

CURSOS PIT

CTIC  UNI

SQL SERVER 2022

IMPLEMENTACIÓN DE LA BASE DE DATOS



Microsoft®
SQL Server®



Capítulo 1

Introducción a Microsoft SQL Server 2022

En este capítulo conoceremos la plataforma Microsoft SQL Server 2022, qué es, cuáles son las tecnologías que la sustentan, qué ediciones están disponibles según nuestras necesidades, y una introducción a su interfaz de usuario.

1. ¿QUÉ ES MICROSOFT SQL SERVER 2022?

Microsoft SQL Server 2022 es un sistema de gestión de bases de datos relacional desarrollado por Microsoft. La versión 2022 es la más reciente en el momento actual, y ofrece una variedad de mejoras y características nuevas en comparación con versiones anteriores. Algunas de estas características incluyen mejoras en la seguridad, la escalabilidad, el rendimiento y la inteligencia artificial. También se puede ejecutar en una variedad de plataformas, incluyendo Windows, Linux y en la nube.

1.1. ¿Qué es OLTP?

OLTP (On-Line Transaction Processing) es una técnica de procesamiento de transacciones en tiempo real que se utiliza para manejar un gran volumen de operaciones de alta velocidad, pequeñas y breves en una base de datos.

En un sistema OLTP, las transacciones se procesan inmediatamente y los resultados se reflejan de manera rápida en la base de datos.

Los sistemas OLTP son comunes en entornos de neGOcios y comerciales, como sistemas de ventas, bancos, aerolíneas y tiendas en línea, donde se requiere un alto rendimiento y una respuesta rápida para las transacciones.

Los sistemas OLTP utilizan una arquitectura de base de datos relacional y se caracterizan por tener un gran número de transacciones, una alta concurrencia y una alta velocidad de acceso a los datos.

1.2. ¿Qué es OLAP?

OLAP (On-Line Analytical Processing) es una técnica de procesamiento analítico en línea que se utiliza para analizar grandes volúmenes de datos multidimensionales.

En un sistema OLAP, los datos se organizan en una estructura de cubo multidimensional, lo que permite a los usuarios analizar los datos desde diferentes perspectivas y niveles de detalle.

Los sistemas OLAP son comunes en entornos de neGOcios y financieros, donde se requiere un análisis detallado de los datos para la toma de decisiones.

Los sistemas OLAP utilizan una arquitectura de base de datos multidimensional y se caracterizan por tener un gran volumen de datos, un acceso lento a los datos y una alta complejidad en las consultas.

1.3. ¿Qué es e-commerce?

E-commerce (comercio electrónico) es la compra y venta de bienes y servicios a través de Internet.

Esto incluye una variedad de actividades comerciales, desde la adquisición de bienes y servicios en línea hasta la venta de productos y servicios en línea, así como la realización de transacciones en línea.

El e-commerce se ha convertido en una parte importante de la economía mundial y ha cambiado la forma en que las empresas y los consumidores interactúan y realizan transacciones comerciales.

Algunos ejemplos de e-commerce son tiendas en línea, compras en línea, paGOs en línea, marketing en línea, entre otros.

Además, se puede clasificar en diferentes tipos, como B2B (business-to-business), B2C (business-to-consumer), C2B (consumer-to-business), C2C (consumer-to-consumer) dependiendo de la relación entre el comprador y el vendedor.

1.4. ¿Qué es Data Warehouse?

Un Data Warehouse es un sistema de almacenamiento de datos diseñado específicamente para soportar el análisis y la toma de decisiones.

Es una colección de datos que se utiliza para ayudar a las empresas a entender y mejorar su rendimiento.

Estos datos se extraen de diferentes fuentes, como sistemas transaccionales, sistemas de gestión de relaciones con clientes (CRM) y otras fuentes de datos, y se integran en un solo lugar para su análisis.

Los Data Warehouses son diferentes a las bases de datos transaccionales, ya que estas últimas están diseñadas para manejar transacciones rápidas y precisas, mientras que los Data Warehouses están diseñados para manejar consultas y análisis complejos y de alto

rendimiento. Los Data Warehouses suelen tener una arquitectura dimensional, que se refiere a una estructura de datos organizada en torno a entidades clave, como tiempo, clientes y productos, para facilitar el análisis.

2. TECNOLOGÍAS DE MICROSOFT SQL SERVER 2022

Microsoft SQL Server 2022 es un conjunto completo de tecnologías y herramientas de administración y análisis de datos para el entorno empresarial.

- Database Engine (Motor de Base de Datos)
- Analysis Services (Servicios de Análisis)
- Reporting Services (Servicios de Generación de Reportes)
- Integration Services (Servicios de Integración)
- Azure-SSIS Integration Runtime (Paquetes de transformación de datos e integración de datos en la nube)

2.1. SQL Server Database Engine (Motor de Bases de Datos)

SQL Server Database Engine Es el Motor de Bases de Datos de SQL Server, también conocido como Database Engine, es el componente principal de SQL Server que se encarga de almacenar, procesar y proteger los datos en un sistema de gestión de bases de datos relacionales (RDBMS). Es el núcleo de SQL Server y proporciona las funciones subyacentes necesarias para que los demás componentes de SQL Server, como Analysis Services, Integration Services y Reporting Services, funcionen correctamente.

Algunas de las funciones clave del Motor de Bases de Datos incluyen:

- **Gestión de bases de datos relacionales:** permite la creación y gestión de bases de datos relacionales, incluyendo tablas, vistas, índices y restricciones.
- **Procesamiento transaccional:** proporciona soporte para el procesamiento transaccional, lo que permite ejecutar múltiples operaciones relacionadas como una sola unidad de trabajo. Esto garantiza que la base de datos permanezca en un estado consistente, incluso en caso de error o fallo del sistema.
- **Seguridad de datos:** proporciona un modelo de seguridad robusto que permite gestionar el acceso de usuarios y roles basados en datos.



- **Recuperación de datos:** ofrece varias opciones para la recuperación de datos, como la posibilidad de realizar recuperación en un momento determinado y copias de seguridad de registros de transacciones.
- **Procesamiento de consultas:** proporciona un potente motor de procesamiento de consultas que permite la ejecución eficiente de consultas complejas en grandes conjuntos de datos.
- **Escalabilidad:** permite la escalabilidad horizontal de las bases de datos, permitiendo distribuirlos a través de varios servidores para mejorar el rendimiento y disponibilidad.

2.2. Analysis Services (SSAS)

SQL Server Analysis Services (Servicios de Análisis) es una plataforma de análisis de datos multidimensionales de Microsoft que permite a los usuarios crear, administrar y consultar cubos de datos multidimensionales.

Es un componente de la plataforma SQL Server. Con SSAS, los desarrolladores y administradores de bases de datos pueden crear cubos de datos multidimensionales que se conectan a una variedad de fuentes de datos, incluyendo bases de datos relacionales, hojas de cálculo y servicios web.

Algunas de las características más importantes de SSAS incluyen:

- Capacidad para crear cubos de datos multidimensionales con tablas, gráficos y grillas de datos.
- Herramientas para crear y personalizar cubos de datos, incluyendo un diseñador de cubos visual y un lenguaje de consulta de cubos (MDX).
- Capacidad para programar tareas y flujos de trabajo utilizando el lenguaje de programación de Microsoft Visual C# o Microsoft Visual Basic.
- Integración con otras tecnologías de Microsoft, como SQL Server, SharePoint, Reporting Services, and Excel
- Capacidad de publicar y compartir cubos a través de un portal web o correo electrónico.

SSAS es utilizado principalmente para crear y administrar cubos de datos multidimensionales en una variedad de organizaciones, desde pequeñas empresas hasta grandes corporaciones, para ayudar a los usuarios a obtener información valiosa de sus datos y realizar análisis complejos.

2.3. Reporting Services (SSRS)

SQL Server Reporting Services (Servicios de Generación de Reportes) es una plataforma de generación de informes de Microsoft que permite a los usuarios crear, publicar y administrar informes en línea y en papel. Es un componente de la plataforma SQL Server.

Con SSRS, los desarrolladores y administradores de bases de datos pueden crear informes que se conectan a una variedad de fuentes de datos, incluyendo bases de datos relacionales, hojas de cálculo y servicios web.

Algunas de las características más importantes de SSRS incluyen:

- Capacidad para crear informes estáticos o interactivos con tablas, gráficos y grillas de datos.
- Herramientas para crear y personalizar informes, incluyendo un diseñador de informes visual y un lenguaje de consulta de informes (RDL).
- Capacidad para programar tareas y flujos de trabajo utilizando el lenguaje de programación de Microsoft Visual C# o Microsoft Visual Basic.
- Integración con otras tecnologías de Microsoft, como SQL Server, SharePoint, y Reporting Services.
- Capacidad de publicar y compartir informes a través de un portal web o correo electrónico.

SSRS es utilizado principalmente para crear y administrar informes en una variedad de organizaciones, desde pequeñas empresas hasta grandes corporaciones, para ayudar a los usuarios a obtener información valiosa de sus datos.



2.4. Integration Services (SSIS)

SQL Server Integration Services (Servicios de Integración) es una plataforma de integración de datos de Microsoft que permite a los usuarios crear soluciones de integración de datos complejas para extraer, transformar y cargar datos (ETL, por sus siglas en inglés).

Con SSIS, los desarrolladores y administradores de bases de datos pueden crear paquetes de integración de datos que contienen una serie de tareas y transformaciones para manejar y procesar datos.

Algunas de las características más importantes de SSIS incluyen:

- Capacidad para conectarse a una amplia variedad de fuentes de datos, como bases de datos relacionales, archivos planos, hojas de cálculo, y servicios web.
- Herramientas de transformación de datos para limpiar, combinar y manipular datos.
- Capacidad de programar tareas y flujos de trabajo utilizando el lenguaje de programación de Microsoft Visual C# o Microsoft Visual Basic.
- Integración con otras tecnologías de Microsoft, como SQL Server, Reporting Services y SharePoint.

SSIS se utiliza principalmente para realizar tareas de integración de datos complejas como migraciones de datos, actualizaciones de datos, y la creación de soluciones de Data Warehousing, así como también para automatizar tareas de mantenimiento de bases de datos y procesos de neGOcio.

3. EDICIONES DE MICROSOFT SQL SERVER 2022

Las diferentes ediciones de SQL Server se adecuan a los requerimientos de rendimiento, tiempo de ejecución y precio de los distintos tipos de organizaciones.

Edición ¹	Descripción
Enterprise	SQL Server Enterprise Edition , proporciona funciones de centro de datos de tecnología avanzada completas con un rendimiento ultrarrápido, virtualización ilimitada e inteligencia empresarial integral, lo que habilita los mayores niveles de servicio para las cargas de trabajo críticas y el acceso del usuario final a información sobre los datos.
Estándar	SQL Server Standard Edition proporciona administración básica de datos y base de datos de inteligencia empresarial para que los departamentos y pequeñas organizaciones ejecuten sus aplicaciones y admite las herramientas de desarrollo comunes, tanto locales como en la nube, que habilitan la administración eficaz de bases de datos con recursos de TI mínimos.
Web	SQL Server Web Edition es una opción con un costo total de propiedad bajo para los hosts de Web y los VAP de Web que proporciona capacidades asequibles de administración y escalabilidad para propiedades web, tanto de pequeña como de gran escala.
Desarrollador	SQL Server Developer Edition permite a los desarrolladores compilar cualquier tipo de aplicación en SQL Server. Incluye toda la funcionalidad de la edición Enterprise, pero tiene licencias para usarse como sistema de prueba y desarrollo, no como un servidor de producción. SQL Server Developer es una opción ideal para las personas que compilan y prueban aplicaciones.
Express	SQL Server 2022 (16.x) Express Edition es una base de datos gratuita para principiantes y es ideal a fin de descubrir cómo compilar pequeñas aplicaciones de servidor y de escritorio orientadas a datos. Es la mejor opción para los fabricantes de

¹ Información tomada de los Libros en Pantalla de Microsoft SQL Server 2022.



	software independientes, los desarrolladores y los aficionados que compilan aplicaciones cliente. Si necesita características de base de datos más avanzadas, SQL Server Express se puede actualizar sin problemas a otras versiones superiores de SQL Server. SQL Server Express LocalDB es una versión ligera de Express que tiene todas sus características de capacidad de programación, se ejecuta en modo usuario y presenta una instalación rápida y sin configuración, y una lista reducida de requisitos previos.
--	--

4. SQL SERVER MANAGEMENT STUDIO

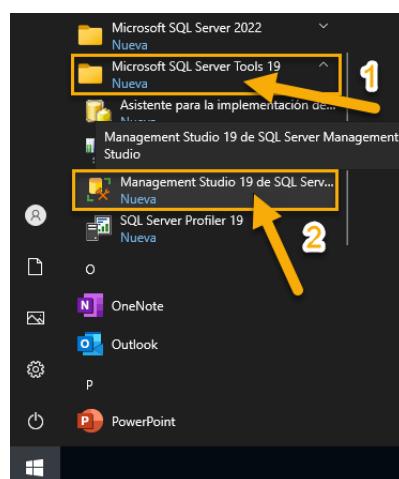
SQL Server Management Studio es una aplicación cliente que proporciona un entorno gráfico para crear y gestionar nuestras bases de datos, los objetos de las bases de datos y desarrollar consultas y scripts en SQL Server en el equipo local o en la nube.

4.1. Iniciar SQL Server Management Studio

SQL Server Management Studio debe instalarse en las computadoras cliente que tendrán acceso al servidor SQL. Se encuentra en el grupo Microsoft SQL Server 2022 del botón Iniciar de Windows.

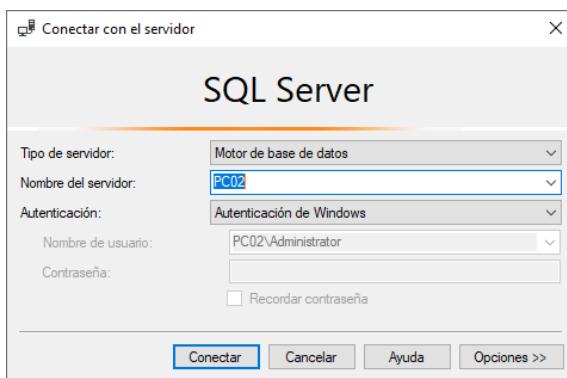
Ejercicio 1.1: Carga de SQL Server Management Studio

1. Para cargar la aplicación cliente SQL Server Management Studio ejecute la secuencia: botón **Iniciar**, **Microsoft SQL Server Tools 19**, **Microsoft SQL Server Management Studio 19**.



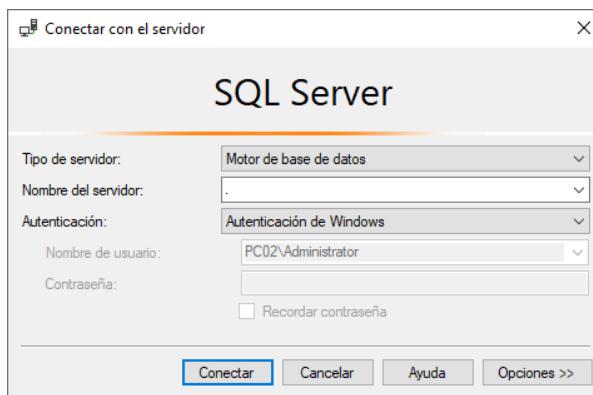


2. Se abre la ventana **Conectar con el servidor**. En **Tipo de servidor** verifique que aparece seleccionado **Motor de base de datos**.



3. En **Nombre del servidor** puede colocar:

- El nombre del servidor: En mi caso PC02
- El IP del servidor: En mi caso 192.168.1.2
- Si el servidor se encuentra en la nube usted colocará el dominio:
 - En mi caso: Luchosoft.com
- Si el servidor es local (está instalado en la máquina donde se ejecuta SQL Server Management Studio) puede digitar:
 - Localhost
 - (local)
 - Si usted es el administrador del servidor: Puede colocar solamente **un punto** tal como se muestra en la siguiente imagen:



4. En **Autenticación** indique el tipo de autenticación con el que se identifica ante el servidor, y haga clic en **Conectar**. Mas adelante, en el capítulo sobre la seguridad en SQL Server se verá con más detalle los tipos de autenticación.

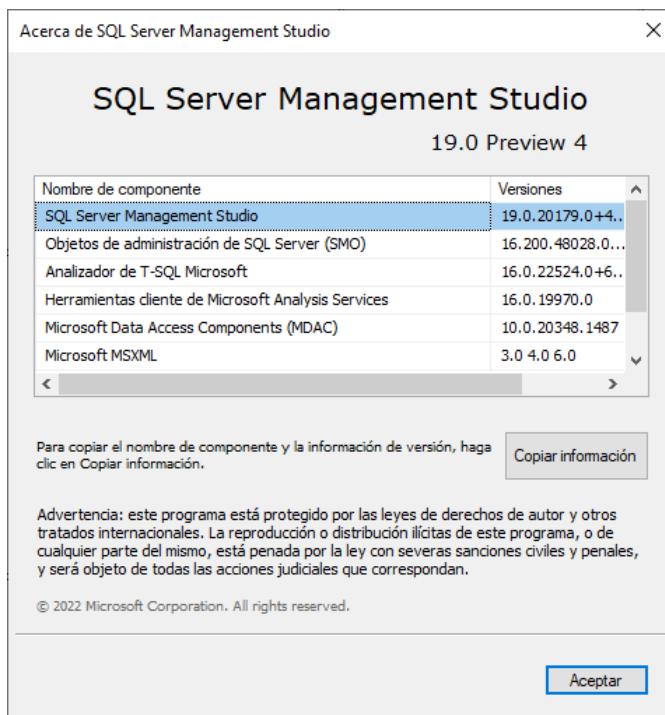


4.2. La caja de diáloGO Acerca de

La caja de diáloGO **Acerca de** proporciona información acerca de **SQL Server Management Studio** y de sus componentes relacionados.

Ejercicio 1.2: Consultar la caja de diáloGO Acerca de

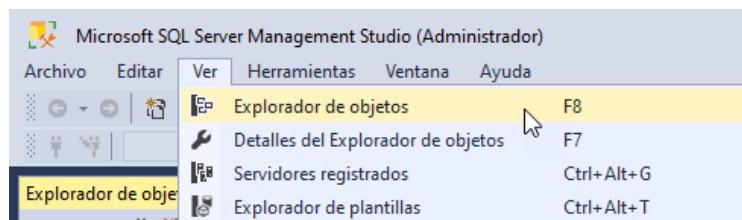
1. En el menú **Ayuda** haga clic en **Acerca de...**



2. Haga clic en **Aceptar** para cerrar la caja de diáloGO.

Ejercicio 1.3: Mostrar/ocultar los elementos de la interfaz de SQL Server Management Studio

1. Si no ve alguno de los elementos de la interfaz de **SQL Server Management Studio**, en el menú **Ver** busque la entrada con el nombre del elemento, y haga clic sobre ella. Por ejemplo, en el **Explorador de objetos** si está no se encuentra visible.



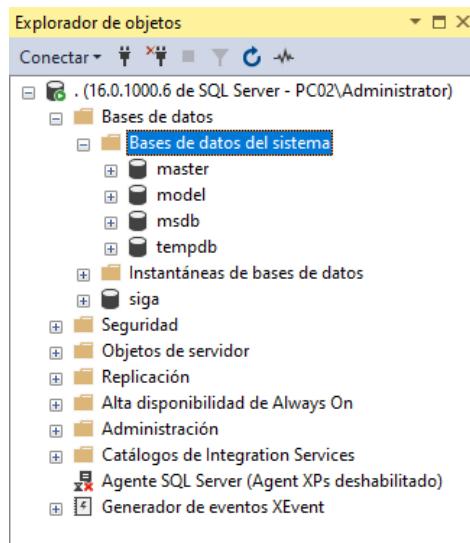
2. Para ocultar el elemento vaya nuevamente al menú **Ver**, o haga clic en el botón **Cerrar** si dispone de él.

4.3. Los paneles Explorador de objetos y Detalles del Explorador de objetos

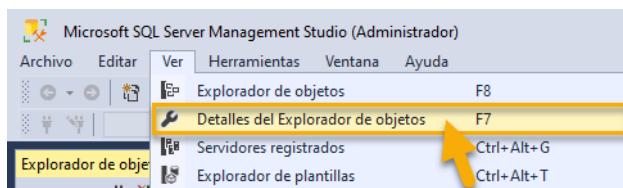
El panel **Explorador de objetos** permite explorar los diferentes objetos accedidos a través de una conexión, tales como **Bases de datos**, **Seguridad**, Objetos de servidor, Replicación Alta disponibilidad de Always On, Administración, etc. El panel **Detalles del Explorador de objetos** muestra el detalle del elemento seleccionado en el Explorador de objetos.

Ejercicio 1.4: Usar los paneles Explorador de objetos y Detalles del Explorador de objetos para ver la estructura de una base de datos

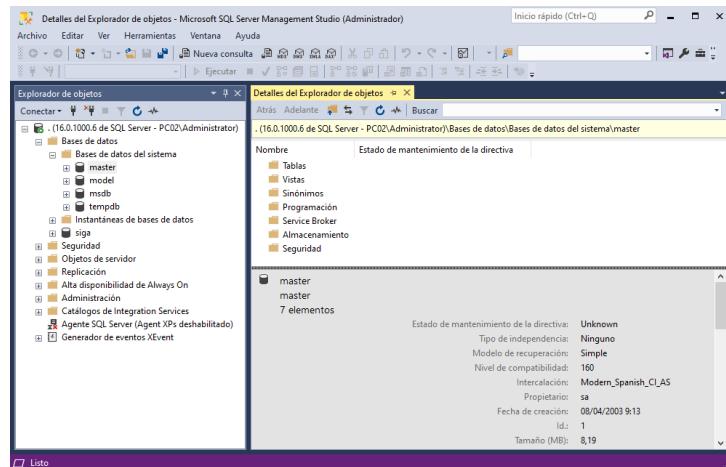
1. En el panel **Explorador de objetos** expanda el nodo **Bases de datos** y Luego expanda **Bases de datos del sistema**.



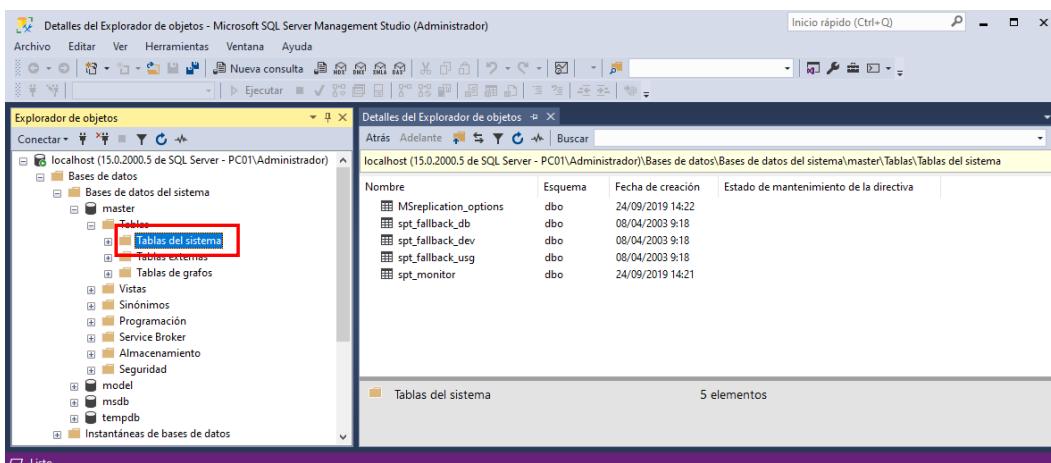
2. Seleccione una de las bases de datos disponible en su conexión, por ejemplo, la base de datos **master**.
3. En el menú principal, haga clic en **Ver/Detalles del Explorador de objetos**



4. En el panel **Detalles del Explorador de objetos** se muestra el listado de los componentes de la base de datos.



5. Expanda el nodo de la base de datos **master**, y Luego expanda la entrada **Tablas** y Luego haga clic en **Tablas del sistema**. En el panel de detalles se muestra la lista de las tablas de la base de datos.

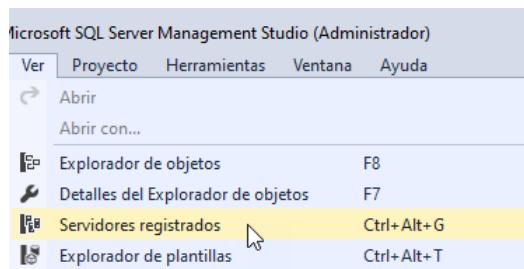


4.4. El panel Servidores registrados

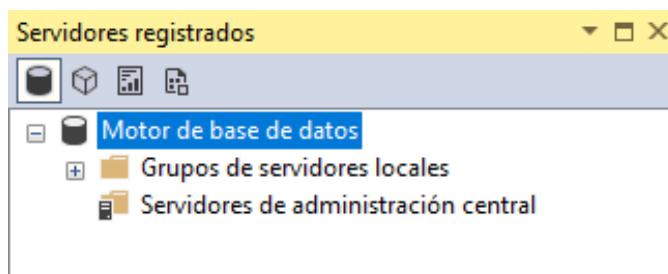
El panel **Servidores registrados** muestra los servidores registrados en el cliente SQL Server Management Studio. Permite registrar servidores, conectarse a un servidor registrado, desconectarse de un servidor registrado, cambiar las propiedades de registro de un servidor.

Ejercicio 1.5: Registrar un servidor en el panel Servidores registrados

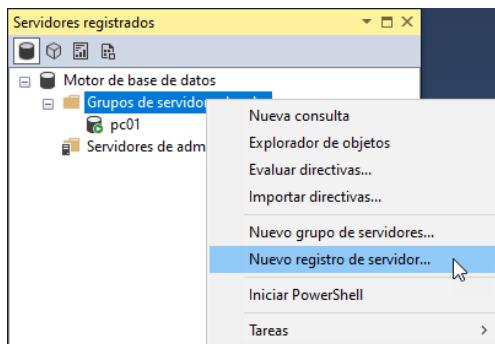
1. Ingrese al panel de Servidores registrados haciendo clic en **Ver** y Luego **Servidores Registrados**:



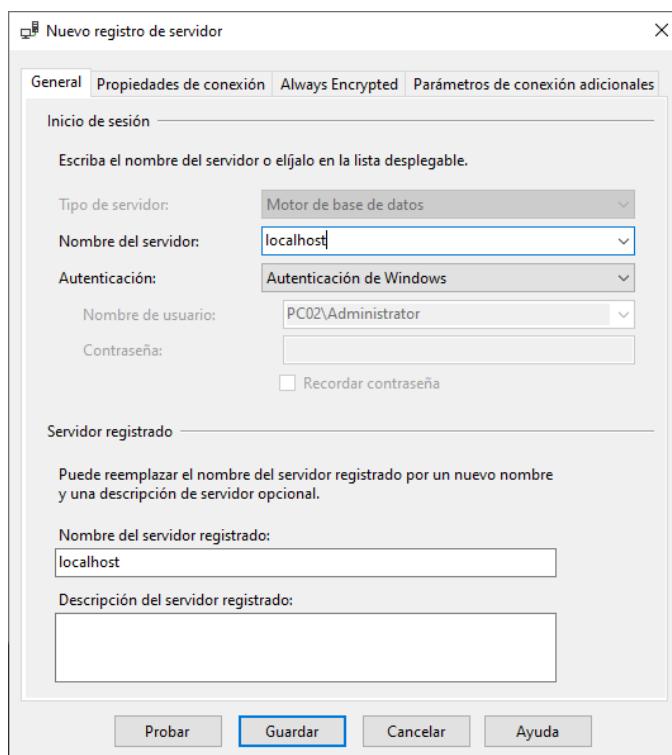
2. En el panel **Servidores registrados** expanda el nodo **Motor de base de datos**. Se muestran los nodos **Grupos de servidores locales** y **Servidores de administración central** para registrar un servidor local o un servidor al que se accede remotamente.



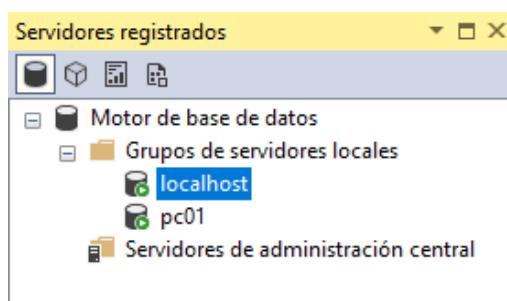
3. Vamos a registrar el servidor local instalado en la máquina con la que se ha desarrollado este manual. Haga un clic secundario sobre **Grupos de servidores locales**, y en el menú contextual ejecute **Nuevo registro de servidor**.



4. En **Nombre del servidor** seleccione **localhost**.
5. En **Autenticación** seleccione **Autenticación de Windows** (más adelante veremos los tipos de autenticación y su uso).
6. En **Nombre del servidor registrado** acepte el nombre sugerido (localhost) para el registro. Si desea, ingrese una descripción en **Descripción del servidor registrado**.



7. Haga clic en el botón **Probar** para probar la conectividad.
8. Haga clic en el botón **Guardar** para registrar el servidor.

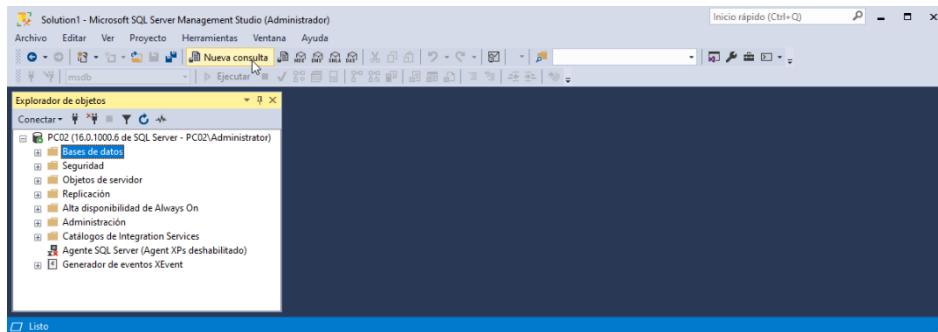


4.5. El panel de Código

Es el área para editar sus consultas o programas. Trabaja como un editor de textos que permite editar, guardar, recuperar y ejecutar su código de programación.

Ejercicio 1.6: Usar el panel de Código para ejecutar tareas de consultas

1. Haga clic en el botón **Nueva consulta** de la barra de herramientas de SQL Server Management Studio para abrir el panel de Código.

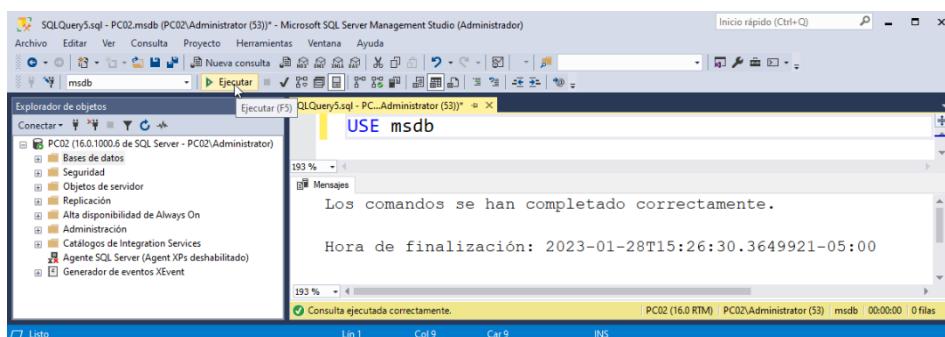


- En el panel digite la siguiente instrucción que establece la base de datos **msdb** como la base de datos con la que se desea trabajar:



- Haga clic en el botón **Ejecutar** de la barra de herramientas del editor, o pulse la tecla [F5]. Si la orden se ejecuta sin problemas, se mostrará el panel Mensajes con el mensaje:

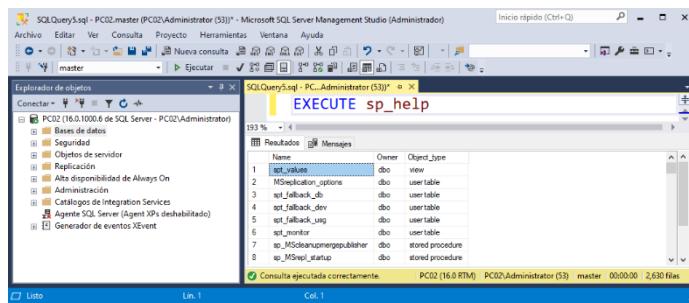
Los comandos se han completado correctamente.



- Añada en el editor la siguiente instrucción que mostrará la lista de objetos contenidos en la base de datos **model**:

EXECUTE sp_help

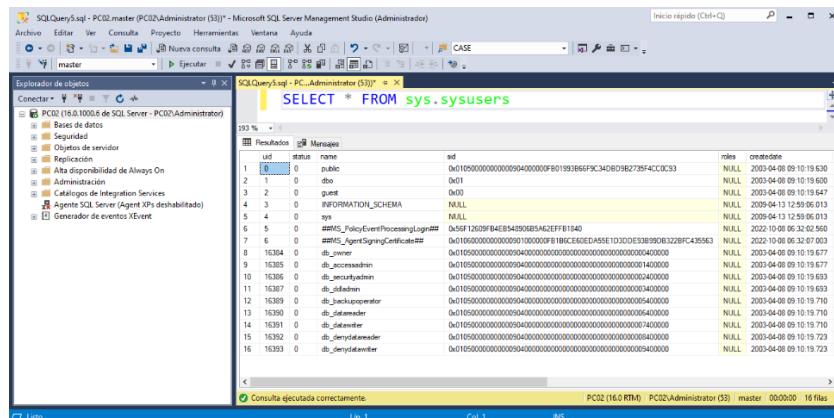
- Seleccione el comando que acaba de digitar, y haga clic en el botón **Ejecutar** de la barra de herramientas.
- En el panel **Resultados** se muestra la lista de objetos de la base de datos.



7. Añada la siguiente instrucción para mostrar los datos de la vista del sistema **sysusers** de la base de datos **msdb**:

```
SELECT * FROM sys.sysusers
```

8. Seleccione el comando y haga clic en el botón **Ejecutar**. El panel **Resultados** muestra el resultado.



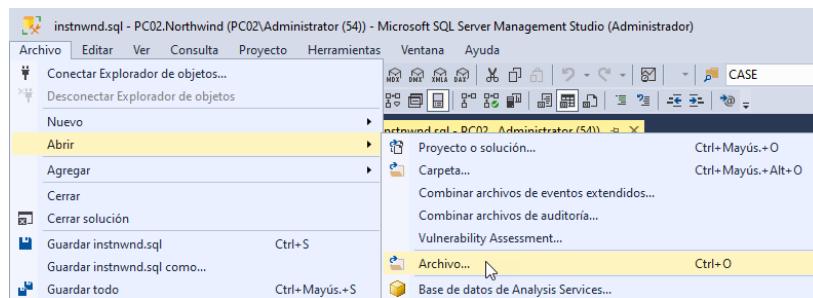
9. Si desea puede guardar estas instrucciones en un script de SQL (archivo de texto con extensión .SQL) para revisarlo y ejecutarlo nuevamente en otra oportunidad. Para ello, con el cursor ubicado en el panel de edición, haga clic en el botón **Guardar** de la barra de herramientas.

Ejercicio 1.7: Ejecutar un script de SQL en SQL Server Management Studio

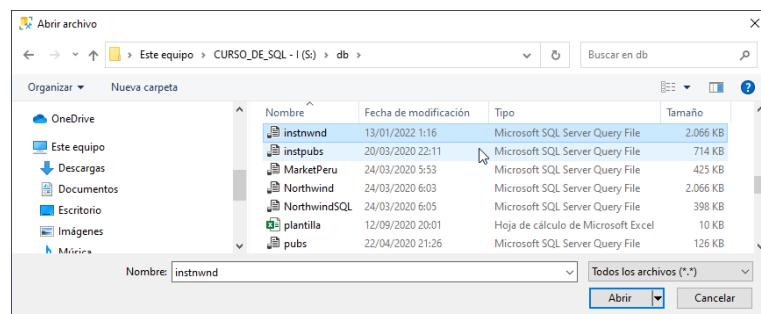
En este ejercicio ejecutaremos un script de SQL (programa de T-SQL) que crea la base de datos **Northwind**. Esta base de datos se proporcionaba como ejemplo en versiones anteriores de Microsoft SQL Server.

El archivo que contiene el script es **instnwnd.sql** y está ubicado en la carpeta **DB**.

1. En el menú **Archivo** ejecute la secuencia **Abrir, Archivo**. Se abre el diálogo **Abrir archivo**.



2. En el diálogo **Abrir archivo** seleccione la carpeta **DB**, y Luego seleccione el archivo **instnwnd.sql**.



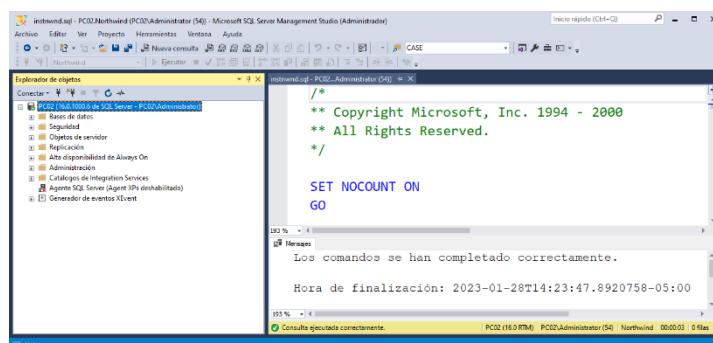
3. Haga clic en el botón **Abrir** para abrir el archivo en el editor de Código.

```
/*
** Copyright Microsoft, Inc. 1994 - 2000
** All Rights Reserved.
*/

SET NOCOUNT ON
GO

USE master
GO
if exists (select * from sysdatabases where name='Northwind')
    drop database Northwind
go
```

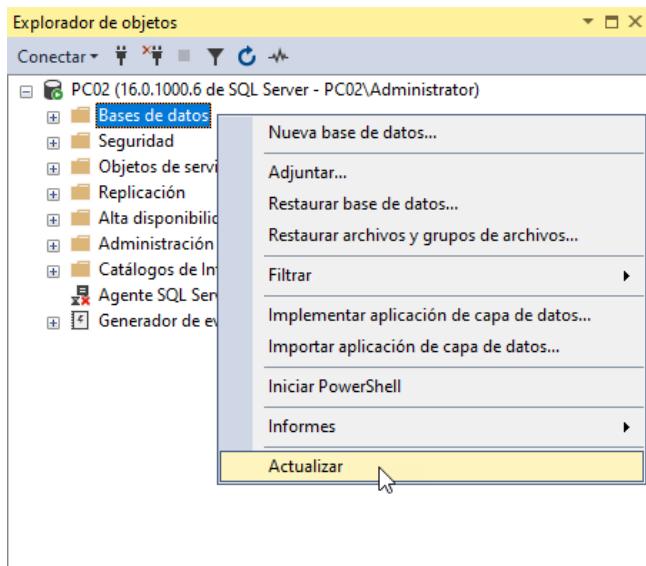
4. Haga clic en el botón **Ejecutar** de la barra de herramientas del editor de Código. Dependiendo de la velocidad de su procesador, la ejecución del script tomará varios segundos.



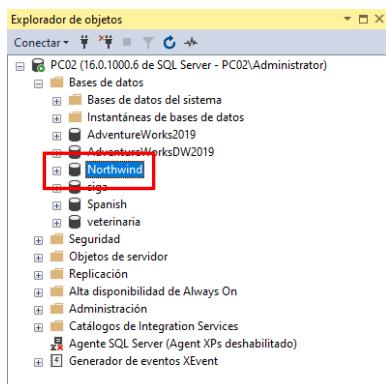
5. Cierre el editor de Código.



6. En el panel **Explorador de objetos** haga un clic secundario sobre el nodo **Bases de datos** y en el menú contextual haga clic en **Actualizar**.



7. En la lista de base de datos se muestra la base de datos **Northwind** que acabamos de crear.





5. EJERCICIOS PROPUESTOS

Estos ejercicios se plantean como un reto. Para resolverlos deberá utilizar las opciones de la interfaz gráfica de SQL Server Management Studio.

Mostrar:

1. La lista de procedimientos almacenados (stored procedures) guardados en la base de datos **Northwind**.
2. La lista de usuarios definidos en la base de datos **Northwind**.
3. El nombre y el número de versión del sistema operativo sobre el que corre su SQL Server.
4. El modo de autenticación del servidor (Server authentication) configurado para su SQL Server.
5. La ubicación del directorio predeterminado de los archivos de bases de datos.
6. El tamaño de la RAM de la máquina en la que ejecuta su SQL Server.
7. La lista de tablas contenidas en la base de datos **Northwind**.
8. La estructura (nombres y tipos de columnas) de la tabla **Customers** de **Northwind**.
9. El contenido (Los datos) de la tabla **Customers** de **Northwind**.
10. Las propiedades de la tabla **Customers** de **Northwind**.



Capítulo II

Bases de Datos de Microsoft SQL Server 2022

En este capítulo conoceremos qué es una base de datos, qué objetos forman parte de ella, qué son las bases de datos de sistema y una introducción al papel que cumplen en SQL Server, cómo se crea una base de datos SQL Server y qué tipos de archivos la conforman, así como la función de cada tipo de archivo.



1. ¿QUÉ ES UNA BASE DE DATOS?

Una base de datos es un conjunto organizado de datos almacenados electrónicamente y accesibles para su uso. Una base de datos permite la recuperación, modificación y administración de los datos de manera eficiente y segura.

Hay muchos tipos diferentes de bases de datos, incluyendo:

- **Bases de datos relacionales:** almacenan los datos en tablas relacionadas y utilizan un lenguaje de consulta estructurado (SQL) para acceder a los datos.
- **Bases de datos NoSQL:** permiten almacenar y acceder a los datos sin un esquema fijo y a menudo se utilizan para aplicaciones de big data y aplicaciones en tiempo real.
- **Bases de datos de documentos:** almacenan los datos en forma de documentos y se utilizan para aplicaciones que requieren flexibilidad en la estructura de los datos.
- **Bases de datos de grafos:** almacenan y permiten acceder a los datos en forma de nodos y relaciones y se utilizan para aplicaciones que requieren modelar relaciones complejas.

En cualquier caso, las bases de datos se utilizan en una amplia variedad de aplicaciones, desde sistemas de información empresarial hasta aplicaciones web y móviles. Su uso permite una mejor organización, accesibilidad y seguridad de los datos, lo que facilita la toma de decisiones y mejora la eficiencia en la gestión de la información.

1.1. Objetos de una base de datos Microsoft SQL Server

Una base de datos relacional está formada por objetos de diversos tipos, cada uno de los cuales cumple un papel específico dentro de la estructura de la base de datos.

Tablas (tables), son los objetos en los que se almacenan los datos. Una tabla tiene un formato fila-columna similar a las tablas o listas de Microsoft Excel. Cada fila es un registro único en la tabla (no existe una fila igual a otra en la misma tabla), y cada columna almacena los valores de propiedades específicas del registro. Por ejemplo, si tenemos una tabla EMPLEADO, cada fila almacena los datos de un único empleado, y en cada columna



podemos almacenar su apellido paterno, su apellido materno, su nombre, su fecha de nacimiento, su estado civil, número de hijos, etc.

Columnas (columns), son las partes de la tabla que almacenan los datos. A cada columna debe asignársele un tipo de dato y un nombre o identificador único. Por ejemplo, en la tabla EMPLEADO podemos definir las columnas: *paterno* de tipo cadena, *materno* de tipo cadena, *nombre* de tipo cadena, *fecha_nac* de tipo fecha, *estado_civil* de tipo cadena, *hijos* de tipo numérico, etc.

Tipos de datos (datatypes), en una base de datos podemos almacenar distintos tipos de datos tales como cadenas, fechas y números; incluso podemos almacenar datos binarios. Cada columna de una tabla debe contener solo un tipo de datos.

