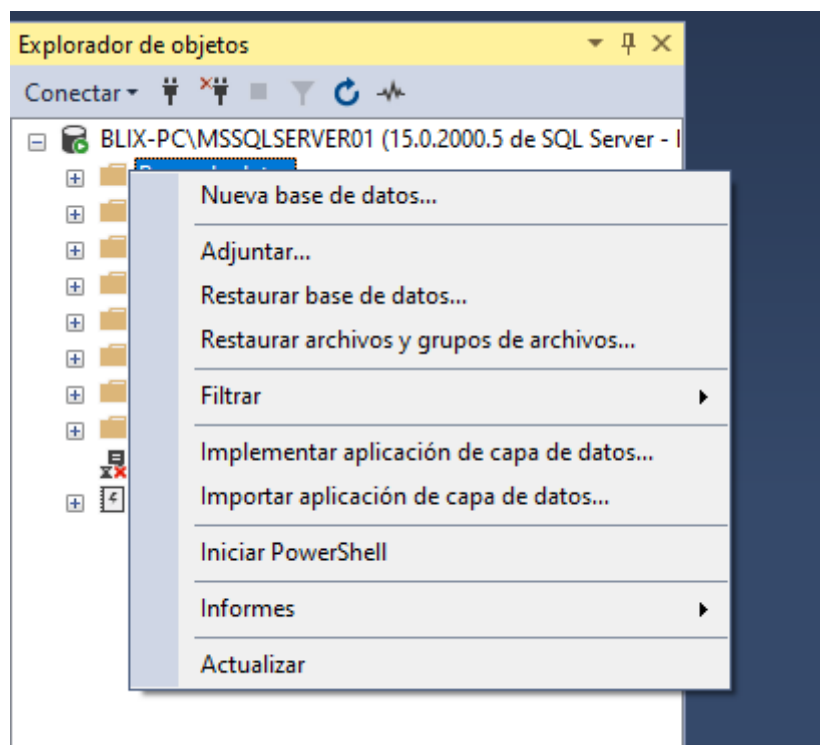


Figura 8: Creación de base de datos.

Asignamos un nombre a la base de datos, aceptamos y **ya tenemos nuestra base de datos**.

Nueva base de datos

Script ? Ayuda

Nombre de la base de datos: MI_BASE_DE_DATOS

Propietario: <predeterminado>

☒ Usar indexación de texto completo

Archivos de la base de datos:

Nombre lógico	Tipo de archivo	Grupo de archivos	Tamaño inicial (MB)	Crecimiento automático
MI_BASE_D...	Datos de FILAS	PRIMARY	8	En 64 MB, sin límite
MI_BASE_D...	REGISTRO	No aplicable	8	En 64 MB, sin límite

Conexión

Servidor:
BLIX-PC\MSSQLSERVER01

Conexión:
BLIX-PC\Blix

[Ver propiedades de conexión](#)

Progreso

Listo

Agregar Quitar

Aceptar Cancelar

Figura 9: Creación de base de datos con nombre.

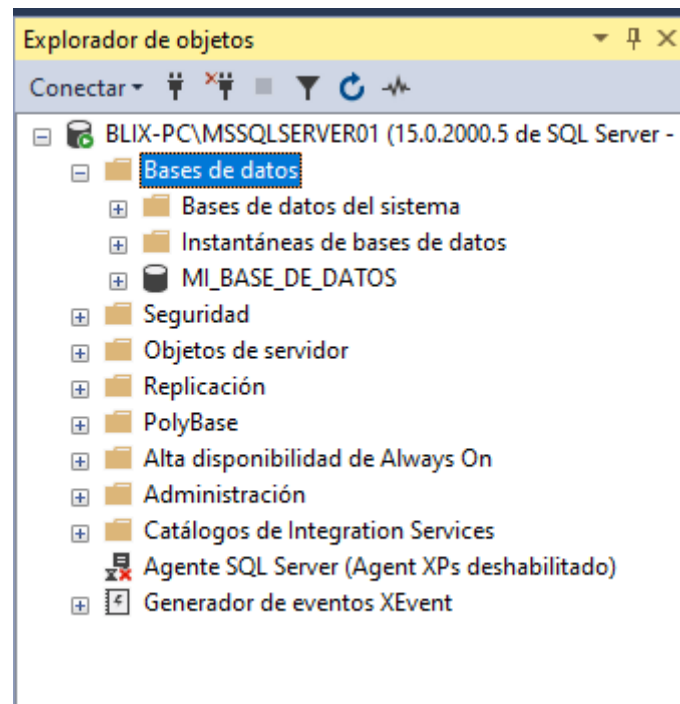


Figura 10: Vista de árbol con base de datos creada.