Claves principales (primary keys), éstas son un elemento muy importante en las bases de datos. Una clave principal es un valor único que permite identificar a un registro de la tabla, diferenciándolo de otros registros. La clave principal de una tabla se almacena en una columna de la misma. Por costumbre, suele ser la primera columna de la tabla.

Claves foráneas (foreign keys), son columnas cuyos valores hacen referencia a las claves principales de otras tablas o de la misma tabla. Las bases de datos relacionales utilizan las claves principales y las claves foráneas para relacionar los datos almacenados en tablas distintas cuando se hacen consultas.

(1) CATEGORIA		
IdCategoria	Nombre	Descripcion
1	GOLOSINAS	GALLETAS,CHOCOLATES,CARAMELOS,TOFFES
2	EMBUTIDOS	JAMONADAS,JAMONES,SALCHICHAS,CHORIZOS
3	HIGIENE PERSONAL	JABONES,P.DENTALES,SHAMPOOS,P.H.
4	LACTEOS	LECHES,YOGURES,QUESOS,MANTEQUILLAS
5	LICORES Y GASEOSAS	VINOS,WHISKIES,BEBIDAS GASIFICADAS
6	LIMPIEZA	DETERGENTES,DESINFECTANTES,ACCESORIOS

(3)



(1) PRODUCTO	(4) IdCategoria	(2) Nombre	UnidadMedida
IdProducto			
11	1	CHOCOLATE BARRA REGULAR	BARRA 2 ONZAS
65	3	JABON ROSAS Y LIMON BLANCO	UNIDAD X 105 GR
66	3	JABON ROSAS Y LIMON ROSADO	UNIDAD X 105 GR
101	4	MANTEQUILLA LAIVE C/SAL	PAQUETE 0.5 LB
102	4	MANTEQUILLA LAIVE	PAQUETE 114 GR
103	4	MANTEQUILLA FERM C/SAL	PAQUETE 227 GR
105	4	MARGARINA ASTRA	PAQUETE 230 GR
106	4	DORINA CLASICA	POTE 225 GR
10	1	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES
13	1	CHOCOLATE BARRA MILKY WAY	BARRA 2.15 ONZAS
76	3	P.H. BLANCO ROLL KLEENEX	PAQUETE X 2 UNIDADES
18	1	GALLETAS CHIPS AHOY	PAQUETE X 6 UNIDADES
2	1	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR
74	3	P.H. BLANCO SUAVE (ROJA)	PAQUETE X 4 UNIDADES
130	6	DETERGENTE LIMON INVICTO	BOLSA 225 GR

- (1) Identificador de tabla.
- (2) Identificador de columna.
- (3) Clave primaria de la tabla CATEGORIA.
- (4) Clave foránea de la tabla PRODUCTO que relaciona a cada producto con los datos de su categoría.

Restricciones (constraints), son elementos de la base de datos que definen mecanismos, con base en el servidor, para controlar la integridad de los datos. Las restricciones que podemos definir en una tabla de la base de datos son: clave principal (primary key), claves foráneas (foreign keys), valores únicos (unique), valores predeterminados (defaults), y reglas de validación (checks).

Procedimientos almacenados (stored procedures), son pequeños programas que se escriben en código Transact-SQL (el lenguaje de programación de Microsoft SQL Server) y se almacenan en el servidor bajo un nombre. Al ejecutar el procedimiento almacenado, se ejecuta el código Transact-SQL contenido en él. Podemos escribir código Transact-SQL que ejecuta un informe semanal, guardarla en la base de datos como un procedimiento almacenado, y ejecutar el procedimiento almacenado cada vez que se necesita generar informes. También pueden emplearse como mecanismos de seguridad.



Desencadenantes o disparadores (triggers), son una forma especial de los procedimientos almacenados que se ejecutan automáticamente cuando se agrega (INSERT), se modifica (UPDATE) o se elimina (DELETE) datos de una base de datos. También puede definirse triggers que se ejecutan cuando se modifica la definición de un objeto de la base de datos. La integridad de datos y las reglas de neGOcios se suelen definir mediante triggers. Por ejemplo, un trigger puede asegurar que todo producto en un almacén tenga una descripción, además del nombre.

Funciones definidas por el usuario (user-defined functions), adicionalmente a las funciones predefinidas proporcionadas por Transact-SQL, podemos crear nuestras propias funciones para ampliar la capacidad del lenguaje, y utilizarlas en nuestras soluciones de bases de datos.

Indices (indexes), son un elemento de la base de datos que cumple una función similar al índice alfabético o temático de un libro. Los índices pueden ayudarle a organizar los datos a efecto de que las consultas se ejecuten con mayor rapidez.

Estadísticas (statistics), representan la distribución estadística de los elementos de una columna o conjunto de columnas. Esta distribución es utilizada para establecer la selectividad de los datos en la columna o conjunto de columnas, o, en otras palabras, la eficiencia de un índice en la recuperación de datos.

Vistas (views), son consultas almacenadas en la base de datos que pueden hacer referencia a una o varias tablas. Pueden considerarse una forma especial de procedimiento almacenado que ejecuta una sola instrucción Transact-SQL, una sentencia SELECT. Puede crearlas y guardarlas a fin de utilizarlas con facilidad en el futuro. Se usan para ocultar a los usuarios columnas sensibles (columnas a las que no deben tener acceso); por lo regular las vistas excluyen ciertas columnas de una tabla, o bien, vinculan varias tablas.

Sinónimos (synonyms), permiten que un objeto pueda ser referenciado utilizando un nombre alternativo. A veces, para referirse a un objeto es necesario calificarlo completamente, lo que genera una referencia larga y compleja para el objeto. En este caso, podemos crear un sinónimo que permita establecer la referencia al objeto utilizando un nombre más corto y sencillo.



Tipos de datos definidos por el usuario (user-defined data types), son tipos de datos con nombre propio creados en base a los tipos de datos estándar de SQL Server. Se usan para estandarizar el proceso de creación de tablas de una base de datos.

Usuarios (users), son elementos de la base de datos que registran los permisos de quienes acceden a una base de datos. Una cuenta de usuario que accede a una base de datos obtiene sus permisos a través de un usuario de dicha base de datos.

2. BASES DE DATOS DEL SISTEMA

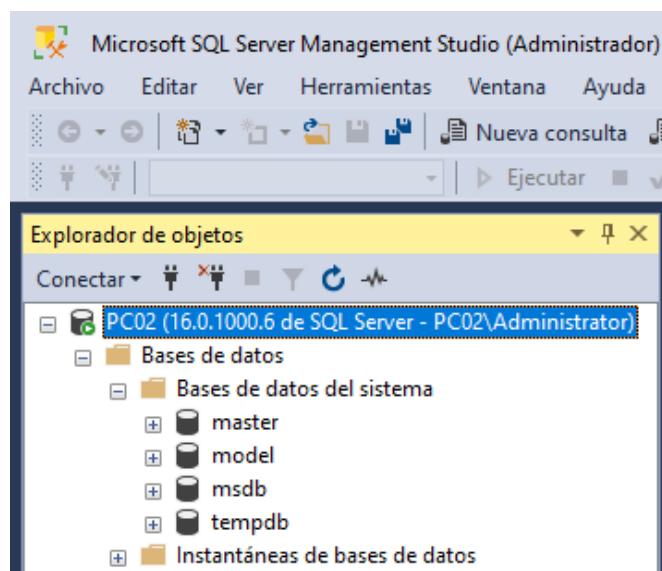
El proceso de instalación de Microsoft SQL Server crea en el servidor 4 bases de datos de sistema: master, model, msdb y tempdb.

Una base de datos de sistema contiene tablas de sistema; éstas contienen la metadata del sistema, es decir, los datos que permiten operar y administrar el sistema.

Las bases de datos que nosotros creamos se conocen como bases de datos de usuario, y contienen tablas de usuario (las tablas que creamos); estas tablas son las que almacenan nuestros datos. Una base de datos de usuario también contiene tablas de sistema que almacenan la metadata de la base de datos, es decir, los datos que permiten operar y administrar la base de datos.

Adicionalmente, el proceso de instalación crea dos bases de datos del Report Server. Estos datos como definición de reportes, reportes almacenados en caché, y la metadata de los reportes, que permiten operar y administrar los reportes.

La siguiente es una descripción breve del papel que cumple en SQL Server cada una de las bases de datos de sistema.



master: Es la principal base de datos del sistema. Controla los usuarios y las operaciones sobre el servidor manteniendo datos como cuentas de usuario, variables de entorno, mensajes de error del sistema, etc.

model: Proporciona una plantilla o modelo para cualquier base de datos nueva. Cuando se crea una base de datos, todo el contenido de la model se copia en la nueva base de datos. Permite personalizar el proceso de creación de bases de datos.

msdb: Almacena toda la data que utiliza el SQL Server Agent para programar alertas y trabajos, y para registrar operadores.

tempdb: Para almacenamiento de tablas temporales.

3. CREACIÓN DE BASES DE DATOS

Tal como hemos mencionado arriba, una base de datos es una colección de tablas que contienen datos, y además otros objetos como vistas, índices, procedimientos almacenados, desencadenantes, y usuarios, definidos para soportar las operaciones a ejecutar con los datos. La data almacenada en una base de datos está generalmente relacionada con un objetivo o proceso en particular, tales como control del inventario en un almacén, el registro del personal de una organización, o el control académico en una institución educativa.

3.1. Archivos que conforman una base de datos

Los datos y todos los objetos de la base de datos se almacenan en archivos de bases de datos. Una base de datos de Microsoft SQL Server está formada por mínimo 2 archivos: un archivo primario y un archivo de registro de transacciones. La base de datos puede tener además varios archivos secundarios y archivos de registro de transacciones adicionales.

Archivo primario

Es el archivo principal de una base de datos. Almacena las tablas de sistema, y puede también almacenar las tablas de usuario y otros objetos de la base de datos. Cada base de datos tiene solo un archivo primario, y se utiliza **.mdf** como la extensión en el identificador del archivo.

Archivos secundarios

Estos archivos se usan para almacenar las tablas de usuario y demás objetos de la base de datos. Cuando la base de datos es muy grande, se puede utilizar archivos secundarios para distribuir los objetos de modo que puedan administrarse mejor. Los archivos secundarios pueden estar almacenados en discos diferentes, lo que hace que el rendimiento entrada/salida de la base de datos mejore. De ser necesario, pueden estar incluso en computadoras distintas. La extensión de los archivos secundarios es **.ndf**.



Archivos de registro de transacciones (transaction log)

Este archivo registra todas las transacciones a ejecutar sobre la base de datos, y es un mecanismo de seguridad para recuperar la base de datos ante la eventualidad de una falla en el sistema. Una base de datos debe tener por lo menos un archivo de registro de transacciones. La extensión del archivo es **.ldf**.

3.2. Definición de transacción

Una transacción es un conjunto de operaciones de modificación de datos que se ejecuta sobre una base de datos y que debe ser procesado como una unidad. Si una transacción está formada, por ejemplo, por 20 operaciones, para que la transacción se considere válida se deben ejecutar las 20 operaciones; si una de las operaciones falla, la transacción se considera inválida, y por lo tanto, debe anularse.

Por ejemplo, se tiene las tablas **Factura**, **Detalle_factura** y **Producto** de una base de datos de ventas. La tabla **Producto** registra el nivel de inventario (stock) de cada producto.

Supongamos que se tiene que registrar la venta de 10 unidades del producto ABC. Para ello, la aplicación de base de datos procede de la siguiente manera:

Operación 1 Registra en la tabla **Factura**, los datos de la cabecera de la factura.

Operación 2 Registra en la tabla **Detalle_factura**, los detalles de la venta.

Operación 3 Actualiza el nivel de inventario del producto ABC en la tabla **Producto**.

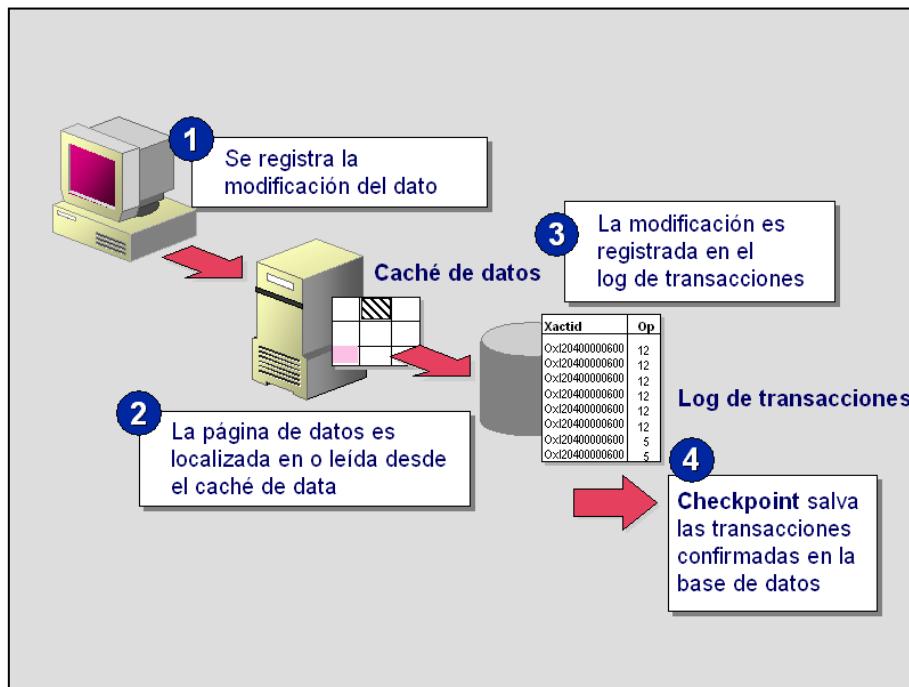
¿Qué ocurriría si durante la ejecución de la aplicación se completan las operaciones 1 y 2, y por una falla del sistema, la operación 3 no se lleva a cabo?

Si no se diseña un mecanismo para corregir el error, la base de datos perdería consistencia. Según las tablas **Factura** y **Detalle_factura** se habrían vendido 10 unidades del producto ABC, por lo que en la tabla **Producto** deberíamos tener 10 unidades menos. Esto último no es cierto al no haberse completado la operación 3.

En otras palabras, las operaciones 1, 2 y 3 forman una transacción, y si ésta no se completa, el sistema debe deshacer todas las operaciones. Una transacción asegura que las operaciones se llevarán a cabo completas para garantizar la consistencia de los datos, o en caso contrario, la transacción debe ser anulada.

3.3. El Registro de Transacciones como mecanismo para garantizar la consistencia de los datos

El archivo de registro de transacciones registra todas las operaciones llevadas a cabo por las sentencias INSERT, UPDATE y DELETE, y otras que producen modificaciones en los datos de la base de datos. El proceso se describe a continuación.



- (1) Una aplicación envía una modificación de datos al servidor SQL.
- (2) Las páginas de datos afectadas por la modificación son localizadas en el caché, o leídas desde el disco al caché, si no se encuentran en ella.
- (3) Cada sentencia de modificación de datos es registrada en el transaction log antes que el cambio se lleve a cabo en la base de datos.
- (4) El punto de control (checkpoint) escribe las transacciones confirmadas (committed) en la base de datos.

Si el sistema falla, el proceso de recuperación automático usa el registro de transacciones para recuperar todas las transacciones confirmadas y anula todas las transacciones incompletas.

3.4. Creación de una base de datos – La instrucción CREATE DATABASE

Debemos tener en cuenta lo siguiente antes de crear una base de datos:

- Solo pueden crear bases de datos las cuentas de inicio de sesión que pertenecen a los roles fijos de servidor sysadmin o **dbcreator**, o que tienen asignado el permiso para ejecutar el comando CREATE DATABASE.
- La cuenta de inicio de sesión que crea la base de datos se convierte en el usuario dueño (dbo) de la base de datos.
- Defina el nombre y estime el tamaño de la base de datos, y las propiedades de cada uno de los archivos en los que residirá la base de datos.

La siguiente es la sintaxis de la forma básica del comando CREATE DATABASE.

Sintaxis

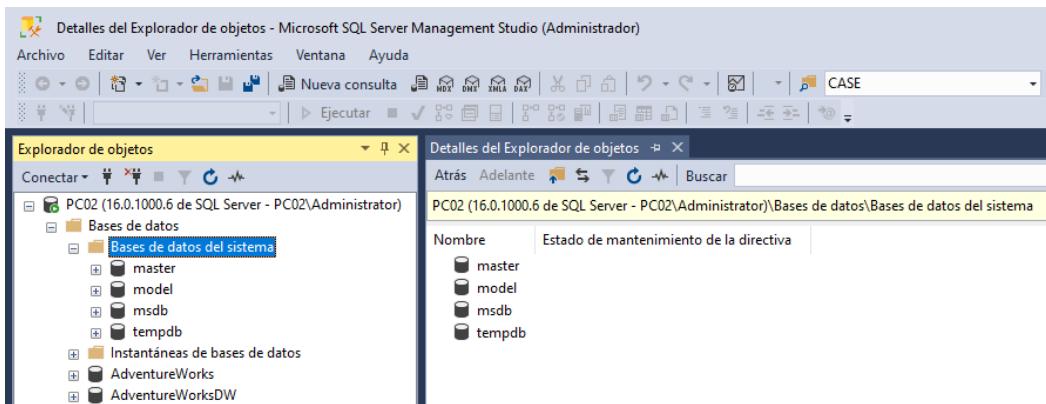
```
CREATE DATABASE nombre_basedatos
ON [ PRIMARY ] (
    NAME = nombre_lógico_data ,
    FILENAME = 'ubicación_y_nombre_archivo_data' ,
    SIZE = tamaño [KB|MB|GB|TB] ,
    MAXSIZE =
        tamaño_máximo [KB|MB|GB|TB|UNLIMITED] ,
    FILEGROWTH = incremento_crecimiento [KB|MB|%] )
LOG ON (
    NAME = nombre_lógico_log ,
    FILENAME = 'ubicación_y_nombre_archivo_log' ,
    SIZE = tamaño [KB|MB|GB|TB] ,
    MAXSIZE =
        tamaño_máximo [KB|MB|GB|TB|UNLIMITED] ,
    FILEGROWTH = incremento_crecimiento [KB|MB|%] )
```

- La cláusula ON PRIMARY define las propiedades del archivo primario. Cada una de las propiedades va separada de la anterior por una coma.
- La cláusula LOG ON define las propiedades del archivo de registro de transacciones.
- **nombre_lógico_data**, **nombre_lógico_log** es el nombre a utilizar cuando en una sentencia SQL se tiene que hacer referencia al archivo de datos o al archivo de log respectivamente.
- **ubicación_y_nombre_archivo** es una cadena que incluye la ruta y el nombre del archivo. La ruta debe especificar una carpeta existente en el servidor en el que está instalado SQL.
- **tamaño** especifica el tamaño inicial del archivo.
- **tamaño_máximo** es el máximo tamaño que puede alcanzar el archivo si se necesitara espacio adicional.

- **incremento_crecimiento** es la cantidad de espacio que se añade al archivo cada vez que se necesita espacio adicional. Se puede especificar como una magnitud constante en KB o MB, o como una tasa de crecimiento (%).

Ejercicio 2.1: Creación de una base de datos con las propiedades predeterminadas

5. En **SQL Server Management Studio** haga clic en el botón **Nueva consulta** de la barra de herramientas.



1. Se abre el panel de Código.

Nota: a partir de este punto, todos los ejercicios referidos a instrucciones Transact-SQL los ejecutaremos en el panel de Código.

2. En el panel de Código escriba las siguientes instrucciones:

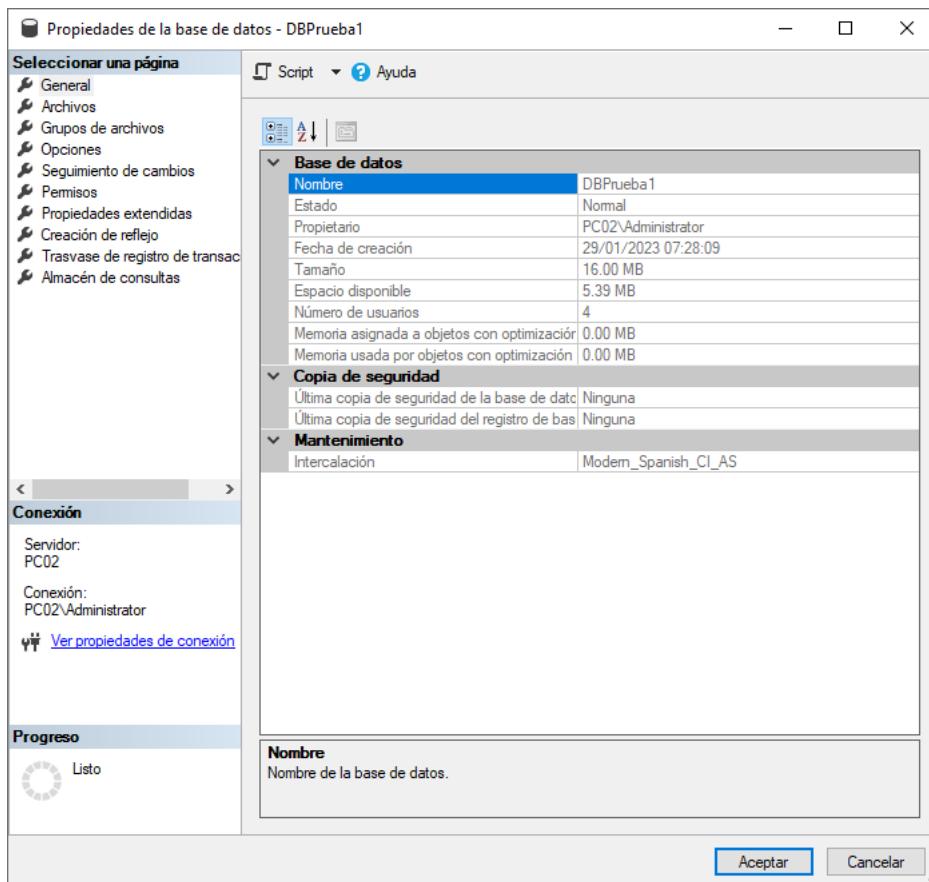
```
USE master
GO
```

```
CREATE DATABASE DBPrueba1
GO
```

3. Haga clic en el botón **Ejecutar** de la barra de herramientas, o pulse la tecla [F5]. Si las instrucciones se ejecutan sin error se muestra el siguiente mensaje:

Comandos completados correctamente.

4. En el **Explorador de objetos** haga un clic secundario sobre **Bases de datos**, y Luego clic en **Actualizar**. Se muestra la base de datos **DBPrueba1** recién creada.
5. Haga un clic secundario sobre el nombre de la base de datos **DBPrueba1**, y Luego clic en **Propiedades**. Se muestra la ventana de propiedades de la base de datos.



6. Todas las propiedades de la base de datos **DBPrueba1** se han copiado de las propiedades de la base de datos de sistema **model**. Seleccione la página **Archivos**. Observe que la ubicación predeterminada de los archivos de la base de datos es la carpeta **Data** en la carpeta de instalación de la instancia de SQL Server.
7. Cierre la ventana de propiedades de la base de datos.

Ejercicio 2.2: Creación de una base de datos especificando las propiedades de los archivos de la base de datos

1. En una nueva instancia del panel de Código digite y ejecute las siguientes instrucciones Transact-SQL:

```
USE master
```

```
GO
```

```
CREATE DATABASE DBPrueba2
ON PRIMARY (
    NAME = DBPrueba2_data ,
    FILENAME =
        'D:\SQLServer2022\Data\DBPrueba2.mdf' ,
```



```
SIZE = 8MB ,  
MAXSIZE = 15MB,  
FILEGROWTH = 1MB )  
LOG ON (  
NAME = DBPrueba2_log ,  
FILENAME =  
      'D:\SQLServer2022\Data\DBPrueba2_log.ldf' ,  
SIZE = 3MB ,  
MAXSIZE = 8MB,  
FILEGROWTH = 10% )  
GO
```

Nota: Se asume que la ruta **D:\SQLServer2022\Data** existe en el disco del servidor.

2. Cuando se crea una base de datos se añade una entrada con el nombre de la base de datos en la tabla de sistema **sysdatabases** de la base de datos de sistema **master**. Puede revisar la tabla **sysdatabases** ejecutando la siguiente consulta (para ejecutar solo esta consulta, selecciónela con el mouse, y Luego haga clic en **Ejecutar**):

```
SELECT name FROM sysdatabases  
WHERE name in ('DBPrueba1', 'DBPrueba2')  
GO
```

	name
1	DBPrueba1
2	DBPrueba2

4. PROPIEDADES Y OPCIONES DE CONFIGURACIÓN DE UNA BASE DE DATOS

Cuando creamos una base de datos, la mayoría de sus propiedades y opciones de configuración son tomadas de la base de datos de sistema **model**. Podemos revisar la configuración de la base de datos usando los procedimientos que mostraremos a continuación.

4.1. El procedimiento de sistema **sp_helpdb**

Este procedimiento entrega información acerca de todas las bases de datos en el servidor o de una base de datos específica.

Sintaxis

```
sp_helpdb [nombre_basedatos]
```

Si el procedimiento se ejecuta sin el argumento, entrega información de todas las bases de datos en la instancia de SQL Server.

Ejercicio 2.3: Propiedades de todas las bases de datos del servidor

1. En el panel de Código escriba y ejecute lo siguiente:

```
EXEC sp_helpdb
GO
```

2. Se muestra la lista de bases de datos en el servidor mostrando sus propiedades básicas.

	name	db_size	owner	dbid	created	status
1	AdventureWorks	336.00 MB	sa	6	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...
2	AdventureWorksDW	208.00 MB	sa	7	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...
3	BibliotecaGyS	40.00 MB	PC02\Administrator	11	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...
4	BookStore	41.00 MB	PC02\Administrator	12	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...
5	DBPrueba1	16.00 MB	PC02\Administrator	22	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...
6	DBPrueba2	11.00 MB	PC02\Administrator	23	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...
7	delttron	43.38 MB	PC02\Administrator	14	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...
8	Educa	40.00 MB	PC02\Administrator	13	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...
9	EduTec	40.00 MB	PC02\Administrator	15	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...
10	EurekaBank	40.00 MB	PC02\Administrator	16	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, User...

Ejercicio 2.4: Propiedades de una base de datos específica

1. En el panel de Código escriba y ejecute lo siguiente:

```
EXEC sp_helpdb DBPrueba1
GO
```

2. Ahora el panel de resultados muestra dos cuadros: una con las propiedades básicas de la base de datos **DBPrueba1**, y otra con las propiedades de sus archivos.

Resultados		Mensajes	
1	name	db_size	owner
1	DBPrueba1	16.00 MB	PC02\Administrator
1	dbid	created	status
1	22	Ene 29 2023	Status=ONLINE, Updateability=READ_WRITE, UserAcc...
1	compatibility_level		160
<hr/>			
1	name	fileid	filename
1	DBPrueba1	1	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...
2	DBPrueba1_log	2	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...
1	filegroup	size	maxsize
1	PRIMARY	8192 KB	Unlimited
2	NULL	8192 KB	2147483648 KB
1	growth	65536 KB	data only
2	usage	65536 KB	log only

4.2. Configuración de una base de datos – La instrucción ALTER DATABASE

Salvo que se especifique explícitamente al momento de crear la base de datos, ésta toma su configuración de la establecida en forma predeterminada en la base de datos de sistema **model**. El siguiente cuadro muestra algunas de las opciones de configuración de una base de datos.



Opción de base de datos	Descripción
autoclose	Cuando está en true , la base de datos es cerrada, y sus recursos liberados, después que el último usuario se desconecta.
autoshrink	Cuando está en true , los archivos de la base de datos reducen su tamaño automáticamente en forma periódica.
concat null yields null	Cuando está en true , si algunos de los operandos de una concatenación es NULL, el resultado es NULL.
dbo use only	Cuando está en true , solo el dueño de la base de datos la puede utilizar.
offline	Cuando está en true (on) , la base de datos está fuera de línea y no puede ser utilizada. Cuando está en false (off) , la base de datos está en línea.
read only	Cuando está en true , la base de datos es de solo lectura y no se permite ningún cambio en ella.
single user	Cuando está en true , solo un usuario a la vez puede acceder a la base de datos.

Ejercicio 2.5: Configuración de una base de datos

En este ejercicio haremos uso de la instrucción ALTER DATABASE para configurar la base de datos **DBPrueba1** como monousuario.

1. En el panel de Código escriba y ejecute lo siguiente:

```
ALTER DATABASE DBPrueba1
SET single_user
GO
```



2. Para verificar la base de datos usaremos la función **DatabasePropertyEx**.

```
SELECT DatabasePropertyEx(  
    'DBPrueba1', 'UserAccess')  
GO
```

La función devuelve como resultado el valor SINGLE_USER.

4.3. La función DatabasePropertyEx

Esta función devuelve el valor actual de una opción o propiedad de la base de datos especificada.

Sintaxis

```
DatabasePropertyEx( 'nombre_basedatos', 'propiedad' )
```

- **propiedad** es el nombre de la opción o propiedad de base de datos cuyo valor se desea conocer.



5. EJERCICIOS PROPUESTOS

Estos ejercicios le permitirán conocer su nivel de asimilación de los temas presentados en este capítulo.

1. Escriba una sentencia Transact-SQL que le permita crear la base de datos de nombre **Reto1** conformada por:

- un archivo primario de 10 MB, de 20 MB de tamaño máximo, y con un factor de crecimiento de 10%.
- un archivo de registro de transacciones de 5 MB, de 8 MB de tamaño máximo, y con un factor de crecimiento de 1 MB.

Ambos archivos estarán ubicados en la carpeta de datos predeterminada de su instancia de SQL Server.

2. Usando la interfaz gráfica de SQL Server Management Studio debe crear una base de datos de nombre **Reto2** conformada por:

- un archivo primario de 10 MB, sin límite de crecimiento, y con un factor de crecimiento de 10%.
- un archivo de registro de transacciones de 5 MB, de 8 MB de tamaño máximo, y con un factor de crecimiento de 10%.

Ambos archivos estarán ubicados en la carpeta **C:\SQLServer2022\Data** de su disco.



Capítulo III

Tablas de Microsoft SQL Server 2022

En este capítulo conoceremos qué es una tabla, cuáles son los tipos de datos disponibles en SQL Server, cómo se crea una tabla, cómo se asegura la integridad de datos y qué herramientas nos proporciona SQL Server para definirla.



1. ¿QUÉ ES UNA TABLA?

Una tabla es un objeto de la base de datos que se utiliza como almacenamiento de los datos. Una tabla tiene un formato fila-columna similar a las tablas o listas de Microsoft Excel. Cada fila es un registro único en la tabla (no existe una fila igual a otra en la misma tabla), y cada columna almacena los valores de propiedades específicas del registro. Por ejemplo, si tenemos una tabla EMPLEADO, cada fila almacena los datos de un único empleado, y en cada columna podemos almacenar su apellido paterno, su apellido materno, su nombre, su fecha de nacimiento, su estado civil y número de hijos, por lo que antes de crear una tabla se debe definir el tipo de dato que almacenará cada columna. Un tipo de dato define el tipo de información (cadena, número o fecha) que la columna almacena.

2. TIPOS DE DATOS EN MICROSOFT SQL SERVER

2022

Los tipos de datos de Microsoft SQL Server 2022 están organizados en las siguientes categorías:

- Numéricos exactos
- Numéricos aproximados
- Fecha y hora
- Cadenas de caracteres
- Cadenas de caracteres UNICODE
- Cadenas binarias
- Tipos especiales

2.1. Tipos de datos numéricos exactos

Los tipos de datos numéricos exactos nos permiten especificar de manera exacta la escala y precisión a utilizar para el dato. Por ejemplo, puede especificar tres dígitos a la derecha del decimal y cuatro a la izquierda. Una consulta siempre devuelve exactamente lo que ingresó. SQL Server soporta dos tipos de datos numéricos exactos compatibles con ANSI: **decimal** y **numeric**.



En general, se usan los datos numéricos exactos para aplicaciones financieras en las que se desea tener los datos de forma consistente, por ejemplo, siempre dos espacios decimales para evitar errores de redondeo.

Tipo de dato	Descripción
Bit	Entero. Solo puede tomar el valor 1, 0 o NULL.
Tinyint	Entero de 1 byte. Puede tomar los valores de 0 a 255.
smallint	Entero de 2 bytes. Puede tomar los valores de -2^{15} (-32.768) a $2^{15}-1$ (32.767).
Int	Entero de 4 bytes. Puede tomar los valores de -2^{31} (-2.147.483.648) a $2^{31}-1$ (2.147.483.647).
Bigint	Entero de 8 bytes. Puede tomar los valores de -2^{63} (-9.223.372.036.854.775.808) a $2^{63}-1$ (9.223.372.036.854.775.807).
decimal(<i>p</i>, <i>s</i>)*	p es la precisión, y va de 1 a 38, siendo 18 el valor predeterminado. s es la escala y va de 0 hasta p.
numeric(<i>p</i>, <i>s</i>)*	p es la precisión, y va de 1 a 38, siendo 18 el valor predeterminado. s es la escala y va de 0 hasta p.
smallmoney	Valor monetario de 4 bytes de - 214.748,3648 a 214.748,3647.
money	Valor monetario de 8 bytes de -922,337,203,685.477,5808 a 922,337,203,685.477,5807.

- * **p** es la precisión y determina el número máximo de dígitos tanto a la izquierda como a la derecha del punto decimal. **s** es la escala y determina el número máximo de dígitos a la derecha del punto decimal.

2.2. Tipos de datos numéricos aproximados

Los tipos de datos numéricos aproximados almacenan los datos sin precisión. Por ejemplo, el fragmento 1/3 se representa en un sistema decimal como 0,33333.... El número no puede guardarse con precisión, por lo que se almacena una aproximación del valor.

Se usan en las aplicaciones científicas en las que la cantidad de decimales de un valor suele ser muy grande.

Tipo de dato	Descripción
float(n)	<p>Si n es de 1 a 24, numérico de 4 bytes con una precisión de 7 dígitos.</p> <p>Si n es de 25 a 53, numérico de 8 bytes con una precisión de 15 dígitos. El valor predeterminado de n es 53.</p> <p>Los valores van de - 1,79E+308 a -2,23E-308, 0 y de 2,23E-308 a 1,79E+308.</p>
real ó float(24)	Numérico de 4 bytes. Los valores van de - 3,40E + 38 a -1,18E - 38, 0 y de 1,18E - 38 a 3,40E + 38.

2.3. Tipos de datos de fecha y hora

La fecha y hora pueden almacenarse en un tipo de dato **datetime** o bien en uno **smalldatetime**. Hasta la versión anterior (SQL Server 2008), la fecha y hora se debían almacenar siempre juntas en un solo valor. Ahora es posible almacenar solo la fecha (tipo de dato **date**) o solo la hora (tipo de dato **time**); estos nuevos tipos de datos, además de los tipos **datetime2** y **datetimeoffset** se alinean con el estándar SQL ANSI.

Los datos de fecha y hora pueden tomar varios formatos diferentes. Puede especificar el mes utilizando el nombre completo o una abreviatura. Se ignora el uso de mayúscula/minúscula y las comas son opcionales.

Los siguientes son algunos de los ejemplos de los formatos alfabéticos para el 15 de abril de 2010.

```
'Abr 15 2010'  
'Abr 15 10'  
'Abr 10 15'
```



'15 Abr 10'
'2010 Abril 15'
'2010 15 Abril'

También puede especificar el valor ordinal del mes. El valor ordinal de un elemento es el valor posicional dentro de una lista de elementos. En los ejemplos anteriores abril es el cuarto mes del año, así que puede usar el número 4 para su designación.

Los siguientes son algunos ejemplos que usan el valor ordinal para el 15 de abril de 2010.

4/15/10 (mm/dd/aa)
4/10/15 (mm/aa/dd)
15/10/04 (dd/aa/mm)
10/15/04 (aa/mm/dd)

Tipo de dato	Descripción
datetime	Fecha en el rango del 1 de enero de 1753 al 31 de diciembre de 9999, con intervalo de horas de 00:00:00 a 23:59:59.997. Longitud de 8 bytes.
smalldatetime	Fecha en el rango del 1 de enero de 1900 al 6 de junio del 2079, con intervalo de horas de 00:00:00 a 23:59:59. Longitud de 4 bytes.
Date	Fecha en el rango del 1 de enero del año 1 al 31 de diciembre de 9999. Longitud de 3 bytes.
datetime2(p)	Fecha en el rango del 1 de enero del año 1 31 de diciembre de 9999, con intervalo de horas de 00:00:00 a 23:59:59.999999; p va de 0 a 7 dígitos, con una precisión de 100 ns. La precisión predeterminada es 7 dígitos. La longitud es 6 bytes para precisiones inferiores a 3, 7 bytes para precisiones 3 y 4. Todas las demás precisiones requieren 8 bytes.
time(p)	Hora en el intervalo de 00:00:00.0000000 a 23:59:59.999999. p es un valor entero de 0 a 7 que especifica el número de dígitos de la parte fraccionaria de los segundos. La precisión predeterminada de las fracciones es 7 (100 ns).



datetimeoffset(p)	Fecha y hora con reconocimiento de zona horaria en el rango del 1 de enero del año 1 31 de diciembre de 9999, con intervalo de horas de 00:00:00 a 23:59:59.9999999; p va de 0 a 7 dígitos, con una precisión de 100 ns. La precisión predeterminada es 7 dígitos. La longitud es 6 bytes para precisiones inferiores a 3, 7 bytes para precisiones 3 y 4. Todas las demás precisiones requieren 8 bytes.
----------------------------	--

2.4. Tipos de datos de cadenas de caracteres

Estos tipos de datos se usan para almacenar cadenas de caracteres no Unicode.

Tipo de dato	Descripción
char(n) ó character(n)	Cadena de longitud fija (n puede ser de 1 a 8000 bytes).
varchar(n max) ó character varying(n max)	Cadena de longitud variable (n puede ser de 1 a 8000 bytes). max indica la longitud máxima que es $2^{31} - 1$ bytes.
text *	Cadena de longitud variable con un máximo de $2^{31} - 1$ caracteres.

- * **text** no será considerado en versiones futuras de SQL Server. En lugar de ella utilice **varchar(max)**.

2.5. Tipos de datos de cadenas de caracteres UNICODE

Estos tipos de datos se utilizan para almacenar cadenas de caracteres UNICODE.

Tipo de dato	Descripción
nchar(n) ó national character (n)	Cadena UNICODE de longitud fija (n puede ser de 1 a 4000 caracteres). La longitud de almacenamiento es $2n$.
nvarchar(n max) ó national character varying (n max)	Cadena UNICODE de longitud variable (n puede ser de 1 a 4000 caracteres). max indica la longitud



	máxima que es $2^{31} - 1$ bytes. La longitud de almacenamiento es 2 veces la real + 2 bytes.
ntext ó national text *	Cadena UNICODE de longitud variable con un máximo de $2^{31} - 1$ caracteres.

- * **ntext** no será considerado en versiones futuras de SQL Server. En lugar de ella utilice **nvarchar(max)**.

2.6. Tipos de datos de cadenas binarias

Tipos de datos para almacenamiento de datos binarios.

Tipo de dato	Descripción
binary(n)	Dato binario de longitud fija (n puede ser de 1 a 8000 bytes).
varbinary(n max) ó binary varying(n max)	Dato binario de longitud variable (n puede ser de 1 a 8000 bytes). max indica la longitud máxima que es $2^{31} - 1$ bytes.
image *	Dato binario de longitud variable que va desde 0 hasta $2^{31} - 1$ bytes.

- * **image** no será considerado en versiones futuras de SQL Server. En lugar de ella utilice **varbinary(max)**.

2.7. Tipos de datos especiales

Tipo de dato	Descripción
cursor *	Tipo de dato para variables o parámetros de salida de procedimientos almacenados.
hierarchyid	Cadena de longitud variable que representa la posición en una jerarquía.
sql_variant	Almacena un valor que puede ser de cualquier tipo soportado por SQL Server, excepto text , ntext , image , timestamp , y sql_variant .

table *	Define una tabla temporal para almacenar un conjunto de resultados que será utilizado posteriormente.
rowversion **	Generación automática de números binarios de modo que son únicos dentro de una base de datos.
uniqueidentifier	Genera un GUID de 16 bytes. Se usa para crear identificadores únicos en el sistema.
xml(xml_esquema)	Almacena data XML.

* **table** y **cursor** no se pueden utilizar para definir columnas con el comando CREATE TABLE:

** El tipo **timestamp** de versiones anteriores se ha reemplazado por el tipo **rowversion**.

3. CREACIÓN DE TABLAS EN MICROSOFT SQL SERVER 2022

Para ilustrar este tema presentaremos la base de datos que usaremos para los ejemplos a lo larGO del libro. En este capítulo describiremos el escenario del caso a estudiar, crearemos la base de datos y sus tablas y relaciones.

3.1. Presentación del caso a estudiar

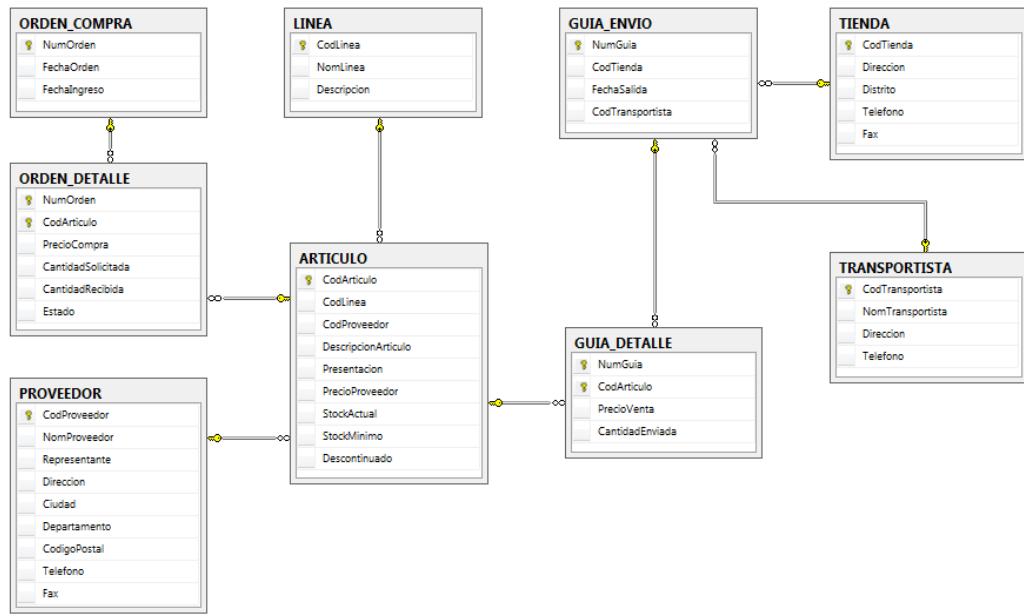
QhatuPERU es una empresa dedicada a la comercialización de productos de consumo masivo que cuenta con una cadena de puntos de venta localizados en varias zonas de Lima Metropolitana.

Los distintos locales de la cadena y las áreas administrativas de la empresa se interconectarán mediante una red metropolitana. Se desea diseñar e implantar las aplicaciones comerciales que permitirán gestionar las operaciones de la empresa. Para efectos del presente libro delimitaremos el área de estudio a las operaciones que se ejecutan en el Almacén Central, por lo que la base de datos se diseñará para registrar dichas operaciones, y estará habilitada para que todas las áreas administrativas la utilicen.

Almacén Central controlará todas las operaciones de entrada y salida de artículos (el inventario), y las otras áreas podrán efectuar consultas a la base de datos.



A continuación, se presenta el diagrama del modelo de datos propuesto para Almacén Central.



La base de datos **QhatuPERU** estará formada por las siguientes tablas:

TIENDA: relación de los puntos de venta de la empresa. Para cada tienda almacena los siguientes datos:

- CodTienda, identificador de la tienda o punto de venta; dato obligatorio de tipo numérico entero.
- Direccion, ubicación de la tienda; cadena de longitud variable de hasta 60 caracteres.
- Distrito, ubicación de la tienda; cadena de longitud variable de hasta 20 caracteres.
- Telefono, número telefónico de la tienda; cadena de longitud variable de hasta 15 caracteres.
- Fax, número de fax de la tienda; cadena de longitud variable de hasta 15 caracteres.

LINEA: relación de las distintas líneas de artículos que comercializa la empresa. Para cada línea se tiene los siguientes datos:

- CodLinea, identificador de la línea de artículos; dato obligatorio de tipo numérico entero autogenerado.
- NomLinea, nombre de la línea de artículos; dato obligatorio de tipo cadena de longitud variable de hasta 20 caracteres.

- Descripcion, descripción de la línea de artículos; cadena de longitud variable de hasta 40 caracteres.

PROVEEDOR: relación de los proveedores de los artículos que comercializa la empresa.

Para cada proveedor se registra los siguientes datos:

- CodProveedor, identificador del proveedor; dato obligatorio de tipo numérico entero autogenerado.
- NomProveedor, nombre del proveedor; dato obligatorio de tipo cadena de longitud variable de hasta 40 caracteres.
- Representante, nombre de la persona de contacto del proveedor; cadena de longitud variable de hasta 30 caracteres.
- Direccion, ubicación del proveedor; cadena de longitud variable de hasta 60 caracteres.
- Ciudad, ubicación del proveedor; cadena de longitud variable de hasta 15 caracteres.
- Departamento, ubicación del proveedor; cadena de longitud variable de hasta 15 caracteres.
- CodiGOPostal, ubicación del proveedor; cadena de longitud variable de hasta 15 caracteres.
- Telefono, número telefónico del proveedor; cadena de longitud variable de hasta 15 caracteres.
- Fax, número de fax del proveedor; cadena de longitud variable de hasta 15 caracteres.

ARTICULO: relación de los artículos que comercializa la empresa. Para cada artículo se registra los siguientes datos:

- CodArticulo, identificador del artículo; dato obligatorio de tipo numérico entero autogenerado.
- CodLinea, identificador de la línea a la que pertenece el artículo; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente línea.
- CodProveedor, identificador del proveedor del artículo; dato obligatorio de tipo numérico entero que relaciona al artículo con su proveedor.
- DescripcionArticulo, nombre del artículo; dato obligatorio de tipo cadena de longitud variable de hasta 40 caracteres.
- Presentacion, forma en la que se presenta el artículo; cadena de longitud variable de hasta 30 caracteres.
- PrecioProveedor, precio al que se le compra el artículo al proveedor; dato numérico de tipo dinero (money).
- StockActual, cantidad del artículo existente en el almacén; dato numérico entero pequeño.
- StockMinimo, cantidad mínima que debe existir en el almacén para atender los pedidos de las tiendas; dato numérico entero pequeño.
- Descontinuado, dato de tipo bit que indica si el artículo ha dejado de ser comercializado por la empresa.

ORDEN_COMPRA: relación de los ingresos de artículos al almacén. Para cada orden de compra se registra los siguientes datos:

- NumOrden, identificador de la orden de compra; dato obligatorio de tipo numérico entero.
- FechaOrden, fecha de la orden de compra; dato obligatorio de tipo fecha-hora.
- FechalIngreso, fecha de ingreso de los artículos al almacén; dato de tipo fecha-hora.

ORDEN_DETALLE: relación de los artículos asociados a cada orden de compra. Para cada artículo se registra los siguientes datos:

- NumOrden, número de la orden de compra a la que pertenece el artículo ingresado al almacén; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente orden de compra.
- CodArticulo, identificador del artículo recibido con la orden de compra especificada en NumOrden; dato obligatorio de tipo numérico entero que relaciona al ítem en la orden de compra con los datos complementarios del artículo.
- PrecioCompra, precio de compra del artículo; dato obligatorio de tipo dinero (money).
- CantidadSolicitada, cantidad que se le solicitó al proveedor en la correspondiente orden de compra; dato obligatorio de tipo numérico entero pequeño.
- CantidadRecibida, cantidad que recibió del proveedor y se ingresó al almacén; dato numérico entero pequeño.
- Estado, estado del artículo en la orden de compra; cadena de longitud variable de hasta 10 caracteres.

TRANSPORTISTA: relación de las personas encargadas de llevar los artículos desde el almacén hasta cada una de las tiendas. Para cada transportista se registra los siguientes datos:

- CodTransportista, identificador del transportista; dato obligatorio de tipo numérico entero.
- NomTransportista, nombre del transportista; dato obligatorio de tipo cadena de longitud variable de hasta 30 caracteres.
- Direccion, dirección del transportista; cadena de longitud variable de hasta 60 caracteres.
- Telefono, número telefónico del transportista; cadena de longitud variable de hasta 15 caracteres.

GUIA_ENVIO: relación de las salidas de artículos al almacén. Para cada guía de envío se registra los siguientes datos:

- NumGuia, identificador de la guía de envío; dato obligatorio de tipo numérico entero.

- CodTienda, identificador de la tienda a la que se envía los artículos; dato obligatorio de tipo numérico entero que relaciona a la guía de envío con la correspondiente tienda.
- FechaSalida, fecha de la guía de envío; dato obligatorio de tipo fecha-hora.
- CodTransportista, identificador del transportista responsable de trasladar el envío; dato obligatorio de tipo numérico entero que relaciona la guía de envío con el transportista a cargo del envío.

GUIA_DETALLE: relación de los artículos asociados a cada guía de envío. Para cada artículo se registra los siguientes datos:

- NumGuia, número de la guía de envío a la que pertenece el artículo que ha salido del almacén; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente guía de envío.
- CodArticulo, identificador del artículo entregado con la guía de envío especificada en NumOrden; dato obligatorio de tipo numérico entero que relaciona al ítem en la guía de envío con los datos complementarios del artículo.
- PrecioVenta, precio de venta al público del artículo en la tienda; dato obligatorio de tipo dinero (money).
- CantidadEnviada, cantidad que se envió a la tienda desde el almacén; dato obligatorio de tipo numérico entero pequeño.

Ejercicio 3.1: Creación de la base de datos QhatuPERU

3. En **SQL Server Management Studio** conéctese a su servidor SQL y abra una nueva consulta.

4. Ejecute las siguientes instrucciones:

```
USE master
GO
```

```
CREATE DATABASE QhatuPERU
GO
```

3.2. Creación de una tabla – La instrucción CREATE TABLE

Sintaxis básica

```
CREATE TABLE nombre_tabla (
    nombre_columna1 tipo_dato1 [NULL|NOT NULL] ,
    nombre_columna2 tipo_dato2 [NULL|NOT NULL] ,
    nombre_columna3 tipo_dato3 [NULL|NOT NULL] ,
    ...
)
```

- NULL|NOT NULL establece si el dato a ingresar en la columna es obligatorio u opcional. Por ejemplo, para los datos de un trabajador podemos establecer que su nombre es obligatorio registrarlo en la tabla, por lo tanto definiríamos la columna **nombreTrabajador** con la propiedad NOT NULL. Adicionalmente, como no todos los trabajadores cuentan con un teléfono celular, definiríamos la columna **telefonoMovil** con la propiedad NULL para indicar que este dato no es obligatorio registrarlo.

Si para una columna no se especifica si es NULL o NOT NULL, el valor predeterminado de la propiedad depende de la opción de configuración **ANSI null default** de la base de datos. Es recomendable establecer de forma explícita el valor de la propiedad; de este modo evitamos conflictos si la configuración de la base de datos cambia.

Significado del valor NULL

Si una columna de una tabla tiene la propiedad NULL, y no se especifica el valor a almacenar en dicha columna, el motor de datos almacena en ella el valor NULL. Con este valor estamos indicando simplemente que para dicha fila el valor de esa columna es desconocido o no está disponible. Un valor NULL no es lo mismo que una cadena de longitud cero, o una cadena de espacios, o el valor 0 (cero). Todos estos representan un dato conocido, el valor NULL significa dato desconocido.

Ejercicio 3.2: Creación de una tabla especificando solo sus columnas – Creación de la tabla TIENDA

Según el modelo de datos visto anteriormente, la tabla TIENDA tiene la siguiente definición:

- CodTienda, identificador de la tienda o punto de venta; dato obligatorio de tipo numérico entero.
- Direccion, ubicación de la tienda; cadena de longitud variable de hasta 60 caracteres.
- Distrito, ubicación de la tienda; cadena de longitud variable de hasta 20 caracteres.
- Telefono, número telefónico de la tienda; cadena de longitud variable de hasta 15 caracteres.
- Fax, número de fax de la tienda; cadena de longitud variable de hasta 15 caracteres.

Ejecute las siguientes instrucciones:

```
USE QhatuPERU
GO
```



```
CREATE TABLE TIENDA (
    CodTienda      int NOT NULL ,
    Direccion      varchar(60) NULL ,
    Distrito       varchar(20) NULL ,
    Telefono        varchar(15) NULL ,
    Fax            varchar(15) NULL )
GO
```

3.3. Consulta de la definición de una tabla – El procedimiento sp_help

El procedimiento de sistema **sp_help** genera un reporte que muestra la definición de un objeto de la base de datos activa. Este procedimiento puede ser ejecutado por cualquier usuario de la base de datos.

Sintaxis

```
sp_help nombre_objeto_basedatos
```

Ejercicio 3.3: Verificación de la definición de la tabla TIENDA

En la ventana query ejecute la siguiente instrucción:

```
sp_help TIENDA
GO
```

Resultados									
Mensajes									
Name	Owner	Type	Created_datetime						
TIENDA	dbo	user table	2013-01-28 21:52:29.340						
Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1 CodTienda	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2 Direccion	varchar	no	60			yes	no	yes	Modem_Spanish_CI_AS
3 Distrito	varchar	no	20			yes	no	yes	Modem_Spanish_CI_AS
4 Telefono	varchar	no	15			yes	no	yes	Modem_Spanish_CI_AS
5 Fax	varchar	no	15			yes	no	yes	Modem_Spanish_CI_AS
Identity	Seed	Increment	Not For Replication						
1 No identity column defined.	NULL	NULL	NULL						
RowGuidCol									
1 No rowguidcol column defined.									
Data_located_on_filegroup									
1 PRIMARY									

El reporte generado muestra:

- Las propiedades principales de la tabla.
- La definición de cada una de sus columnas.



- Si la tabla tiene definida una columna IDENTITY. Una columna IDENTITY es una columna de tipo numérica entera cuyo valor es generado por el sistema en forma secuencial siguiendo una regla de incremento específica. Se verá más adelante.
- Si la tabla tiene definida una columna RowGuidCol. Una columna RowGuidCol es una columna de tipo uniqueidentifier que almacena un GUID (Global Unique Identifier) de 16 bytes como identificador único en el Sistema.
- El grupo de archivos en el que está localizada la tabla. De modo predeterminado, la base de datos y sus objetos se localizan en el grupo de archivos PRIMARY.

4. MODIFICACIÓN DE LA DEFINICIÓN DE UNA TABLA

4.1. La instrucción ALTER TABLE

Sintaxis

```
ALTER TABLE nombre_tabla
    ADD nombre_columna propiedades_columna
    | DROP COLUMN nombre_columna
    | ALTER COLUMN nombre_columna
        nuevas_propiedades_columna
    | ADD CONSTRAINT nombre_restricción
        PRIMARY KEY... | UNIQUE... | FOREIGN KEY...
        | DEFAULT... | CHECK...
    | DROP CONSTRAINT nombre_restricción
```

- ADD **nombre_columna** permite crear una nueva columna en la tabla.
- DROP COLUMN se utiliza para eliminar una columna.
- ALTER COLUMN permite cambiar las propiedades de una columna.
- ADD CONSTRAINT permite añadir una restricción PRIMARY KEY, UNIQUE, FOREIGN KEY, DEFAULT o CHECK a la definición de una tabla. Una restricción define las reglas que debe cumplir un dato para ser aceptable.
- DROP CONSTRAINT se utiliza para eliminar una restricción.

Ejercicio 3.4: Adición de una columna a una tabla

- Para ilustrar el uso de la instrucción ALTER TABLE, en la base de datos **QhatuPERU** vamos a crear una tabla denominada **TablaPrueba**, a la que Luego modificaremos su definición.

```
USE QhatuPERU
GO
```

```
CREATE TABLE TablaPrueba (
campo1 INT NOT NULL ,
campo2 CHAR(10) NOT NULL ,
campo3 DATETIME NULL )
GO
```

- A **TablaPrueba** vamos a añadirle una columna de tipo MONEY con la propiedad NULL.

```
ALTER TABLE TablaPrueba
ADD campo4 MONEY NULL
GO
```

- Utilizando el procedimiento **sp_help** revisamos la definición de la tabla **TablaPrueba**.

```
sp_help TablaPrueba
GO
```

Ejercicio 3.5: Adición de una columna NOT NULL a una tabla

- Ahora, añadiremos a la tabla **TablaPrueba** una columna de tipo VARCHAR con la propiedad NOT NULL.

```
ALTER TABLE TablaPrueba
ADD campo5 VARCHAR(20) NOT NULL
GO
```

- La columna se añade sin problemas. A continuación ingresaremos 2 filas de datos a **TablaPrueba**. Para ello ejecute las siguientes instrucciones (más adelante veremos de forma más detallada la instrucción INSERT):

```
INSERT INTO TablaPrueba
VALUES(1, 'RMM0123456', '19/03/2022',
      1375.50, 'Rosa')
INSERT INTO TablaPrueba
VALUES(2, 'JCS9876543', '21/09/2022',
      1250.50, 'Juvenal')
GO
```

3. Las 2 filas se insertan sin problemas en la tabla **TablaPrueba**. Para ver el contenido de la tabla ejecute la instrucción:

```
SELECT * FROM TablaPrueba  
GO
```

4. En el panel de **Resultados** se muestra el contenido de la tabla. Ahora, trataremos de añadir otra columna NOT NULL a la tabla; esta nueva columna será de tipo DECIMAL.

```
ALTER TABLE TablaPrueba  
ADD campo6 DECIMAL(12,3) NOT NULL  
GO
```

5. Obtenemos el siguiente mensaje de error:

Mens. 4901, Nivel 16, Estado 1, Línea 1
ALTER TABLE sólo permite agregar columnas que contengan valores NULL, que tengan la definición DEFAULT, que la columna que se agrega sea una columna de identidad o de marca de tiempo, o si ninguna de las condiciones anteriores se cumplen, la tabla debe estar vacía para que se pueda agregar esta columna. La columna 'campo6' no se puede agregar a la tabla 'TablaPrueba' no vacía porque no cumple estas condiciones.

El mensaje nos dice que ALTER TABLE solo permite añadir columnas que pueden contener valores nulos (columnas NULL) o que tengan especificado un valor predeterminado, salvo que la tabla esté vacía.

Nota: ¿Qué pasaría en una tabla que ya tiene algunas filas registradas si tratamos de añadirle una nueva columna que no permita valores nulos? ¿Qué valor debería colocar SQL Server en la nueva columna para las filas que ya están registradas?

Es evidente que si la nueva columna no permite valores nulos, SQL Server se vería obligado a ingresar un valor para la nueva columna de las filas ya existentes. El problema es que SQL Server no conoce cuál es ese valor, por lo que se produciría un conflicto entre la definición de la columna y el estado actual de los datos.

Para añadir una columna NOT NULL a una tabla que ya tiene datos, primero debemos añadir la columna con la propiedad NULL; a continuación, debemos

registrar los valores de esta nueva columna para las filas ya existentes; finalmente cambiamos la propiedad de la columna a NOT NULL.

6. Añadimos la nueva columna como NULL:

```
ALTER TABLE TablaPrueba
ADD campo6 DECIMAL(12,3) NULL
GO
```

7. Establecemos los valores en la columna **campo6** para las 2 filas existentes (más adelante veremos con detalle el uso de la instrucción UPDATE):

```
UPDATE TablaPrueba SET campo6 = 123.456
WHERE campo1 = 1
UPDATE TablaPrueba SET campo6 = 678.123
WHERE campo1 = 2
GO
```

8. Redefinimos la columna **campo6** como NOT NULL:

```
ALTER TABLE TablaPrueba
ALTER COLUMN campo6 DECIMAL(12,3) NOT NULL
GO
```

5. INTEGRIDAD DE DATOS

Se conoce como integridad de datos al mecanismo utilizado en la base de datos para garantizar la consistencia y exactitud de los datos. La base de datos permite definir un conjunto de herramientas para manejar la integridad de los datos.

5.1. Los niveles de la integridad de datos

La integridad de los datos se define en distintos niveles dependiendo de la parte de la base de datos a la que afecta:

Integridad de entidad

Una tabla almacena los datos de cada una de las ocurrencias o instancias de una entidad. Por ejemplo, si la entidad es ARTICULO, cada uno de los artículos representa una ocurrencia o instancia de la entidad.

La entidad (o tabla) requiere que todas sus filas sean únicas. Esto se garantiza definiendo para cada fila de la entidad un identificador único (llave primaria). Por ejemplo, en la tabla ARTICULO, cada fila de la tabla debe representar a solo un artículo; un artículo no puede aparecer registrado en dos filas de la tabla.

Integridad de dominio

Al conjunto de valores aceptables de un atributo (o columna) se le conoce como el dominio del atributo. La integridad de dominio determina las condiciones que deben cumplir los valores a almacenar en una columna.

La integridad de dominio se define mediante reglas de validación, valores predeterminados, conjunto de valores permitidos en la columna (llave foránea), tipo y formato de los datos. Por ejemplo, en la tabla ARTICULO se puede requerir que el valor de la columna **PrecioProveedor** no puede ser negativo; cualquier valor menor a 0 debe ser rechazado por la base de datos. También podemos establecer que los valores en la columna **DescripcionArticulo** deben ser valores únicos (no duplicados), es decir que no puede haber más de un artículo con la misma descripción.

Integridad referencial

La integridad referencial garantiza que la relación entre la llave primaria (en la tabla referenciada) y la llave foránea (en la tabla de referencia) siempre se mantiene. En otras palabras, que cada una de las filas de la tabla de referencia o tabla secundaria está relacionada con una fila de la tabla referenciada o tabla primaria.

Una fila en una tabla referenciada no puede anularse, ni cambiar su valor de la llave primaria, si una llave foránea se refiere a la fila. Por ejemplo, si tenemos las tablas LINEA y ARTICULO, cada artículo debe pertenecer a solo una línea; es decir, cada fila de la tabla ARTICULO debe estar relacionada con una fila de la tabla LINEA, pero una fila de la tabla LINEA puede estar relacionada con muchas filas de la tabla ARTICULO ya que una línea puede tener muchos artículos.

6. LAS RESTRICCIONES (CONSTRAINTS)

El lenguaje SQL proporciona un método declarativo para definir la integridad de los datos. Este método consiste en la creación de restricciones de datos al momento de crear la

tabla con la instrucción CREATE TABLE, o al momento de modificar su definición con la sentencia ALTER TABLE.

Las restricciones o constraints forman parte de la definición de las tablas, y son el método preferido para forzar la integridad de los datos.

6.1. Tipos de restricciones

Las restricciones o constraints son un método estándar ANSI para forzar la integridad de los datos. Garantizan que los datos ingresados en las columnas de las tablas son valores válidos y que se mantengan las relaciones entre las tablas.

La siguiente es una relación de los diferentes tipos de restricciones que podemos definir y sus descripciones.

PRIMARY KEY (clave primaria)

Garantiza que cada fila o registro en una tabla es único(a). Se aplica sobre una columna (clave primaria simple) o combinación de columnas (clave primaria compuesta). La columna o combinación de columnas definida como clave primaria no permite valores duplicados.

UNIQUE (valor no duplicado)

Garantiza que cada valor en una columna es único. Se utiliza para asegurar que una columna que no es la clave primaria contenga valores únicos. Permite valores nulos.

FOREIGN KEY (clave foránea)

Define la columna o combinación de columnas de una tabla secundaria cuyos valores dependen de la clave primaria de una tabla primaria. Establece la relación entre la tabla primaria y su tabla secundaria.

DEFAULT (valor predeterminado)

Establece el valor predeterminado para una columna cuando al insertar una fila no se especifica el valor para dicha columna. Es útil cuando la columna no permite valores nulos y no se conoce el valor que debe tener dicha columna para la fila que se está insertando.

CHECK (regla de validación)

Establece la regla que debe cumplir un valor para que sea un valor aceptable en una columna. Es útil para implementar reglas de neGOcios sencillas en la base de datos.

Ejercicio 3.6: Creación de una tabla especificando su clave primaria – Creación de la tabla TRANSPORTISTA

Según el modelo de la base de datos **QhatuPERU**, la tabla TRANSPORTISTA tiene la siguiente definición:

- CodTransportista, identificador del transportista; dato obligatorio de tipo numérico entero.
 - NomTransportista, nombre del transportista; dato obligatorio de tipo cadena de longitud variable de hasta 30 caracteres.
 - Direccion, dirección del transportista; cadena de longitud variable de hasta 60 caracteres.
 - Telefono, número telefónico del transportista; cadena de longitud variable de hasta 15 caracteres.
1. En su ventana Query ejecute las siguientes instrucciones:

```
USE QhatuPERU
GO
```

```
CREATE TABLE TRANSPORTISTA (
    CodTransportista      int NOT NULL PRIMARY KEY,
    NomTransportista      varchar(30) NOT NULL,
    Direccion              varchar(60) NULL,
    Telefono               varchar(15) NULL )
GO
```

2. Se crea en la base de datos **QhatuPERU** la tabla TRANSPORTISTA en la que la columna **CodTransportista** es la clave primaria o identificador de cada transportista. Para probar el efecto de la clave primaria ejecutaremos las siguientes sentencias INSERT:

```
INSERT INTO TRANSPORTISTA
VALUES(1, 'VELASQUEZ ORTIZ, FRANCISCO',
      'Av. Los Alamos 1234 San Luis',
      '999-123-456')
INSERT INTO TRANSPORTISTA
VALUES(2, 'ALIAGA VIDAL, JEREMIAS',
      'Jr. Tarma 456 Callao',
      '991-234-456')
INSERT INTO TRANSPORTISTA
```

```
VALUES (2, 'CASTRO SOLIS, JUAN JOSE',
       'Jr. Las Novicias 123 Chorrillos',
       '993-000-111')
```

GO

- La dos primeras instrucciones INSERT se ejecutan sin problemas. La tercera genera el mensaje de error 2627 debido a que se viola la clave primaria de la tabla ya que el valor de la columna **CodTransportista** se está duplicando (valor 2).

(1 filas afectadas)

(1 filas afectadas)

Mens. 2627, Nivel 14, Estado 1, Línea 5
 Infracción de la restricción PRIMARY KEY
 'PK_TRANSPOR_E42881D5DD5000DD'. No se puede insertar
 una clave duplicada en el objeto 'dbo.TRANSPORTISTA'.
 El valor de la clave duplicada es (2).
 Se terminó la instrucción.

- Observe que SQL Server ha definido la cadena **PK_TRANSPOR_E42881D5DD5000DD** como nombre de la restricción clave primaria. El nombre está formado por el prefijo PK_ seguido de los 8 primeros caracteres del nombre de la tabla a la que pertenece, y Luego de una cadena hexadecimal. Para revisar la tabla y verificar la data ingresada ejecute:

```
SELECT * FROM TRANSPORTISTA
GO
```

	CodTransportista	Nom Transportista	Direccion	Telefono
1	1	VELASQUEZ ORTIZ, FRANCISCO	Av. Los Alamos 1234 San Luis	999-123-456
2	2	ALIAGA VIDAL, JEREMIAS	Jr. Tama 456 Callao	991-234-456

6.2. Creación de una clave primaria en una tabla existente

Si la tabla ya ha sido creada y deseamos añadirle su clave primaria, podemos usar la instrucción ALTER TABLE.

Sintaxis

```
ALTER TABLE nombre_tabla
ADD CONSTRAINT PK_nombre_tabla
PRIMARY KEY( columnaX, columnaP, ... )
```

- **PK_nombre_tabla** es el nombre de la restricción clave primaria. Se recomienda definir como nombre de la clave primaria, el nombre de la tabla con el prefijo PK_. Si no se especifica el nombre de la restricción, SQL Server le asigna un nombre.



- **columnaX, columnaP**, es la columna o combinación de columnas que se define como clave primaria. Las columnas involucradas no deben permitir valores nulos, y además, no deben tener valores duplicados. En el caso de una combinación de columnas, la combinación vista como una unidad no debe tener valores duplicados.

Ejercicio 3.7: Creación de una clave primaria en una tabla existente – Creación de la clave primaria de la tabla TIENDA

Para la tabla TIENDA creada en el ejercicio 3.2 defina la columna **CodTienda** como su clave primaria.

1. Ejecute las siguientes instrucciones:

```
USE QhatuPERU
GO
```

```
ALTER TABLE TIENDA
ADD CONSTRAINT PK_TIENDA
PRIMARY KEY( CodTienda )
GO
```

2. Ejecute el procedimiento **sp_help** para verificar la definición de la tabla.

```
sp_help TIENDA
GO
```

Resultados									
	Name	Owner	Type	Created_datetime					
1	TIENDA	dbo	user table	2023-01-29 07:14:27.503					
1	CodTienda	int	no	4	10	0	no	(n/a)	(n/a)
2	Direccion	var...	no	60			yes	no	yes
3	Distrito	var...	no	20			yes	no	yes
4	Telefono	var...	no	15			yes	no	yes
5	Fax	var...	no	15			yes	no	yes
1	Identity	Seed	Increment	Not For Replication					
1	No identity column defined.	NULL	NULL	NULL					
1	RowGuidCol								
1	No rowguidcol column defined.								
1	Data_located_on_filegroup								
1	PRIMARY								
1	index_name	index_description		index_keys					
1	PK_TIENDA_62AA333138E277A4	clustered, unique, primary key located on PRIMARY		CodTienda					
1	constraint_type	constraint_name		delete_action	update_action	status_enabled	status_for_replication	constraint_keys	
1	PRIMARY KEY (clustered)	PK_TIENDA_62AA333138E277A4		(n/a)	(n/a)	(n/a)	(n/a)	CodTienda	
1	Table is referenced by foreign key								
1	QhatuPERU.dbo.GUIA_ENVIO: FK_GUIA_ENVI_CodTi_4CA06362								

6.3. Creación de una clave primaria numérica con valores secuenciales autogenerados – Uso de la propiedad IDENTITY

La propiedad IDENTITY permite definir una columna numérica entera en la que el valor de la columna es autogenerado a medida que se va insertando filas. Es útil cuando se desea emplear claves primarias numéricas.

Ejercicio 3.8: Uso de la propiedad IDENTITY – Creación de la tabla LINEA

Según el modelo de datos de la base de datos **QhatuPERU**, la tabla LINEA tiene la siguiente definición:

- **CodLinea**, identificador de la línea de artículos; dato obligatorio de tipo numérico entero autogenerado.
- **NomLinea**, nombre de la línea de artículos; dato obligatorio de tipo cadena de longitud variable de hasta 20 caracteres.
- **Descripcion**, descripción de la línea de artículos; cadena de longitud variable de hasta 40 caracteres.

En la tabla LINEA, la columna **CodLinea** no solo es su clave primaria sino que además su contenido es autogenerado en forma de secuencia a medida que se va insertando las filas.

1. Para crear la tabla LINEA ejecute las siguientes instrucciones:

```
USE QhatuPERU
GO
CREATE TABLE LINEA (
    CodLinea      int IDENTITY(1,1) PRIMARY KEY,
    NomLinea     varchar(20) NOT NULL,
    Descripcion  varchar(40) NULL )
GO
```

2. Ejecute las siguientes instrucciones para insertar filas en la tabla y vea el efecto de la propiedad IDENTITY:

```
INSERT LINEA
    VALUES ('GOLOSINAS',
            'GALLETAS, CHOCOLATES, CARAMELOS, TOFFES')
INSERT LINEA
    VALUES ('EMBUTIDOS',
            'JAMONADAS, JAMONES, SALCHICHAS, CHORIZOS')
INSERT LINEA
    VALUES ('HIGIENE PERSONAL',
            'JABONES, P.DENTALES, SHAMPOOS, P.H.')
GO
```

3. Para ver el contenido de la tabla ejecute:

```
SELECT * FROM LINEA
GO
```

En IDENTITY(1,1), el primer argumento conocido como "semilla" establece el primer valor de identidad a generar, y el segundo argumento es el "incremento" a utilizar para generar los siguientes valores.

Ejercicio 3.9: Creación de una clave primaria compuesta – Creación de la tabla ORDEN_DETALLE y de su clave primaria

Según el modelo de la base de datos, la tabla ORDEN_DETALLE tiene la siguiente definición:

- NumOrden, número de la orden de compra a la que pertenece el artículo ingresado al almacén; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente orden de compra.
- CodArticulo, identificador del artículo recibido con la orden de compra especificada en NumOrden; dato obligatorio de tipo numérico entero que relaciona al ítem en la orden de compra con los datos complementarios del artículo.
- PrecioCompra, precio de compra del artículo; dato obligatorio de tipo dinero (money).
- CantidadSolicitada, cantidad que se le solicitó al proveedor en la correspondiente orden de compra; dato obligatorio de tipo numérico entero pequeño.
- CantidadRecibida, cantidad que recibió del proveedor y se ingresó al almacén; dato numérico entero pequeño.
- Estado, estado del artículo en la orden de compra; cadena de longitud variable de hasta 10 caracteres.

La clave primaria de la tabla ORDEN_DETALLE está formada por las columnas **NumOrden** y **CodArticulo**, ya que para identificar el ingreso de un artículo al Almacén se necesita conocer el artículo ingresado y con cuál orden de compra se ingresó.

1. Para crear la tabla ORDEN_DETALLE ejecute las siguientes instrucciones:

```
USE QhatuPERU
GO
CREATE TABLE ORDEN_DETALLE (
    NumOrden      int NOT NULL,
    CodArticulo   int NOT NULL,
    PrecioCompra  money NOT NULL,
    CantidadSolicitada smallint NOT NULL,
    CantidadRecibida smallint NULL,
    Estado        varchar(10) NULL )
```



GO

2. Para añadir la clave primaria a la tabla ejecute:

```
ALTER TABLE ORDEN_DETALLE
ADD CONSTRAINT PK_ORDEN_DETALLE
PRIMARY KEY( NumOrden, CodArticulo )
GO
```

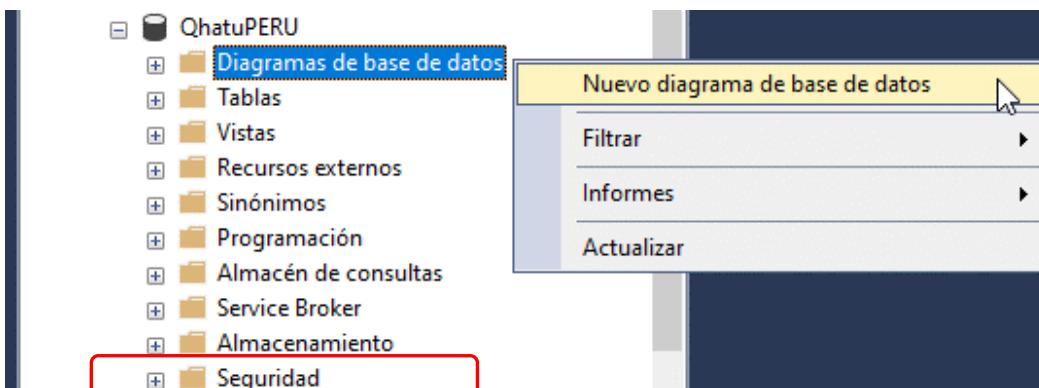
6.4. Creación de un diagrama de base de datos

SQL Server Management Studio posee una herramienta que permite ver gráficamente su modelo de datos, ya sea en una vista parcial o en una vista completa.

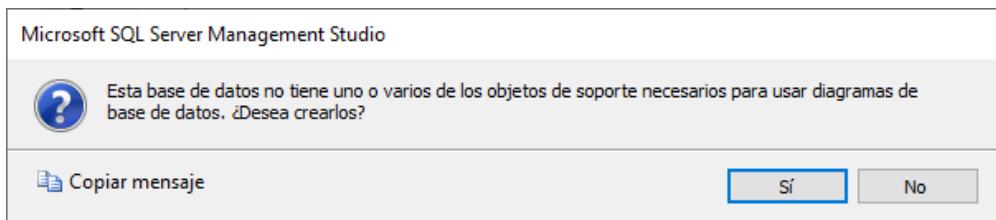
Ejercicio 3.10: Creación de un diagrama de la base de datos

Si deseamos ver gráficamente qué tablas forman parte de nuestra base de datos y cómo se relacionan entre sí, podemos crear un diagrama de la base de datos. Para ello:

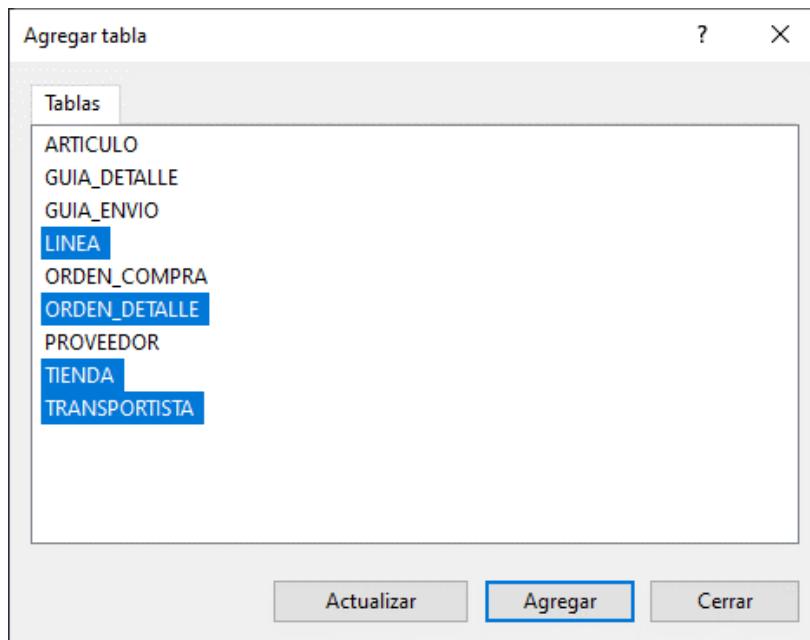
1. En el **Explorador de objetos** de SQL Server Management Studio, expanda el nodo **Bases de datos** de su servidor, Luego expanda el nodo de la base de datos **QhatuPERU**, y haga un clic secundario en **Diagramas de base de datos**.



2. En el menú contextual haga clic en **Nuevo diagrama de base de datos**. Si se muestra la ventana de mensajes que se muestra a continuación, haga clic en **Si**.



3. Se muestra la ventana **Agregar tabla** con la lista de tablas presentes en la base de datos.



4. Seleccione las tablas LINEA, ORDEN_DETALLE, TIENDA y TRANSPORTISTA, y Luego haga clic en **Agregar**. Luego clic en **Cerrar**. Se crea un diagrama que muestra las tablas añadidas a él. Por el momento, no hay definidas relaciones entre las tablas.



5. Guarde el diagrama bajo el nombre **Diagrama01**.

Ejercicio 3.11: Creación de la tabla ARTICULO y de su clave primaria

El modelo de datos de la base de datos QhatuPERU establece que la tabla ARTICULO tiene la siguiente definición:

- CodArticulo, identificador del artículo; dato obligatorio de tipo numérico entero autogenerado.
- CodLinea, identificador de la línea a la que pertenece el artículo; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente línea.
- CodProveedor, identificador del proveedor del artículo; dato obligatorio de tipo numérico entero que relaciona al artículo con su proveedor.
- DescripcionArticulo, nombre del artículo; dato obligatorio de tipo cadena de longitud variable de hasta 40 caracteres.
- Presentacion, forma en la que se presenta el artículo; cadena de longitud variable de hasta 30 caracteres.
- PrecioProveedor, precio al que se le compra el artículo al proveedor; dato numérico de tipo dinero (money).
- StockActual, cantidad del artículo existente en el almacén; dato numérico entero pequeño.
- StockMinimo, cantidad mínima que debe existir en el almacén para atender los pedidos de las tiendas; dato numérico entero pequeño.
- Descontinuado, dato de tipo bit que indica si el artículo ha dejado de ser comercializado por la empresa.

1. Para crear la tabla ARTICULO ejecute lo siguiente:

```
USE QhatuPERU
GO

CREATE TABLE ARTICULO (
    CodArticulo           int IDENTITY
                           PRIMARY KEY,
    CodLinea              int NOT NULL,
    CodProveedor          int NOT NULL,
    DescripcionArticulo  varchar(40)
                           NOT NULL,
    Presentacion          varchar(30) NULL,
    PrecioProveedor       money NULL,
    StockActual           smallint NULL,
    StockMinimo           smallint NULL,
    Descontinuado         bit )
GO
```

2. Para ver qué tablas tiene la base de datos ejecute la siguiente consulta:

```
SELECT name FROM sys.tables
GO
```

6.5. Creación de una clave foránea

Sintaxis

```
ALTER TABLE nombre_tabla
ADD CONSTRAINT FK_nombre_tabla_tabla_referenciada
FOREIGN KEY( columnaX, columnaP, ... )
REFERENCES tabla_referenciada
```

- **FK_nombre_tabla_tabla_referenciada** es el nombre de la restricción clave foránea. Se recomienda definir como nombre de la clave foránea, el nombre de la tabla seguido del nombre de la tabla referenciada, todo con el prefijo **FK_**. Si no se especifica el nombre de la restricción, SQL Server le asigna un nombre.
- **columnaX, columnaP**, es la columna o combinación de columnas que se define como clave foránea.
- **tabla_referenciada** es el nombre de la tabla primaria con la que se relaciona la tabla secundaria que tiene la clave foránea. De modo predeterminado, la clave foránea hace referencia a la clave primaria de la tabla primaria.

Ejercicio 3.12: Creación de la clave foránea en la tabla ARTICULO que referencia a la tabla LINEA

1. En su ventana Query ejecute las siguientes instrucciones para crear la clave foránea de la tabla ARTICULO que referencia a la tabla LINEA:

```
USE QhatuPERU
GO
```

```
ALTER TABLE ARTICULO
ADD CONSTRAINT fk_Articulo_Linea
FOREIGN KEY( CodLinea )
REFERENCES LINEA
GO
```

La columna **CodLinea** de la tabla ARTICULO establece la relación de esta tabla con la tabla LINEA. La clave foránea en **CodLinea** de ARTICULO apunta a la clave primaria en **CodLinea** de la tabla LINEA.



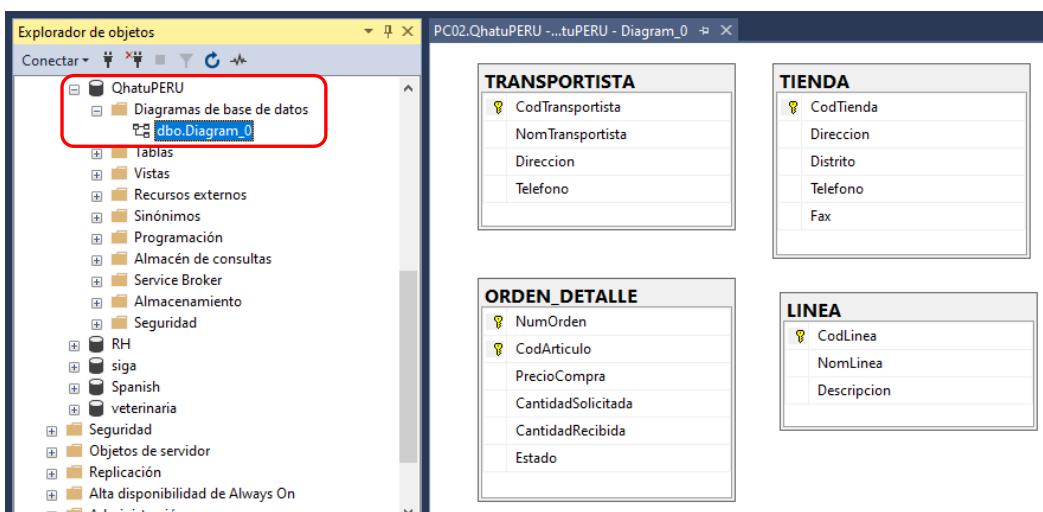
2. Para probar la integridad referencial (la restricción clave foránea) ejecute las siguientes instrucciones:

```
INSERT INTO ARTICULO
VALUES(1,14,
      'CARAMELOS BASTON VIENA ARCOR',
      'PAQUETE 454 GR',1.50,200,50,0)
INSERT INTO ARTICULO
VALUES(3,5,
      'ACEITE BABY JOHNSONS C/ALOE Y VIT. E',
      'FRASCO 100 ML',3.90,175,100,0)
INSERT INTO ARTICULO
VALUES(2,1,
      'JAMONADA LAIVE',
      'KILOGRAMO',12.50,80,75,0)
INSERT INTO ARTICULO
VALUES(4,1,
      'CREMA DE LECHE LAIVE',
      'ENVASE 160 GR',2.00,250,250,0)
GO
```

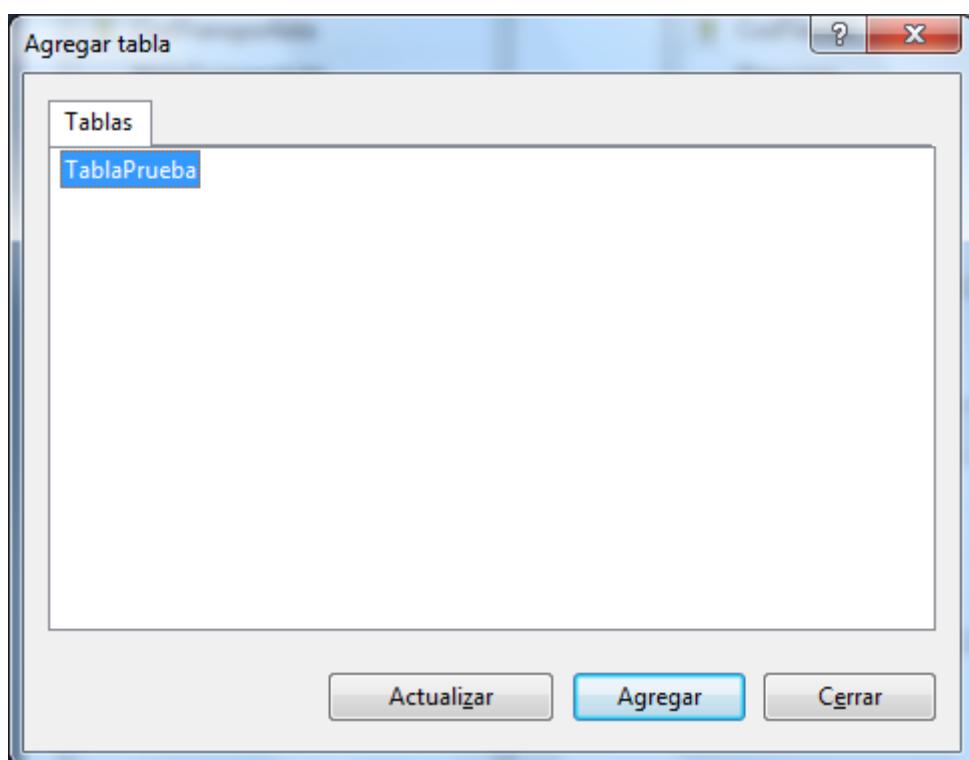
3. Las 3 primeras instrucciones INSERT se ejecutan sin problemas. La última genera un error debido a que la línea cuyo **CodLinea** es 4 aún no se ha registrado.

Mens. 547, Nivel 16, Estado 0, Línea 7
Instrucción INSERT en conflicto con la restricción
FOREIGN KEY "fk_Articulo_Linea". El conflicto ha
aparecido en la base de datos "QhatUPERU", tabla
"dbo.LINEA", column 'CodLinea'.
Se terminó la instrucción.