Para crear las tablas, hacemos click derecho en el menú *Tablas* dentro del árbol de nuestra base de datos. Seleccionamos *Nuevo* y hacemos click en *Tabla...*

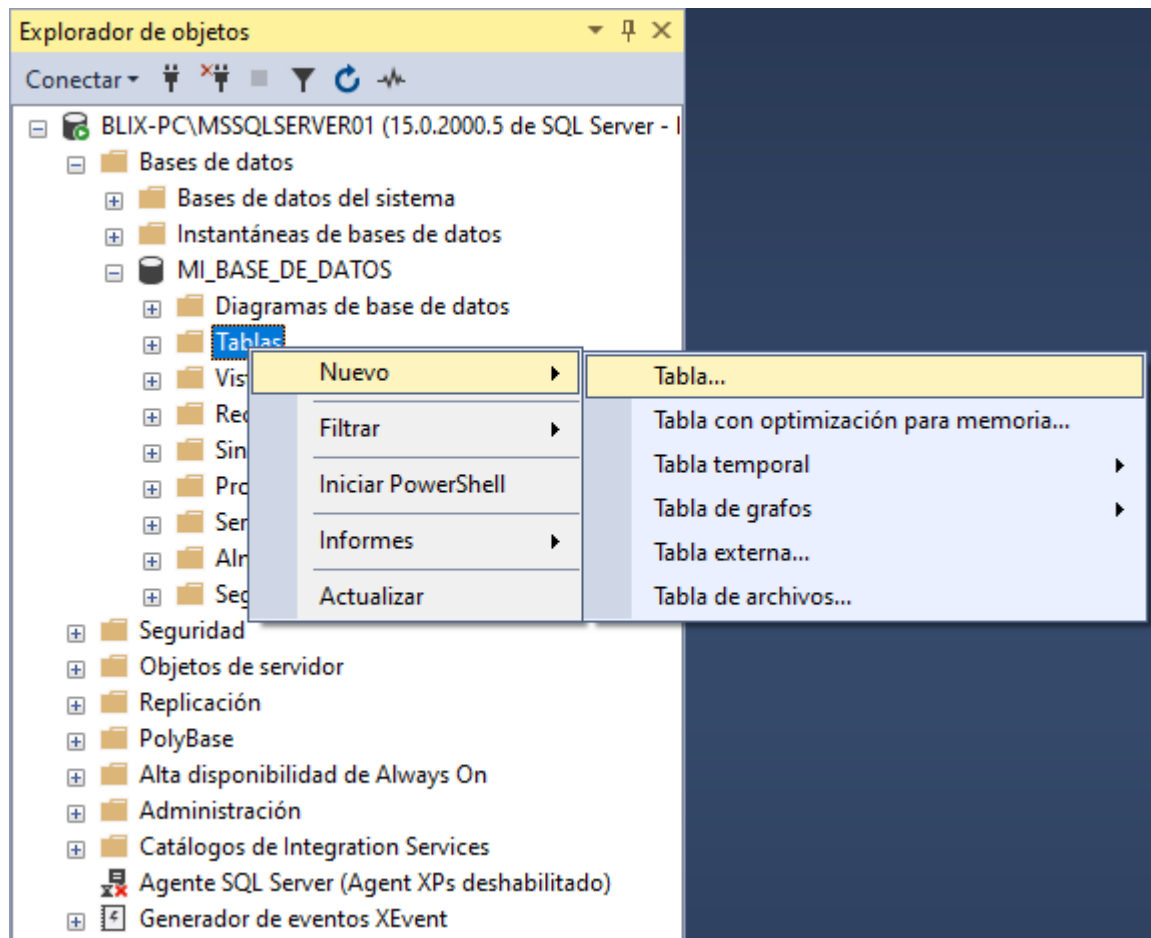


Figura 11: Creación de tabla.

La pestaña muestra el nombre de nuestra tabla. Una vez que guardemos los cambios, se mostrará el nombre que le asignemos. El asterisco simboliza que tenemos cambios sin guardar.

BLIX-PC\MSSQLSERV...TOS - dbo.Table_1*

Nombre de columna	Tipo de datos	Permitir valores NULL
Id_Cliente	int	<input type="checkbox"/>
Nombre	varchar(50)	<input type="checkbox"/>
Apellido	varchar(50)	<input type="checkbox"/>
Id_Localidad	int	<input type="checkbox"/>

Campos

Tipo de datos de cada campo

Permite o no valores NULOS

Figura 12: Creación de tablas agregando campos con tipos de datos.

Para asignar la clave primaria, hacemos click derecho en el campo que será PK (Primary Key - en inglés) y seleccionamos *Establecer clave principal*.