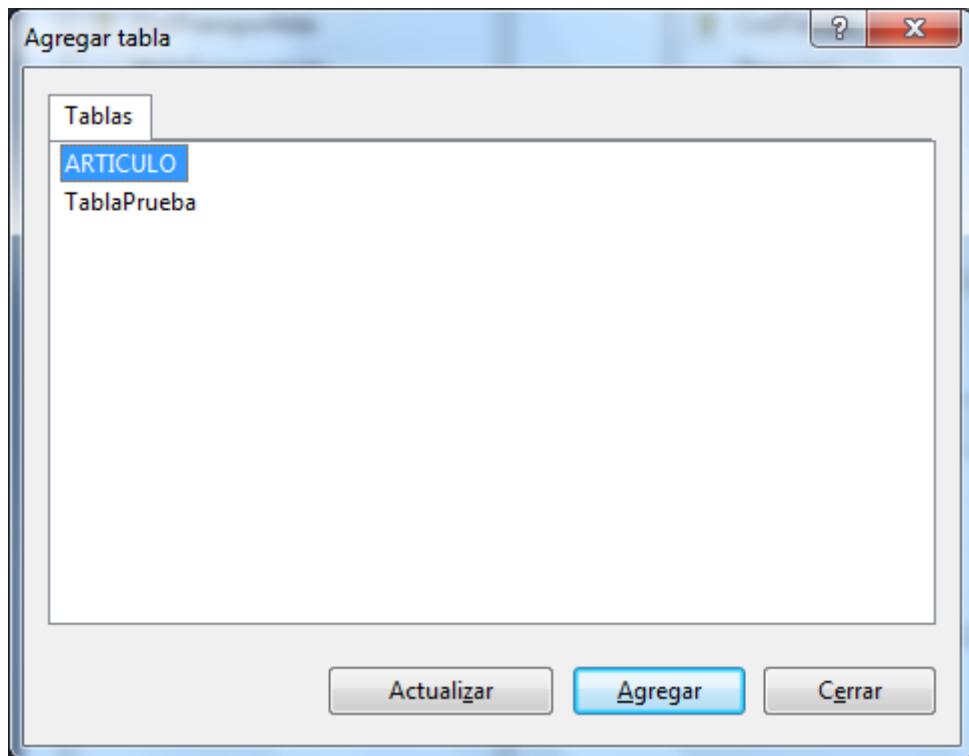
4. Abra el diagrama **Diagrama01** (si no lo tiene abierto) que creó en uno de los ejercicios anteriores. Para ello, en el nodo de la base de datos **QhatUPERU**, expanda **Diagramas de base de datos**, y haga doble clic en **Diagrama01**.



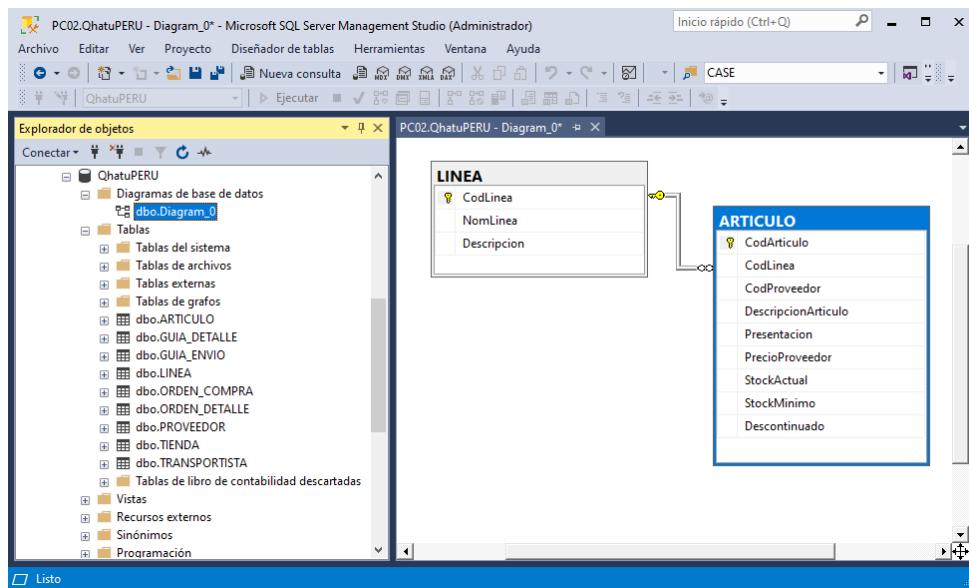
5. Haga clic secundario sobre un área libre del diagrama, y en el menú contextual seleccione **Agregar tabla**. Se abre la ventana **Agregar tabla**.



6. Haga clic en el botón **Actualizar** para que la ventana muestre las tablas creadas después de la creación del diagrama.



7. Seleccione la tabla ARTICULO y haga clic en **Agregar**, y Luego en **Cerrar**. La tabla ARTICULO se añade al diagrama mostrando su relación con la tabla LINEA como consecuencia de la creación de su llave foránea.



8. Guarde el diagrama.

6.6. Creación de una restricción UNIQUE (valor no duplicado)

Sintaxis

```
ALTER TABLE nombre_tabla
    ADD CONSTRAINT U_nombre_tabla_nombre_columna
        UNIQUE( columnaX, columnaP, ... )
```

- **U_nombre_tabla_nombre_columna** es el nombre de la restricción valor no duplicado o UNIQUE. Se recomienda definir como nombre de la restricción, el nombre de la tabla seguido del nombre de la columna afectada, todo con el prefijo **U_**. Si no se especifica el nombre de la restricción, SQL Server le asigna un nombre.
- **columnaX, columnaP**, es la columna o combinación de columnas a la que se aplica la restricción.

Ejercicio 3.13: Creación de restricción UNIQUE para la columna NomLinea en la tabla LINEA

Se desea que el nombre de una línea no se duplique. Para garantizar el cumplimiento de esta regla creamos una restricción UNIQUE en la columna **NomLinea** de la tabla LINEA.

1. En su ventana Query ejecute las siguientes instrucciones:

```
USE QhatuPERU
GO
```

```
ALTER TABLE LINEA
    ADD CONSTRAINT U_Linea_NomLinea
        UNIQUE( NomLinea )
GO
```

2. Pruebe la restricción ejecutando la siguiente instrucción:

```
INSERT INTO LINEA
VALUES('HIGIENE PERSONAL', 'Por definir')
GO
```

3. Se recibe el siguiente mensaje de error debido a que la línea de nombre HIGIENE PERSONAL ya está registrada.

Mens. 2627, Nivel 14, Estado 1, Línea 1
Infracción de la restricción UNIQUE KEY
'U_Linea_NomLinea'. No se puede insertar una clave
duplicada en el objeto 'dbo.LINEA'. El valor de la clave
duplicada es (HIGIENE PERSONAL).
Se terminó la instrucción.

6.7. Creación de una restricción DEFAULT (valor predeterminado)

Sintaxis

```
ALTER TABLE nombre_tabla
    ADD CONSTRAINT DF_nombre_tabla_nombre_columna
        DEFAULT valor_predeterminado FOR columnaX
```

- **DF_nombre_tabla_nombre_columna** es el nombre de la restricción valor predeterminado o DEFAULT. Se recomienda definir como nombre de la restricción, el nombre de la tabla seguido del nombre de la columna afectada, todo con el prefijo **DF_**. Si no se especifica el nombre de la restricción, SQL Server le asigna un nombre.
- **valor_predeterminado** es el valor que se almacena en *columnaX* cuando al insertar una fila no se especifica el valor para esa columna.
- **columnaX** es la columna a la que se aplica la restricción.

Ejercicio 3.14: Creación de restricción DEFAULT para la columna Descontinuado en la tabla ARTICULO

Para la columna **Descontinuado** el valor por defecto es **0** (cero).

1. Ejecute en su ventana Query las siguientes instrucciones:

```
USE QhatuPERU
GO

ALTER TABLE ARTICULO
    ADD CONSTRAINT DF_Articulo_Descontinuado
        DEFAULT 0 FOR Descontinuado
GO
```

2. Para probar la restricción ejecute la siguiente instrucción:

```
INSERT INTO ARTICULO(CodLinea,
                      CodProveedor, DescripcionArticulo)
VALUES(1,15,'CARAMELOS SURTIDO DE FRUTAS')
GO
```

3. Consulte la tabla para verificar el valor en la columna **Descontinuado** para el artículo CARAMELOS SURTIDO DE FRUTAS.

```
SELECT DescripcionArticulo, Descontinuado
FROM ARTICULO
GO
```



	DescripcionArticulo	Descontinuado
1	CARAMELOS BASTON VIENA ARCOR	0
2	CARAMELOS SURTIDO DE FRUTAS	0
3	CARAMELOS FRUTAS SURTIDA ARCOR	0
4	CARAMELOS FRUTAS MASTICABLES	0

6.8. Creación de una restricción CHECK (regla de validación)

Sintaxis

```
ALTER TABLE nombre_tabla
    ADD CONSTRAINT CK_nombre_tabla_nombre_columna
    CHECK( condición )
```

- **CK_nombre_tabla_nombre_columna** es el nombre de la restricción regla de validación o CHECK. Se recomienda definir como nombre de la restricción, el nombre de la tabla seguido del nombre de la columna afectada, todo con el prefijo **CK_**. Si no se especifica el nombre de la restricción, SQL Server le asigna un nombre.
- **condición** es la expresión que determina cómo debe ser el valor a ingresar en la columna afectada por la restricción.

Ejercicio 3.15: Creación de restricción CHECK para la columna PrecioProveedor de la tabla ARTICULO

El precio de un artículo no puede ser menor que 0 (cero)

1. En su ventana Query escriba la siguiente instrucción:

```
USE QhatuPERU
GO
```

```
ALTER TABLE ARTICULO
    ADD CONSTRAINT CK_Articulo_PrecioProveedor
    CHECK( PrecioProveedor >= 0 )
GO
```

2. Pruebe la regla ejecutando la siguiente instrucción:

```
INSERT INTO ARTICULO(CodLinea, CodProveedor,
DescripciónArtículo, Presentación,
PrecioProveedor)
VALUES(2,12,
'JAMONADA ESPECIAL LA SEGOVIANA',
'KILOGRAMO', -15)
GO
```



3. La instrucción genera error debido a que el precio no puede ser negativo. Se recibe el siguiente mensaje:

Mens. 547, Nivel 16, Estado 0, Línea 1
Instrucción INSERT en conflicto con la restricción
CHECK "CK_Articulo_PrecioProveedor". El conflicto ha
aparecido en la base de datos "QhatUPERU", tabla
"dbo.ARTICULO", column 'PrecioProveedor'.
Se terminó la instrucción.

7. INTEGRIDAD REFERENCIAL EN CASCADA

Cuando se intenta eliminar una fila de una tabla a la que apuntan claves foráneas, la eliminación falla debido a que las filas que contienen las claves foráneas no pueden quedar "huérfanas". Por ejemplo, cuando intentamos eliminar a una línea que tiene artículos registrados.

La integridad referencial en cascada permite controlar las acciones que lleva a cabo SQL Server cuando se intenta actualizar o eliminar una clave primaria a la que apuntan claves foráneas existentes. Esto se controla mediante las cláusulas ON DELETE y ON UPDATE en la cláusula REFERENCES de las instrucciones CREATE TABLE y ALTER TABLE.

La cláusula ON DELETE controla las acciones que se llevarán a cabo si intenta eliminar una fila a la que apuntan las claves foráneas existentes. A partir de SQL Server 2005 la cláusula ON DELETE tiene cuatro opciones:

- NO ACTION especifica que la eliminación produce un error.
- CASCADE especifica que también se eliminan todas las filas con claves foráneas que apuntan a la fila eliminada.
- SET NULL especifica que todas las filas con claves foráneas que apuntan a la fila eliminada tendrán el valor NULL en la clave foránea.
- SET DEFAULT especifica que todas las filas con claves foráneas que apuntan a la fila eliminada se configurarán al valor predeterminado en la clave foránea.

La cláusula ON UPDATE define las acciones que se llevarán a cabo si intenta actualizar un valor de clave candidata a la que apuntan las claves foráneas existentes. También acepta las opciones NO ACTION, CASCADE, SET NULL y SET DEFAULT.



Ejercicio 3.16: Definición de integridad referencial en cascada para las tablas LINEA y ARTICULO

Para la base de datos **QhatuPERU** defina que al eliminar una línea en la tabla LINEA se eliminen también todos los artículos registrados en la tabla ARTICULO asociados a dicha línea.

1. Ejecute las siguientes instrucciones:

```
USE QhatuPERU
```

```
GO
```

```
SELECT * FROM LINEA
```

```
SELECT * FROM ARTICULO
```

```
GO
```

2. Observe que la línea 1 tiene registrados 2 artículos.

	CodLinea	NomLinea	Descripcion
1	1	GOLOSINAS	GALLETAS,CHOCOLATES,CARAMELOS,TOFFES
2	2	EMBUTIDOS	JAMONADAS,JAMONES,SALCHICHAS,CHORIZOS
3	3	HIGIENE PERSONAL	JABONES,P.DENTALES,SHAMPOOS,P.H.

	CodArticulo	CodLinea	CodProveedor	DescripcionArticulo
1	1	1	14	CARAMELOS BASTON VIENA ARCOR
2	2	3	5	ACEITE BABY JOHNSONS C/ALOE Y VIT. E
3	3	2	1	JAMONADA LAIVE
4	5	1	15	CARAMELOS SURTIDO DE FRUTAS

3. Procedemos a definir la integridad referencial en cascada. Para ello vamos a eliminar la llave foránea de la tabla ARTICULO que apunta a la tabla LINEA para volverla a definir.

```
-- Eliminando la clave foránea de la
-- tabla ARTICULO para volverla a crear
-- estableciendo la eliminación en cascada
ALTER TABLE ARTICULO
DROP CONSTRAINT FK_Articulo_Linea
GO
```

4. Ahora creamos la clave foránea definiendo la eliminación en cascada:



```
ALTER TABLE ARTICULO
ADD CONSTRAINT FK_Articulo_Linea
FOREIGN KEY( CodLinea )
REFERENCES LINEA
ON DELETE CASCADE
GO
```

5. Vamos a eliminar la línea 1 para ver el efecto de la integridad referencial en cascada.

```
-- Eliminando la línea 1
DELETE FROM LINEA
WHERE CodLinea = 1
GO
```

6. Comprobamos que se ha eliminado la línea 1 y además los artículos que tenía registrados.

```
SELECT * FROM LINEA
SELECT * FROM ARTICULO
GO
```

	CodLinea	NomLinea	Descripcion
1	2	EMBUTIDOS	JAMONADAS,JAMONES,SALCHICHAS,CHORIZOS
2	3	HIGIENE PERSONAL	JABONES,P.DENTALES,SHAMPOOS,P.H.

	CodArticulo	CodLinea	CodProveedor	DescripcionArticulo
1	2	3	5	ACEITE BABY JOHNSONS C/ALOE Y VIT. E
2	3	2	1	JAMONADA LAIVE



8. ELIMINACIÓN DE TABLAS (LA INSTRUCCIÓN DROP TABLE)

Sintaxis

```
DROP TABLE nombre_tabla
```

Elimina la definición de una tabla, y todos los datos, índices, desencadenantes, restricciones y permisos especificados para dicha tabla.

Si la tabla es referenciada por una clave foránea, no se podrá eliminar. Primero debe eliminarse la tabla que contiene la clave foránea, y luego la tabla referenciada.

Ejercicio 3.17: Eliminación de tablas

1. Ejecute la siguiente instrucción en su ventana Query:

```
DROP TABLE LINEA  
GO
```

2. Se recibe el siguiente mensaje de error:

Mens. 3726, Nivel 16, Estado 1, Línea 1
No se puede quitar el objeto 'LINEA'. Hay una referencia
a él en una restricción FOREIGN KEY.

3. Ahora, ejecute la siguiente instrucción:

```
DROP TABLE ORDEN_DETALLE  
GO
```

La tabla ORDEN_DETALLE se elimina sin problemas.



9. EJERCICIO PROPUESTO

Con las especificaciones presentadas en el punto 3.1 de este capítulo escriba todas las instrucciones necesarias para crear la base de datos QhatuPERU y todas sus tablas y relaciones. Previamente deberá eliminar la base de datos creada en este capítulo. Para ello, en la ventana Query ejecute las siguientes instrucciones:

```
USE master
GO
```

```
DROP DATABASE QhatuPERU
GO
```



Capítulo IV

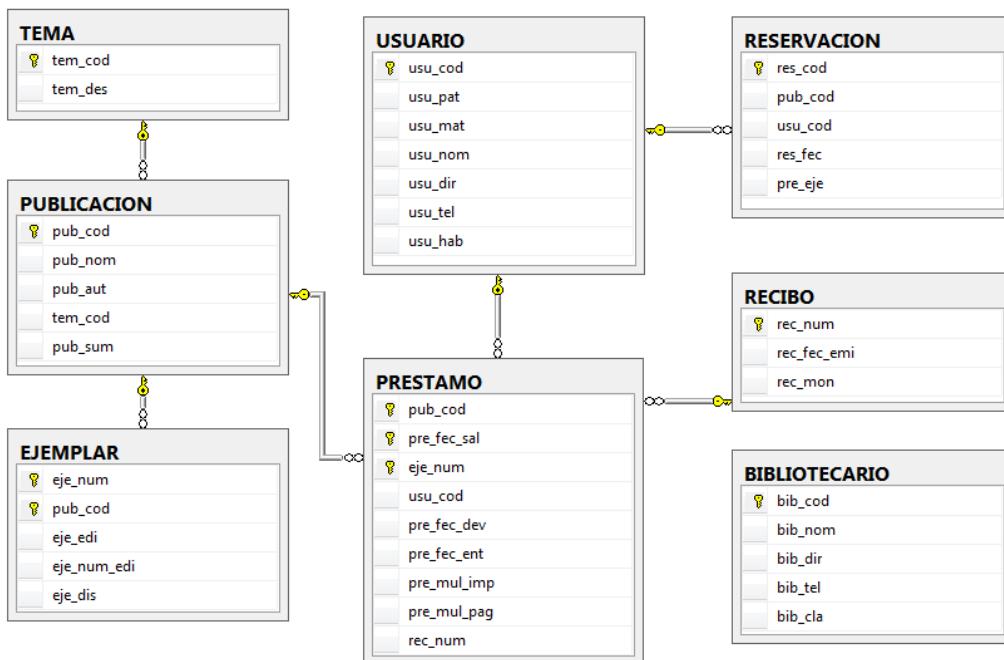
Operaciones de Mantenimiento de Datos

En este capítulo conoceremos de modo más detallado cómo operar con los datos en las tablas SQL Server: ingresar nuevos datos (sentencia INSERT), actualizar los datos existentes (sentencia UPDATE), y eliminar registros (sentencia DELETE).



1. PREPARACIÓN DEL ESCENARIO

Para desarrollar este capítulo crearemos previamente una base de datos **BibliotecaGyS** cuyo modelo se muestra a continuación, y en la que ejecutaremos las operaciones de mantenimiento de datos:



A continuación, una descripción de lo que almacenará cada tabla:

PUBLICACION: relación de todas las publicaciones registradas en la biblioteca. Las publicaciones están clasificadas por temas.

TEMA: relación de todos los temas disponibles en la biblioteca.

EJEMPLAR: para cada publicación puede haber más de un ejemplar de acuerdo con la demanda de la misma.

BIBLIOTECARIO: relación del personal que atiende a los usuarios de la biblioteca.

USUARIO: para hacer uso de los servicios de la biblioteca se requiere estar registrado como usuario.

PRESTAMO: registra los movimientos de las publicaciones.

RESERVACION: si para una publicación no hubiera un ejemplar disponible, el usuario puede hacer la reserva correspondiente.

Ejercicio 4.1: Creación de la base de datos BibliotecaGyS

1. Inicie su sesión en **SQL Server Management Studio**.
2. En el menú **Archivo** ejecute **Abrir, Archivo**.
3. Ubique en su CD del libro la carpeta **ScriptsBD** y abra el archivo **BibliotecaGyS-Tablas.sql**.
4. Haga clic en el botón **Ejecutar** de la barra de herramientas o pulse la tecla [F5].

2. INSERCIÓN DE FILAS – INSTRUCCIÓN INSERT

Sintaxis

```
INSERT [INTO] nombre_tabla[ ( lista_de_columnas ) ]
VALUES( lista_de_valores )
```

- **lista_de_columnas** es la relación de columnas en las que se almacenarán los valores especificados en *lista_de_valores*.
- **lista_de_valores** es la relación de los valores a almacenar en la fila a insertar.

Ejercicio 4.2: Inserción de una fila con lista de valores completa

4. En su ventana de consulta ejecute las siguientes instrucciones:

```
USE BibliotecaGyS
GO

INSERT Usuario(usu_nom, usu_pat, usu_mat,
    usu_cod, usu_dir, usu_tel, usu_hab)
VALUES('ABELARDO', 'BARRIONUEVO',
    'CASTILLO', '961002',
    'LOS PINOS 234 - LOS OLIVOS', NULL, 1)
GO
```

En este caso, cuando se entrega valores para cada una de las columnas de una tabla, el orden de los valores en la lista de valores debe coincidir con el orden de las columnas en la lista de columnas.



5. También se puede insertar una fila con lista de valores completa usando la siguiente Sintaxis:

```
INSERT usuario
VALUES ('961003', 'ANTAYHUA', 'OSORIO',
        'ANA ROSA',
        'AV. ARICA 1312 - BREÑA', '4321234', 1)
GO
```

En este caso, cuando en la instrucción no se establece la lista de columnas, pero se está entregando la lista de valores completa, los valores de la lista se ordenan según el orden natural de las columnas en la tabla.

6. También es posible insertar una fila con lista de valores completa que contengan valores NULL, siempre y cuando la columna correspondiente los permita.

```
INSERT usuario
VALUES( '961004', 'QUISPE', 'CUYUBAMBA',
       'JUAN JOSE',
       'AV. URUGUAY 546 - CERCADO - LIMA', NULL, 1)
GO
```

En este caso, no se conoce el teléfono del usuario, pero como la columna permite nulos registramos NULL como valor de dicha columna.

Ejercicio 4.3: Inserción de una fila con lista de valores incompleta

Cuando la lista de valores está incompleta es obligatorio establecer en qué columna va cada valor de la lista, por lo que la lista de columnas es obligatoria en la instrucción INSERT.

```
INSERT usuario(usu_cod, usu_pat, usu_mat,
               usu_nom, usu_dir, usu_hab)
VALUES('961005', 'ALTAMIRANO', 'CASTRILLON',
      'MARIO', 'COSME BUENO 256 - BELLAVISTA', 1)
GO
```

El orden de las columnas en la lista de columnas, y el orden de los valores en la lista de valores se deben corresponder.

2.1. Uso de DEFAULT VALUES

Si cada una de las columnas de la tabla tiene definido un valor por defecto, o permite valores nulos, o tiene la propiedad IDENTITY, podemos insertar filas sin tener que especificar los valores a insertar tal como se muestra en la Sintaxis siguiente:

```
INSERT [INTO] nombre_tabla DEFAULT VALUES
```

Esta orden inserta una fila de la siguiente manera:

- Si la columna permite nulos, inserta NULL en dicha columna.
- Si la columna tiene definido un valor por defecto, inserta el valor por defecto.
- Si la columna tiene la propiedad IDENTITY, inserta el valor de identidad correspondiente.

2.2. La propiedad IDENTITY

Permite definir una columna numérica entera en la que el valor de la columna es autogenerado a medida que se van insertando filas. Es útil cuando se desea emplear claves primarias numéricas.

Ejercicio 4.4: Uso de la propiedad IDENTITY

1. Cree una tabla que tenga la propiedad IDENTITY.

```
USE BibliotecaGyS
GO

CREATE TABLE pruebaIdentidad(
    codIGO int IDENTITY(1,1) PRIMARY KEY,
    nombre varchar(15) not null,
    telefono varchar(15) null )
GO
```

2. Inserte filas en la tabla ejecutando las siguientes instrucciones:

```
INSERT INTO pruebaIdentidad(nombre, telefono)
VALUES ('Gustavo', '99981234')

INSERT INTO pruebaIdentidad(nombre, telefono)
VALUES ('Ricardo', '2211234')

INSERT INTO pruebaIdentidad(nombre, telefono)
```

```
VALUES ('Gloria', '3493212')
GO
```

```
SELECT * FROM pruebaIdentidad
GO
```

En IDENTITY(1,1), el primer argumento establece el primer valor de identidad a generar, y el segundo argumento es el incremento a utilizar para generar los siguientes valores.

Ejercicio 4.5: Inserción de la fecha del sistema

La función **getdate()** entrega la fecha y hora del sistema. La puede utilizar para insertar la fecha y hora del sistema en una columna de tipo fecha-hora.

```
USE BibliotecaGyS
GO

INSERT INTO recibo(rec_num, rec_fec_emi, rec_mon)
VALUES(1001, getdate(), 10)
GO

SELECT * FROM recibo
GO
```

Ejercicio 4.6: Inserción de un valor de fecha específico

Para enviar un dato fecha al servidor, la fecha se envía como una cadena con formato de fecha. Por lo general, cuando la conexión es a un servidor SQL remoto, no sabemos cuál es el formato de fecha predeterminado que está utilizando el servidor, por lo que en ocasiones podemos tener problemas con la manipulación de las fechas.

5. Cree una tabla para hacer algunas pruebas con las fechas:

```
USE BibliotecaGyS
GO

CREATE TABLE PruebaFechas (
    fecha datetime not null )
GO
```



2. Ahora, inserte la fecha 26 de noviembre de 1980 mediante la siguiente instrucción:

```
-- Prueba 1: insertar 26 de noviembre de 1980
INSERT INTO PruebaFechas VALUES( '26 Nov 1980' )
GO

SELECT * FROM PruebaFechas
GO
```

Observe que la fecha se registró correctamente.

3. Ahora, vamos a suponer que la conexión al servidor SQL se realiza desde algún lugar de USA, donde se acostumbra a utilizar el formato mes/día/año para expresar las fechas.

```
-- Prueba 2: insertar la fecha 3 de enero
-- del 2001 usando el formato mes/día/año
INSERT INTO PruebaFechas VALUES( '01/03/2001' )
GO

SELECT * FROM PruebaFechas
GO
```

Resultados	
	Mensajes
1	fecha 1980-11-26 00:00:00.000
2	2001-03-01 00:00:00.000

Note que se ha insertado incorrectamente. La fecha registrada es 1 de marzo del 2001. El servidor con el que se ha desarrollado este libro tiene como formato predeterminado de fecha 'dd/mm/aaaa'.

4. Ahora, inserte la fecha 26 de noviembre de 1980 con la siguiente instrucción:

```
-- Prueba 3: insertar la fecha 26 de noviembre
-- de 1980 usando el formato mes/día/año
INSERT INTO PruebaFechas VALUES( '11/26/1980' )
GO
```

5. Se genera el siguiente mensaje de error:

Mens. 242, Nivel 16, Estado 3, Línea 1
La conversión del tipo de datos varchar en datetime
produjo un valor fuera de intervalo.
Se terminó la instrucción.



El valor de fecha se ha leído como el día 11 del mes 26 del año 1980, lo que produce un valor de fecha fuera de rango (fecha inválida). Recuerde que mi servidor lee las fechas en el formato día/mes/año.

Cuando se envía fechas al servidor se recomienda informar al servidor en qué formato se le está enviando las fechas para que las lea correctamente.

6. Infórmeme al servidor acerca del formato de sus fechas:

```
-- Informando al servidor en qué formato
-- se envían las cadenas de fecha
SET DATEFORMAT MDY
GO
```

7. Ahora, trate de insertar nuevamente la fecha 26 de noviembre de 1980 ejecutando la instrucción que generó el error.

```
-- Prueba 4: insertar la fecha 26 de noviembre
-- de 1980 usando el formato mes/día/año
INSERT INTO PruebaFechas VALUES( '11/26/1980' )
GO

SELECT * FROM PruebaFechas
GO
```

La fecha se inserta correctamente.



3. ACTUALIZACIÓN DE DATOS – INSTRUCCIÓN UPDATE)

Sintaxis

```
UPDATE nombre_tabla
    SET     columnaX = expresiónX,
            columnaP = expresiónP, ...
    [ WHERE condición_de_las_filas_a_actualizar ]
```

- **columnaX, columnaP** son las columnas cuyo contenido se actualizará.
- **expresiónX, expresiónP** establecen los nuevos valores a almacenar en las columnas *columnaX* y *columnaP* respectivamente.
- **condición_de_las_filas_a_actualizar** es una expresión lógica que determina las filas en las que la actualización se debe llevar a cabo.

Ejercicio 4.7: Uso de UPDATE

6. Ejecute las siguientes instrucciones en su ventana de consulta:

```
USE BibliotecaGyS
GO
```

```
SELECT * FROM usuario
GO
```

8. Note que el usuario "BARRIONUEVO" no tiene registrado su teléfono. Procederemos a hacerlo.

```
UPDATE usuario
SET usu_tel = '5714315'
WHERE usu_cod = '961002'
GO
```

```
SELECT * FROM usuario
WHERE usu_cod = '961002'
GO
```



4. ELIMINACIÓN DE FILAS – INSTRUCCIÓN DELETE)

Sintaxis

```
DELETE [ FROM ] nombre_tabla
      [ WHERE condición_de_las_filas_a_eliminar ]
```

- **condición_de_las_filas_a_eliminar** es una expresión lógica que determina las filas en las que la eliminación se debe llevar a cabo.

Ejercicio 4.8: Uso de DELETE

7. Ejecute las siguientes instrucciones en su ventana de código:

```
USE BibliotecaGyS
GO
```

```
SELECT * FROM usuario
GO
```

9. Elimine el registro del usuario cuyo código es 961004.

```
DELETE FROM usuario
WHERE usu_cod = '961004'
GO
```

```
SELECT * FROM usuario
GO
```



5. ELIMINACIÓN DE TODAS LAS FILAS DE UNA TABLA - **INSTRUCCIÓN TRUNCATE TABLE)**

Si bien la sentencia DELETE utilizada sin la cláusula WHERE permite eliminar todas las filas de una tabla, se recomienda el uso de TRUNCATE TABLE porque esta instrucción se ejecuta más rápido debido a que no registra la eliminación de cada fila en el log de transacciones de la base de datos como lo hace DELETE. La Sintaxis es:

```
TRUNCATE TABLE nombre_tabla
```

Sin embargo, esta instrucción no puede ejecutarse sobre una tabla que es referenciada por una clave foránea, independientemente de que la tabla que hace la referencia (la tabla que posee la clave foránea) tenga filas registradas o no.

Ejercicio 4.9: Uso de TRUNCATE TABLE

1. Ejecute las siguientes instrucciones en su ventana de consulta:

```
USE BibliotecaGyS
GO
```

```
TRUNCATE TABLE usuario
GO
```

2. Se genera el siguiente mensaje de error:

Mens. 4712, Nivel 16, Estado 1, Línea 1
No se puede truncar la tabla 'usuario'. Una restricción
FOREIGN KEY hace referencia a ella.



6. EJERCICIOS PROPUESTOS

1. Ejecute el script **BibliotecaGyS-Tablas.sql** para recrear la base de datos **BibliotecaGyS**.
2. Ejecute las instrucciones necesarias para registrar los siguientes datos en las tablas indicadas. Si alguna instrucción falla, explique porqué, y qué acción tomaría para corregir la falla.

Tabla TEMA

	Resultados	Mensajes
	tem_cod	tem_des
1	1	ANALISIS Y DISEÑO DE SISTEMAS
2	2	ADMINISTRADORES DE BASES DE DATOS
3	3	LENGUAJES DE PROGRAMACION

Tabla PUBLICACION

	Resultados	Mensajes			
	pub_cod	pub_nom	pub_aut	tem_cod	pub_sum
1	57	FOUNDATIONS OF POWERBUILDER PROGRAMMING	SMITH, BRIAN - SCHAAD, GORDON	4	NULL
2	55	MS VISUAL BASIC 6 - MANUAL DEL PROGRAMADOR	MICROSOFT PRESS	4	NULL
3	54	APRENDIENDO MS SQL SERVER 2000 EN 21 DIAS	WAYMIRE, RICHARD - SAWTELL, RICK	2	NULL
4	49	IMPLEMENTING A DATABASE ON MS SQL SERVER 7 - MOC ...	MICROSOFT	2	NULL
5	48	ORACLE 9i - GUIA DE APRENDIZAJE	ABBEY - COREY - ABRAMSIN	2	NULL
6	46	JAVA - GUIA DE DESARROLLO	JAWORSKI, JAMIE	3	NULL
7	44	DESARROLLO DE APLICACIONES C/S CON ERWIN 3. SQL S...	CORONEL CASTILLO, GUSTAVO	4	NULL
8	40	ANALISIS Y DISEÑO ORIENTADO A OBJETOS	MARTIN, JAMES - ODELL, JAMES	1	NULL
9	29	ORACLE - INTRODUCCION A PL/SQL V.2.0	ORACLE CORPORATION	2	NULL
10	28	MICROSOFT SQL SERVER - ADMINISTRATOR COMPANION	MICROSOFT	2	NULL

Tabla EJEMPLAR

	eje_num	pub_cod	eje_edi	eje_num_edi	eje_dis
1	1	28	MICROSFOT CORPORATION	1	N
2	1	29	ORACLE CORPORATION	1	S
3	1	40	PRENTICE HALL HISPANOAMERICANA S.A.	1	N
4	2	40	PRENTICE HALL HISPANOAMERICANA S.A.	1	S
5	3	40	PRENTICE HALL HISPANOAMERICANA S.A.	1	N



Tabla USUARIO

	usu_cod	usu_pat	usu_mat	usu_nom	usu_dir	usu_tel	usu_hab
1	971017	HUERTAS	MARCELO	ANA MARIA	JR. RECUAY 397 - BREÑA	NULL	1
2	971018	ARRIOLA	EVANS	MARCELA	LAS AMAZONAS S/N MZA. Z LOTE 2 - LOS OLIVOS	4523489	1
3	971019	JIMENEZ	MAC ALLISTER	DIANA	JR. GENERAL VIDAL 348 - SAN ISIDRO	NULL	1
4	971020	APOLAYA	CASTRO	IVONNE	BARRIO OBRERO 1378 - LA VICTORIA	4352378	1

Tabla PRESTAMO

	pub_cod	pre_fec_sal	eje_num	usu_cod	pre_fec_dev	pre_fec_ent	pre_mul_imp	pre_mul_pag	rec_num
1	40	2013-03-14 21:03:15.553	1	971020	2013-03-16 21:03:15.553	2013-03-16 21:03:15.553	NULL	NULL	NULL
2	40	2013-04-01 21:03:15.567	3	971017	2013-04-04 21:03:15.567	2013-04-04 21:03:15.567	NULL	NULL	NULL



Capítulo V

Introducción a las Consultas

En este capítulo conoceremos cómo leer los datos de una tabla y organizarlos en el informe de salida usando la instrucción SELECT.



1. PREPARACIÓN DEL ESCENARIO

Para desarrollar los temas de este capítulo utilizaremos la base de datos **QhatuPERU**, cuyo modelo de datos se presentó en el capítulo 3. En la carpeta **ScriptsBD** del CD incluido en el libro encontrará el script **CreaBaseDatosQhatuPERU.sql**; al ejecutarlo se creará la base de datos, sus tablas y relaciones, y además insertará una cantidad adecuada de registros de datos, necesarios para ilustrar los temas a presentar.

Ejercicio 5.1: Ejecución del script que crea la base de datos QhatuPERU

1. Inicie una sesión SQL Server Management Studio en su servidor SQL.
2. En el menú **Archivo**, seleccione **Abrir**, y Luego **Archivo**.
3. Abra el archivo **CreaBaseDatosQhatuPERU.sql**.
4. Para ejecutar el script, haga clic en **Ejecutar** de la barra de herramientas o pulse la tecla [F5].

```
-- Empresa      : QhatuPERU SAC
-- Software     : Sistema de Control de Inventarios
-- DBMS         : MS SQL Server
-- Base de Datos : QhatuPERU
-- Script        : Crea la base de datos QhatuPERU

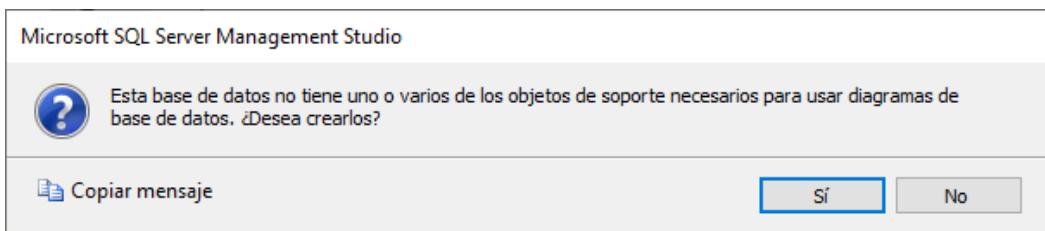
-----
-- Creación de la base de datos
-----
USE [master]
GO
```

Ejercicio 5.2: Creación de un diagrama de la base de datos

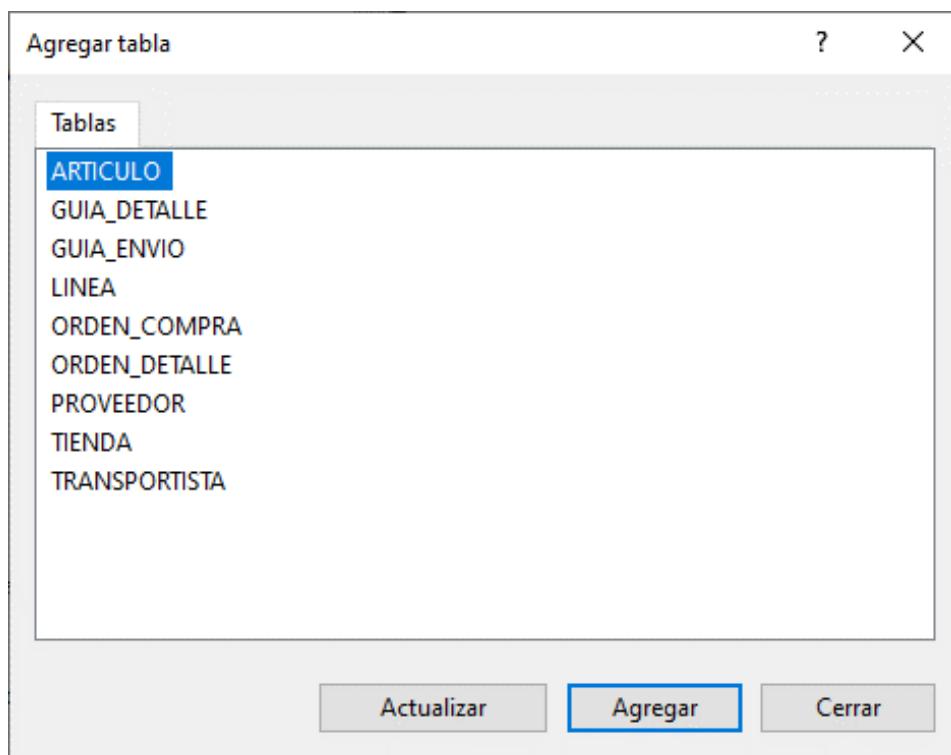
1. Si bajo el nodo **Bases de datos** de su servidor no ve la base de datos **QhatuPERU**, haga clic secundario sobre **Bases de datos** y ejecute **Actualizar**.
2. Expanda el nodo de la base de datos, haga clic secundario sobre **Diagramas de bases de datos**, y Luego clic sobre **Nuevo diagrama de base de datos**.



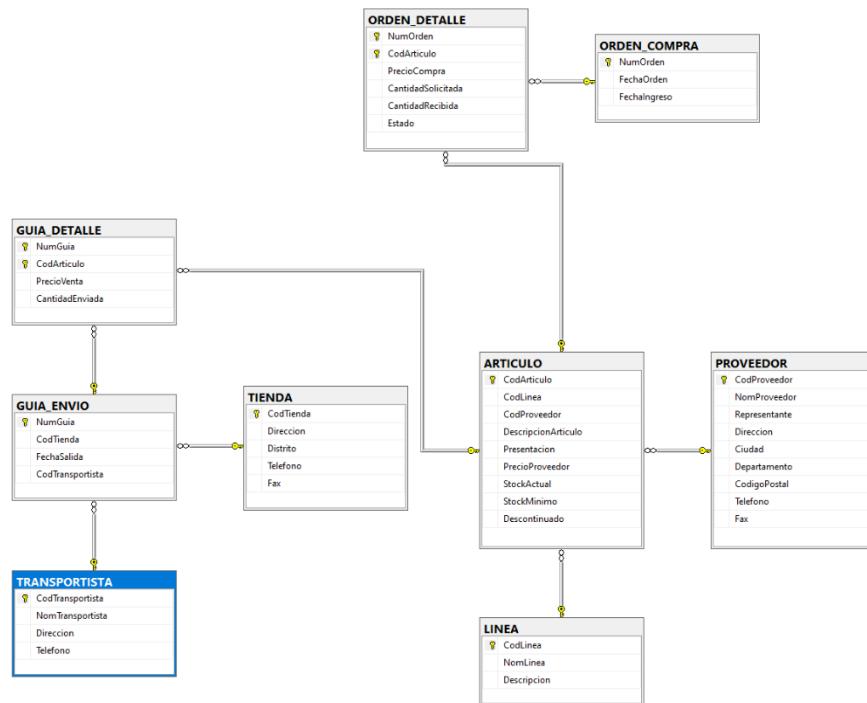
3. Si se muestra el siguiente mensaje, haga clic en **Sí**.



4. Se muestra el diálogo **Agregar tabla**.

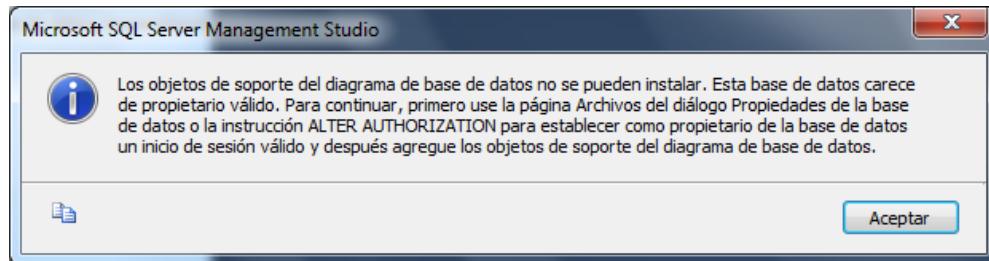


5. Seleccione las tablas que desea mostrar en el diagrama y haga clic en **Agregar**, y para finalizar haga clic en **Cerrar**.



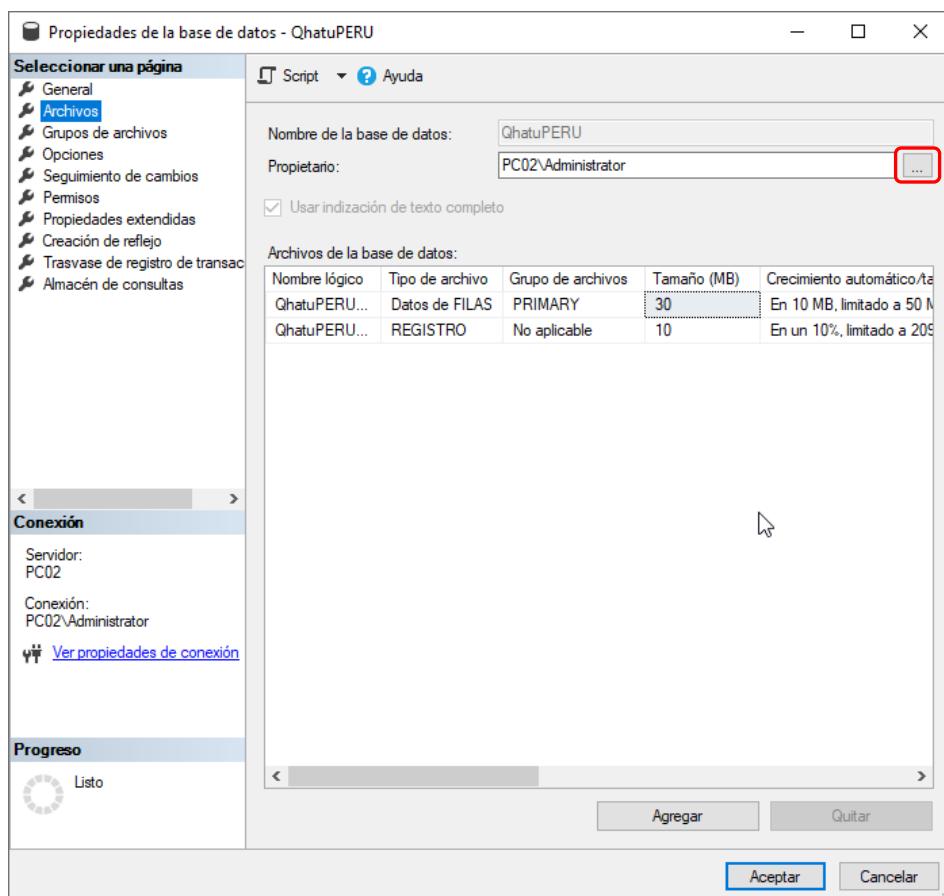
6. Guarde el diagrama bajo el nombre **ModeloCompleto**.