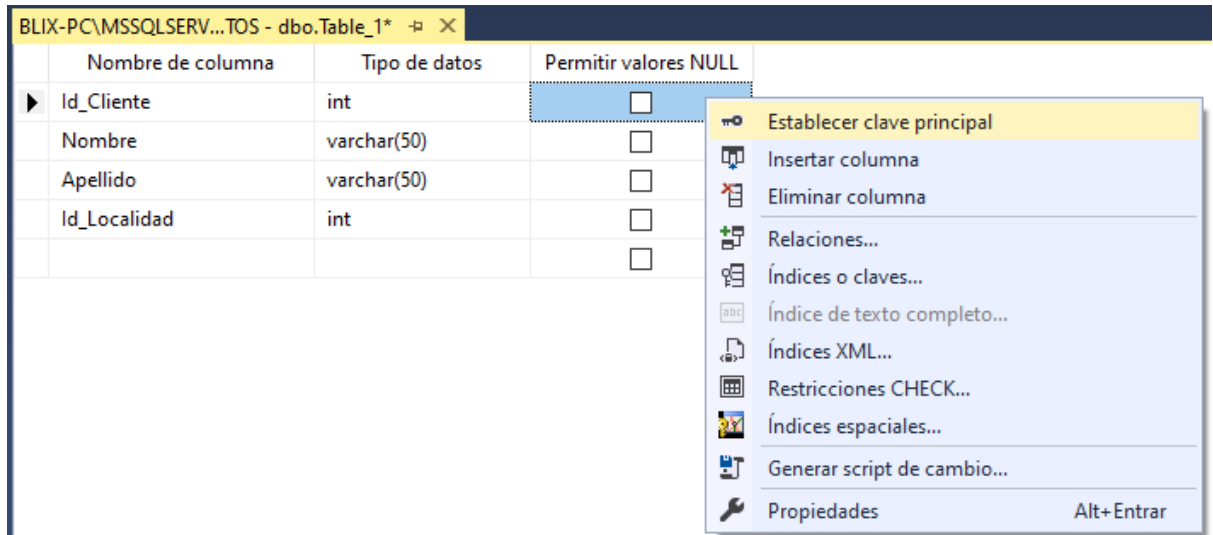


Figura 13: Establecimiento de clave principal.

Además, a nuestra clave principal, podemos asignarle *especificación de identidad* (opcional) para que autoincremente su valor y lo maneje el motor de bases de datos. Podemos asignar el valor con el que comienza la identidad y el incremento que tendrá a medida que se van agregando registros.

Una vez seleccionada la clave principal, se simboliza con una llave como muestra la siguiente imagen.

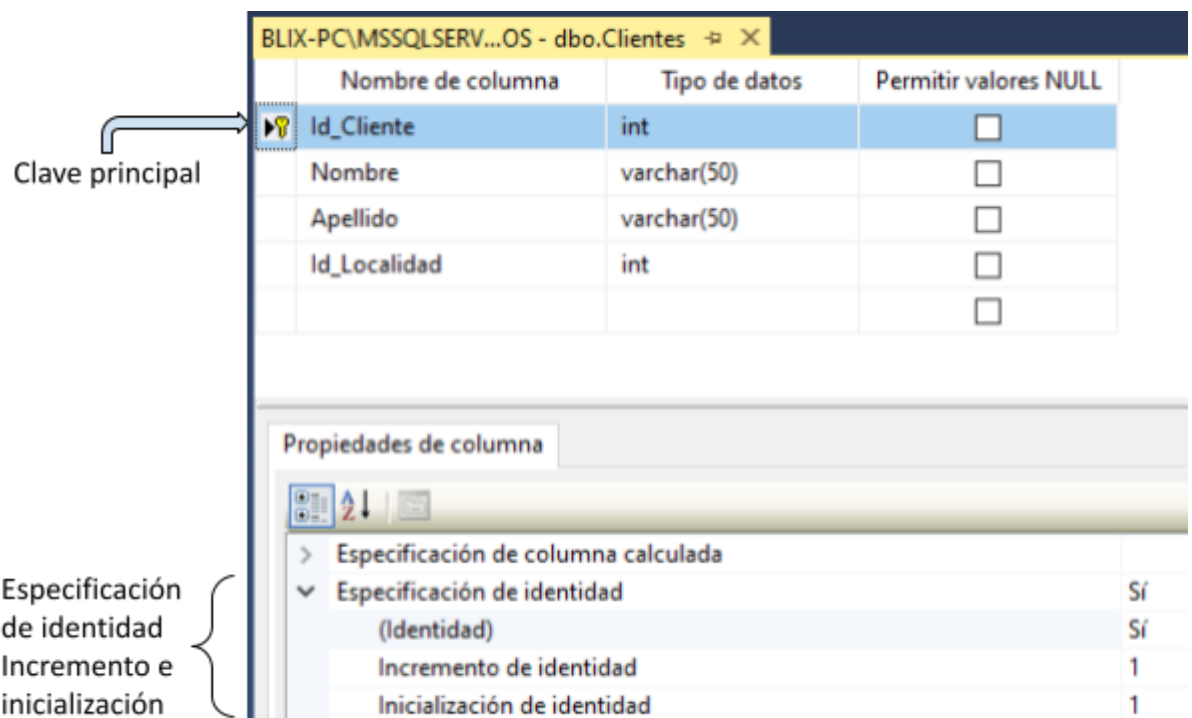


Figura 14: Especificación de identidad con inicialización e incremento.

Asignada la clave principal y agregadas las columnas necesarias, guardamos nuestra tabla con el nombre de la entidad. Luego veremos nuestra tabla creada en el árbol de la base de datos. (Si no se refresca automáticamente, deberemos actualizarla nosotros haciendo click derecho en el nombre de la base de datos, luego en *Actualizar*).

Para crear nuestra clave foránea, debemos crear primero la tabla con su clave primaria (Para ello repetimos los pasos de creación de tablas) y luego debemos asignar en la tabla secundaria la clave foránea.

Para ello, vamos a modificar la tabla que contiene la clave foránea y hacemos click derecho y luego hacemos click en *Relaciones*.

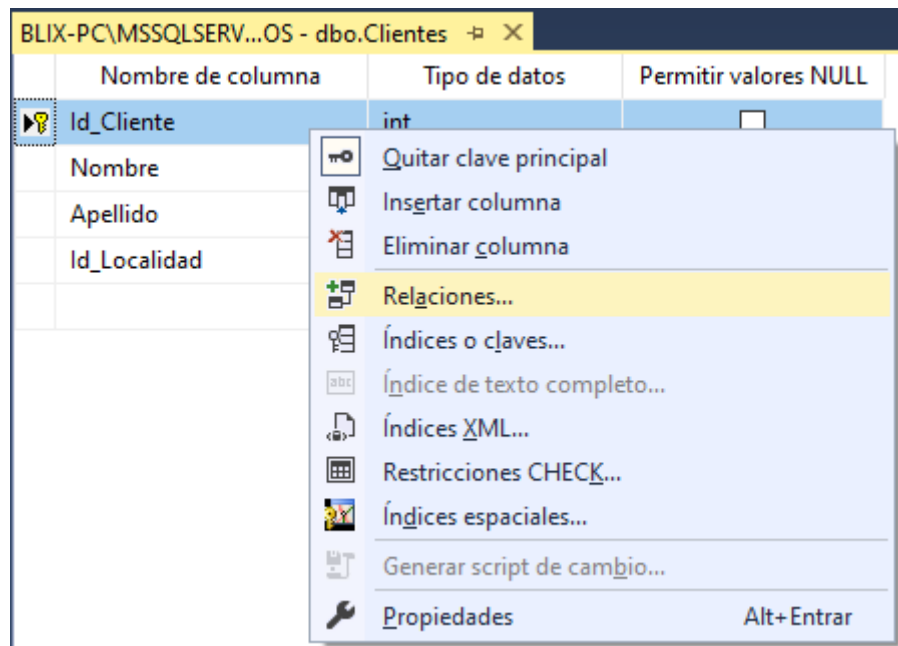


Figura 15: Creación de relaciones entre tablas.

Una vez en la pantalla de relaciones, desde el botón *agregar* creamos una nueva relación, de allí seleccionamos la opción *especificación de tablas y columnas* y seleccionamos el botón