Nota: Si al intentar crear un diagrama para una base de datos recibe el siguiente mensaje:

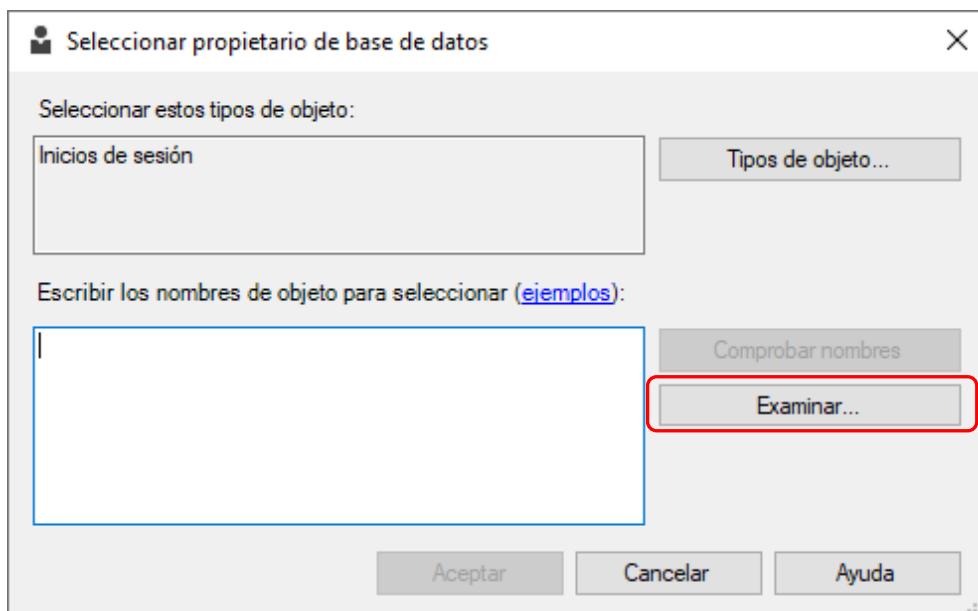


Proceda del siguiente modo:

1. Haga clic secundario sobre la base de datos, y en el menú contextual seleccione **Propiedades**.
2. En la ventana **Propiedades de la base de datos** seleccione la página **Archivos**.
3. En el campo **Propietario** haga clic sobre el botón **Buscar**.

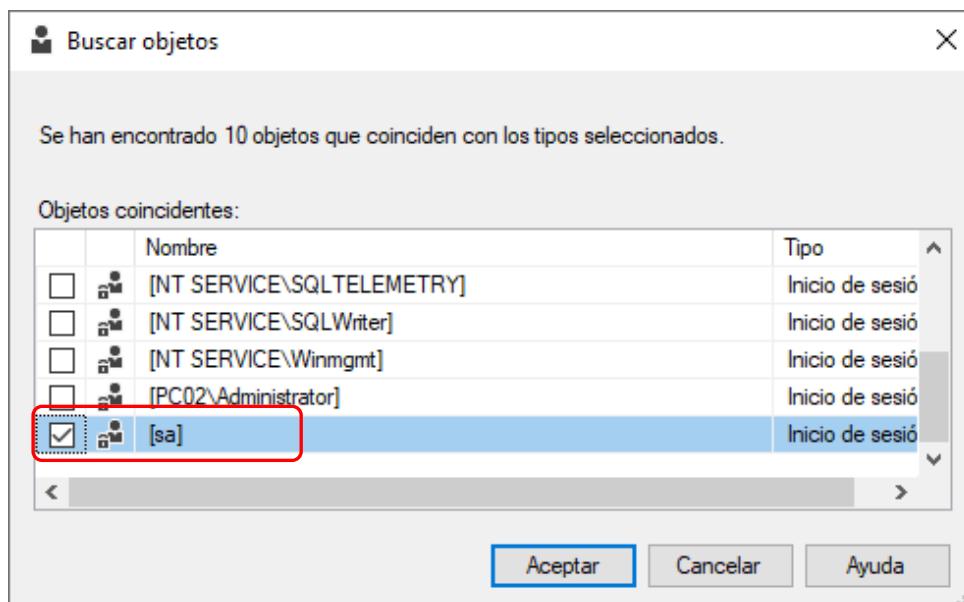


4. En la ventana **Seleccionar propietario de base de datos** haga clic en **Examinar**:

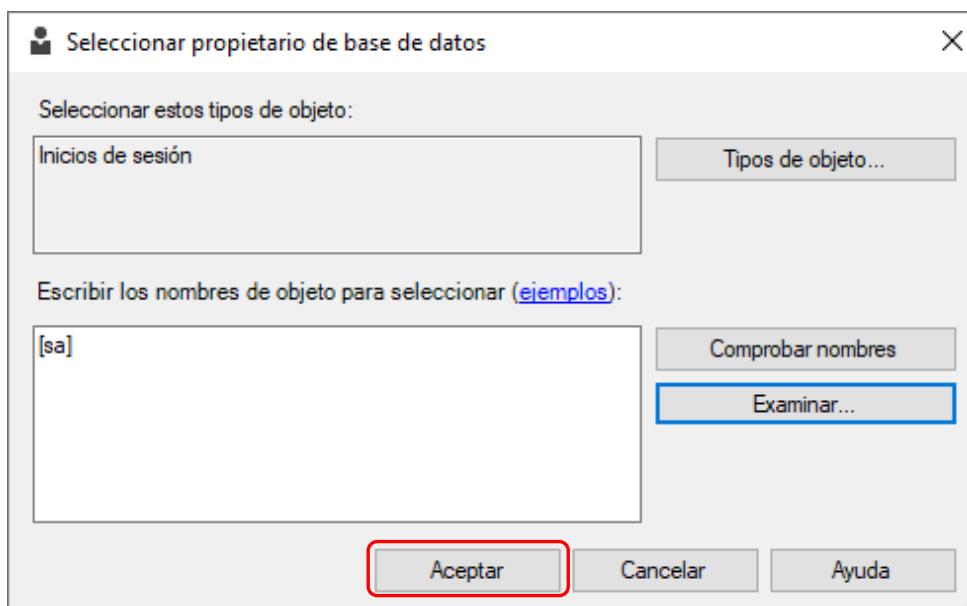




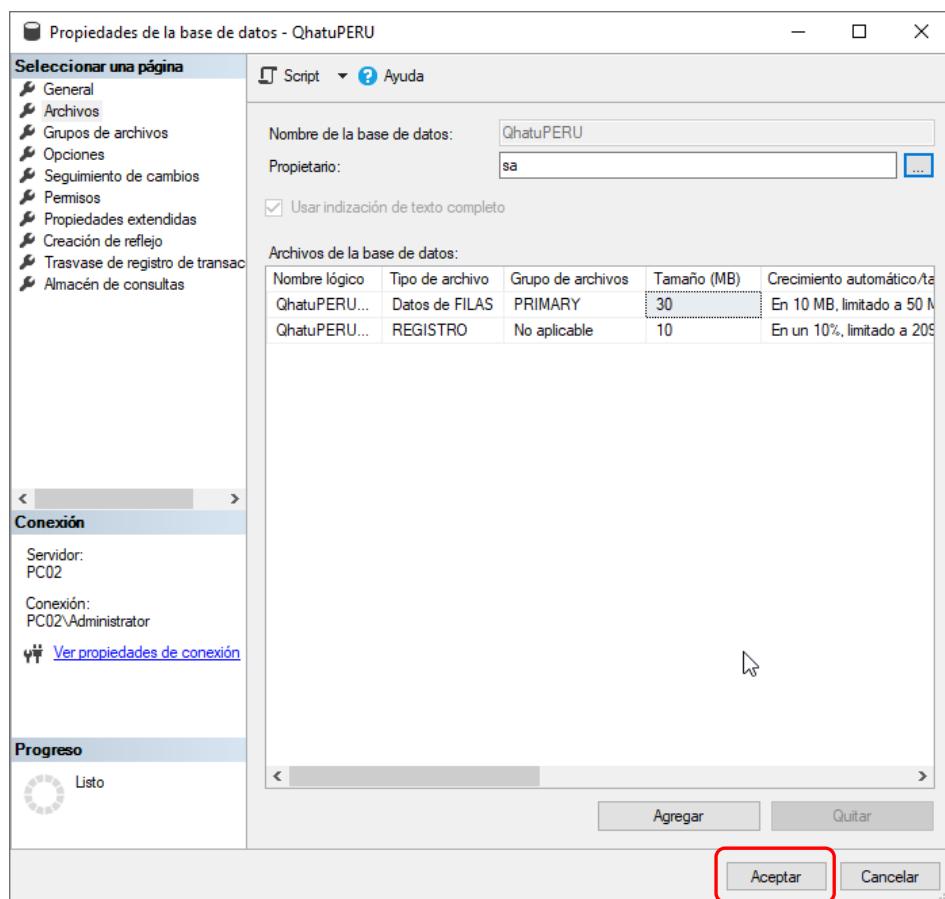
5. Busque la cuenta **sa**, selecciónela y haga clic en **Aceptar**. **sa** es la cuenta **System Administrator**.



6. Luego haga clic en el botón **Aceptar**.



7. Haga clic en **Aceptar** para regresar a la página **Archivos**, y Luego en **Aceptar** para cerrar la ventana **Propiedades de la base de datos**.



2. CONSULTAS A LAS TABLAS – INSTRUCCIÓN

SELECT

Sintaxis básica

```
SELECT * | lista_columnas
FROM nombre_tabla
[ WHERE condición_filas ]
```

- **lista_columnas** es la lista de columnas a mostrar en el resultado de la consulta. Si se especifica * se mostrarán todas las columnas de la tabla.
- **condición_filas** es una expresión lógica que indica que las filas a mostrar son aquellas para las que el valor de la expresión es verdadero.

Ejercicio 5.3: Lectura de todos los datos de una tabla

```
USE QhatuPERU
GO
```



```
SELECT * FROM ARTICULO  
GO
```

Muestra todas las columnas y todas las filas de la tabla ARTICULO de la base de datos **QhatuPERU**.

	CodArticulo	CodLinea	CodProveedor	DescripcionArticulo	Presentacion	PrecioProveedor
1	1	1	14	CARAMELOS BASTON VIENA ARCOR	PAQUETE 454 GR	1,50
2	2	1	15	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	1,00
3	3	1	14	CARAMELOS FRUTAS SURTIDA ARCOR	PAQUETE 520 GR	1,50
4	4	1	14	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	1,30
5	5	1	15	CHIQUETES LOLY AMBROSOLI	KILOGRAMO	1,20
6	6	1	15	FRUNA SURTIDA DONOFRIO	PAQUETE X 24 UNIDADES	1,80
7	7	1	15	CHOCOLATE DOÑA PEPA FIELD	PAQUETE X 6 UNIDADES	2,20
8	8	1	15	CHOCOLATE CUA CUA FIELD	PAQUETE X 6 UNIDADES	1,60
9	9	1	15	MELLOWS FAMILIAR FIELD	PAQUETE 454 GR	2,10
10	10	1	15	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES	0,70

Ejercicio 5.4: Lectura de columnas seleccionadas de una tabla

```
SELECT CodProveedor, NomProveedor,  
       Direccion, Ciudad  
FROM PROVEEDOR  
GO
```

El resultado muestra las columnas **CodProveedor**, **NomProveedor**, **Direccion** y **Ciudad** de la tabla PROVEEDOR.

	CodProveedor	NomProveedor	Direccion	Ciudad
1	1	LACTEOS DEL CENTRO	LIBERTAD 345 URB. EL PINO	HUANCAYO
2	2	DISTRIBUIDORA ALEMANA	AV. SAN VICENTE 1276 ATE-VITARTE	LIMA
3	3	EMBUTIDOS EL GORDITO	AV. VENEZUELA 5434 BELLAVISTA	CALLAO
4	4	DISTRIBUIDORA NANDO	LOS CONDORES 345 PARQUE INDUSTRIAL EL ALAMO	CALLAO
5	5	DISTRIBUIDORA ALBRICIAS	LEONCIO PRADO 625 MAGDALENA	LIMA
6	6	DISTRIBUIDORA DEL HOGAR	SAN MARTIN 1187 SMP	LIMA
7	7	PAPELERA PACHACAMAC	LOTIZACION INDUSTRIAL MZA. H LOTE 34	CAÑETE
8	8	DISTRIBUIDORA SAN ANTONIO	AV. DEL PACIFICO 5634 SAN JUAN	LIMA
9	9	EMBOTELLADORA LA PREFERIDA	HEROES DEL CENEPA 342 CERCADO	TRUJILLO
10	10	DROGUERIA MAHAN	AV. CIRCUNVALACION 625 ZONA INDUSTRIAL	AREQUIPA
11	11	QUIMICA DEL NORTE	JOSE CARLOS MARIEGUE 473 CERCADO	CHICLAYO



Ejercicio 5.5: Uso de alias para los títulos de columnas

```
SELECT      CodArticulo AS Código,  
            DescripcionArticulo AS Descripción,  
            Presentacion AS Unidad,  
            PrecioProveedor AS 'Precio unitario'  
FROM ARTICULO  
GO
```

Permite mostrar títulos alternativos para las columnas. Si el alias contiene espacios se debe escribir entre comillas.

	Código	Descripción	Unidad	Precio unitario
1	1	CARAMELOS BASTON VIENA ARCOR	PAQUETE 454 GR	1,50
2	2	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	1,00
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	PAQUETE 520 GR	1,50
4	4	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	1,30
5	5	CHUPETES LOLY AMBROSOLI	KILOGRAMO	1,20
6	6	FRUNA SURTIDA DONOFRIO	PAQUETE X 24 UNIDADES	1,80
7	7	CHOCOLATE DOÑA PEPA FIELD	PAQUETE X 6 UNIDADES	2,20
8	8	CHOCOLATE CUA CUA FIELD	PAQUETE X 6 UNIDADES	1,60
9	9	MELLOWS FAMILIAR FIELD	PAQUETE 454 GR	2,10
10	10	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES	0,70
11	11	CHOCOLATE BARRA REGULAR	BARRA 2 ONZAS	0,40

Las siguientes son formas alternas para definir los títulos de columnas usando alias.

```
SELECT      Código = CodArticulo,  
            Descripción = DescripcionArticulo,  
            Unidad = Presentacion,  
            'Precio unitario' = PrecioProveedor  
FROM ARTICULO  
GO
```

```
SELECT      CodArticulo Código,  
            DescripcionArticulo Descripción,  
            Presentacion Unidad,  
            PrecioProveedor 'Precio unitario'  
FROM ARTICULO  
GO
```



Ejercicio 5.6: Uso de columnas calculadas

Una columna computada es una columna que se muestra en el resultado de una consulta, pero que no existe físicamente como tal en la tabla. La columna computada muestra el resultado de ejecutar alguna operación con las columnas de la tabla.

```
SELECT CodArticulo, PrecioVenta, CantidadEnviada,  
Monto = PrecioVenta * CantidadEnviada  
FROM GUIA_DETALLE  
GO
```

El contenido de la columna calculada **Monto** se obtiene operando con los valores de las columnas **PrecioVenta** y **CantidadEnviada**.

	CodArticulo	PrecioVenta	CantidadEnviada	Monto
1	1	2,25	20	45,00
2	2	1,50	20	30,00
3	3	2,25	20	45,00
4	4	1,95	20	39,00
5	6	2,70	30	81,00
6	7	3,30	25	82,50
7	8	2,40	25	60,00
8	9	3,15	20	63,00
9	10	1,05	20	21,00
10	11	0,60	50	30,00
11	12	2,25	50	112,50

Ejercicio 5.7: Uso del operador + para concatenar cadenas en el resultado

```
SELECT CodTienda,  
Direccion + ' - ' + Distrito AS Ubicación  
FROM TIENDA  
GO
```

En el ejemplo se concatena las cadenas almacenadas en los campos **Direccion** y **Distrito**, y se muestran como una sola cadena en la columna **Ubicación** del resultado.



Resultados		
	CodTienda	Ubicación
1	1	AV. LA PAZ 659 - MIRAFLORES
2	2	AV. BOLIVAR 1789 - PUEBLO LIBRE
3	3	AV. SAENZ PEÑA 590 - CALLAO
4	4	PANAMERICANA NORTE KM. 17.5 - LOS OLIVOS
5	5	AV. ESPAÑA 775 - BREÑA

3. ESTABLECIENDO FILTROS DE FILA EN LA INSTRUCCIÓN SELECT

En la cláusula WHERE, **condición_filas** es una expresión lógica que establece la condición que deben cumplir las filas a mostrar en el resultado de la consulta. Para construir la expresión lógica utilice operadores relacionales o de comparación, operadores lógicos como NOT, AND y OR, y operadores lógicos SQL como LIKE, BETWEEN e IN.

Ejercicio 5.8: Uso del operador de igualdad (=)

```
USE QhatuPERU
GO

SELECT CodArticulo, DescripcionArticulo,
       Presentacion, PrecioProveedor
FROM ARTICULO
WHERE CodLinea = 5
GO
```

Muestra la lista de artículos de la línea 5 (licores y gaseosas).

Resultados		
	CodArticulo	DescripcionArticulo
1	109	INCA KOLA DIET DESCARTABLE
2	110	INCA KOLA DESCARTABLE
3	111	INCA KOLA DIET
4	112	INCA KOLA PLASTIFORMA DESCARTABLE
5	113	SPRITE DESCARTABLE
6	114	SPRITE CONTOUR
7	115	SPRITE DESCARTABLE
8	116	SPRITE RETORNABLE
9	117	TRIPLE DIET NO RETORNABLE
10	118	7 UP DESCARTABLE
11	119	WHISKY JOHNNIE WALKER ETIQUETA ROJA

Ejercicio 5.9: Uso del operador diferente (<>, !=)

```
SELECT CodProveedor, NomProveedor,
Telefono, Departamento
FROM PROVEEDOR
WHERE Departamento <> 'Lima'
GO
```

Muestra a los proveedores que no están ubicados en el departamento de **Lima**.

	CodProveedor	NomProveedor	Telefono	Departamento
1	1	LACTEOS DEL CENTRO	NULL	JUNIN
2	9	EMBOTELLADORA LA PREFERIDA	NULL	LA LIBERTAD
3	10	DROGUERIA MAHAN	NULL	AREQUIPA
4	11	QUIMICA DEL NORTE	NULL	LAMBAYEQUE
5	14	GOLOSINAS Y ANTOJOS	NULL	AREQUIPA

Ejercicio 5.10: Uso del operador menor que (<)

```
SELECT CodArticulo, DescripcionArticulo,
Presentacion, PrecioProveedor
FROM ARTICULO
WHERE PrecioProveedor < 1.50
GO
```

Muestra los artículos con un precio menor a **1.50**.

	CodArticulo	DescripcionArticulo	Presentacion	PrecioProveedor
1	2	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	1,00
2	4	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	1,30
3	5	CHUPETES LOLY AMBROSOLI	KILOGRAMO	1,20
4	10	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES	0,70
5	11	CHOCOLATE BARRA REGULAR	BARRA 2 ONZAS	0,40
6	13	CHOCOLATE BARRA MILKY WAY	BARRA 2.15 ONZAS	0,80
7	14	SNICKERS BAR KING SIZE	BARRA 3.7 ONZAS	1,20
8	15	CHOCOLATE BARRA MILK DOVE	UNIDAD	1,30
9	16	CHOCOLATE BARRA DARK DOVE	UNIDAD	1,30
10	18	GALLETAS CHIPS AHOY	PAQUETE X 6 UNIDADES	1,00
11	19	GALLETAS TUAREG COSTA	PAQUETE X 6 UNIDADES	1,20



Ejercicio 5.11: Uso del operador mayor que (>)

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo > 'T'
GO
```

Muestra los artículos cuya descripción es posterior a t cuando todos los artículos se ordenan alfabéticamente por su descripción.

Resultados		Mensajes
	CodArticulo	DescripcionArticulo
1	117	TRIPLE DIET NO RETORNABLE
2	10	WAFER CHOCOLATE FIELD
3	122	WHISKY CHIVAS REGAL
4	120	WHISKY JOHNNIE WALKER ETIQUETA NEGRA
5	119	WHISKY JOHNNIE WALKER ETIQUETA ROJA
6	121	WHISKY SOMETHING SPECIAL
7	123	WHISKY YE MONKS
8	95	YOGURT GLORIA FRESA
9	97	YOGURT LAIVE FRESA
10	96	YOGURT YOLEIT FRESA

Ejercicio 5.12: Manipulación de datos de tipo fecha-hora con los operadores de comparación

1. Ejecute la siguiente instrucción en su ventana de consulta:

```
SET DATEFORMAT DMY
GO
```

```
SELECT NumOrden, FechaOrden
FROM ORDEN_COMPRA
GO
```

La lista muestra 2 órdenes cuya fecha de emisión es el **11 de abril del 2013**.



Resultados		
	NumOrden	FechaOrden
1	1	2013-04-09 20:38:06.400
2	2	2013-04-09 20:38:06.410
3	3	2013-04-09 20:38:06.417
4	4	2013-04-09 20:38:06.420
5	5	2013-04-11 20:38:06.423
6	6	2013-04-11 20:38:06.440
7	7	2013-04-12 20:38:06.440
8	8	2013-04-12 20:38:06.443
9	9	2013-04-12 20:38:06.443
10	10	2013-04-12 20:38:06.453
11	11	2013-04-12 20:38:06.477

2. Ahora, ejecute la siguiente instrucción:

```
SELECT NumOrden, FechaOrden
FROM ORDEN_COMPRA
WHERE FechaOrden = '11/04/2013'
GO
```

El listado resultante no muestra ninguna orden para el **11 de abril del 2013**. El problema no es de formato de fecha ya que previamente se le informó al servidor que la cadena de fecha se le está enviando en formato **DMY**.

La columna **FechaOrden** es de tipo fecha-hora; por lo tanto almacena la fecha y además una hora, es decir, que representa un instante en el tiempo. La instrucción SELECT que se ha enviado no especifica la hora, por lo que el motor de búsqueda asume que estamos buscando la fecha '11/04/2013 00:00:00.000'.

3. Ahora, ejecute la siguiente consulta:

```
SELECT NumOrden, FechaOrden
FROM ORDEN_COMPRA
WHERE FechaOrden <= '11/04/2013'
GO
```

El listado resultante muestra las órdenes cuya fecha de salida es anterior al **11 de abril del 2013**, pero las órdenes de dicha fecha no se muestran.

Al no indicar la hora en la fecha de búsqueda, el motor asume que la condición de búsqueda es la siguiente:

```
WHERE FechaOrden <= '11/04/2013 00:00:00.000'
GO
```

Entonces, ¿cómo debemos especificar la búsqueda sin tener que indicar un instante preciso del tiempo como valor de búsqueda?

3.1. Uso de la función CONVERT() con datos de tipo fecha-hora

Sintaxis

```
CONVERT( char(n) , expresión_fecha , estilo )
```

Convierte **expresión_fecha** a una cadena con formato de fecha. La cadena tiene longitud **n**, y el formato del dato se establece con el valor de **estilo**.

- **char(n)**, es el tipo y la longitud de la cadena con formato de fecha resultante.
- **expresión_fecha**, representa la fecha cuyo formato de presentación se desea cambiar.
- **estilo**, indica el formato de presentación de la cadena resultante.

El siguiente cuadro muestra algunos valores de estilo para la función CONVERT.

Año con 2 dígitos	Año con 4 dígitos	Formato	Salida
1	101	USA	mm/dd/aa
2	102	ANSI	aa.mm.dd
3	103	Británico/Francés	dd/mm/aa
4	104	Alemán	dd.mm.aa
5	105	Italiano	dd-mm-aa
6	106		dd mes aa
7	107		Mes dd, aa

Ejercicio 5.13: Búsqueda en columna de tipo fecha-hora

Se desea obtener el listado de las órdenes cuya fecha de emisión es anterior o igual al **11 de abril del 2013**.



```
SELECT NumOrden, FechaOrden
FROM ORDEN_COMPRA
WHERE CONVERT(CHAR(10), FechaOrden, 102)
    <= '2013.04.11'
GO
```

```
SELECT NumOrden, FechaOrden
FROM ORDEN_COMPRA
WHERE CONVERT(CHAR(10), FechaOrden, 103)
    <= '11/04/2013'
GO
```

	NumOrden	FechaOrden
1	1	2013-04-09 20:38:06.400
2	2	2013-04-09 20:38:06.410
3	3	2013-04-09 20:38:06.417
4	4	2013-04-09 20:38:06.420
5	5	2013-04-11 20:38:06.423
6	6	2013-04-11 20:38:06.440

Los datos de tipo fecha-hora se convierten a cadenas y se comparan con una cadena con el formato de fecha adecuado.

Nota: a partir de la versión 2022 de MS SQL Server ya puede crear columnas que solo almacenen la fecha ya que se ha incluido el tipo de dato **date**.

3.2. Búsqueda basada en cadena de caracteres – El operador LIKE

Sintaxis

```
SELECT * | lista_columnas
FROM nombre_tabla
WHERE columna LIKE expresión_cadena_a_buscar
```

- **columna**, es la columna en la que se busca la cadena de caracteres.
- **expresión_cadena_a_buscar**, indica como debe ser la cadena que se está buscando en columna. La expresión admite comodines.

Los comodines en el operador LIKE

El siguiente cuadro muestra los comodines que puede utilizar con el operador LIKE.

Comodín	Descripción
%	Indica que en la posición del comodín puede ir cualquier cadena de caracteres, incluso una cadena nula.
_	Indica que en la posición del comodín puede ir cualquier carácter no nulo.
[abc]	Establece el conjunto de caracteres válidos en la posición del comodín.
[a-b]	Establece el rango de caracteres válidos en la posición del comodín.
^	Excluir. Indica que el carácter, conjunto de caracteres, o rango de caracteres que sigue al símbolo ^ no debe figurar en el resultado de la consulta.

Ejercicio 5.14: Uso del comodín %

Se desea obtener una relación de los artículos que contienen la cadena '**gloria**' en su descripción.

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE 'gloria'
GO
```

El resultado tiene 0 filas. La búsqueda especifica que el contenido de la columna **DescripcionArticulo** debe ser exactamente la cadena '**gloria**'.

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE 'gloria%'
GO
```

El resultado tiene 0 filas. La expresión de búsqueda indica que el contenido de la columna **DescripcionArticulo** debe empezar con la cadena '**gloria**'.

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE '%gloria'
GO
```

El resultado tiene 0 filas. La expresión de búsqueda indica que el contenido de la columna **DescripcionArticulo** debe terminar con la cadena '**gloria**'.

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE '%gloria%'
GO
```

La expresión de búsqueda indica que la cadena '**gloria**' puede encontrarse en cualquier posición del contenido de la columna **DescripcionArticulo**.

Resultados		Mensajes
	CodArticulo	DescripcionArticulo
1	95	YOGURT GLORIA FRESA

Ejercicio 5.15: Uso del comodín _

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE '%blanc_'
GO
```

La expresión de búsqueda indica que el contenido de la columna **DescripcionArticulo** debe tener la cadena '**blanc**' antes del último carácter de la descripción. El último carácter puede ser cualquiera.

Resultados		Mensajes
	CodArticulo	DescripcionArticulo
1	126	DETERGENTE LIMON ECOLOGICO MAGIA BLANCA
2	64	JABON DOVE BLANCO
3	65	JABON ROSAS Y LIMON BLANCO
4	68	PASTA DENTAL KOLYNOS SUPER BLANCO

Ejercicio 5.16: Uso del comodín [abc]

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE '[pdf]%'
ORDER BY DescripcionArticulo
GO
```



La expresión de búsqueda indica que el primer carácter en el contenido de la columna **DescripcionArticulo** puede ser cualquiera del conjunto (**pdf**); es decir, que el primer carácter puede ser **p**, o **d**, o **f**.

	CodArticulo	DescripcionArticulo
1	124	DETERGENTE C/BLANQUEADOR ARIEL
2	131	DETERGENTE C/BLANQUEADOR ARIEL
3	128	DETERGENTE LIMON ARIEL
4	125	DETERGENTE LIMON ARIEL
5	129	DETERGENTE LIMON ARIEL
6	126	DETERGENTE LIMON ECOLOGICO MAGIA BLANCA
7	130	DETERGENTE LIMON INVICTO
8	132	DETERGENTE LIMON OPAL
9	127	DETERGENTE PODER LIMON ACE
10	106	DORINA CLASICA
11	6	FRUNA SURTIDA DONOFRIO
12	23	FUDGE SHOPPE DELUXE GRAHAMS
13	24	FUDGE SHOPPE STICKS KEEB
14	77	P.H. BLANCO ROLL KLEENEX
15	76	P.H. BLANCO ROLL KLEENEX
16	74	P.H. BLANCO SUAVE (ROJA)
17	75	P.H. BLANCO SUAVE DOBLE HOJA (AZUL)
18	81	P.H. ELITE BLANCO EXTRA

Ejercicio 5.17: Uso del comodín [a-b]

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE '[d-p]%'
ORDER BY DescripcionArticulo
GO
```

La expresión de búsqueda indica que el primer carácter en el contenido de la columna **DescripcionArticulo** puede ser cualquiera del rango que va desde el carácter **d** hasta el carácter **p**.



	CodArticulo	DescripcionArticulo
1	124	DETERGENTE C/BLANQUEADOR ARIEL
2	131	DETERGENTE C/BLANQUEADOR ARIEL
3	128	DETERGENTE LIMON ARIEL
4	125	DETERGENTE LIMON ARIEL
5	129	DETERGENTE LIMON ARIEL
6	126	DETERGENTE LIMON ECOLOGICO MAGIA BLANCA
7	130	DETERGENTE LIMON INVICTO
8	132	DETERGENTE LIMON OPAL
9	127	DETERGENTE PODER LIMON ACE
10	106	DORINA CLASICA
11	6	FRUNA SURTIDA DONOFRIO
12	23	FUDGE SHOPPE DELUXE GRAHAMS
13	24	FUDGE SHOPPE STICKS KEEB
14	18	GALLETAS CHIPS AHYO
15	25	GALLETAS DELICE
16	21	GALLETAS SURTIDAS BUTTER COOKIES
17	19	GALLETAS TUAREG COSTA
18	20	GALLETAS VAINILLA COSTA

Ahora, repita la consulta anterior, pero invierta el rango en la expresión de búsqueda.

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE '[p-d]%'
ORDER BY DescripcionArticulo
GO
```

El resultado no devuelve filas, pero el sistema no genera ningún mensaje de error. ¿Por qué? Revise el tema **Intercalación o Server Collation** en la ayuda del producto.

Ejercicio 5.18: Uso del comodín ^

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE '[^pdf]%'
ORDER BY DescripcionArticulo
GO
```



La expresión de búsqueda indica que el primer carácter en el contenido de la columna **DescripcionArticulo** puede ser cualquiera menos los especificados en el conjunto (**pdf**); es decir, que el primer carácter no puede ser **p**, o **d**, o **f**.

	CodArticulo	DescripcionArticulo
1	118	7 UP DESCARTABLE
2	53	ACEITE BABY JOHNSONS
3	51	ACEITE BABY JOHNSONS C/ALOE Y VIT. E
4	54	ACEITE BABY JOHNSONS C/ALOE Y VIT. E
5	52	ACEITE JOHNSONS
6	56	ACEITE JOHNSONS CREMOSO
7	58	ACEITE P/BEBES CHICCO
8	55	ACEITE P/BEBES DR. ZAIDMAN
9	57	ACEITE P/BEBES NINET
10	1	CARAMELOS BASTON VIENA ARCOR
11	4	CARAMELOS FRUTAS MASTICABLES
12	3	CARAMELOS FRUTAS SURTIDA ARCOR
13	2	CARAMELOS SURTIDO DE FRUTAS
14	62	CEPILLO ADVANTAGE-60 CONTROL
15	61	CEPILLO DENTAL FLEX ADULTO
16	63	CEPILLO MASTER ADULTO
17	59	CEPILLO ODONTOLOGICA EXTRA MEDIA...
18	60	CEPILLO ODONTOLOGICA EXTRA SUAVE
19	16	CHOCOLATE BARRA DARK DOVE

Ahora, ejecute la siguiente consulta:

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo LIKE '[^d-p]%'
ORDER BY DescripcionArticulo
GO
```

La expresión de búsqueda indica que el primer carácter en el contenido de la columna **DescripcionArticulo** puede ser cualquiera menos los que se encuentran en el rango que va desde el carácter **d** hasta el carácter **p**.



Resultados		Mensajes
	CodArticulo	DescripcionArticulo
34	5	CHUPETES LOLY AMBROSOLI
35	98	CREAM CHESSE LAIVE
36	93	CREMA DE LECHE DUPRE
37	92	CREMA DE LECHE LAIVE
38	94	CREMA DE LECHE NESTLE
39	99	CREMA DE QUESO LAIVE
40	100	QUESO CREMA MILKITO
41	42	SALCHICHA DE HUACHO
42	84	SHAMPOO ALBERTO VO5 CABELLO NOR...
43	86	SHAMPOO ALBERTO VO5 JOJOBA
44	85	SHAMPOO HERBAL CABELLO FINO
45	87	SHAMPOO HERBAL CABELLO GRASO
46	83	SHAMPOO HERBAL CABELLO NORMAL
47	82	SHAMPOO HERBAL CABELLO SECO/DA...
48	91	SHAMPOO PANTENE CABELLO GRASO
49	89	SHAMPOO PANTENE CABELLO NORMAL

3.3. Búsqueda basada en rango de valores – El operador BETWEEN

El operador BETWEEN permite ejecutar consultas que ejecutan búsquedas basadas en rango de valores numéricos, valores de cadena, o valores de fecha.

Sintaxis

```
SELECT * | lista_columnas
FROM nombre_tabla
WHERE columna BETWEEN valor_inicial AND valor_final
```

BETWEEN indica que el valor en columna debe encontrarse en el rango definido por **valor_inicial** y **valor_final**.

- **columna**, es la columna en la que se busca según el rango especificado por **valor_inicial** y **valor_final**.
- **valor_inicial, valor_final**, establecen los límites del rango de valores en el que se basa la búsqueda.

BETWEEN define siempre un intervalo cerrado.



Ejercicio 5.19: Búsqueda basada en rango numérico

```
USE QhatuPERU
GO
```

```
SELECT CodArticulo, DescripcionArticulo,
Presentacion, PrecioProveedor
FROM ARTICULO
WHERE PrecioProveedor BETWEEN 5.50 AND 7
GO
```

Entrega una lista de artículos cuyo **PrecioProveedor** se encuentra en el rango que va desde **5.50** hasta **7.00** incluyendo a los límites.

	CodArticulo	DescripcionArticulo	Presentacion	PrecioProveedor
1	37	HOT DOG LAIVE PELADO	KILOGRAMO	5,50
2	38	HOT DOG LA SEGOVIANA	KILOGRAMO	6,80
3	84	SHAMPOO ALBERTO VO5 CABELLO NORMAL	FRASCO 15 ONZAS	5,50
4	86	SHAMPOO ALBERTO VO5 JOJOBA	FRASCO 15 ONZAS	5,50
5	87	SHAMPOO HERBAL CABELLO GRASO	FRASCO 355 ML	6,90
6	88	SHAMPOO SEDAL CERAMIDAS 2 EN 1	FRASCO 315 ML	5,90
7	90	SHAMPOO SEDAL DUO	FRASCO 315 ML	5,90
8	108	LECHE CULTIVADA MILKITO FRESA	ENVASE 1 LT	6,00

Ejercicio 5.20: Búsqueda basada en rango de valores cadena

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE DescripcionArticulo BETWEEN 'fru' AND 'hot'
GO
```

Entrega una lista de artículos cuya descripción se encuentra en el rango que va desde la cadena **'fru'** hasta la cadena **'hot'**.



	CodArticulo	DescripcionArticulo
1	6	FRUNA SURTIDA DONOFRIO
2	23	FUDGE SHOPPE DELUXE GRAHAMS
3	24	FUDGE SHOPPE STICKS KEEB
4	18	GALLETAS CHIPS AHoy
5	25	GALLETAS DELICE
6	21	GALLETAS SURTIDAS BUTTER COOKIES
7	19	GALLETAS TUAREG COSTA
8	20	GALLETAS VAINILLA COSTA

Observe que el resultado no incluye al producto 'HOT DOG' ya que esta cadena se encuentra después de la cadena 'HOT', por lo tanto está fuera del rango especificado.

Ejercicio 5.21: Búsqueda basada en rango de valores fecha para columnas fecha-hora

Primero, veamos que fechas tenemos registradas en la tabla GUIA_ENVIO.

	NumGuia	FechaSalida
56	14	2013-04-04 20:38:04.663
57	19	2013-04-04 20:38:04.733
58	24	2013-04-04 20:38:04.770
59	29	2013-04-04 20:38:04.833
60	34	2013-04-05 20:38:04.880
61	39	2013-04-05 20:38:04.913
62	44	2013-04-05 20:38:04.960
63	49	2013-04-05 20:38:05.023
64	54	2013-04-05 20:38:05.070
65	59	2013-04-05 20:38:05.130
66	64	2013-04-06 20:38:05.197

Observe que tenemos varias guías cuya FechaSalida es **5 de abril del 2013**.

Ahora, trate de obtener una lista de las guías emitidas desde el **25 de marzo del 2013** al **5 de abril del 2013**.

```
SET DATEFORMAT DMY
GO
SELECT NumGuia, FechaSalida
FROM GUIA_ENVIO
WHERE FechaSalida BETWEEN '25/03/2013'
    AND '05/04/2013'
ORDER BY FechaSalida
GO
```



	NumGuia	FechaSalida
29	53	2013-03-31 20:38:05.060
30	58	2013-03-31 20:38:05.113
31	63	2013-04-01 20:38:05.180
32	68	2013-04-01 20:38:05.220
33	73	2013-04-01 20:38:05.283
34	78	2013-04-01 20:38:05.330
35	83	2013-04-01 20:38:05.400
36	88	2013-04-01 20:38:05.487
37	9	2013-04-04 20:38:04.597
38	14	2013-04-04 20:38:04.663
39	19	2013-04-04 20:38:04.733
40	24	2013-04-04 20:38:04.770
41	29	2013-04-04 20:38:04.833

Las guías del **5 de abril del 2013** no se muestran en el listado. ¿Por qué?

Para la consulta ejecutada, el rango establecido señala que empieza a las **00:00 horas del 25 de marzo del 2013**, y termina a las **00:00 horas del 5 de abril del 2013**. Por lo tanto, cualquier guía emitida después de las 00:00 horas del día 5 de abril se encuentra fuera del rango.

Para incluir las guías del día 5 de abril podemos escribir la consulta así:

```
SELECT NumGuia, FechaSalida
FROM GUIA_ENVIO
WHERE FechaSalida BETWEEN '25/03/2013'
    AND '05/04/2013 23:59:59.999'
ORDER BY FechaSalida
GO
```

	NumGuia	FechaSalida
35	83	2013-04-01 20:38:05.400
36	88	2013-04-01 20:38:05.487
37	9	2013-04-04 20:38:04.597
38	14	2013-04-04 20:38:04.663
39	19	2013-04-04 20:38:04.733
40	24	2013-04-04 20:38:04.770
41	29	2013-04-04 20:38:04.833
42	34	2013-04-05 20:38:04.880
43	39	2013-04-05 20:38:04.913
44	44	2013-04-05 20:38:04.960
45	49	2013-04-05 20:38:05.023
46	54	2013-04-05 20:38:05.070
47	59	2013-04-05 20:38:05.130

3.4. Búsqueda basada en conjunto de valores – El operador IN

El operador IN permite ejecutar consultas que ejecutan búsquedas basadas en conjuntos de valores numéricos, valores de cadena, o valores de fecha.

Sintaxis

```
SELECT * | lista_columnas
FROM nombre_tabla
WHERE columna [NOT] IN ( conjunto_de_valores )
```

IN indica que el valor en **columna** debe encontrarse (IN), o no debe encontrarse (NOT IN) en el conjunto definido por **conjunto_de_valores**.

- **columna**, es la columna en la que se busca según el conjunto de valores especificado en **conjunto_de_valores**.
- **conjunto_de_valores**, establece el conjunto de valores en el que se basa la búsqueda.

Ejercicio 5.22: Búsqueda basada en conjunto de valores

```
SELECT CodArticulo, DescripcionArticulo, CodProveedor
FROM ARTICULO
WHERE CodProveedor IN (7,1,3)
GO
```

Muestra un listado de los artículos registrados para los proveedores cuyo **CodProveedor** es **7 o 1 o 3**.

Resultados			Mensajes
	CodArticulo	DescripcionArticulo	CodProveedor
1	26	JAMONADA LAIVE	1
2	32	JAMON INGLES LAIVE	1
3	36	JAMON YORK SALCHICHERIA ALEMANA	3
4	37	HOT DOG LAIVE PELADO	1
5	40	HOT DOG CERDEÑA	3
6	44	CHORIZO PARRILLERO LAIVE	1
7	47	CHORIZO PARRILLERO CERDEÑA	3
8	50	CHORIZO PARRILLERO CATALANES	3
9	74	P.H. BLANCO SUAVE (ROJA)	7
10	75	P.H. BLANCO SUAVE DOBLE HOJA (AZUL)	7
11	78	P.H. ELITE DOBLE HOJA	7
12	79	P.H. ELITE BLANCO EXTRA	7
13	81	P.H. ELITE BLANCO EXTRA	7
14	92	CREMA DE LECHE LAIVE	1
15	97	YOGURT LAIVE FRESCA	1
16	98	CREAM CHESSE LAIVE	1
17	99	CREMA DE QUESO LAIVE	1



4. MANIPULACIÓN DE VALORES NULL

Un valor NULL indica que el valor es desconocido, no aplicable, o que simplemente se registrará posteriormente. Un valor NULL es distinto a una cadena vacía o al valor cero (0), y también es distinto a cualquier otro valor NULL.

La comparación ú operación entre dos valores NULL, o entre un valor NULL y cualquier otro valor retorna un valor desconocido (otro valor NULL), ya que cada valor NULL es desconocido y distinto a cualquier otro NULL.

Ejercicio 5.23: Búsqueda de valores NULL

Para ilustrar este tema necesitamos una tabla que contenga una columna NULL, y que además dicha columna tenga registrados algunos valores distintos a NULL. Vamos a crear dicha tabla ejemplo.

```
USE QhatuPERU
GO

CREATE TABLE Empleado (
    IdEmpleado int PRIMARY KEY,
    Apellido varchar(30) not null,
    HaberBasico money not null,
    PorcentajeComision decimal(3,1) null )
GO

INSERT INTO Empleado
    VALUES(1, 'CASTRO ARENAS', 1200, 5)
INSERT INTO Empleado
    VALUES(2, 'LUNA ESPEJO', 1000, 10)
INSERT INTO Empleado
    VALUES(3, 'SOTO BUENO', 1400, NULL)
INSERT INTO Empleado
    VALUES(4, 'MARQUEZ ARIZAGA', 1500, NULL)
INSERT INTO Empleado
    VALUES(5, 'DAVILA SANCHEZ', 1200, 7.5)
GO
```

La siguiente consulta ejecuta una búsqueda basada en valores NULL.

```
SELECT idEmpleado, apellido, haberBasico,
       porcentajeComision
  FROM Empleado
```



```
WHERE porcentajeComision IS NOT NULL
GO
```

Genera una lista de empleados cuyo contenido en la columna **porcentajeComision** es distinto a NULL.

	Resultados	Mensajes		
	idEmpleado	apellido	haberBasico	porcentajeComision
1	1	CASTRO ARENAS	1200,00	5.0
2	2	LUNA ESPEJO	1000,00	10.0
3	5	DAVILA SANCHEZ	1200,00	7.5

Ejercicio 5.24: Cálculos con columnas que contienen valores NULL

Asumiendo que cada empleado obtuvo en este mes, ventas por 10,000 Nuevos Soles, genere un listado que muestre el monto total a abonarle a cada uno por su remuneración del mes. El monto total es la suma del haber básico más el monto a pagarle por comisión de las ventas.

```
SELECT idEmpleado, apellido, haberBasico,
       porcentajeComision,
       monto = haberBasico + 10000 *
                  porcentajeComision / 100
FROM Empleado
GO
```

	Resultados	Mensajes			
	idEmpleado	apellido	haberBasico	porcentajeComision	monto
1	1	CASTRO ARENAS	1200,00	5.0	1700.000000
2	2	LUNA ESPEJO	1000,00	10.0	2000.000000
3	3	SOTO BUENO	1400,00	NULL	NULL
4	4	MARQUEZ ARIZAGA	1500,00	NULL	NULL
5	5	DAVILA SANCHEZ	1200,00	7.5	1950.000000

Observe que el monto resulta NULL para aquellos empleados cuyo **porcentajeComision** es NULL.



Ejercicio 5.25: Cálculos con columnas que contienen valores NULL – Uso de ISNULL

Sintaxis

```
ISNULL(expresión_a_verificar, valor_de_reemplazo)
```

- **expresión_a_verificar**, es una expresión cuyo valor puede ser NULL.
- **valor_de_reemplazo**, es el valor con que se reemplaza expresión_a_verificar en el caso que ésta fuera NULL.

Calculemos el monto a abonarle a cada empleado haciendo uso de ISNULL.

```
SELECT idEmpleado, apellido, haberBasico,  
       porcentajeComision,  
       monto = haberBasico +  
              10000*ISNULL(porcentajeComision, 0)/100  
FROM Empleado  
GO
```

	idEmpleado	apellido	haberBasico	porcentajeComision	monto
1	1	CASTRO ARENAS	1200,00	5.0	1700.000000
2	2	LUNA ESPEJO	1000,00	10.0	2000.000000
3	3	SOTO BUENO	1400,00	NULL	1400.000000
4	4	MARQUEZ ARIZAGA	1500,00	NULL	1500.000000
5	5	DAVILA SANCHEZ	1200,00	7.5	1950.000000

5. FUNCIONES PARA MANIPULACIÓN DE FECHAS

El siguiente cuadro muestra una lista de las funciones para manipular fechas.

Función	Descripción
getdate()	Retorna la fecha y hora del sistema.
getutcdate()	Retorna la fecha y hora del Meridiano de Greenwich. El valor se obtiene a partir de la configuración regional del sistema.
dateadd(parte_fecha, n, fecha)	Genera una nueva fecha a partir de <i>fecha</i> , añadiéndole a <i>fecha</i> <i>n</i> unidades de <i>parte_fecha</i> .



datediff (<i>parte_fecha, fecha1, fecha2</i>)	Entrega la diferencia entre <i>fecha1</i> y <i>fecha2</i> en las unidades indicadas en <i>parte_fecha</i> .
datepart (<i>parte_fecha, fecha</i>)	Devuelve, a partir de <i>fecha</i> , un valor entero con la <i>parte_fecha</i> especificada.
datename (<i>parte_fecha, fecha</i>)	Devuelve, a partir de <i>fecha</i> , una cadena con la <i>parte_fecha</i> especificada.
day (<i>fecha</i>)	Devuelve un entero con la parte del día de <i>fecha</i> .
month (<i>fecha</i>)	Devuelve un entero con la parte del mes de <i>fecha</i> .
year (<i>fecha</i>)	Devuelve un entero con la parte del año de <i>fecha</i> .
datefromparts (<i>año, mes, día</i>)	Devuelve un dato de tipo date para el <i>año, mes</i> y <i>día</i> especificados.
datetimefromparts (<i>año, mes, día, hora, minutos, segundos, milisegundos</i>)	Devuelve un dato de tipo datetime para los valores especificados.
eomonth (<i>fecha</i>)	Devuelve un dato de tipo date que representa el último día del mes para la <i>fecha</i> especificada.
isdate (<i>expresión</i>)	Devuelve 1 si <i>expresión</i> es un valor de tipo date, time o datetime válido.

El siguiente cuadro muestra las **parte_fecha** que podemos especificar en las funciones de fecha listadas arriba.

parte_fecha	Abreviatura	Descripción
year	yy, yyyy	Año de la fecha
quarter	q, qq	Trimestre del año de la fecha
month	m, mm	Mes del año de la fecha
week	ww, wk	Semana del año de la fecha
day	d, dd	Día de la fecha
dayofyear	y, dy	Día del año de la fecha

weekday	dw	Día de la semana de la fecha
hour	hh	Hora de la fecha
minute	n, mi	Minutos de la hora de la fecha
second	s, ss	Segundos de la hora de la fecha
millisecond	ms	Milisegundos de la hora de la fecha

Ejercicio 5.26: Uso de las funciones de fecha y hora

Uso de GETDATE()

```
SELECT GETDATE()
GO
```

Entrega la fecha y hora del sistema.

Uso de GETUTCDATE()

```
SELECT GETUTCDATE()
GO
```

Entrega la fecha y hora del Meridiano de Greenwich.

Uso de DATEADD()

```
SELECT NumOrden, FechaOrden,
FechaPaGO = DATEADD(day, 30, FechaOrden)
FROM ORDEN_COMPRA
GO
```

Genera una nueva fecha añadiéndole 30 días a FechaOrden.

Uso de DATEDIFF()

```
SELECT NumOrden, FechaOrden,
SemanasTranscurridas =
DATEDIFF(week, FechaOrden, GETDATE())
FROM ORDEN_COMPRA
GO
```



Obtiene la diferencia en semanas entre FechaOrden y la fecha del sistema.

Uso de DATEPART() y DATENAME()

```
SELECT DATEPART(month, GETDATE())
SELECT DATENAME(month, GETDATE())
GO
```

Entrega el número y el nombre del mes de la fecha del sistema

Uso de DAY(), MONTH() y YEAR()

```
SELECT DAY(GETDATE())
SELECT MONTH(GETDATE())
SELECT YEAR(GETDATE())
GO
```

Entrega el día, el mes y el año de la fecha del sistema.

Uso de EOMONTH()

```
SELECT EOMONTH(GETDATE())
GO
```

Entrega la fecha del último día del mes para la fecha del sistema.

6. EJERCICIOS PROPUESTOS

Para los siguientes ejercicios debe utilizar las bases de datos RH y EDUCA. Los scripts para crearlas en su servidor los encontrará en el CD que acompaña al libro.

Escriba las instrucciones SELECT para obtener:

1. (RH) El nombre, apellido y teléfono de los empleados.
2. (EDUCA) El nombre, vacantes y precio de cada curso.
3. (EDUCA) El importe que se obtendría si se logra vender todas las vacantes por cada curso.
4. (EDUCA) El importe de lo recaudado hasta el momento de los cursos vendidos.
5. (EDUCA) El importe de lo que se tiene comprometido (cobrado y no cobrado) por los cursos vendidos hasta el momento.
6. (RH) El nombre y apellido de un empleado en una sola columna.
7. (RH) El nombre completo, sueldo, comisión y el total a abonar de cada empleado.
8. (RH) Los empleados del departamento de contabilidad.
9. (RH) Los empleados que se desempeñan como gerentes.
10. (RH) Los empleados de contabilidad cuyo sueldo es mayor a 5,000.00.
11. (EDUCA) Los cursos que aún no tienen profesor.
12. (EDUCA) Los cursos que aún no tienen alumnos matriculados.
13. (RH) Todos los empleados cuyos sus ingresos totales están entre 6,000.00 y 15,000.00, debe incluir los extremos.
14. (RH) Los empleados del departamento de VENTAS cuyo sueldo es menor a 5,000.00.
15. (RH) Los empleados cuyo nombre finaliza con la letra "O".
16. (RH) Los empleados cuyo apellido tiene en la segunda posición la letra "A" u "O".
17. (RH) Los empleados que tienen un sueldo mayor a 3,000 y menor de 10,000.
18. (RH) Los empleados de los departamentos de contabilidad y ventas.
19. (RH) Una lista de los empleados ordenada por fecha de ingreso.
20. (RH) Los empleados cuyos ingresos totales son menores a 8,000.



21. (RH) Los empleados que ingresaron cuyo mes de ingreso a la empresa fue ENERO.
22. (EDUCA) Las matrículas del último mes.
23. (RH) Los locales (ubicaciones) que se encuentran en LIMA.
24. (RH) El importe de la planilla del departamento de ventas, debe incluir el sueldo y la comisión.
25. (RH) Los departamentos que tienen por lo menos un trabajador.



Capítulo VI

Agrupamiento y Agregación de Datos

En este capítulo conoceremos cómo agrupar los datos utilizando la cláusula GROUP BY para obtener consolidados por los grupos y acumulados usando las funciones de agregación.



Uno de los requerimientos más comunes en el análisis de datos es la generación de reportes con datos consolidados: totales, subtotales, promedios, acumulados, etc. El uso de las funciones de agregación y la cláusula GROUP BY de la sentencia SELECT nos ayudan a cumplir con dicho requerimiento.

7. FUNCIONES DE AGREGACIÓN

Una función de agregación permite efectuar una operación aritmética que consolida los valores de una columna para toda la tabla o para los mismos valores agrupados según determinado criterio. La función devuelve un solo valor que es el consolidado para la tabla o para cada uno de los grupos.

El siguiente cuadro muestra las funciones de agregación más utilizadas.

Función	Descripción y Sintaxis
AVG()	Retorna el promedio de los valores de una columna o expresión. AVG ([DISTINCT] <i>expresión_numérica</i>)
COUNT()	Retorna la cuenta del número de valores distintos a NULL en una columna o expresión. Devuelve un valor de tipo int . COUNT ([DISTINCT] <i>expresión</i>) COUNT (*)
COUNT_BIG()	Funciona como COUNT(), con la diferencia de que devuelve un valor de tipo bigrat . COUNT_BIG ([DISTINCT] <i>expresión</i>) COUNT_BIG(*)
MAX()	Retorna el valor máximo de una columna o expresión. MAX(<i>expresión</i>)
MIN()	Retorna el valor mínimo de una columna o expresión. MIN(<i>expresión</i>)
STDEV()	Retorna la desviación estándar típica de los valores de una columna o expresión.



	STDEV([DISTINCT] <i>expresión_numérica</i>)
STDEVP()	Retorna la desviación estándar para la población de los valores de una columna o expresión. STDEVP([DISTINCT] <i>expresión_numérica</i>)
SUM()	Retorna la suma de los valores de una columna o expresión. SUM([DISTINCT] <i>expresión_numérica</i>)
VAR()	Retorna la varianza de los valores de una columna o expresión. VAR([DISTINCT] <i>expresión_numérica</i>)
VARP()	Retorna la varianza para la población de los valores de una columna o expresión. VARP([DISTINCT] <i>expresión_numérica</i>)

Para todas las funciones, DISTINCT indica que debe eliminarse los valores duplicados de **expresión** o **expresión_numérica** antes de evaluar la función.

Para consolidar los datos de una columna puede usar las funciones de agregación con la declaración SELECT; si desea consolidar los datos agrupándolos por determinado criterio añada la cláusula GROUP BY.

Con excepción de la función COUNT(*), todas las funciones de agregación retornan NULL si ninguna fila satisface la cláusula WHERE. La función COUNT(*) retorna un valor de cero si ninguna fila satisface la cláusula WHERE.

Ejercicio 6.1: Uso de la función COUNT()

1. Cuenta de los artículos registrados en la base de datos.

```
USE QhatuPERU
GO
```

```
SELECT COUNT(*) FROM ARTICULO
GO
```



Resultados	
(Sin nombre de columna)	
1	138

2. Cuenta de los artículos despachados a las diferentes tiendas de la empresa.

```
SELECT COUNT(DISTINCT CodArticulo)
FROM GUIA_DETALLE
GO
```

Resultados	
(Sin nombre de columna)	
1	66

Ejercicio 6.2: Uso de la funciones MAX() y MIN()

1. Precio más alto y más bajo de los artículos registrados en la tabla ARTICULO.

```
SELECT MAX(precioProveedor) AS 'Precio Alto',
       MIN(precioProveedor) AS 'Precio Bajo'
FROM ARTICULO
GO
```

Resultados		
	Precio Alto	Precio Bajo
1	24,40	0,40

2. Guía de envío más reciente y más antigua.

```
SELECT MAX(fechaSalida), MIN(fechaSalida)
FROM GUIA_ENVIO
GO
```



Resultados		Mensajes
(Sin nombre de columna)	(Sin nombre de columna)	
1	2013-04-19 20:38:06.400	2013-03-20 20:38:04.447

3. Nombre del primer artículo y del último artículo si se ordenaran en base a su descripción.

```
SELECT MIN(descripcionArticulo),  
       MAX(descripcionArticulo)  
FROM ARTICULO  
GO
```

Resultados		Mensajes
(Sin nombre de columna)	(Sin nombre de columna)	
1	7 UP DESCARTABLE	YOGURT YOLEIT FRESA

Ejercicio 6.3: Uso de la funciones de agregación para cálculos estadísticos

El siguiente ejemplo muestra el uso de las funciones estadísticas.

```
SELECT AVG(PrecioProveedor) AS 'Promedio',  
       STDEV(PrecioProveedor)  
             AS 'Desviación estándar',  
       STDEVP(PrecioProveedor)  
             AS 'Desviación estándar población',  
       VAR(PrecioProveedor) AS 'Varianza',  
       VARP(PrecioProveedor)  
             AS 'Varianza población'  
FROM ARTICULO  
GO
```

Resultados		Mensajes		
Promedio	Desviación estándar	Desviación estándar población	Varianza	Varianza población
1 5,7468	5,54546632749484	5,52533752667282	30,7521967893791	30,5293547836589



Ejercicio 6.4: Uso de la función SUM()

Monto total de los artículos enviados a las tiendas.

```
SELECT SUM(precioVenta * cantidadEnviada)
FROM GUIA_DETALLE
GO
```

Resultados	
(Sin nombre de columna)	
1	378095,00

Total, de unidades despachadas del producto **27**.

```
SELECT SUM(cantidadEnviada)
FROM GUIA_DETALLE
WHERE codArticulo = 27
GO
```

Resultados	
(Sin nombre de columna)	
1	300

8. LA CLÁUSULA GROUP BY

La cláusula GROUP BY se utiliza para agrupar los registros en base a determinado criterio, y Luego ejecutar cálculos sobre las columnas para consolidar los datos para cada uno de los grupos obtenidos. Por ejemplo, puede utilizar GROUP BY para agrupar todas las guías de envío por tienda, y Luego calcular el monto total enviado a cada una de ellas.

Sintaxis

```
SELECT lista_columnas,
       función_agregación(columna),
       función_agregación(columna), ...
FROM nombre_tabla
[ WHERE condición_filas ]
```



```
GROUP BY lista_columnas
      [ HAVING condición_grupos ]
```

- Las columnas presentes en **lista_columnas** de la cláusula GROUP BY deben necesariamente estar presentes en la **lista_columnas** de SELECT.
- Cualquier columna presente en SELECT, y que no se encuentre en la **lista_columnas** de GROUP BY debe estar afectada por una **función_agregación**.
- **condición_grupos** en la cláusula HAVING permite establecer una expresión lógica que indica que los grupos a mostrar en el resultado son aquellos para los que el valor de la expresión es verdadero.
- Una consulta GROUP BY solo entrega una fila por cada grupo generado. Esta fila muestra el resultado de la **función_agregación** aplicada sobre el grupo. No muestra el contenido del grupo.
- Cuando se utiliza GROUP BY sobre una columna que contiene valores NULL, éstos se procesan como un grupo.

Ejercicio 6.5: Uso de la cláusula GROUP BY

1. Cantidad de artículos registrados para cada línea.

```
SELECT codLinea,
       COUNT(codArticulo) AS Artículos
  FROM ARTICULO
 GROUP BY codLinea
 GO
```

código de línea	Artículos
1	25
2	25
3	41
4	17
5	15
6	15

2. Cantidad de artículos por proveedor para las líneas 1 y 5.

```
SELECT codLinea, codProveedor,
       COUNT(codArticulo) AS Artículos
  FROM ARTICULO
 WHERE codLinea IN (1, 5)
 GROUP BY codLinea, codProveedor
```



```
ORDER BY codLinea
GO
```

	Resultados	Mensajes	
	codLinea	codProveedor	Artículos
1	1	14	13
2	1	15	12
3	5	8	4
4	5	9	6
5	5	10	5

3. Monto total enviado por artículo.

```
SELECT codArticulo,
       SUM(precioVenta * cantidadEnviada)
     AS 'Monto total'
  FROM GUIA_DETALLE
 GROUP BY codArticulo
 ORDER BY 'Monto total' DESC
GO
```

	Resultados	Mensajes
	codArticulo	Monto total
1	130	27000,00
2	124	22500,00
3	125	22500,00
4	129	21150,00
5	64	21000,00
6	127	20250,00
7	126	16875,00
8	132	15750,00
9	66	15000,00
10	65	15000,00

4. Artículos cuyo monto total enviado es mayor a **20,000**.

```
SELECT codArticulo,
       SUM(precioVenta * cantidadEnviada)
     AS 'Monto total'
  FROM GUIA_DETALLE
 GROUP BY codArticulo
```



```
HAVING SUM(precioVenta * cantidadEnviada) > 20000
ORDER BY 'Monto total' DESC
GO
```

	codArticulo	Monto total
1	130	27000,00
2	124	22500,00
3	125	22500,00
4	129	21150,00
5	64	21000,00
6	127	20250,00

8.1. Uso de GROUP BY con los operadores ROLLUP y CUBE

El operador ROLLUP permite resumir los valores de grupo. El operador ROLLUP se usa normalmente para producir promedios acumulados o sumas acumuladas. Puede hacer esto aplicando la función de agregación en la lista de columnas de SELECT para cada columna en la cláusula GROUP BY moviéndose de izquierda a derecha.

El operador CUBE permite crear y resumir todas las posibles combinaciones de grupos basadas en la cláusula GROUP BY.

El siguiente ejercicio ilustra el uso de estos operadores.

Ejercicio 6.6: Uso de los operadores ROLLUP y CUBE

1. Ejecute la siguiente consulta:

```
SELECT codLinea, codProveedor,
       AVG(precioProveedor) AS 'Precio promedio'
  FROM ARTICULO
 GROUP BY codLinea, codProveedor
 ORDER BY codLinea, codProveedor
 GO
```



	codLinea	codProveedor	Precio promedio
1	1	14	3,1538
2	1	15	1,325
3	2	1	12,50
4	2	2	14,25
5	2	3	12,875
6	2	4	10,625
7	2	12	9,60
8	3	5	4,875
9	3	6	4,395
10	3	7	2,53
11	4	1	1,8333

La consulta muestra el precio promedio para cada pareja **línea-proveedor**. En la imagen se destaca las parejas **línea1-proveedor14** con su precio promedio **3.1538**, y **línea1-proveedor15** con su precio promedio **1.325**.

5. Ahora, ejecute la siguiente consulta usando el operador ROLLUP.

```
SELECT codLinea, codProveedor,
       AVG(precioProveedor) AS 'Precio promedio'
  FROM ARTICULO
 GROUP BY codLinea, codProveedor
    WITH ROLLUP
      GO
```

	codLinea	codProveedor	Precio promedio
1	1	14	3,1538
2	1	15	1,325
3	1	NULL	2,276
4	2	1	12,50
5	2	2	14,25
6	2	3	12,875
7	2	4	10,625
8	2	NULL	12,612
9	3	5	4,875



18	5	8	1,96
19	5	9	2,2066
20	5	10	16,768
21	5	NULL	6,9946
22	6	5	4,6866
23	6	6	4,576
24	6	11	13,00
25	6	NULL	6,8666
26	NULL	NULL	5,7468

La consulta muestra el precio promedio para cada pareja **línea-proveedor**. En la imagen se destaca las parejas **línea1-proveedor14** con su precio promedio **3.1538**, y **línea1-proveedor15** con su precio promedio **1.325**.

También muestra el precio promedio de cada línea tomando en cuenta a todos los proveedores. En la imagen se destaca la **línea1** cuyo precio promedio es **2.276**. En el caso de las consultas GROUP BY, el valor NULL que se muestra en la columna **codProveedor** debe interpretarse como "todos".

En la última fila del resultado de la consulta se muestra el precio promedio de todos los productos. Observe que en la columna **codLinea** el valor es NULL ("todas" las líneas), y en la columna **codProveedor** el valor también es NULL ("todos" los proveedores). El precio promedio de todos los productos es **5.7468**.

- Finalmente ejecute la siguiente consulta con el operador CUBE.

```
SELECT codLinea, codProveedor,
       AVG(precioProveedor) AS 'Precio promedio'
  FROM ARTICULO
 GROUP BY codLinea, codProveedor
 WITH CUBE
 GO
```



	codLinea	codProveedor	Precio promedio
1	NULL	NULL	5,7468
2	NULL	1	6,10
3	NULL	2	9,3312
4	NULL	3	12,875
5	NULL	4	10,625
6	NULL	5	4,8236
7	NULL	6	4,4312
8	NULL	7	2,53
9	NULL	8	1,96
10	NULL	9	2,2066

Si recorre el resultado de la consulta notará que la consulta con CUBE entrega los mismos resultados que la consulta con ROLLUP, pero además proporciona el precio promedio para cada proveedor. En la imagen se destaca al **proveedor2** cuyo precio promedio para "todas" las líneas es **6.10**.

Las siguientes son formas alternas de especificación de los operadores ROLLUP y CUBE.

```
SELECT codLinea, codProveedor,
       AVG(precioProveedor) AS 'Precio promedio'
  FROM ARTICULO
 GROUP BY ROLLUP(codLinea, codProveedor)
 GO
```

```
SELECT codLinea, codProveedor,
       AVG(precioProveedor) AS 'Precio promedio'
  FROM ARTICULO
 GROUP BY CUBE(codLinea, codProveedor)
 GO
```



9. LA CLÁUSULA OVER CON FUNCIONES DE AGREGACIÓN

La cláusula OVER usada con las funciones de agregación permite crear particiones de datos bajo determinado criterio, y calcular la función de agregación para cada partición con la ventaja de que permite mostrar los elementos que forman el consolidado de la partición.

Ejercicio 6.7: Uso de la cláusula OVER con la función SUM()

2. Ejecute la siguiente consulta:

```
SELECT CodLinea, DescripcionArticulo,  
StockActual AS Unidades  
FROM ARTICULO  
GO
```

	CodLinea	DescripcionArticulo	Unidades
1	1	CARAMELOS BASTON VIENA ARCOR	200
2	1	CARAMELOS SURTIDO DE FRUTAS	300
3	1	CARAMELOS FRUTAS SURTIDA ARCOR	250
4	1	CARAMELOS FRUTAS MASTICABLES	250
5	1	CHUPETES LOLY AMBROSOLI	150
6	1	FRUNA SURTIDA DONOFRIO	500
7	1	CHOCOLATE DOÑA PEPA FIELD	500
8	1	CHOCOLATE CUA CUA FIELD	500
9	1	MELLOWS FAMILIAR FIELD	100
10	1	WAFER CHOCOLATE FIELD	900

La consulta muestra el stock de cada artículo registrado en la tabla ARTICULO.

7. Ahora, ejecute la siguiente consulta de agregación:

```
SELECT CodLinea,  
SUM(StockActual) AS 'Total unidades'  
FROM ARTICULO  
GROUP BY CodLinea  
GO
```



	CodLinea	Total unidades
1	1	7016
2	2	1990
3	3	38259
4	4	6415
5	5	4770
6	6	42556

La consulta acumula el stock de los artículos que pertenecen a la misma línea.

8. Finalmente, ejecute la siguiente consulta usando la cláusula OVER:

```
SELECT CodLinea, DescripcionArticulo,
StockActual AS Unidades,
SUM(StockActual) OVER(PARTITION BY CodLinea)
AS 'Total unidades'
FROM ARTICULO
GO
```

	CodLinea	DescripcionArticulo	Unidades	Total unidades
1	1	CARAMELOS BASTON VIENA ARCOR	200	7016
2	1	CARAMELOS SURTIDO DE FRUTAS	300	7016
3	1	CARAMELOS FRUTAS SURTIDA ARCOR	250	7016
4	1	CARAMELOS FRUTAS MASTICABLES	250	7016
5	1	CHUPETES LOLY AMBROSOLI	150	7016
6	1	FRUNA SURTIDA DONOFRIO	500	7016
7	1	CHOCOLATE DOÑA PEPA FIELD	500	7016
8	1	CHOCOLATE CUA CUA FIELD	500	7016
9	1	MELLOWS FAMILIAR FIELD	100	7016
10	1	WAFER CHOCOLATE FIELD	900	7016

Esta última consulta muestra la data entregada por las 2 consultas anteriores: el acumulado del stock de los artículos que son de la misma línea, y el stock de cada artículo por separado.



10. EL OPERADOR PIVOT

Con mucha frecuencia necesitamos ver los datos en base a múltiples dimensiones (ó entidades). En las versiones antiguas de SQL Server disponemos de los operadores ROLLUP y CUBE que se utilizan con la cláusula GROUP BY, y permiten ver data resumida en base a distintas dimensiones.

Las últimas versiones de SQL Server incorporan el operador PIVOT que es más fácil de entender e implementar que los operadores ROLLUP y CUBE. Por ejemplo, con el operador PIVOT podemos rotar filas y mostrarlas como columnas para obtener una vista diferente de los datos. El resultado de PIVOT es una tabla de doble entrada.

Sintaxis

```
SELECT...
...
PIVOT( función_agregación( columna_numérica )
      FOR columna_dimensión
      IN ( lista_valores_columna_dimensión ) )
```

- **columna_numérica** es la columna cuyos valores se desea mostrar en una tabla de doble entrada.
- **columna_dimensión** es la columna cuyos valores en una consulta normal se ven como filas, y que en la consulta PIVOT se desea ver como columnas.
- **lista_valores_columna_dimensión** son los valores de **columna_dimensión** a mostrar como columnas en la tabla de doble entrada.

Ejercicio 6.8: Uso del operador PIVOT

Escriba y ejecute la siguiente consulta:

```
SELECT codLinea, codProveedor,
AVG(precioProveedor) AS 'Precio promedio'
FROM ARTICULO
GROUP BY codLinea, codProveedor
ORDER BY 1,2
GO
```



	codLinea	codProveedor	Precio promedio
1	1	14	3,1538
2	1	15	1,325
3	2	1	12,50
4	2	2	14,25
5	2	3	12,875
6	2	4	10,625
7	2	12	9,60
8	3	5	4,875
9	3	6	4,395
10	3	7	2,53

La consulta muestra el precio promedio por cada pareja línea-proveedor.

A continuación, crearemos un reporte a modo de tabla de doble entrada, pero solo con el precio promedio de las parejas para los proveedores 1, 14 y 15.

```
SELECT codLinea,
       [1] AS Proveedor1, [14] AS Proveedor14,
       [15] AS Proveedor15
  FROM
    (SELECT precioProveedor, codLinea, codProveedor
      FROM ARTICULO) origen
 PIVOT (AVG(precioProveedor)
    FOR codProveedor
   IN ([1], [14], [15])) AS destino
 GO
```

	codLinea	Proveedor1	Proveedor14	Proveedor15
1	1	NULL	3,1538	1,325
2	2	12,50	NULL	NULL
3	3	NULL	NULL	NULL
4	4	1,8333	NULL	NULL
5	5	NULL	NULL	NULL
6	6	NULL	NULL	NULL

Muestra el precio promedio de todas las líneas para los proveedores 1, 14 y 15.



Finalmente, "pivotaremos" el resultado anterior para que la consulta muestre a los proveedores 1, 14 y 15 como filas, y a las líneas como columnas, pero solo las líneas 1, 2 y 4.

```
SELECT codProveedor,
       [1] AS Linea1, [2] AS Linea2,
       [4] AS Linea4
  FROM
    (SELECT precioProveedor, codLinea, codProveedor
      FROM ARTICULO
     WHERE codProveedor IN (1, 14, 15)) origen
   PIVOT (AVG(precioProveedor)
        FOR codLinea
        IN ([1], [2], [4])) AS destino
    GO
```

	codProveedor	Linea1	Linea2	Linea4
1	1	NULL	12,50	1,8333
2	14	3,1538	NULL	NULL
3	15	1,325	NULL	NULL

11. EJERCICIOS PROPUESTOS

Para los siguientes ejercicios debe utilizar las bases de datos RH y EDUCA. Los scripts para crearlas en su servidor los encontrará en el CD que acompaña al libro.

Escriba las instrucciones SELECT para obtener:

9. (RH) Calcular el importe de la planilla del departamento de ventas. Debe incluir el sueldo y la comisión.
10. (RH) Encontrar el mayor y menor sueldo en el departamento de ventas.
11. (RH) Encontrar el salario promedio en la empresa.
12. (RH) Conocer la cantidad de empleados que hay en el departamento de ventas.
13. (RH) Conocer el importe de la planilla del departamento de ventas, con comisión y sin comisión.
14. (EDUCA) Conocer para el curso SQL Server Administración la cantidad de alumnos matriculados y a cuánto asciende el importe que se proyecta recaudar hasta el momento.
15. (EDUCA) Conocer cuál es el importe recaudado hasta el momento para el curso SQL Server Administración.
16. (RH) Encontrar el sueldo promedio por departamento.
17. (EDUCA) Encontrar el importe recaudado por curso.
18. (RH) Conocer el sueldo máximo, sueldo mínimo y el sueldo promedio por departamento.
19. (RH) Conocer cuántos empleados hay por departamento.
20. (RH) Conocer cuántos empleados han ingreso por año en cada departamento.
21. (RH) Conocer la cantidad de empleados, el importe de la planilla y el sueldo promedio.
22. (EDUCA) Conocer para cada curso la cantidad de alumnos matriculados y el importe que se tiene proyectado recaudar por los alumnos matriculados.
23. (RH) Encontrar los departamentos que tienen más de 3 trabajadores.
24. (RH) Conocer cuáles son los puestos de trabajo que tienen más de 2 empleados.
25. (EDUCA) Encontrar los ingresos por mes y los ingresos totales.



26. (RH) Encontrar el importe de la planilla por puesto de trabajo en cada departamento, el total por departamento y el total general.
27. (RH) Encontrar el importe de la planilla por cargo y departamento, encontrando resúmenes por todas las combinaciones posibles de estos datos.



Capítulo VII

Consultas a más de una tabla

En este capítulo conoceremos cómo diseñar consultas cuyo resultado implica leer datos de más de unas tablas: las combinaciones o joins y las subconsultas correlacionadas, así como las subconsultas simples.

Por lo general, la generación de reportes útiles y fáciles de entender para los usuarios requiere que la consulta lea los datos de varias tablas. En este capítulo veremos cómo diseñar instrucciones SELECT que permitan recuperar datos de múltiples tablas en un solo conjunto de resultados.

1. COMBINACIONES O JOINS

Una combinación, join o consulta correlacionada es la consulta que muestra columnas de dos tablas o conjuntos de filas y las entrega en un único conjunto de resultados. Típicamente, la combinación se lleva a cabo relacionando valores comunes en los dos conjuntos de resultados, tales como los valores de clave primaria y clave foránea.

Sintaxis

```
SELECT lista_columnas
FROM tabla1 | conjunto_de_filas1
tipo_join JOIN tabla2 | conjunto_de_filas2
ON condición_del_join
```

- **lista_columnas** es la lista de columnas a mostrar en el resultado de la consulta. Se recomienda que cada columna sea calificada con el alias de la tabla a la cual pertenece.
- **tipo_join** indica si el join es interior (INNER), exterior (OUTER) o irrestricto (CROSS).
- **condición_del_join** es una expresión que indica en base a qué columnas de cada una de las tablas se establece la relación entre ellas.

Una combinación (join) puede ser de cualquiera de los siguientes tipos:

- inner join
- outer join
 - left outer join
 - right outer join
 - full outer join
- cross join
- autojoin

2. INNER JOIN

Un **inner join** es la consulta correlacionada cuyo resultado muestra todas las filas que están relacionadas entre dos tablas o conjuntos de filas.



Ejercicio 7.1: Uso de INNER JOIN

Se desea obtener un listado de los artículos que comercializa **QhatuPERU**. La lista debe mostrar: el código del artículo, su descripción, su presentación, su precio unitario, y quién es el proveedor del artículo.

Ejecute las siguientes instrucciones:

```
USE QhatuPERU
GO
```

```
SELECT CodArticulo, DescripcionArticulo,
       Presentacion, PrecioProveedor, CodProveedor
  FROM ARTICULO
  GO
```

Resultados				
CodArticulo	DescripcionArticulo	Presentacion	PrecioProveedor	CodProveedor
1	CARAMELOS BASTON VIENA ARCOR	PAQUETE 454 GR	1,50	14
2	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	1,00	15
3	CARAMELOS FRUTAS SURTIDA ARCOR	PAQUETE 520 GR	1,50	14
4	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	1,30	14
5	CHUPETES LOLY AMBROSOLI	KILOGRAMO	1,20	15
6	FRUNA SURTIDA DONOFRIO	PAQUETE X 24 UNIDADES	1,80	15
7	CHOCOLATE DOÑA PEPA FIELD	PAQUETE X 6 UNIDADES	2,20	15
8	CHOCOLATE CUA CUA FIELD	PAQUETE X 6 UNIDADES	1,60	15
9	MELLOWS FAMILIAR FIELD	PAQUETE 454 GR	2,10	15
10	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES	0,70	15

El resultado de la consulta muestra al proveedor del artículo (**CodProveedor**), pero no es muy cómodo o útil para el usuario final ya que no muestra, por ejemplo, el nombre del proveedor. Este último dato, no es posible obtenerlo de la tabla ARTICULO. El dato se encuentra en la tabla PROVEEDOR. Felizmente hay una relación entre el artículo y su proveedor tal como se muestra en el diagrama siguiente:



Si para un artículo conocemos su **CodProveedor**, a través de la relación

ARTICULO.CodProveedor = PROVEEDOR.CodProveedor

podemos conocer no solo el nombre del proveedor, sino también sus demás datos.

Vamos a modificar la consulta anterior para que usando la relación existente combine los datos de las tablas ARTICULO y PROVEEDOR.

```
SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       ARTICULO.Presentacion,
       ARTICULO.PrecioProveedor,
       ARTICULO.CodProveedor,
       PROVEEDOR.NomProveedor,
       PROVEEDOR.Departamento
  FROM ARTICULO INNER JOIN PROVEEDOR
    ON ARTICULO.CodProveedor =
       PROVEEDOR.CodProveedor
 ORDER BY ARTICULO.CodArticulo
 GO
```



Resultados		Mensajes				
CodArticulo	DescripcionArticulo	Presentacion	PrecioProveedor	CodProveedor	NomProveedor	Departamento
1	CARAMELOS BASTON VIENA ARCOR	PAQUETE 454 GR	1,50	14	GOLOSINAS Y ANTOJOS	AREQUIPA
2	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	1,00	15	DISTRIBUIDORA DE GOLOSINAS FENIX	LIMA
3	CARAMELOS FRUTAS SURTIDA ARCOR	PAQUETE 520 GR	1,50	14	GOLOSINAS Y ANTOJOS	AREQUIPA
4	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	1,30	14	GOLOSINAS Y ANTOJOS	AREQUIPA
5	CHUPETES LOLY AMBROSOLI	KILOGRAMO	1,20	15	DISTRIBUIDORA DE GOLOSINAS FENIX	LIMA
6	FRUNA SURTIDA DONOFRIO	PAQUETE X 24 UNIDADES	1,80	15	DISTRIBUIDORA DE GOLOSINAS FENIX	LIMA
7	CHOCOLATE DOÑA PEPA FIELD	PAQUETE X 6 UNIDADES	2,20	15	DISTRIBUIDORA DE GOLOSINAS FENIX	LIMA
8	CHOCOLATE CUA CUA FIELD	PAQUETE X 6 UNIDADES	1,60	15	DISTRIBUIDORA DE GOLOSINAS FENIX	LIMA
9	MELLOWS FAMILIAR FIELD	PAQUETE 454 GR	2,10	15	DISTRIBUIDORA DE GOLOSINAS FENIX	LIMA
10	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES	0,70	15	DISTRIBUIDORA DE GOLOSINAS FENIX	LIMA

Como el resultado de la consulta debe mostrar la relación que hay entre cada artículo y su correspondiente proveedor, la consulta debe ser de tipo INNER JOIN. Observe que en la consulta, cada columna ha sido "calificada", es decir, se ha indicado de qué tabla debe ser leída. Si bien no es obligatorio calificar todas las columnas que participan en la consulta, se recomienda hacerlo ya que es una buena práctica que facilita el comprender la lógica de la misma.

2.1. Uso de alias para hacer referencia a las tablas

Un alias es un nombre alterno con el que se hace referencia a una tabla y que permite simplificar la escritura de las consultas. Por ejemplo, para la tabla ARTICULO podemos definir que su alias es ART, de modo que cuando en la consulta deseamos hacer referencia a la tabla usamos su alias ART en vez de su nombre.

Ejercicio 7.2: Uso de alias

En la consulta del ejercicio anterior

```
SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       ARTICULO.Presentacion,
       ARTICULO.PrecioProveedor,
       ARTICULO.CodProveedor,
       PROVEEDOR.NomProveedor,
       PROVEEDOR.Departamento
  FROM ARTICULO INNER JOIN PROVEEDOR
    ON ARTICULO.CodProveedor =
       PROVEEDOR.CodProveedor
 ORDER BY ARTICULO.CodArticulo
 GO
```

cada columna se ha calificado con el nombre de la tabla en la que se lee la columna.

Si bien calificar las columnas hace que la consulta sea más fácil de entender para el usuario, hacerlo no es obligatorio, salvo en los casos en los que no hacerlo puede conducir a ambigüedad en la lectura de la instrucción.

Por ejemplo, la ejecución de la consulta

```
SELECT CodArticulo, DescripcionArticulo,  
Presentacion, PrecioProveedor,  
CodProveedor, NomProveedor,  
Departamento  
FROM ARTICULO INNER JOIN PROVEEDOR  
ON ARTICULO.CodProveedor = PROVEEDOR.CodProveedor  
ORDER BY CodArticulo  
GO
```

genera el error 209

```
Mens. 209, Nivel 16, Estado 1, Línea 3  
El nombre de columna 'CodProveedor' es ambiguo.
```

debido a que la columna CodProveedor mencionada en la instrucción SELECT existe tanto en la tabla ARTICULO como en la tabla PROVEEDOR, por lo que el motor de datos no sabe de qué tabla debe leer la columna. No ocurre lo mismo con el resto de identificadores de columna porque éstos solo existen en una de las tablas.

Para evitar que el error se produzca, la consulta se debe escribir como sigue

```
SELECT CodArticulo, DescripcionArticulo,  
Presentacion, PrecioProveedor,  
ARTICULO.CodProveedor, NomProveedor,  
Departamento  
FROM ARTICULO INNER JOIN PROVEEDOR  
ON ARTICULO.CodProveedor = PROVEEDOR.CodProveedor  
ORDER BY CodArticulo  
GO
```

También es posible simplificar la digitación de los nombres de tabla definiendo un alias para cada tabla. El alias es una cadena corta que podemos definir para hacer referencia a la tabla con un nombre corto.

```

SELECT Art.CodArticulo, Art.DescripcionArticulo,
Art.Presentacion, Art.PrecioProveedor,
Art.CodProveedor, Prov.NomProveedor,
Prov.Departamento
FROM ARTICULO Art INNER JOIN PROVEEDOR Prov
ON Art.CodProveedor = Prov.CodProveedor
ORDER BY Art.CodArticulo
GO

```

Ya sea que utilice o no los alias para tablas, la recomendación al escribir una consulta JOIN es que califique todas las columnas ya que así es más fácil de entender para el usuario que no ha escrito la consulta.

Ejercicio 7.3: Cabecera y detalle de la Guía de Envío X

Escriba una consulta que muestre los datos de la cabecera de la guía de envío número 27, y además su detalle.

```

SELECT GUIA_ENVIO.NumGuia,
       GUIA_ENVIO.FechaSalida,
       GUIA_ENVIO.CodTienda,
       GUIA_DETALLE.CodArticulo,
       GUIA_DETALLE.PrecioVenta,
       GUIA_DETALLE.CantidadEnviada
  FROM GUIA_ENVIO INNER JOIN GUIA_DETALLE
  ON GUIA_ENVIO.NumGuia = GUIA_DETALLE.NumGuia
 WHERE GUIA_ENVIO.NumGuia = 27
GO

```

	NumGuia	FechaSalida	CodTienda	CodArticulo	PrecioVenta	CantidadEnviada
1	27	2013-03-25 20:38:04.813	2	124	3,00	500
2	27	2013-03-25 20:38:04.813	2	125	3,00	500
3	27	2013-03-25 20:38:04.813	2	126	2,25	500
4	27	2013-03-25 20:38:04.813	2	127	2,70	500
5	27	2013-03-25 20:38:04.813	2	129	5,64	250
6	27	2013-03-25 20:38:04.813	2	130	1,80	1000
7	27	2013-03-25 20:38:04.813	2	132	2,10	500
8	27	2013-03-25 20:38:04.813	2	133	17,65	10
9	27	2013-03-25 20:38:04.813	2	134	17,40	10
10	27	2013-03-25 20:38:04.813	2	135	17,40	10
11	27	2013-03-25 20:38:04.813	2	136	25,55	10



Ejercicio 7.4: Cabecera y monto total de la Guía de Envío X

Modifique la consulta del ejercicio anterior para que muestre los datos de la cabecera de la guía de envío número 27 y además su monto total.

```
SELECT GUIA_ENVIO.NumGuia,
       GUIA_ENVIO.FechaSalida,
       GUIA_ENVIO.CodTienda,
       SUM(GUIA_DETALLE.CantidadEnviada *
            GUIA_DETALLE.PrecioVenta) AS MontoTotal
  FROM GUIA_ENVIO INNER JOIN GUIA_DETALLE
    ON GUIA_ENVIO.NumGuia = GUIA_DETALLE.NumGuia
 GROUP BY GUIA_ENVIO.NumGuia,
          GUIA_ENVIO.FechaSalida,
          GUIA_ENVIO.CodTienda
 HAVING GUIA_ENVIO.NumGuia = 27
    GO
```

NumGuia	FechaSalida	CodTienda	MontoTotal	
1	27	2013-03-25 20:38:04.813	2	10515.00

Ejercicio 7.5: JOIN de 3 tablas

Se desea generar el catálogo de artículos de QhatuPERU que debe mostrar:

- línea del artículo (tabla LINEA, columna NomLinea)
- código del artículo (tabla ARTICULO, columna CodArticulo)
- descripción del artículo (tabla ARTICULO, columna DescripcionArticulo)
- presentación del artículo (tabla ARTICULO, columna Presentacion)
- precio unitario del artículo (tabla ARTICULO, columna PrecioProveedor)
- nombre del proveedor (tabla PROVEEDOR, columna NomProveedor)

Vamos a construir la consulta paso a paso, de modo que aprovechemos la característica "autocompletear" del editor de consultas de SQL Server Management Studio.