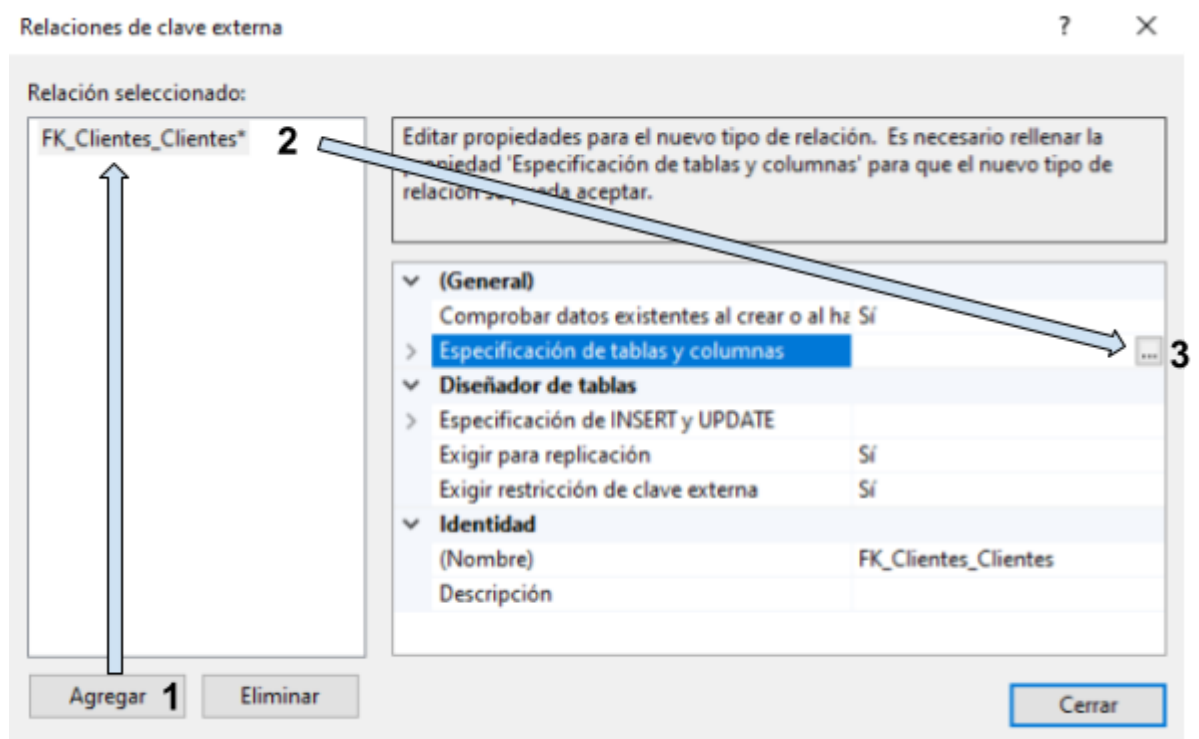


Figura 16: Creación de nombre de relaciones.

En la pantalla de *Tablas y columnas* seleccionaremos la tabla de clave principal y la columna clave principal y la tabla de clave externa con la columna clave externa o foránea.

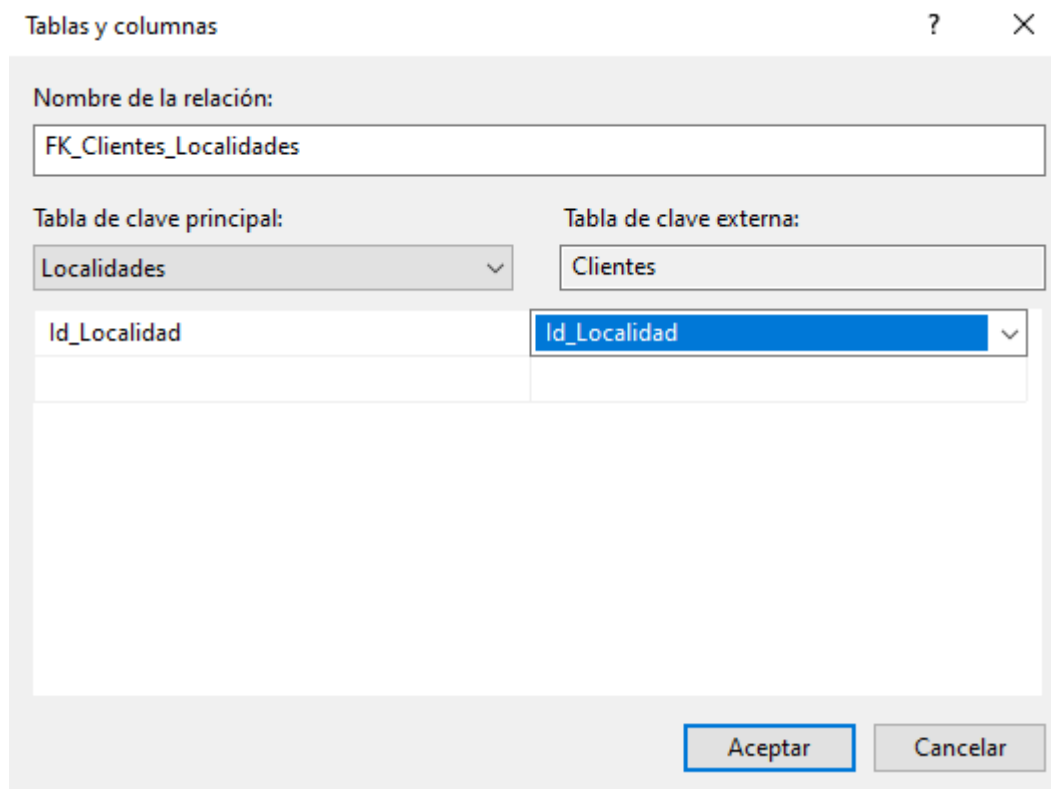


Figura 17: Selección de tablas a relacionar.

Aceptamos, guardamos los cambios y ya tenemos nuestras dos tablas unidas por una clave principal (en su tabla principal) y una clave foránea (en su tabla secundaria).