En el editor digite SELECT y Luego la cláusula FROM que define el primer join entre las tablas LINEA y ARTICULO:



```
USE QhatuPERU
go

SELECT
    FROM LINEA INNER JOIN ARTICULO
        ON LINEA.CodLinea = ARTICULO.
```

CodArticulo
CodLinea
CodProveedor
Descontinuado
DescripcionArticulo
PrecioProveedor
Presentacion
StockActual
StockMinimo

columna CodLinea(int, not null)

A continuación, especifique en el SELECT las columnas que provienen de las tablas LINEA y ARTICULO en el orden que se desea mostrar en el resultado.

```
USE QhatuPERU
go

SELECT LINEA.NomLinea, ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       ARTICULO.Presentacion, ARTICULO.PrecioProveedor
  FROM LINEA INNER JOIN ARTICULO
    ON LINEA.CodLinea = ARTICULO.CodLinea
```

Si desea puede ejecutar la consulta a modo de comprobación.

Resultados		Mensajes		
NomLinea	CodArticulo	DescripcionArticulo	Presentacion	PrecioProveedor
1 GOLOSINAS	1	CARAMELOS BASTON VIENA ARCOR	PAQUETE 454 GR	1,50
2 GOLOSINAS	2	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	1,00
3 GOLOSINAS	3	CARAMELOS FRUTAS SURTIDA ARCOR	PAQUETE 520 GR	1,50
4 GOLOSINAS	4	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	1,30
5 GOLOSINAS	5	CHUPETES LOLY AMBROSOLI	KILOGRAMO	1,20
6 GOLOSINAS	6	FRUNA SURTIDA DONOFRIO	PAQUETE X 24 UNIDADES	1,80
7 GOLOSINAS	7	CHOCOLATE DOÑA PEPA FIELD	PAQUETE X 6 UNIDADES	2,20
8 GOLOSINAS	8	CHOCOLATE CUA CUA FIELD	PAQUETE X 6 UNIDADES	1,60
9 GOLOSINAS	9	MELLOWS FAMILIAR FIELD	PAQUETE 454 GR	2,10
10 GOLOSINAS	10	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES	0,70



Ahora añada el segundo JOIN, que combina el resultado anterior con la tabla PROVEEDOR.

```
USE QhatuPERU
go

SELECT LINEA.NomLinea, ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       ARTICULO.Presentacion, ARTICULO.PrecioProveedor
  FROM LINEA INNER JOIN ARTICULO
    ON LINEA.CodLinea = ARTICULO.CodLinea
   INNER JOIN PROVEEDOR
    ON ARTICULO.CodProveedor = PROVEEDOR.
```

columna CodProveedor(PK, int, not null)

<input type="checkbox"/>	Ciudad
<input type="checkbox"/>	CodigoPostal
<input checked="" type="checkbox"/>	CodProveedor
<input type="checkbox"/>	Departamento
<input type="checkbox"/>	Direccion
<input type="checkbox"/>	Fax
<input type="checkbox"/>	NomProveedor
<input type="checkbox"/>	Representante
<input type="checkbox"/>	Telefono

Añada al SELECT las columnas a leer de la tabla PROVEEDOR.

```
USE QhatuPERU
go

SELECT LINEA.NomLinea, ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       ARTICULO.Presentacion, ARTICULO.PrecioProveedor,
       PROVEEDOR.NomProveedor
  FROM LINEA INNER JOIN ARTICULO
    ON LINEA.CodLinea = ARTICULO.CodLinea
   INNER JOIN PROVEEDOR
    ON ARTICULO.CodProveedor = PROVEEDOR.CodProveedor
```



Ejecute la consulta.

Resultados Mensajes						
	NomLinea	CodArticulo	DescripcionArticulo	Presentacion	PrecioProveedor	NomProveedor
1	GOLOSINAS	1	CARAMELOS BASTON VIENA ARCOR	PAQUETE 454 GR	1,50	GOLOSINAS Y ANTOJOS
2	GOLOSINAS	2	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	1,00	DISTRIBUIDORA DE G...
3	GOLOSINAS	3	CARAMELOS FRUTAS SURTIDA AR...	PAQUETE 520 GR	1,50	GOLOSINAS Y ANTOJOS
4	GOLOSINAS	4	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	1,30	GOLOSINAS Y ANTOJOS
5	GOLOSINAS	5	CHUPETES LOLY AMBROSOLI	KILOGRAMO	1,20	DISTRIBUIDORA DE G...
6	GOLOSINAS	6	FRUNA SURTIDA DONOFRIO	PAQUETE X 24 ...	1,80	DISTRIBUIDORA DE G...
7	GOLOSINAS	7	CHOCOLATE DOÑA PEPA FIELD	PAQUETE X 6 U...	2,20	DISTRIBUIDORA DE G...
8	GOLOSINAS	8	CHOCOLATE CUA CUA FIELD	PAQUETE X 6 U...	1,60	DISTRIBUIDORA DE G...
9	GOLOSINAS	9	MELLOWS FAMILIAR FIELD	PAQUETE 454 GR	2,10	DISTRIBUIDORA DE G...
10	GOLOSINAS	10	WAFER CHOCOLATE FIELD	PAQUETE X 9 U...	0,70	DISTRIBUIDORA DE G...

Ejercicio 7.6: Monto total enviado a cada tienda

Escriba una consulta que muestre el monto total enviado a cada tienda.

```
SELECT TIENDA.CodTienda, TIENDA.Distrito,
SUM(GUIA_DETALLE.CantidadEnviada *
    GUIA_DETALLE.PrecioVenta)
AS MontoTotal
FROM TIENDA INNER JOIN GUIA_ENVIO
ON TIENDA.CodTienda = GUIA_ENVIO.CodTienda
INNER JOIN GUIA_DETALLE
ON GUIA_ENVIO.NumGuia = GUIA_DETALLE.NumGuia
GROUP BY TIENDA.CodTienda, TIENDA.Distrito
GO
```

Resultados Mensajes			
	CodTienda	Distrito	MontoTotal
1	1	MIRAFLORES	76363,75
2	2	PUEBLO LIBRE	76363,75
3	3	CALLAO	75122,50
4	4	LOS OLIVOS	75122,50
5	5	BREÑA	75122,50

Ejercicio 7.7: Total de unidades despachadas por mes del artículo X

Escriba una consulta que muestre el total de unidades mensuales despachadas del artículo 37.

```
SELECT YEAR(GUIA_ENVIO.FechaSalida) AS Año,
```



```
MONTH(GUIA_ENVIO.FechaSalida) AS Mes,
SUM(GUIA_DETALLE.CantidadEnviada) AS
    UnidadesEnviadas
FROM GUIA_ENVIO INNER JOIN GUIA_DETALLE
ON GUIA_ENVIO.NumGuia = GUIA_DETALLE.NumGuia
WHERE GUIA_DETALLE.CodArticulo = 37
GROUP BY YEAR(GUIA_ENVIO.FechaSalida),
MONTH(GUIA_ENVIO.FechaSalida)
GO
```

	Año	Mes	UnidadesEnviadas
1	2013	3	200
2	2013	4	300

3. OUTER JOIN

En ocasiones es necesario obtener un reporte que muestre qué filas de una tabla no tienen ninguna relación con otra tabla. En estas situaciones, podemos emplear una consulta outer join.

Un **outer join** es la consulta correlacionada que entrega todas las filas que están relacionadas, y además:

- las filas no relacionadas de la tabla izquierda (LEFT OUTER JOIN), ó
- las filas no relacionadas de la tabla derecha (RIGHT OUTER JOIN), ó
- las filas no relacionadas de ambas tablas (FULL OUTER JOIN)

Se considera como la tabla izquierda, a aquella que se menciona primero en la cláusula FROM.

Ejercicio 7.8: Uso de OUTER JOIN

En la base de datos **QhatuPERU** la tabla ARTICULO es una lista de todos los artículos comercializados por la empresa. Cuando un artículo se despacha a una tienda, el movimiento se registra en el detalle de una guía de envío. El siguiente diagrama muestra la relación entre las tablas ARTICULO y GUIA_DETALLE.



Se desea conocer si existen artículos que aún no han sido despachados ni una sola vez a las tiendas; es decir, artículos que están registrados en la tabla ARTICULO, pero que no tienen registro en la tabla GUIA_DETALLE.

La siguiente consulta muestra que artículos están registrados en la tabla GUIA_DETALLE; es decir, artículos que han sido enviados a las tiendas.

```
SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       GUIA_DETALLE.CantidadEnviada
  FROM ARTICULO INNER JOIN GUIA_DETALLE
    ON ARTICULO.CodArticulo =
       GUIA_DETALLE.CodArticulo
 ORDER BY ARTICULO.CodArticulo
 GO
```

Resultados		
CodArticulo	DescripcionArticulo	CantidadEnviada
1	CARAMELOS BASTON VIENA ARCOR	20
2	CARAMELOS BASTON VIENA ARCOR	20
3	CARAMELOS BASTON VIENA ARCOR	20
4	CARAMELOS BASTON VIENA ARCOR	20
5	CARAMELOS BASTON VIENA ARCOR	20
6	CARAMELOS BASTON VIENA ARCOR	20
7	CARAMELOS BASTON VIENA ARCOR	20
8	CARAMELOS BASTON VIENA ARCOR	20
9	CARAMELOS BASTON VIENA ARCOR	20
10	CARAMELOS BASTON VIENA ARCOR	20



El resultado muestra que en el caso del artículo 1, éste se ha enviado en más de una ocasión a las tiendas. Observe que en este resultado no se muestra el artículo 5.

Como el requerimiento es obtener una lista de los artículos que no se ha despachado a las tiendas, vamos a convertir la consulta anterior en una consulta OUTER JOIN. La consulta debe ser LEFT OUTER JOIN porque la tabla izquierda es la tabla ARTICULO, y lo que buscamos son registros de ARTICULO que no tengan relación con los registros de GUIA_DETALLE.

```
SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       GUIA_DETALLE.CantidadEnviada
  FROM ARTICULO LEFT OUTER JOIN GUIA_DETALLE
    ON ARTICULO.CodArticulo =
       GUIA_DETALLE.CodArticulo
 ORDER BY ARTICULO.CodArticulo
 GO
```

	CodArticulo	DescripcionArticulo	CantidadEnviada
75	4	CARAMELOS FRUTAS MASTICABLES	20
76	4	CARAMELOS FRUTAS MASTICABLES	20
77	4	CARAMELOS FRUTAS MASTICABLES	20
78	4	CARAMELOS FRUTAS MASTICABLES	20
79	4	CARAMELOS FRUTAS MASTICABLES	20
80	4	CARAMELOS FRUTAS MASTICABLES	20
81	5	CHUPETES LOLY AMBROSOLI	NULL
82	6	FRUNA SURTIDA DONOFRIO	30
83	6	FRUNA SURTIDA DONOFRIO	30
84	6	FRUNA SURTIDA DONOFRIO	30

Por definición, una consulta OUTER JOIN muestra las filas relacionadas y las filas no relacionadas. Observe que para el artículo 5 el valor en la columna **CantidadEnviada** es NULL (dato desconocido o dato no disponible).

Filtraremos el resultado de la consulta para que solo muestre los artículos cuyo valor en **CantidadEnviada** es NULL.

```
SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
```



```
GUIA_DETALLE.CantidadEnviada
FROM ARTICULO LEFT OUTER JOIN GUIA_DETALLE
ON ARTICULO.CodArticulo =
    GUIA_DETALLE.CodArticulo
WHERE GUIA_DETALLE.CantidadEnviada IS NULL
ORDER BY ARTICULO.CodArticulo
GO
```

	CodArticulo	DescripcionArticulo	CantidadEnviada
1	5	CHUPETES LOLY AMBROSOLI	NULL
2	13	CHOCOLATE BARRA MILKY WAY	NULL
3	14	SNICKERS BAR KING SIZE	NULL
4	15	CHOCOLATE BARRA MILK DOVE	NULL
5	16	CHOCOLATE BARRA DARK DOVE	NULL
6	17	MILKY WAY BAR KING SIZE	NULL
7	18	GALLETAS CHIPS AHOY	NULL
8	19	GALLETAS TUAREG COSTA	NULL
9	20	GALLETAS VAINILLA COSTA	NULL
10	21	GALLETAS SURTIDAS BUTTER COOKIES	NULL

Hay 72 artículos que NO tienen registrada salida del almacén, que sumados a los 66 artículos que tienen salida registrada, nos da un total de 138 artículos.

Eliminamos de la consulta la columna **CantidadEnviada**. La consulta queda como sigue:

```
SELECT ARTICULO.CodArticulo,
ARTICULO.DescripcionArticulo
FROM ARTICULO LEFT OUTER JOIN GUIA_DETALLE
ON ARTICULO.CodArticulo =
    GUIA_DETALLE.CodArticulo
WHERE GUIA_DETALLE.CantidadEnviada IS NULL
ORDER BY ARTICULO.CodArticulo
GO
```

Ejercicio 7.9: Reporte de unidades enviadas de cada artículo

Diseñe una consulta que muestre cuántas unidades se han enviado en total a las tiendas para cada uno de los artículos. El resultado debe mostrar dicho valor para todos los artículos.



```
SELECT ARTICULO.CodArticulo,
ARTICULO.DescripcionArticulo,
SUM(GUIA_DETALLE.CantidadEnviada)
AS TotalEnviado
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
ARTICULO.CodArticulo
GROUP BY ARTICULO.CodArticulo,
ARTICULO.DescripcionArticulo
ORDER BY ARTICULO.CodArticulo
GO
```

	CodArticulo	DescripcionArticulo	TotalEnviado
1	1	CARAMELOS BASTON VIENA ARCOR	400
2	2	CARAMELOS SURTIDO DE FRUTAS	400
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	400
4	4	CARAMELOS FRUTAS MASTICABLES	400
5	6	FRUNA SURTIDA DONOFRIO	600
6	7	CHOCOLATE DOÑA PEPA FIELD	500
7	8	CHOCOLATE CUA CUA FIELD	500
8	9	MELLOWS FAMILIAR FIELD	400
9	10	WAFER CHOCOLATE FIELD	400
10	11	CHOCOLATE BARRA REGULAR	1000

Note que el resultado no muestra todos los artículos. Para que se muestren todos los artículos convierta la consulta INNER JOIN en una consulta OUTER JOIN, y para los artículos en los que **CantidadEnviada** es NULL que se muestre 0 (cero) como el valor en **TotalEnviado**.

```
SELECT ARTICULO.CodArticulo,
ARTICULO.DescripcionArticulo,
ISNULL(SUM(GUIA_DETALLE.CantidadEnviada), 0)
AS TotalEnviado
FROM GUIA_DETALLE RIGHT OUTER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
ARTICULO.CodArticulo
GROUP BY ARTICULO.CodArticulo,
ARTICULO.DescripcionArticulo
ORDER BY ARTICULO.CodArticulo
GO
```



	CodArticulo	DescripcionArticulo	TotalEnviado
1	1	CARAMELOS BASTON VIENA ARCOR	400
2	2	CARAMELOS SURTIDO DE FRUTAS	400
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	400
4	4	CARAMELOS FRUTAS MASTICABLES	400
5	5	CHUPETES LOLY AMBROSOLI	0
6	6	FRUNA SURTIDA DONOFRIO	600
7	7	CHOCOLATE DOÑA PEPA FIELD	500
8	8	CHOCOLATE CUA CUA FIELD	500
9	9	MELLOWS FAMILIAR FIELD	400
10	10	WAFER CHOCOLATE FIELD	400

La consulta debe ser RIGHT OUTER JOIN porque la tabla ARTICULO se encuentra a la derecha y se desea mostrar todos los artículos: los que tienen relación con la tabla GUIA_DETALLE, y los que no la tienen.

4. CROSS JOIN

Un **cross join** es la consulta correlacionada que combina cada una de las filas de una de las tablas con todas las filas de la otra tabla.

No es necesario que exista una columna común para ejecutar cross join. Esta consulta también se conoce como el **producto cartesiano** de dos tablas.

Ejercicio 7.10: Consulta CROSS JOIN

```
SELECT LINEA.Descripcion,
       ARTICULO.DescripcionArticulo
  FROM LINEA CROSS JOIN ARTICULO
 ORDER BY LINEA.Descripcion
        GO
```



	Descripcion	DescripcionArticulo
1	DETERGENTES,DESINFECTANTES,ACCESORIOS	7 UP DESCARTABLE
2	DETERGENTES,DESINFECTANTES,ACCESORIOS	ACEITE BABY JOHNSONS
3	DETERGENTES,DESINFECTANTES,ACCESORIOS	ACEITE BABY JOHNSONS C/ALOE Y VIT. E
4	DETERGENTES,DESINFECTANTES,ACCESORIOS	ACEITE BABY JOHNSONS C/ALOE Y VIT. E
5	DETERGENTES,DESINFECTANTES,ACCESORIOS	ACEITE JOHNSONS
6	DETERGENTES,DESINFECTANTES,ACCESORIOS	ACEITE JOHNSONS CREMOSO
7	DETERGENTES,DESINFECTANTES,ACCESORIOS	ACEITE P/BEBES CHICCO
8	DETERGENTES,DESINFECTANTES,ACCESORIOS	ACEITE P/BEBES DR. ZAIMAN
9	DETERGENTES,DESINFECTANTES,ACCESORIOS	ACEITE P/BEBES NINET
10	DETERGENTES,DESINFECTANTES,ACCESORIOS	CARAMELOS BASTON VIENA ARCOR

La tabla LINEA tiene 6 líneas:

```
SELECT COUNT (*) FROM LINEA
GO
```

La tabla ARTICULO tiene 138 artículos:

```
SELECT COUNT (*) FROM ARTICULO
GO
```

El resultado de la consulta CROSS JOIN entrega $6 \times 138 = 828$ filas.

5. AUTOJOIN

Es una consulta correlacionada en la que una tabla se combina consiGO misma para obtener un nuevo conjunto de resultados.

Ejercicio 7.11: Consulta AUTOJOIN

Para ilustrar este caso vamos a crear una tabla que tenga una autorrelación; es decir, que tenga una clave foránea que apunte a la clave primaria de la misma tabla.

Ejecute las siguientes instrucciones para crear la tabla y cargar sus datos:

```
USE MarketPERU
GO

-- Consulta autojoin,
```



```
-- creación de la tabla con autorelación
CREATE TABLE Trabajador(
    idTrabajador int PRIMARY KEY,
    Apellidos varchar(30) not null,
    Jefe int null )
GO

ALTER TABLE Trabajador
ADD CONSTRAINT fk_Trabajador_Trabajador
FOREIGN KEY(Jefe)
REFERENCES Trabajador
GO

INSERT INTO Trabajador
VALUES(102, 'Ardiles Soto', NULL)
INSERT INTO Trabajador
VALUES(101, 'Camacho Saravia', 102)
INSERT INTO Trabajador
VALUES(105, 'Vilchez Santos', 102)
INSERT INTO Trabajador
VALUES(103, 'Sánchez Aliaga', 101)
INSERT INTO Trabajador
VALUES(104, 'Castro Avila', 101)
INSERT INTO Trabajador
VALUES(107, 'Urrunaga Tapia', 101)
INSERT INTO Trabajador
VALUES(106, 'Juárez Pinto', 105)
GO

SELECT * FROM Trabajador
GO
```

	Resultados	Mensajes	
	idTrabajador	Apellidos	Jefe
1	101	Camacho Saravia	102
2	102	Ardiles Soto	NULL
3	103	Sánchez Aliaga	101
4	104	Castro Avila	101
5	105	Vilchez Santos	102
6	106	Juárez Pinto	105
7	107	Urrunaga Tapia	101



La columna **Jefe** de la tabla **Trabajador** registra el código del jefe de un trabajador. Por ejemplo, el trabajador **101 (Camacho Saravia)** es el jefe de los trabajadores **103, 104** y **107**.

Se desea crear una consulta que muestre una lista de trabajadores. La lista debe mostrar los **apellidos del jefe** de cada trabajador.

```
SELECT T1.idTrabajador, T1.apellidos,
       T2.apellidos AS Jefe
  FROM Trabajador T1 INNER JOIN Trabajador T2
    ON T1.jefe = T2.idTrabajador
GO
```

	Resultados	Mensajes	
	idTrabajador	apellidos	Jefe
1	101	Camacho Saravia	Ardiles Soto
2	103	Sánchez Aliaga	Camacho Saravia
3	104	Castro Avila	Camacho Saravia
4	105	Vilchez Santos	Ardiles Soto
5	106	Juárez Pinto	Vilchez Santos
6	107	Umunaga Tapia	Camacho Saravia

Note que el resultado muestra a todos los trabajadores con su respectivo jefe, pero el trabajador **102 (Ardiles Soto)** no aparece en la lista porque él no tiene **jefe**.

Modifique la consulta para que también se muestre al trabajador **102**.

```
SELECT T1.idTrabajador, T1.apellidos,
       T2.apellidos AS Jefe
  FROM Trabajador T1 LEFT OUTER JOIN Trabajador T2
    ON T1.jefe = T2.idTrabajador
GO
```



Resultados		Mensajes	
	idTrabajador	apellidos	Jefe
1	101	Camacho Saravia	Ardiles Soto
2	102	Ardiles Soto	NULL
3	103	Sánchez Aliaga	Camacho Saravia
4	104	Castro Avila	Camacho Saravia
5	105	Vilchez Santos	Ardiles Soto
6	106	Juárez Pinto	Vilchez Santos
7	107	Umunaga Tapia	Camacho Saravia

6. SUBCONSULTAS

Una subconsulta es una declaración SELECT anidada dentro una sentencia SELECT, INSERT, UPDATE o DELETE o dentro de otra subconsulta.

Si la respuesta a un requerimiento de datos requiere la ejecución de una serie de pasos lógicos, utilice subconsultas para tratar de resolver el requerimiento con una sola sentencia.

Las subconsultas son de los tipos siguientes:

- Subconsulta que entrega un solo valor (1 fila, 1 columna)
- Subconsulta que entrega un conjunto de valores (varias filas, 1 columna)

Una subconsulta se especifica entre paréntesis, y se puede especificar en cualquier donde la Sintaxis permite una expresión.

6.1. Subconsulta que entrega un solo valor (1 fila, 1 columna)

Cuando la subconsulta se especifica:

- en la lista de columnas del SELECT externo, ó
- en la cláusula WHERE del SELECT externo usando un operador relacional (test de comparación),

la subconsulta debe ser una que entregue un solo valor.



Ejercicio 7.12: Subconsulta definida en la lista de columnas del SELECT externo

Genere una consulta que entregue la lista de precios de todos los artículos, especificando en una columna adicional la diferencia entre el precio de cada artículo y el precio promedio de todos los artículos.

Primero, especifique la consulta que entrega el precio promedio de todos los artículos.

```
SELECT AVG(PrecioProveedor) FROM ARTICULO  
GO
```

Resultados	
(Sin nombre de columna)	
1	5,7468

Ahora, escriba la consulta que entrega la lista de precios solicitada teniendo en cuenta la fórmula que determina la diferencia entre el precio de cada artículo y el precio promedio de todos los artículos.

```
SELECT CodArticulo, DescripcionArticulo,  
       PrecioProveedor,  
       Diferencia = PrecioProveedor -  
           (SELECT AVG(PrecioProveedor) FROM ARTICULO)  
FROM ARTICULO  
GO
```

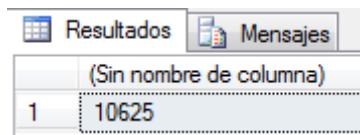
	CodArticulo	DescripcionArticulo	PrecioProveedor	Diferencia
1	1	CARAMELOS BASTON VIENA ARCOR	1,50	-4,2468
2	2	CARAMELOS SURTIDO DE FRUTAS	1,00	-4,7468
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	1,50	-4,2468
4	4	CARAMELOS FRUTAS MASTICABLES	1,30	-4,4468
5	5	CHUPETES LOLY AMBROSOLI	1,20	-4,5468
6	6	FRUNA SURTIDA DONOFRIO	1,80	-3,9468
7	7	CHOCOLATE DOÑA PEPA FIELD	2,20	-3,5468
8	8	CHOCOLATE CUA CUA FIELD	1,60	-4,1468
9	9	MELLOWS FAMILIAR FIELD	2,10	-3,6468
10	10	WAFER CHOCOLATE FIELD	0,70	-5,0468

Ejercicio 7.13: Porcentaje despachado de cada artículo respecto al total despachado para la línea X

Escriba una consulta que determine el porcentaje de unidades despachadas de cada artículo de la línea 4 respecto al total despachado de la línea.

Primero, escriba la consulta que calcula el total despachado para la línea 4.

```
SELECT SUM(CantidadEnviada)
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
ARTICULO.CodArticulo
WHERE ARTICULO.CodLinea = 4
GO
```



(Sin nombre de columna)
10625

Ahora, escriba la consulta que, utilizando la consulta anterior, presente el listado requerido.

```
SELECT ARTICULO.CodArticulo,
ARTICULO.DescripcionArticulo,
Despachado =
ISNULL(SUM(GUIA_DETALLE.CantidadEnviada), 0),
Porcentaje =
CONVERT(float,
ISNULL(SUM(GUIA_DETALLE.CantidadEnviada), 0))
/
(SELECT SUM(CantidadEnviada)
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
ARTICULO.CodArticulo
WHERE ARTICULO.CodLinea = 4) * 100
FROM ARTICULO LEFT OUTER JOIN GUIA_DETALLE
ON ARTICULO.CodArticulo =
GUIA_DETALLE.CodArticulo
WHERE ARTICULO.CodLinea = 4
GROUP BY ARTICULO.CodArticulo,
ARTICULO.DescripcionArticulo
GO
```



	CodArticulo	DescripcionArticulo	Despachado	Porcentaje
1	92	CREMA DE LECHE LAIVE	425	4
2	93	CREMA DE LECHE DUPRE	425	4
3	94	CREMA DE LECHE NESTLE	0	0
4	95	YOGURT GLORIA FRESA	850	8
5	96	YOGURT YOLEIT FRESA	0	0
6	97	YOGURT LAIVE FRESA	850	8
7	98	CREAM CHESSE LAIVE	0	0
8	99	CREMA DE QUESO LAIVE	0	0
9	100	QUESO CREMA MILKITO	425	4
10	101	MANTEQUILLA LAIVE C/SAL	1700	16

Ejercicio 7.14: Subconsulta definida en el WHERE del SELECT externo

Escriba una consulta que entregue una lista de los artículos que se despacharon en la fecha que se despachó la última salida del almacén. Tenga en cuenta que en dicha fecha se puede haber registrado más de una salida.

Primero, obtenga la fecha de la última salida

```
SELECT MAX(FechaSalida) FROM GUIA_ENVIO
GO
```

	Resultados	Mensajes
(Sin nombre de columna)		
1	2013-04-19 20:38:06.400	

Ahora, utilizando adecuadamente la consulta anterior, escriba la consulta que responde al requerimiento especificado.

```
SELECT DISTINCT GUIA_DETALLE.CodArticulo,
    ARTICULO.DescripcionArticulo
FROM GUIA_DETALLE INNER JOIN ARTICULO
    ON GUIA_DETALLE.CodArticulo =
        ARTICULO.CodArticulo
INNER JOIN GUIA_ENVIO
    ON GUIA_DETALLE.NumGuia = GUIA_ENVIO.NumGuia
WHERE CONVERT(CHAR(10),
    GUIA_ENVIO.FechaSalida, 103) =
    (SELECT CONVERT(CHAR(10), MAX(FechaSalida),
    103) FROM GUIA_ENVIO)
GO
```



	CodArticulo	DescripcionArticulo
1	1	CARAMELOS BASTON VIENA ARCOR
2	2	CARAMELOS SURTIDO DE FRUTAS
3	3	CARAMELOS FRUTAS SURTIDA ARCOR
4	4	CARAMELOS FRUTAS MASTICABLES
5	6	FRUNA SURTIDA DONOFRIO
6	7	CHOCOLATE DOÑA PEPA FIELD
7	8	CHOCOLATE CUA CUA FIELD
8	9	MELLOWS FAMILIAR FIELD
9	10	WAFER CHOCOLATE FIELD
10	11	CHOCOLATE BARRA REGULAR

Es necesario utilizar la función CONVERT() con las fechas a comparar, ya que al ser los datos de tipo **datetime**, se requiere eliminar la parte de la hora de las fechas para que la comparación sea solo de las fechas y no de la fecha y la hora.

6.2. Subconsulta que entrega un conjunto de valores (varias filas, 1 columna)

Cuando la subconsulta se define en la cláusula WHERE del SELECT externo utilizando el operador IN (test de pertenencia), puede ser una subconsulta que entrega un conjunto de valores.

Ejercicio 7.15: Test de pertenencia

Escriba una consulta que entregue una lista de los artículos que no registran envíos a las tiendas. Recuerde que este requerimiento fue resuelto líneas arriba utilizando una consulta OUTER JOIN.

```
SELECT CodArticulo, DescripcionArticulo
FROM ARTICULO
WHERE CodArticulo NOT IN
    (SELECT CodArticulo FROM GUIA_DETALLE)
ORDER BY CodArticulo
GO
```



	CodArticulo	DescripcionArticulo
1	5	CHUPETES LOY AMBROSOLI
2	13	CHOCOLATE BARRA MILKY WAY
3	14	SNICKERS BAR KING SIZE
4	15	CHOCOLATE BARRA MILK DOVE
5	16	CHOCOLATE BARRA DARK DOVE
6	17	MILKY WAY BAR KING SIZE
7	18	GALLETAS CHIPS AHoy
8	19	GALLETAS TUAREG COSTA
9	20	GALLETAS VAINILLA COSTA
10	21	GALLETAS SURTIDAS BUTTER COOKIES

6.3. Subconsulta correlacionada

Se presenta cuando la consulta externa debe entregar datos a la consulta interna para que se pueda ejecutar.

- La consulta interna se evalúa repetidamente, una vez por cada fila de la consulta externa.
- Se puede definir en la cláusula WHERE de la consulta externa usando el operador EXISTS (Test de existencia).

Ejercicio 7.16: Test de existencia – Uso de EXISTS

Genere la lista de artículos que registran envíos a las tiendas.

```
SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo
  FROM ARTICULO
 WHERE EXISTS
   (SELECT * FROM GUIA_DETALLE
    WHERE ARTICULO.CodArticulo =
      GUIA_DETALLE.CodArticulo)
 ORDER BY ARTICULO.CodArticulo
 GO
```



	CodArticulo	DescripcionArticulo
1	1	CARAMELOS BASTON VIENA ARCOR
2	2	CARAMELOS SURTIDO DE FRUTAS
3	3	CARAMELOS FRUTAS SURTIDA ARCOR
4	4	CARAMELOS FRUTAS MASTICABLES
5	6	FRUNA SURTIDA DONOFRIO
6	7	CHOCOLATE DOÑA PEPA FIELD
7	8	CHOCOLATE CUA CUA FIELD
8	9	MELLOWS FAMILIAR FIELD
9	10	WAFER CHOCOLATE FIELD
10	11	CHOCOLATE BARRA REGULAR

7. EJERCICIOS PROPUESTOS

Para los siguientes ejercicios debe utilizar las bases de datos RH y EDUCA. Los scripts para crearlas en su servidor los encontrará en el CD que acompaña al libro.

Escriba las instrucciones SELECT para obtener:

1. (EDUCA) Un listado que incluya el nombre del curso con sus respectivos nombres de alumnos.
2. (EDUCA) Un reporte que muestre el nombre del alumno y la suma de todos sus paGOs.
3. (EDUCA) Un reporte que muestre el nombre del curso y el importe de todos sus paGOs.
4. (RH) Un reporte que muestre el nombre del departamento y el importe de su planilla.
5. (RH) La cantidad de trabajadores en cada ciudad.
6. (RH) Un listado de todos los departamentos y la cantidad de trabajadores en cada uno de ellos.
7. (EDUCA) La cantidad de alumnos matriculados en cada curso, debe incluir en el listado todos los cursos.
8. (EDUCA) Un reporte que para cada curso muestre la cantidad de alumnos matriculados y el importe recaudado por los paGOs realizados por los alumnos.
9. (RH) Todas las posibles combinaciones entre las tabla departamento y carGO.
10. (RH) Un listado de los empleados con el respectivo nombre de su superior inmediato.

Utilizando subconsultas escriba las instrucciones para obtener:

11. (RH) Quiénes son los empleados que tienen el menor sueldo.
12. (RH) Un listado de los empleados mostrando su sueldo y la diferencia con el sueldo promedio de la empresa.
13. (RH) Un reporte que muestre las personas que tienen el menor sueldo por departamento.
14. (RH) Un listado que muestre la cantidad de empleados y el importe de la planilla por departamento.
15. (EDUCA) Un reporte que muestre los alumnos matriculados en el curso SQL Server Implementación.



16. (RH) Un reporte que muestre los empleados que ocupan alguna gerencia.
17. (RH) Un reporte que muestre a los empleados que laboran en Trujillo.
18. (RH) Un reporte que muestre a los empleados que no laboran en Lima.



Capítulo VIII

Consultas: Casos Especiales

En este capítulo complementaremos el tema consultas de SQL Server mostrando casos especiales como las consultas DISTINCT, la creación de rankings, la creación de tablas a partir de las consultas, el uso de los operadores de conjuntos, entre otros.



8. CLÁUSULA DISTINCT

La cláusula DISTINCT establece que el conjunto de resultados de la consulta solo debe incluir filas únicas. Las filas duplicadas no se muestran como parte del resultado.

Sintaxis

```
SELECT [ DISTINCT ] ...  
FROM ...
```

Ejercicio 8.1: Uso de DISTINCT

Se desea obtener un listado de los artículos enviados a las tiendas.

```
USE QhatuPERU  
GO  
  
-- Artículos enviados a las tiendas  
SELECT GUIA_DETALLE.CodArticulo,  
ARTICULO.DescripcionArticulo  
FROM GUIA_DETALLE INNER JOIN ARTICULO  
ON GUIA_DETALLE.CodArticulo =  
ARTICULO.CodArticulo  
ORDER BY 1  
GO
```

	Resultados	Mensajes
	CodArticulo	DescripcionArticulo
1	1	CARAMELOS BASTON VIENA ARCOR
2	1	CARAMELOS BASTON VIENA ARCOR
3	1	CARAMELOS BASTON VIENA ARCOR
4	1	CARAMELOS BASTON VIENA ARCOR
5	1	CARAMELOS BASTON VIENA ARCOR
6	1	CARAMELOS BASTON VIENA ARCOR
7	1	CARAMELOS BASTON VIENA ARCOR
8	1	CARAMELOS BASTON VIENA ARCOR
9	1	CARAMELOS BASTON VIENA ARCOR
10	1	CARAMELOS BASTON VIENA ARCOR



El resultado muestra varios artículos más de una vez ya que éstos han salido del almacén en varias oportunidades. Para eliminar las filas duplicadas del resultado ejecute la consulta con la cláusula DISTINCT.

```
-- Eliminando filas duplicadas
SELECT DISTINCT GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
    ARTICULO.CodArticulo
ORDER BY 1
GO
```

Resultados		Mensajes
	CodArticulo	DescripcionArticulo
1	1	CARAMELOS BASTON VIENA ARCOR
2	2	CARAMELOS SURTIDO DE FRUTAS
3	3	CARAMELOS FRUTAS SURTIDA ARCOR
4	4	CARAMELOS FRUTAS MASTICABLES
5	6	FRUNA SURTIDA DONOFRIO
6	7	CHOCOLATE DOÑA PEPA FIELD
7	8	CHOCOLATE CUA CUA FIELD
8	9	MELLOWS FAMILIAR FIELD
9	10	WAFER CHOCOLATE FIELD
10	11	CHOCOLATE BARRA REGULAR

9. CLÁUSULA TOP

Especifica que el resultado de la consulta solo debe mostrar un tramo compuesto por las filas iniciales o por un porcentaje del total de filas recuperadas. Cuando la cláusula TOP se utiliza junto con la cláusula ORDER BY es posible generar un ranking en base a los valores de una columna numérica.

Sintaxis

```
SELECT TOP (cantidad_filas_iniciales_a_mostrar)
    [ PERCENT ] [ WITH TIES ] ...
FROM ...
```

- **cantidad_filas_iniciales_a_mostrar** indica la cantidad o porcentaje de filas a mostrar en el resultado.
- PERCENT indica que debe recuperarse el porcentaje de filas especificado por **cantidad_filas_iniciales_a_mostrar**.
- WITH TIES permite recuperar filas adicionales cuando en el último lugar del conjunto se produce un empate entre varias filas. Requiere que se utilice la cláusula ORDER BY.

Ejercicio 8.2: Uso de TOP

Se desea generar un ranking de los artículos más solicitados por las tiendas. Para ello, diseñe una consulta que muestre el total de unidades enviadas a las tiendas para cada artículo.

```
USE QhatuPERU
GO

-- Ranking de los artículos más solicitados
-- por las tiendas
SELECT GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo,
SUM(GUIA_DETALLE.CantidadEnviada)
AS TotalUnidades
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
ARTICULO.CodArticulo
GROUP BY GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo
ORDER BY TotalUnidades DESC
GO
```

Resultados

CodArticulo	DescripcionArticulo	TotalUnidades
1 66	JABON ROSAS Y LIMON ROSADO	20000
2 65	JABON ROSAS Y LIMON BLANCO	20000
3 130	DETERGENTE LIMON INVICTO	15000
4 127	DETERGENTE PODER LIMON ACE	7500
5 124	DETERGENTE C/BLANQUEADOR ARIEL	7500
6 126	DETERGENTE LIMON ECOLOGICO MAGIA BLANCA	7500
7 132	DETERGENTE LIMON OPAL	7500
8 125	DETERGENTE LIMON ARIEL	7500
9 64	JABON DOVE BLANCO	7000
10 74	P.H. BLANCO SUAVE (ROJA)	4000



Modifique la consulta para que muestre los 5 artículos más solicitados.

```
-- Ranking de los 5 primeros
SELECT TOP 5 GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo,
SUM(GUIA_DETALLE.CantidadEnviada)
AS TotalUnidades
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
ARTICULO.CodArticulo
GROUP BY GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo
ORDER BY TotalUnidades DESC
GO
```

	CodArticulo	DescripcionArticulo	TotalUnidades
1	66	JABON ROSAS Y LIMON ROSADO	20000
2	65	JABON ROSAS Y LIMON BLANCO	20000
3	130	DETERGENTE LIMON INVICTO	15000
4	127	DETERGENTE PODER LIMON ACE	7500
5	124	DETERGENTE C/BLANQUEADOR ARIEL	7500

Observe que el 4to y 5to lugares están empatados. Modifique la consulta para que verifique si hay más artículos empuestos con 7500 unidades despachadas.

```
-- Ranking de los 5 primeros con empates
SELECT TOP 5 WITH TIES GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo,
SUM(GUIA_DETALLE.CantidadEnviada)
AS TotalUnidades
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
ARTICULO.CodArticulo
GROUP BY GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo
ORDER BY TotalUnidades DESC
GO
```



	Resultados	Mensajes	
	CodArticulo	DescripcionArticulo	TotalUnidades
1	66	JABON ROSAS Y LIMON ROSADO	20000
2	65	JABON ROSAS Y LIMON BLANCO	20000
3	130	DETERGENTE LIMON INVICTO	15000
4	127	DETERGENTE PODER LIMON ACE	7500
5	124	DETERGENTE C/BLANQUEADOR ARIEL	7500
6	126	DETERGENTE LIMON ECOLOGICO MAGIA BLANCA	7500
7	132	DETERGENTE LIMON OPAL	7500
8	125	DETERGENTE LIMON ARIEL	7500

Observe que hay 5 artículos empuestos en el último lugar de este ranking de los 5 primeros.

Modifique la consulta para que muestre el quinto superior de todos los artículos enviados a las tiendas.

```
-- Quinto superior
SELECT TOP 20 PERCENT GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo,
SUM(GUIA_DETALLE.CantidadEnviada)
    AS TotalUnidades
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
    ARTICULO.CodArticulo
GROUP BY GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo
ORDER BY TotalUnidades DESC
GO
```



	CodArticulo	DescripcionArticulo	TotalUnidades
1	66	JABON ROSAS Y LIMON ROSADO	20000
2	65	JABON ROSAS Y LIMON BLANCO	20000
3	130	DETERGENTE LIMON INVICTO	15000
4	127	DETERGENTE PODER LIMON ACE	7500
5	124	DETERGENTE C/BLANQUEADOR ARIEL	7500
6	126	DETERGENTE LIMON ECOLOGICO MAGIA BLANCA	7500
7	132	DETERGENTE LIMON OPAL	7500
8	125	DETERGENTE LIMON ARIEL	7500
9	64	JABON DOVE BLANCO	7000
10	74	P.H. BLANCO SUAVE (ROJA)	4000
11	68	PASTA DENTAL KOLYNOS SUPER BLANCO	4000
12	129	DETERGENTE LIMON ARIEL	3750
13	106	DORINA CLASICA	1700
14	101	MANTEQUILLA LAIVE C/SAL	1700

Verifique si hay más artículos empuestos en el puesto 14.

```
-- Quinto superior
SELECT TOP 20 PERCENT WITH TIES
    GUIA_DETALLE.CodArticulo,
    ARTICULO.DescripcionArticulo,
    SUM(GUIA_DETALLE.CantidadEnviada)
        AS TotalUnidades
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
    ARTICULO.CodArticulo
GROUP BY GUIA_DETALLE.CodArticulo,
    ARTICULO.DescripcionArticulo
ORDER BY TotalUnidades DESC
GO
```



	CodArticulo	DescripcionArticulo	TotalUnidades
1	66	JABON ROSAS Y LIMON ROSADO	20000
2	65	JABON ROSAS Y LIMON BLANCO	20000
3	130	DETERGENTE LIMON INVICTO	15000
4	127	DETERGENTE PODER LIMON ACE	7500
5	124	DETERGENTE C/BLANQUEADOR ARIEL	7500
6	126	DETERGENTE LIMON ECOLOGICO MAGIA BLANCA	7500
7	132	DETERGENTE LIMON OPAL	7500
8	125	DETERGENTE LIMON ARIEL	7500
9	64	JABON DOVE BLANCO	7000
10	74	P.H. BLANCO SUAVE (ROJA)	4000
11	68	PASTA DENTAL KOLYNOS SUPER BLANCO	4000
12	129	DETERGENTE LIMON ARIEL	3750
13	106	DORINA CLASICA	1700
14	101	MANTEQUILLA LAIVE C/SAL	1700
15	103	MANTEQUILLA FERM C/SAL	1700
16	105	MARGARINA ASTRA	1700

Hay 4 artículos empataos con 1700 unidades enviadas en el último lugar de este ranking del quinto superior.

10. CLÁUSULAS OFFSET Y FETCH

Estas cláusulas permiten recuperar un intervalo específico de filas del resultado de la consulta.

Sintaxis

```
... ORDER BY ...
[ OFFSET cantidad_filas_iniciales_a_excluir
    ROW | ROWS ]
[ FETCH FIRST | NEXT cantidad_filas_a_mostrar
    ROW | ROWS ONLY ]
```

- **cantidad_filas_iniciales_a_excluir** especifica el número de filas iniciales que no se mostrarán en el resultado de la consulta.
- **cantidad_filas_a_mostrar** especifica el número de filas que se mostrará en el resultado de la consulta Luego de procesar la cláusula OFFSET.