Scripts

Los scripts son comandos que se ejecutan en un intérprete siguiendo un orden. Éstos no requieren compilación, sino que se pueden escribir en cualquier editor de texto y luego ejecutar en un motor de bases de datos, por ejemplo.

Se utilizan para gestionar bases de datos, dominios, tablas, índices, procedimientos almacenados, vistas, insertar, borrar, modificar y consultar registros, etc.

Dentro de las sentencias SQL tenemos dos grupos: *Lenguaje de definición de datos* (DDL por sus siglas en inglés *Data Definition Language*) y *Lenguaje de manipulación de datos* (DML por sus siglas en inglés *Data Manipulation Language*).

Dentro del primer grupo tenemos las sentencias para crear y modificar la estructura de las tablas y los objetos. Podremos encontrar las siguientes sentencias:

CREATE: Sirve para crear objetos en la base de datos;

ALTER: Sirve para modificar nuestros objetos ya creados;

DROP: Sirve para eliminar los objetos creados;

TRUNCATE: sirve para eliminar los registros de una tabla, incluyendo los espacios asignados a los registros.

Continuando con nuestra base de datos de ejemplo, podríamos realizar la creación de los objetos con *Scripts*, incluyendo la propia base de datos.

Para generar o ejecutar nuestro/s script/s en Microsoft SQL Server Management Studio, hacemos click en *nueva consulta* y nos abrirá una ventana de consulta la cual podremos escribir nuestro script y ejecutarlo, además de guardar el archivo con extensión **.sql**.

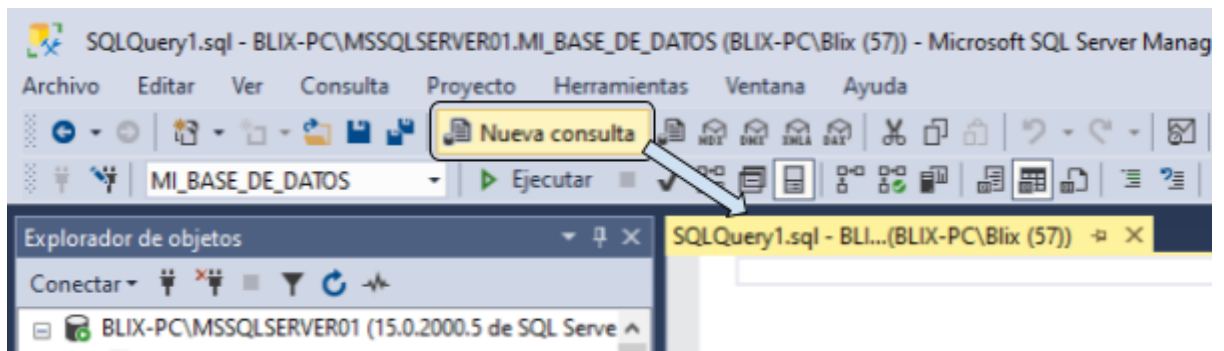


Figura 18: Creación de nueva consulta.

Situados en la ventana de consulta, vamos a ejecutar los siguientes scripts:

```
--SCRIPT PARA CREAR BASE DE DATOS  
CREATE DATABASE MI_BASE_DE_DATOS
```

Como resultado final obtendremos nuestra base de datos de la siguiente manera:

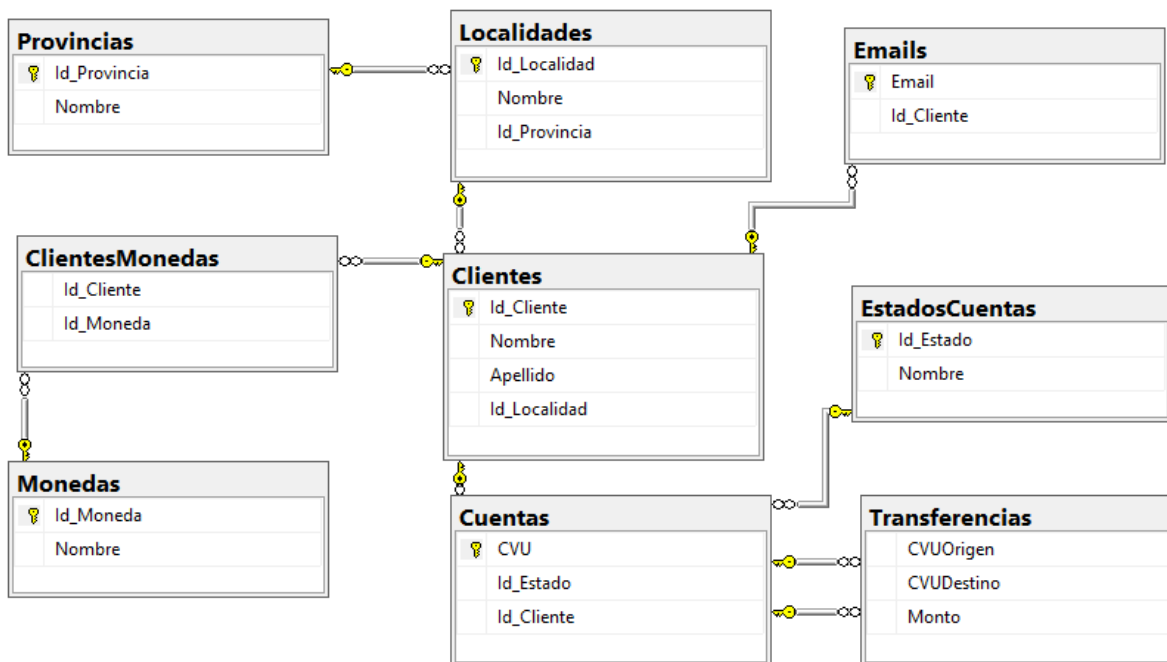


Figura 19: Diagrama de base de datos.

En el segundo grupo tenemos las sentencias para gestionar los datos dentro de las tablas, algunos de ellos son:

SELECT: sirve para obtener los datos de una tabla;

Ejemplo: Para consultar *todas las columnas* de la tabla *Clientes* la sentencia debería ser de la siguiente manera: **SELECT * FROM CLIENTES.**

En caso de querer consultar las columnas *Nombre* y *Apellido* de la tabla *Clientes* separamos las columnas que queremos obtener con *coma*, en ese caso la sentencia debería ser de la siguiente manera: **SELECT Nombre, Apellido FROM Clientes.**

INSERT: Sirve para insertar registros en una tabla;

Ejemplo: Si queremos ingresar datos a la tabla Clientes, se deben respetar el orden de las columnas con los datos de entrada, separando por comas las columnas y pudiendo desestimar los campos que permiten valores nulos. En ese caso, la consulta debería ser de la siguiente manera: **INSERT INTO Clientes (Nombre, Apellido, Id_Localidad) VALUES ('Juan', 'Gomez', 15).**

UPDATE: Sirve para actualizar los registros existentes;

Ejemplo: Suponiendo que el registro recientemente insertado nos brinde el Id_Cliente = 10, si queremos modificar ese registro, debemos realizar la sentencia con la condición WHERE. De no tener la condición, se modificarán todos los registros con los nuevos valores. En ese caso, la sentencia quedará de la siguiente manera: **UPDATE Clientes SET Nombre = 'Juan Carlos', Apellido = 'Gómez' WHERE Id Cliente = 10**

DELETE: Sirve para eliminar los registros existentes.

Ejemplo: De la misma forma que la sentencia UPDATE, de no tener la condición WHERE, se eliminarán todos los registros de la tabla. Si deseamos eliminar el registro recientemente modificado, podremos realizar la siguiente consulta: **DELETE FROM Clientes WHERE Id_Cliente = 10**

Cláusulas

Las cláusulas son condiciones para modificar la sentencia y son utilizadas para definir los datos que se desean seleccionar.

Dentro de las principales encontramos las siguientes:

WHERE: Es la condición para seleccionar los registros que cumplen una condición precisa. Por ejemplo si quisiéramos obtener los clientes que tienen el valor 50 en el campo Id_Localidad, deberíamos escribir la siguiente consulta: **SELECT * FROM Clientes WHERE Id_Localidad = 50.**

También se pueden combinar distintas condiciones, por ejemplo, a la condición anterior le agregamos que el campo *Apellido* tenga el valor 'Gómez'. En ese caso, nuestra consulta sería la siguiente: **SELECT * FROM Clientes WHERE Id_Cliente = 50 AND Apellido = 'Gómez'.** Podemos reemplazar las condiciones como se necesite, especificando una o más de ellas.

ORDER BY: Es utilizada para darle un orden a los resultados, especificando la columna y de qué manera se ordenarán. Se pueden especificar una o más columnas. Por ejemplo, si quisiéramos obtener todos los registros de la tabla Clientes ordenados por Apellido ascendente nuestra consulta quedaría de la siguiente manera: **SELECT * FROM Clientes ORDER BY Apellido ASC.** Si a la condición anterior le agregamos que ordene también por el campo Nombre nuestra consulta quedaría de la siguiente manera: **SELECT * FROM Clientes ORDER BY Apellido ASC, Nombre ASC.**

HAVING: A esta cláusula la utilizaremos de manera similar a la cláusula WHERE, con la diferencia que ésta se utiliza con las funciones de agregación SUM, MAX, AVG, etc. Por ejemplo, si quisiéramos obtener las CVUOrigen las cuales la suma de todas las transferencias haya sido mayor a 10000 nuestra consulta quedaría de la siguiente manera: **SELECT * FROM Transferencias HAVING SUM(Monto) > 10000**

Funciones de agregación

Una función de agregación es un cálculo sobre un conjunto de valores, devolviendo un sólo valor, excepto de la función COUNT, ignorando los valores NULL. Por lo general, debemos combinar las funciones de agregado con la cláusula GROUP BY.

Algunas de ellas son MAX (Máximo), MIN (Mínimo), AVG (Promedio), SUM (Suma), COUNT (Cantidad) entre otras.

Por ejemplo, si quisiéramos obtener el monto máximo de las transferencias nuestra consulta sería la siguiente: **SELECT MAX(Monto) FROM Transferencias.**

Si quisiéramos obtener todas las CVUOrigen con un promedio de transferencias que realizó cada una, nuestra consulta se verá de la siguiente manera: **SELECT CVUOrigen, AVG(Monto) AS Promedio FROM Transferencias GROUP BY CVUOrigen**

En esta consulta podemos realizar dos observaciones. Por un lado, tenemos un *ALIAS* el cual le asignaremos a nuestro campo que contiene la función AVG (O cualquier otra función) sucediendo el nombre de la columna con el nombre del alias (*Columna AS 'Alias'*). Éste nos sirve para ponerle un título a la columna.

Como segunda observación, tenemos una cláusula GROUP BY, la cual la utilizamos con la columna que no tiene ninguna función (En este caso, es la columna CVUOrigen) ya que cada promedio de Monto quedaría como único resultado y el campo CVUOrigen se repetiría.

En el ejemplo de la tabla siguiente tenemos el que el **CVUOrigen 1234** realizó **3** transferencias con **1** promedio de **Monto** de **\$200**, si no tuviésemos la cláusula GROUP BY tendría como resultado **3** registros de CVUOrigen y **1** registro de promedio. Esta cláusula nos agrupa los resultados repetidos.

CVUOrigen	Monto
1234	100
1234	200
1234	300

Subconsultas

Una subconsulta es una instrucción SELECT anidada dentro de otra instrucción SELECT, las cuales pueden contener condiciones como las mencionadas anteriormente.

Para generar una subconsulta debemos tener presentes ciertas reglas como escribir la subconsulta entre paréntesis, especificar en ella sólo una columna (si no se utiliza IN, ANY, ALL o EXISTS), que no contenga la cláusula GROUP BY, además de no poder recuperar datos de la misma tabla a la cual se realice la sentencia UPDATE o DELETE.

Por ejemplo, si quisiéramos obtener los Clientes que tienen en el campo Id_Localidad cualquier ID que esté en la tabla Provincias siendo *Córdoba* el nombre de la misma, nuestra consulta sería la siguiente:

SELECT * FROM Clientes WHERE Id_Localidad IN(SELECT Id_Localidad FROM Provincias WHERE Nombre = 'Córdoba')

La subconsulta obtendrá todos los valores de Id_Localidad que contenga la provincia *Córdoba* y la consulta obtendrá los Clientes que tengan el valor de Id_Localidad que estén en la lista de la subconsulta.

Procedimientos almacenados

Como su nombre lo describe, los procedimientos almacenados son instrucciones guardadas en la base de datos, las cuales podemos reutilizar en cualquier momento. Los SP (Por sus siglas en inglés Stored Procedures) pueden devolver valores, o no.

Por ejemplo, si quisiéramos hacer muchas veces la misma consulta podemos generar un SP y llamarlo por su nombre, generando en una nueva ventana de consulta como mostramos a continuación:

```
CREATE PROCEDURE ObtenerClientes
```

```
AS
```

```
SELECT * FROM CLIENTES
```

Luego de crear el procedimiento, en vez de realizar la consulta SELECT, en la ventana de consultas escribiremos ObtenerClientes.

Haciendo más complejo el ejemplo anterior, nuestro SP recibirá el parámetro 'NombreProvincia' de tipo VARCHAR para poder filtrar la subconsulta por nombre de Provincia como mostramos en el siguiente ejemplo:

```
CREATE PROCEDURE ObtenerClientesPorProvincia
```

```
@NombreProvincia AS VARCHAR(50)
```

```
AS
```

```
SELECT * FROM Clientes WHERE Id_Localidad IN(SELECT Id_Localidad FROM Provincias  
WHERE Nombre = @NombreProvincia)
```

Para ejecutar nuestro procedimiento almacenado, en una nueva ventana de consulta ejecutaremos lo siguiente **ObtenerClientesPorProvincia 'Córdoba'** pudiendo reemplazar 'Córdoba' por cualquier otra provincia.

Referencias

Stack overflow: <https://es.stackoverflow.com/>

Microsoft docs: <https://docs.microsoft.com/es-es/>

W3Schools: <https://www.w3schools.com/>