Ejercicio 8.3: Uso de las cláusulas OFFSET y FETCH

Para ilustrar el uso de OFFSET y FETCH seguiremos revisando el ranking de los artículos despachados a las tiendas.

```
USE QhatuPERU
GO

-- Ranking de los artículos más solicitados
-- por las tiendas
SELECT GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo,
SUM(GUIA_DETALLE.CantidadEnviada)
AS TotalUnidades
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
ARTICULO.CodArticulo
GROUP BY GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo
ORDER BY TotalUnidades DESC
GO
```

	CodArticulo	DescripcionArticulo	TotalUnidades
1	66	JABON ROSAS Y LIMON ROSADO	20000
2	65	JABON ROSAS Y LIMON BLANCO	20000
3	130	DETERGENTE LIMON INVICTO	15000
4	127	DETERGENTE PODER LIMON ACE	7500
5	124	DETERGENTE C/BLANQUEADOR ARIEL	7500
6	126	DETERGENTE LIMON ECOLOGICO MAGIA BLANCA	7500
7	132	DETERGENTE LIMON OPAL	7500
8	125	DETERGENTE LIMON ARIEL	7500
9	64	JABON DOVE BLANCO	7000
10	74	P.H. BLANCO SUAVE (ROJA)	4000

Modifique la consulta para que el resultado muestre el ranking a partir del puesto 11.

```
-- Ranking a partir del puesto 11
SELECT GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo,
SUM(GUIA_DETALLE.CantidadEnviada)
AS TotalUnidades
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
ARTICULO.CodArticulo
```



```
GROUP BY GUIA_DETALLE.CodArticulo,  
ARTICULO.DescripcionArticulo  
ORDER BY TotalUnidades DESC  
OFFSET 10 ROWS  
GO
```

	CodArticulo	DescripcionArticulo	TotalUnidades
1	68	PASTA DENTAL KOLYNOS SUPER BLANCO	4000
2	129	DETERGENTE LIMON ARIEL	3750
3	106	DORINA CLASICA	1700
4	101	MANTEQUILLA LAIVE C/SAL	1700
5	103	MANTEQUILLA FERM C/SAL	1700
6	105	MARGARINA ASTRA	1700
7	116	SPRITE RETORNABLE	1500
8	117	TRIPLE DIET NO RETORNABLE	1500
9	111	INCA KOLA DIET	1500
10	114	SPRITE CONTOUR	1500

Ahora, permita que el resultado muestre los 5 artículos que se encuentran después del puesto 10.

```
-- Ranking del puesto 11 al puesto 15  
SELECT GUIA_DETALLE.CodArticulo,  
ARTICULO.DescripcionArticulo,  
SUM(GUIA_DETALLE.CantidadEnviada)  
    AS TotalUnidades  
FROM GUIA_DETALLE INNER JOIN ARTICULO  
ON GUIA_DETALLE.CodArticulo =  
    ARTICULO.CodArticulo  
GROUP BY GUIA_DETALLE.CodArticulo,  
ARTICULO.DescripcionArticulo  
ORDER BY TotalUnidades DESC  
OFFSET 10 ROWS  
FETCH FIRST 5 ROWS ONLY  
GO
```



Resultados			
	CodArticulo	DescripcionArticulo	TotalUnidades
1	68	PASTA DENTAL KOLYNOS SUPER BLANCO	4000
2	129	DETERGENTE LIMON ARIEL	3750
3	106	DORINA CLASICA	1700
4	101	MANTEQUILLA LAIVE C/SAL	1700
5	103	MANTEQUILLA FERM C/SAL	1700

11. CREACIÓN DE TABLA A PARTIR DE UNA CONSULTA

Usando la cláusula INTO es posible guardar el resultado de una consulta en una tabla nueva.

Sintaxis

```
SELECT ...
INTO nombre_tabla_nueva ...
```

- **nombre_tabla_nueva** especifica el nombre de la tabla que guardará el resultado de la consulta. Si la tabla ya existe, se producirá un error.

Ejercicio 8.4: Creación de una tabla con SELECT ... INTO

Diseñar una consulta que genere el catálogo de artículos y lo guarde en la tabla **CatalogoArticulos**.

```
-- Crear una tabla con columnas
-- de distintas tablas origen
SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       ARTICULO.Presentacion,
       ARTICULO.PrecioProveedor,
       LINEA.NomLinea, PROVEEDOR.NomProveedor
INTO CatalogoArticulos
FROM ARTICULO INNER JOIN LINEA
ON ARTICULO.CodLinea = LINEA.CodLinea
INNER JOIN PROVEEDOR
ON ARTICULO.CodProveedor =
   PROVEEDOR.CodProveedor
GO
```



```
SELECT * FROM CatalogoArticulos
```

Resultados		Mensajes			
CodArticulo	DescripcionArticulo	Presentacion	PrecioProveedor	NomLinea	NomProveedor
1	1	CARAMELOS BASTON VIENA ARCOR	PAQUETE 454 GR	1,50	GOLOSINAS GOLOSINAS Y ANTOJOS
2	2	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	1,00	GOLOSINAS DISTRIBUIDORA DE GOLOSINAS FENIX
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	PAQUETE 520 GR	1,50	GOLOSINAS GOLOSINAS Y ANTOJOS
4	4	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	1,30	GOLOSINAS GOLOSINAS Y ANTOJOS
5	5	CHUPETES LOLY AMBROSOLI	KILOGRAMO	1,20	GOLOSINAS DISTRIBUIDORA DE GOLOSINAS FENIX
6	6	FRUNA SURTIDA DONOFRIO	PAQUETE X 24 UNIDADES	1,80	GOLOSINAS DISTRIBUIDORA DE GOLOSINAS FENIX
7	7	CHOCOLATE DOÑA PEPA FIELD	PAQUETE X 6 UNIDADES	2,20	GOLOSINAS DISTRIBUIDORA DE GOLOSINAS FENIX
8	8	CHOCOLATE CUA CUA FIELD	PAQUETE X 6 UNIDADES	1,60	GOLOSINAS DISTRIBUIDORA DE GOLOSINAS FENIX
9	9	MELLOWS FAMILIAR FIELD	PAQUETE 454 GR	2,10	GOLOSINAS DISTRIBUIDORA DE GOLOSINAS FENIX
10	10	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES	0,70	GOLOSINAS DISTRIBUIDORA DE GOLOSINAS FENIX

11.1. Uso de la función IDENTITY

La función IDENTITY permite definir una columna con la propiedad IDENTITY cuando la tabla se crea con SELECT ... INTO.

Ejercicio 8.5: Creación de una columna IDENTITY en una tabla creada con SELECT ... INTO

Guardar el ranking creado por la consulta del ejercicio 8.3 en una tabla de nombre **RankingArticulos**. Esta tabla deberá tener una columna IDENTITY que indique la posición de cada artículo en el ranking.

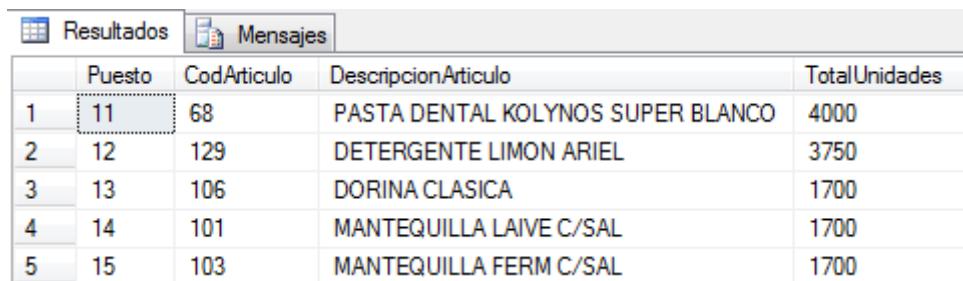
```
USE QhatuPERU
GO

-- Ranking de los artículos
-- más solicitados por las tiendas
-- guardado en la tabla RankingArticulos
SELECT IDENTITY(integer, 1, 1) AS Puesto,
GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo,
SUM(GUIA_DETALLE.CantidadEnviada)
    AS TotalUnidades
INTO RankingArticulos
FROM GUIA_DETALLE INNER JOIN ARTICULO
ON GUIA_DETALLE.CodArticulo =
    ARTICULO.CodArticulo
GROUP BY GUIA_DETALLE.CodArticulo,
ARTICULO.DescripcionArticulo
ORDER BY TotalUnidades DESC
GO
```



Consulte el ranking para que muestre los puestos del 11 al 15.

```
-- Mostrar los puestos del 11 al 15 en el ranking
SELECT * FROM RankingArticulos
ORDER BY Puesto
OFFSET 10 ROWS
FETCH NEXT 5 ROWS ONLY
GO
```



	Puesto	CodArticulo	DescripcionArticulo	TotalUnidades
1	11	68	PASTA DENTAL KOLYNOS SUPER BLANCO	4000
2	12	129	DETERGENTE LIMON ARIEL	3750
3	13	106	DORINA CLASICA	1700
4	14	101	MANTEQUILLA LAIVE C/SAL	1700
5	15	103	MANTEQUILLA FERM C/SAL	1700

12. OPERADORES DE CONJUNTOS: UNION, EXCEPT, INTERSECT

Los operadores de conjuntos permiten combinar los resultados de dos o más consultas en un solo conjunto de resultados. Microsoft SQL Server proporciona 3 operadores para dichas operaciones: UNION, EXCEPT e INTERSECT.

12.1. Operador UNION

Este operador reúne los resultados de dos consultas en un solo conjunto de resultados.

Sintaxis

```
Consulta1
UNION [ ALL ]
Consulta2
```

- ALL establece que el conjunto resultante debe incluir todas las filas de ambas consultas incluso las duplicadas.



Ejercicio 8.6: Uso del operador UNION

Se desea obtener un listado de las fechas en las que se ha producido movimientos en el almacén de QhatuPERU.

```
USE QhatuPERU
GO

SELECT COUNT(DISTINCT FechaIngreso)
      FROM ORDEN_COMPRA
GO
-- 16 fechas distintas para FechaIngreso

SELECT COUNT(DISTINCT FechaSalida)
      FROM GUIA_ENVIO
GO
-- 107 fechas distintas para FechaSalida

-- Uso de UNION ALL
-- Todas las fechas en las que hubo
-- ingreso y/o salida
-- incluyendo las filas duplicadas
SELECT CONVERT(CHAR(10), FechaIngreso, 103)
      AS FechaMovimiento,
      'Entrada' AS Movimiento
FROM ORDEN_COMPRA
UNION ALL
SELECT CONVERT(CHAR(10), FechaSalida, 103),
      'Salida' AS Movimiento
FROM GUIA_ENVIO
ORDER BY FechaMovimiento
GO
-- 125 movimientos
```

	FechaMovimiento	Movimiento
1	01/04/2013	Salida
2	01/04/2013	Salida
3	01/04/2013	Salida
4	01/04/2013	Salida
5	01/04/2013	Salida
6	01/04/2013	Salida
7	04/04/2013	Salida
8	04/04/2013	Salida
9	04/04/2013	Salida
10	04/04/2013	Salida



```
-- UNION
-- Todas las fechas en las que hubo
-- ingreso y/o salida
-- sin incluir las filas duplicadas
SELECT CONVERT(CHAR(10), FechaIngreso, 103)
      AS FechaMovimiento,
'Entrada' AS Movimiento
FROM ORDEN_COMPRA
UNION
SELECT CONVERT(CHAR(10), FechaSalida, 103),
'Salida' AS Movimiento
FROM GUIA_ENVIO
ORDER BY FechaMovimiento
GO
-- 21 combinaciones fecha-movimiento
```

	FechaMovimiento	Movimiento
1	01/04/2013	Salida
2	04/04/2013	Salida
3	05/04/2013	Salida
4	06/04/2013	Salida
5	09/04/2013	Salida
6	10/04/2013	Entrada
7	10/04/2013	Salida
8	11/04/2013	Salida
9	12/04/2013	Entrada
10	13/04/2013	Entrada

12.2. Operador EXCEPT

Este operador muestra en el conjunto de resultados las filas que solo están presentes en la primera consulta, y no se encuentran en la segunda consulta.

Sintaxis

```
Consulta1
EXCEPT
Consulta2
```



Ejercicio 8.7: Uso del operador EXCEPT

Se desea obtener un listado de las fechas en las que solo se ha ejecutado movimientos de entrada en el almacén de QhatuPERU.

```
SELECT COUNT(DISTINCT FechaIngreso)
      FROM ORDEN_COMPRA
GO
-- 16 fechas distintas para FechaIngreso

SELECT COUNT(DISTINCT FechaSalida)
      FROM GUIA_ENVIO
GO
-- 107 fechas distintas para FechaSalida

-- EXCEPT
-- Todas las fechas en las que hubo entradas,
-- pero no hubo salidas
SELECT CONVERT(CHAR(10), FechaIngreso, 103)
      AS FechaMovimiento,
      'Entrada' AS Movimiento
FROM ORDEN_COMPRA
EXCEPT
SELECT CONVERT(CHAR(10), FechaSalida, 103),
      'Salida' AS Movimiento
FROM GUIA_ENVIO
ORDER BY FechaMovimiento
GO
-- 4 fechas en las que solo hubo entradas
```

	FechaMovimiento	Movimiento
1	10/04/2013	Entrada
2	12/04/2013	Entrada
3	13/04/2013	Entrada
4	15/04/2013	Entrada

12.3. Operador Intersect

Este operador muestra en el conjunto de resultados únicamente las filas que se encuentran en las dos consultas sobre las que opera.



Sintaxis

```
Consulta1
INTERSECT
Consulta2
```

Ejercicio 8.8: Uso del operador INTERSECT

Se desea obtener un listado de las fechas en las que se ha ejecutado movimientos de entrada, y también movimientos de salida.

```
SELECT COUNT(DISTINCT FechaIngreso)
      FROM ORDEN_COMPRA
GO
-- 16 fechas distintas para FechaIngreso

SELECT COUNT(DISTINCT FechaSalida)
      FROM GUIA_ENVIO
GO
-- 107 fechas distintas para FechaSalida

-- INTERSECT
-- Fechas en las que hubo entrada y salida
SELECT CONVERT(CHAR(10), FechaIngreso, 103)
      AS FechaMovimiento,
      'Entrada' AS Movimiento
     FROM ORDEN_COMPRA
INTERSECT
SELECT CONVERT(CHAR(10), FechaSalida, 103),
      'Salida' AS Movimiento
     FROM GUIA_ENVIO
    ORDER BY 1
GO
-- 0 filas
```

Resultados	Mensajes
FechaMovimiento	Movimiento

13. EXPRESIÓN CASE

Se utiliza para evaluar una lista de expresiones lógicas y tomar alguna acción en función a los resultados.

13.1. Expresión CASE simple

Compara una expresión contra un conjunto de expresiones simples para determinar la acción a seguir.

Sintaxis

```
CASE expresión_de_entrada
    WHEN expresión_a_evaluar1
        THEN acción_a_seguir1,
    WHEN expresión_a_evaluar2
        THEN acción_a_seguir2,
    ...
    [ ELSE
        acción_cuando_ninguna_anterior_cumple ]
END
```

Ejercicio 8.9: Uso de expresión CASE simple

```
USE QhatuPERU
GO

SELECT NomProveedor, Ciudad,
Ubicacion =
CASE Ciudad
    WHEN 'Lima' THEN 'Capital'
    WHEN 'Callao' THEN 'Puerto'
    ELSE 'Provincias'
END
FROM PROVEEDOR
GO
```

En este ejercicio, la expresión CASE evalúa el contenido de la columna **Ciudad** de la tabla **PROVEEDOR**, y según su valor almacena la cadena '**Capital**', '**Puerto**' o '**Provincias**' en la columna **Ubicación** del resultado de la consulta.



Resultados			
	NomProveedor	Ciudad	Ubicacion
1	LACTEOS DEL CENTRO	HUANCAYO	Provincias
2	DISTRIBUIDORA ALEMANA	LIMA	Capital
3	EMBUTIDOS EL GORDITO	CALLAO	Puerto
4	DISTRIBUIDORA NANDO	CALLAO	Puerto
5	DISTRIBUIDORA ALBRICIAS	LIMA	Capital
6	DISTRIBUIDORA DEL HOGAR	LIMA	Capital
7	PAPELERA PACHACAMAC	CAÑETE	Provincias
8	DISTRIBUIDORA SAN ANTONIO	LIMA	Capital
9	EMBOTELLADORA LA PREFERIDA	TRUJILLO	Provincias
10	DROGUERIA MAHAN	AREQUIPA	Provincias

13.2. Expresión CASE de búsqueda

En este caso, la expresión CASE evalúa un conjunto de expresiones lógicas, y determina para cada una de ellas una acción a seguir.

Sintaxis

```
CASE
    WHEN expresión_lógica1 THEN acción_a_seguir1
        WHEN expresión_lógica1 THEN acción_a_seguir2
        ...
    [ELSE acción_cuando_ninguna_anterior_cumple]
END
```

Ejercicio 8.10: Uso de expresión CASE de búsqueda

```
USE QhatuPERU
GO

SELECT CodArticulo, DescripcionArticulo,
'Acción a tomar' = CASE
    WHEN StockActual <= StockMinimo
        THEN 'URGENTE: Colocar pedido'
    WHEN StockActual < StockMinimo * 1.1
        THEN 'Stock cerca al mínimo'
    ELSE 'Stock adecuado'
END
FROM ARTICULO
GO
```

	CodArticulo	DescripcionArticulo	Acción a tomar
23	23	FUDGE SHOPPE DELUXE GRAHAMS	Stock adecuado
24	24	FUDGE SHOPPE STICKS KEEB	Stock adecuado
25	25	GALLETAS DELICE	Stock adecuado
26	26	JAMONADA LAIVE	Stock cerca al mínimo
27	27	JAMONADA ESPECIAL LA SEGOVIANA	URGENTE: Colocar pedido
28	28	JAMONADA POLACA OTTO KUNZ	Stock adecuado
29	29	JAMONADA DE POLLO SAN FERNANDO	Stock cerca al mínimo
30	30	JAMONADA ESPECIAL OTTO KUNZ	Stock adecuado
31	31	JAMON INGLES SAN FERNANDO	Stock adecuado
32	32	JAMON INGLES LAIVE	Stock adecuado

En este ejercicio se compara para cada producto, el **StockActual** contra el **StockMinimo**, y de acuerdo con el resultado se coloca la cadena correspondiente en la columna **Acción a tomar** del resultado de la consulta.

14. OPERADOR PIVOT

Permite crear una tabla de doble entrada haciendo que los valores que se muestran como filas en un conjunto de datos, se puedan mostrar como títulos de columna.

Sintaxis

```
SELECT * FROM
(
    Consulta_Origen:
    consulta con la data a agregar y pivotear
) AS alias1
PIVOT( función_agregación( columna_numérica )
    FOR columna_dimensión
    IN ( lista_valores_columna_dimensión ) )
AS alias2
```

- **consulta_origen** es la consulta que tiene los datos que normalmente se muestran como filas, y se desea consolidar y pivotear.
- **columna_numérica** es la columna cuyos valores se desea mostrar en una tabla de doble entrada.
- **columna_dimensión** es la columna cuyos valores en una consulta normal se ven como filas, y que en la consulta PIVOT se desea ver como columnas.



- **lista_valores_columna_dimensión** son los valores de *columna_dimensión* a mostrar como columnas en la tabla de doble entrada.

Ejercicio 8.11: Uso del operador PIVOT

Se tiene la siguiente consulta que muestra todos los despachos enviados a las tiendas especificando la línea del artículo enviado, el año y mes de envío, y el monto de lo enviado.

```
USE QhatuPERU
GO

-- Consulta que muestra el monto despachado
-- de todos los envíos a las tiendas.
SELECT LINEA.NomLinea AS Linea,
CAST(YEAR(GUIA_ENVIO.FechaSalida) AS CHAR(4))
    + '-' +
DATENAME(MONTH, GUIA_ENVIO.FechaSalida)
    AS Periodo,
GUIA_DETALLE.CantidadEnviada *
    GUIA_DETALLE.PrecioVenta AS Monto
FROM LINEA INNER JOIN ARTICULO
ON LINEA.CodLinea = ARTICULO.CodLinea
INNER JOIN GUIA_DETALLE
ON ARTICULO.CodArticulo =
    GUIA_DETALLE.CodArticulo
INNER JOIN GUIA_ENVIO
ON GUIA_DETALLE.NumGuia = GUIA_ENVIO.NumGuia
GO
```

	Linea	Periodo	Monto
1	GOLOSINAS	2013-Marzo	45,00
2	GOLOSINAS	2013-Marzo	30,00
3	GOLOSINAS	2013-Marzo	45,00
4	GOLOSINAS	2013-Marzo	39,00
5	GOLOSINAS	2013-Marzo	81,00
6	GOLOSINAS	2013-Marzo	82,50
7	GOLOSINAS	2013-Marzo	60,00
8	GOLOSINAS	2013-Marzo	63,00
9	GOLOSINAS	2013-Marzo	21,00
10	GOLOSINAS	2013-Marzo	30,00



Se desea un reporte que consolide el monto enviado por **Línea** y por **Periodo**, solo que los períodos deben mostrarse como columnas, tal como se observa a continuación.

	Linea	2013-Marzo	2013-Abril
1	EMBUTIDOS	24312,00	36468,00
2	GOLOSINAS	4872,00	7308,00
3	HIGIENE PERSONAL	30938,00	46407,00
4	LACTEOS	9930,00	11171,25
5	LICORES Y GASEOSAS	26114,00	22849,75
6	LIMPIEZA	84120,00	73605,00

```
SELECT * FROM
(
SELECT LINEA.NomLinea AS Linea,
CAST(YEAR(GUIA_ENVIO.FechaSalida) AS CHAR(4))
+ '-' +
DATENAME(MONTH, GUIA_ENVIO.FechaSalida)
AS Periodo,
GUIA_DETALLE.CantidadEnviada *
    GUIA_DETALLE.PrecioVenta AS Monto
FROM LINEA INNER JOIN ARTICULO
ON LINEA.CodLinea = ARTICULO.CodLinea
INNER JOIN GUIA_DETALLE
ON ARTICULO.CodArticulo =
    GUIA_DETALLE.CodArticulo
INNER JOIN GUIA_ENVIO
ON GUIA_DETALLE.NumGuia = GUIA_ENVIO.NumGuia
) AS Origen
PIVOT (SUM(Monto) FOR Periodo
IN ( [2013-Marzo], [2013-Abril] ))
AS Destino
GO
```

15. COMMON TABLE EXPRESSION (CTE)

Un CTE es un conjunto de resultados temporal derivado de una consulta que puede ser utilizado como una tabla derivada. Su comportamiento es muy similar al de una vista. Una CTE puede contener referencias a ella misma, por lo que permite diseñar consultas recursivas.

Sintaxis

```
WITH nombre_CTE( lista_columnas )
AS
(
SELECT_que_puebla_la_CTE
)
SELECT_que_muestra_la_CTE
```

- **SELECT_que_puebla_la_CTE** es la consulta que carga los datos en la CTE.
- **SELECT_que_muestra_la_CTE** es la consulta que muestra la data cargada en la CTE.

Ejercicio 8.12: Uso de Common Table Expression

Crear una CTE que contenga el proveedor, fecha y monto de cada una de las órdenes de envío. Luego consultarla para que muestre las órdenes del 11 de abril del 2013.

```
USE QhatuPERU
GO

WITH ComprasCTE(Orden, Fecha, Proveedor, Monto)
AS
(
SELECT ORDEN_COMPRA.NumOrden,
CONVERT(CHAR(10), ORDEN_COMPRA.FechaOrden, 102),
PROVEEDOR.NomProveedor,
SUM(ORDEN_DETALLE.CantidadRecibida *
    ORDEN_DETALLE.PrecioCompra)
FROM ORDEN_COMPRA INNER JOIN ORDEN_DETALLE
ON ORDEN_COMPRA.NumOrden =
    ORDEN_DETALLE.NumOrden
INNER JOIN ARTICULO
ON ORDEN_DETALLE.CodArticulo =
    ARTICULO.CodArticulo
INNER JOIN PROVEEDOR
ON ARTICULO.CodProveedor =
    PROVEEDOR.CodProveedor
GROUP BY ORDEN_COMPRA.NumOrden,
CONVERT(CHAR(10),
    ORDEN_COMPRA.FechaOrden, 102),
PROVEEDOR.NomProveedor
)
SELECT * FROM ComprasCTE
WHERE Fecha = '2013.04.11'
GO
```



Resultados		Mensajes		
	Orden	Fecha	Proveedor	Monto
1	5	2013.04.11	DISTRIBUIDORA DE GOLOSINAS FENIX	12525,00
2	6	2013.04.11	GOLOSINAS Y ANTOJOS	4700,00

Ejercicio 8.13: Consultas recursivas usando CTE

Ejecute las siguientes instrucciones (la tabla TRABAJADOR fue creada en el capítulo anterior):

```
USE QhatuPERU
GO
```

```
SELECT * FROM Trabajador
GO
```

Resultados		Mensajes	
	idTrabajador	Apellidos	Jefe
1	101	Camacho Saravia	102
2	102	Ardiles Soto	NULL
3	103	Sánchez Aliaga	101
4	104	Castro Avila	101
5	105	Vilchez Santos	102
6	106	Juárez Pinto	105
7	107	Urrunaga Tapia	101

El árbol de jerarquías de los trabajadores tiene la siguiente estructura:

```
Nivel 0           Nivel 1           Nivel 2
Ardiles Soto
                  Camacho Saravia
                           Sánchez Aliaga
                           Castro Avila
                           Urrunaga Tapia
                  Vilchez Santos
                           Juárez Pinto
```

Mediante una consulta recursiva a una CTE obtener un listado de los trabajadores que muestre en qué nivel de la jerarquía se encuentra cada uno.



```
WITH EmpleadosCTE(CodIGO, Nombre, Nivel)
AS
(
-- miembro ancla - consulta no recursiva
SELECT idTrabajador, Apellidos, 0 AS Nivel
FROM Trabajador
WHERE Jefe IS NULL
UNION ALL
-- miembro recursivo - consulta recursiva
SELECT t.idTrabajador, t.Apellidos, Nivel+1
FROM Trabajador t INNER JOIN EmpleadosCTE e
ON t.Jefe = e.CodIGO
)
-- consulta externa
SELECT * FROM EmpleadosCTE
GO
```

	Codigo	Nombre	Nivel
1	102	Ardiles Soto	0
2	101	Camacho Saravia	1
3	105	Vilchez Santos	1
4	106	Juárez Pinto	2
5	103	Sánchez Aliaga	2
6	104	Castro Avila	2
7	107	Umunaga Tapia	2

Una CTE recursiva es construida desde al menos dos consultas. Una, es una consulta no recursiva conocida como el miembro ancla. La segunda, es la consulta recursiva conocida como el miembro recursivo. Estas consultas van separadas por el operador UNION ALL.

La CTE es inicialmente cargada con el resultado de la consulta del miembro ancla. A continuación, el operador UNION ALL combina el resultado de la primera consulta con el resultado de la consulta recursiva.

16. EJERCICIOS PROPUESTOS

Para los siguientes ejercicios debe utilizar las bases de datos RH y EDUCA. Los scripts para crearlas en su servidor los encontrará en el CD que acompaña al libro.

SELECT ... INTO

1. (RH) Crear una tabla de nombre PLANILLA que contenga la siguiente información:
 - Código de departamento
 - Nombre del departamento
 - Importe de planilla
 - Importe de planilla proyectada con un aumento de 15%

Operador UNION

2. (EDUCA) En una sola consulta combine los datos de los alumnos y profesores. El resultado debe mostrar una columna adicional cuyo valor indica si los datos del registro corresponden a un Alumno o a un Profesor.

Operador EXCEPT

3. (EDUCA) Se necesita un listado de los alumnos (solo el código) que no se han matriculado en el curso **SQL Server Implementación**.

Operador INTERSECT

4. (EDUCA) Se necesita un listado de los alumnos (solo el código) que están matriculados en los cursos **SQL Server Implementación** y **SQL Server Administración**.

Cláusula TOP

5. (RH) Escriba una consulta para averiguar quiénes son los trabajadores que tienen el sueldo más bajo.
6. (EDUCA) Escriba una consulta para averiguar quiénes son los alumnos con la menor nota.

Cláusula DISTINCT

7. (EDUCA) Escriba una consulta que muestre el listado de los profesores.

Función CASE

8. (EDUCA) Escriba una consulta que califique la nota de cada alumno según el siguiente cuadro:



Nota	Calificación
[0, 10>	Malo
[10, 14>	Regular
[14, 18>	Bueno
[18,20]	Excelente

Operador PIVOT

9. (EDUCA) Encontrar el ingreso por mes de cada curso.
10. (EDUCA) Encontrar el ingreso por trimestre de cada año.
11. (RH) Encontrar la cantidad de empleados que han ingresado por trimestre en cada año.

Common Table Expression (CTE)

12. (RH) Encontrar el empleado que tiene el menor salario por departamento.
13. (EDUCA) Encontrar el alumno con la mejor nota por curso.



Capítulo IX

Vistas

En este capítulo aprenderemos a construir vistas, un objeto de la base de datos que almacena una consulta predefinida. Las vistas ocultan la complejidad del modelo de datos y permiten que el usuario se enfoque en la data que es relevante para él, además de simplificar el diseño de consultas complejas y facilitar la administración de los permisos.

En ocasiones, una consulta compleja se puede simplificar si previamente se diseñan vistas que contienen consultas parciales que combinadas permiten llegar al resultado final requerido. Adicionalmente, las vistas mejoran el rendimiento de las consultas ya que la instrucción SELECT asociada se guarda compilada y con su plan de ejecución ya definido.

17. VISTA

Una **vista** es un objeto que almacena una consulta predefinida y que proporciona un modo alternativo de visualización de datos sin tener que redefinir la consulta. Las tablas requeridas en una vista se llaman tablas base. Con algunas excepciones, cualquier declaración SELECT puede nombrarse y guardarse como una vista. Los ejemplos comunes de vistas incluyen:

- Un subconjunto de filas o columnas de una tabla base.
- Una unión de dos o más tablas base.
- Un join de dos o más tablas base.
- Un resumen estadístico de una tabla base.
- Un subconjunto de otra vista, o alguna combinación de vistas y tablas base.

Ejercicio 9.1: Ejemplo de vista

En este ejercicio se crea la vista **v_ListaPrecios** en la base de datos **QhatuPERU**.

```
USE QhatuPERU
GO

CREATE VIEW v_ListaPrecios
AS
SELECT LINEA.NomLinea, ARTICULO.CodArticulo,
ARTICULO.DescripcionArticulo,
ARTICULO.Presentacion,
ARTICULO.PrecioProveedor
FROM LINEA INNER JOIN ARTICULO
ON LINEA.CodLinea = ARTICULO.CodLinea
GO

SELECT * FROM v_ListaPrecios
GO
```

Resultados		Mensajes		
NomLinea	CodArticulo	DescripcionArticulo	Presentacion	PrecioProveedor
1	GOLOSINAS 1	CARAMELOS BASTON VIENA ARCOR	PAQUETE 454 GR	1,50
2	GOLOSINAS 2	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	1,00
3	GOLOSINAS 3	CARAMELOS FRUTAS SURTIDA ARCOR	PAQUETE 520 GR	1,50
4	GOLOSINAS 4	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	1,30
5	GOLOSINAS 5	CHUPETES LOLY AMBROSOLI	KILOGRAMO	1,20
6	GOLOSINAS 6	FRUNA SURTIDA DONOFRIO	PAQUETE X 24 UNIDADES	1,80
7	GOLOSINAS 7	CHOCOLATE DOÑA PEPA FIELD	PAQUETE X 6 UNIDADES	2,20
8	GOLOSINAS 8	CHOCOLATE CUA CUA FIELD	PAQUETE X 6 UNIDADES	1,60
9	GOLOSINAS 9	MELLOWS FAMILIAR FIELD	PAQUETE 454 GR	2,10
10	GOLOSINAS 10	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES	0,70

Observe que para ejecutar la vista se utiliza la instrucción SELECT. Una vista se manipula como si fuera una tabla; es decir, que se le puede aplicar las declaraciones SELECT, INSERT, UPDATE y DELETE. Sin embargo, no debe ver la vista como una tabla, sino pensar en ella como un programa que ejecuta una sola instrucción (la instrucción SELECT).

18. CREACIÓN DE VISTAS – INSTRUCCIÓN CREATE VIEW

Sintaxis

```
CREATE VIEW nombre_vista [ ( lista_columnas ) ]
[ WITH ENCRYPTION, SCHEMABINDING ]
AS
sentencia_select
[ WITH CHECK OPTION ]
```

- WITH ENCRYPTION indica que se debe encriptar la sentencia con la que se define la vista, de modo tal que cuando se utiliza el procedimiento **sp_helptext** no se pueda observar la instrucción que creó la vista.
- WITH SCHEMABINDING enlaza la vista al esquema de las tablas subyacentes. Cuando se crea una vista con enlace de esquema, las tablas subyacentes no se pueden modificar ni eliminar.
- WITH CHECK OPTION indica que si la definición de la vista contiene la cláusula WHERE, y la vista se utiliza para operaciones de mantenimiento de datos, estas operaciones deben respetar la condición definida en el WHERE.

Las siguientes restricciones se aplican a la creación de vistas:

- No se puede definir una vista con ORDER BY o SELECT INTO.
- No pueden hacer referencia a tablas temporales o variables que hacen referencia a tablas.
- No pueden hacer referencia a más de 1024 columnas.
- La sentencia CREATE VIEW no puede combinarse con otras sentencias Transact-SQL en el mismo batch.

18.1. Consideraciones al crear las vistas

Tenga en cuenta lo siguiente al momento de crear una vista:

- Para ejecutar la declaración CREATE VIEW, debe ser miembro de alguno de los siguientes roles: **sysadmin**, **db_owner**, **db_ddladmin**, o debe tener el permiso CREATE VIEW. Además, también debe tener el permiso SELECT en todas las tablas o vistas que son referenciadas dentro de la vista.
- Para evitar situaciones en las que el dueño de una vista y el dueño de las tablas subyacentes son distintos, se recomienda que el usuario **dbo** tenga todos los objetos de una base de datos.
- Si la declaración SELECT de la vista contiene columnas computadas o constantes, éstas deben tener un nombre asignado.

Recomendación: Antes de crear la vista, es importante probar la declaración SELECT que define la vista para asegurar que el servidor devuelve el resultado esperado.

Ejercicio 9.2: Reporte que muestra el balance del almacén de QhatuPERU

Se desea obtener un reporte que muestre el balance entrada/salida de todos los artículos registrados en la base de datos QhatuPERU. El reporte debe mostrar para cada artículo: el código y la descripción del artículo, la cantidad de unidades entrantes en el almacén, y la cantidad de unidades salientes del almacén.

Este problema se resuelve fácilmente si crea una vista para las entradas, y una segunda vista para las salidas. Luego, debe combinar las vistas anteriores para obtener el reporte final.

Creación de la vista para el reporte de entradas

```
USE QhatuPERU
GO

-- Vista para el listado de entradas
CREATE VIEW v_UnidadesEntrantes
AS
```



```
SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       ISNULL(SUM(
           ORDEN_DETALLE.CantidadRecibida ), 0)
           AS Entradas
  FROM ARTICULO LEFT OUTER JOIN ORDEN_DETALLE
  ON ARTICULO.CodArticulo =
     ORDEN_DETALLE.CodArticulo
 GROUP BY ARTICULO.CodArticulo,
          ARTICULO.DescripcionArticulo
 GO

-- Ejecución de la vista de entradas
SELECT * FROM v_UnidadesEntrantes
GO
```

	CodArticulo	DescripcionArticulo	Entradas
1	1	CARAMELOS BASTON VIENA ARCOR	0
2	2	CARAMELOS SURTIDO DE FRUTAS	1500
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	1250
4	4	CARAMELOS FRUTAS MASTICABLES	3000
5	5	CHUPETES LOLY AMBROSOLI	3000
6	6	FRUNA SURTIDA DONOFRIO	3000
7	7	CHOCOLATE DOÑA PEPA FIELD	3000
8	8	CHOCOLATE CUA CUA FIELD	3000
9	9	MELLOWS FAMILIAR FIELD	0
10	10	WAFER CHOCOLATE FIELD	3000

Creación de la vista para el reporte de salidas

```
-- Vista para el listado de salidas
CREATE VIEW v_UnidadesSalientes
AS
SELECT ARTICULO.CodArticulo,
       ARTICULO.DescripcionArticulo,
       ISNULL(SUM(
           GUIA_DETALLE.CantidadEnviada ), 0)
           AS Salidas
  FROM ARTICULO LEFT OUTER JOIN GUIA_DETALLE
  ON ARTICULO.CodArticulo =
     GUIA_DETALLE.CodArticulo
 GROUP BY ARTICULO.CodArticulo,
          ARTICULO.DescripcionArticulo
 GO
```



```
-- Ejecución de la vista de salidas
SELECT * FROM v_UnidadesSalientes
GO
```

	CodArticulo	DescripcionArticulo	Salidas
1	1	CARAMELOS BASTON VIENA ARCOR	400
2	2	CARAMELOS SURTIDO DE FRUTAS	400
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	400
4	4	CARAMELOS FRUTAS MASTICABLES	400
5	5	CHUPETES LOLY AMBROSOLI	0
6	6	FRUNA SURTIDA DONOFRIO	600
7	7	CHOCOLATE DOÑA PEPA FIELD	500
8	8	CHOCOLATE CUA CUA FIELD	500
9	9	MELLOWS FAMILIAR FIELD	400
10	10	WAFER CHOCOLATE FIELD	400

Combinación de las vistas para obtener el reporte final

```
-- Combinación de las vistas para el balance
SELECT v_UnidadesEntrantes.CodArticulo,
v_UnidadesEntrantes.DescripcionArticulo,
v_UnidadesEntrantes.Entradas,
v_UnidadesSalientes.Salidas
FROM v_UnidadesEntrantes
    INNER JOIN v_UnidadesSalientes
ON v_UnidadesEntrantes.CodArticulo =
    v_UnidadesSalientes.CodArticulo
ORDER BY v_UnidadesEntrantes.CodArticulo
GO
```

	CodArticulo	DescripcionArticulo	Entradas	Salidas
1	1	CARAMELOS BASTON VIENA ARCOR	0	400
2	2	CARAMELOS SURTIDO DE FRUTAS	1500	400
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	1250	400
4	4	CARAMELOS FRUTAS MASTICABLES	3000	400
5	5	CHUPETES LOLY AMBROSOLI	3000	0
6	6	FRUNA SURTIDA DONOFRIO	3000	600
7	7	CHOCOLATE DOÑA PEPA FIELD	3000	500
8	8	CHOCOLATE CUA CUA FIELD	3000	500
9	9	MELLOWS FAMILIAR FIELD	0	400
10	10	WAFER CHOCOLATE FIELD	3000	400

18.2. Ventajas del uso de las vistas

El uso de las vistas proporciona las siguientes ventajas:

- El usuario accede a la data importante o apropiada para él. Limita el acceso a datos sensibles.
- Oculta la complejidad del modelo de datos. Un join de múltiples tablas se convierte en un simple SELECT para el usuario.
- Desde el punto de vista del usuario, una vista es una "tabla" pues puede ejecutar en ella todas las operaciones de datos: SELECT, INSERT, UPDATE y DELETE.
- Debido a que una vista es un objeto de la base de datos, puede asignarle permisos de usuario. Esto es mucho más eficiente que colocar los mismos permisos sobre columnas individuales en una tabla.
- Los datos pueden exportarse desde una vista por medio de la utilidad **bcp**.

18.3. Averiguar cuál es la consulta que ejecuta una vista

Para averiguar cómo está definida una vista, puede ejecutar los siguientes procedimientos almacenados del sistema para consultar las siguientes tablas del catálogo de una base de datos:

Vista del sistema	Almacena	Procedimiento
sys.sysobjects	Nombre de la vista	sp_help <i>nombre_vista</i>
sys.sysdepends	Nombre de los objetos dependientes de la vista	sp_depends <i>nombre_vista</i>
sys.syscomments	Sentencia que definió la vista	sp_helptext <i>nombre_vista</i>
sys.syscolumns	Columnas definidas en la vista	sp_columns <i>nombre_vista</i>

Ejercicio 9.3: Consultando la metadata de una vista

```

USE QhatuPERU
GO

-- Consultado que objetos de tipo Vista contiene
-- la base de datos QhatuPERU
SELECT name FROM sys.sysobjects
WHERE type = 'V'
GO

```



Resultados		Mensajes
1		v_ListaPrecios
2		v_UnidadesEntrantes
3		v_UnidadesSalientes

```
-- Obteniendo las propiedades
-- de la vista v_ListaPrecios
sp_help v_ListaPrecios
GO
```

Resultados								
Mensajes								
	Name	Owner	Type	Created_datetime				
1	v_ListaPrecios	dbo	view	2013-05-16 14:53:37.080				
1	NomLinea	varchar	no	20		no	no	no
2	CodArticulo	int	no	4	10	0	(n/a)	(n/a)
3	DescripcionArticulo	varchar	no	40		no	no	no
4	Presentacion	varchar	no	30		yes	no	yes
5	PrecioProveedor	money	no	8	19	4	yes	(n/a)
Identity								
1	No identity column defined.	NULL	NULL	NULL				
RowGuidCol								
1	No rowguidcol column defined.							

```
-- Mostrando el código que ejecuta la vista
sp_helptext v_ListaPrecios
GO
```

Resultados		Mensajes
1		
2		CREATE VIEW v_ListaPrecios
3		AS
4		SELECT LINEA.NomLinea, ARTICULO.CodArticulo,
5		ARTICULO.DescripcionArticulo,
6		ARTICULO.Presentacion,
7		ARTICULO.PrecioProveedor
8		FROM LINEA INNER JOIN ARTICULO
9		ON LINEA.CodLinea = ARTICULO.CodLinea



```
-- Mostrando los objetos que dependen de la vista
sp_depends v_ListaPrecios
GO
```

	name	type	updated	selected	column
1	dbo.LINEA	user table	no	yes	CodLinea
2	dbo.LINEA	user table	no	yes	NomLinea
3	dbo.ARTICULO	user table	no	yes	CodArticulo
4	dbo.ARTICULO	user table	no	yes	CodLinea
5	dbo.ARTICULO	user table	no	yes	DescripcionArticulo
6	dbo.ARTICULO	user table	no	yes	Presentacion
7	dbo.ARTICULO	user table	no	yes	PrecioProveedor

19. MODIFICACIÓN Y ELIMINACIÓN DE UNA VISTA

A menudo, en respuesta a las demandas de los usuarios por información adicional, o a los cambios en la definición de la tabla subyacente, es necesario modificar la definición de una vista. Por ejemplo, si la tabla a la que una vista hace referencia se ha eliminado, los usuarios recibirán un mensaje del error cuando ellos intenten utilizar la vista. Se puede modificar una vista eliminándola y Luego recreándola, o ejecutando la sentencia ALTER VIEW.

19.1. Modificación de una vista – La instrucción ALTER VIEW

La sentencia ALTER VIEW cambia la definición de una vista permitiéndole retener los permisos para la vista. Esta sentencia está sujeta a las mismas restricciones que la sentencia CREATE VIEW. Si en vez de modificar la vista, la elimina y Luego la recrea, se verá obligado a recrear los permisos.

Sintaxis

```
ALTER VIEW nombre_vista [ ( lista_columnas ) ]
[ WITH ENCRYPTION ]
AS
nueva_sentencia_select
[ WITH CHECK OPTION ]
```

19.2. Eliminación de una vista – La instrucción DROP VIEW

La siguiente sentencia se utiliza para eliminar una vista.

Sintaxis

```
DROP VIEW nombre_vista
```

19.3. Ocultando la definición de una vista

El texto de la sentencia que define una vista se almacena en la tabla de sistema **syscomments**. Si al crear la vista utiliza la opción WITH ENCRYPTION, el texto se almacenará en dicha tabla en forma encriptada.

Recomendación

Antes de encriptar una vista, asegúrese que la definición de vista (el script) se guarde en un archivo. Para desencriptar el texto de una vista, debe eliminar y Luego recrear o modificar la vista utilizando la Sintaxis original.

Ejercicio 9.4: Uso de WITH ENCRYPTION

La siguiente instrucción crea la vista **v_Inventario** cuya definición se guarda encriptada.

```
USE QhatuPERU
GO

CREATE VIEW v_Inventario
WITH ENCRYPTION
AS
SELECT LINEA.NomLinea, ARTICULO.CodArticulo,
ARTICULO.DescripcionArticulo,
ARTICULO.Presentacion,
ARTICULO.stockActual, ARTICULO.stockMinimo
FROM LINEA INNER JOIN ARTICULO
ON LINEA.CodLinea = ARTICULO.CodLinea
GO

-- Consultando la vista
SELECT * FROM v_Inventario
GO
```

NomLinea	CodArticulo	DescripcionArticulo	Presentacion	stockActual	stockMinimo
1 GOLOSINAS	1	CARAMELOS BASTON VIENA ARCOR	PAQUETE 454 GR	200	50
2 GOLOSINAS	2	CARAMELOS SURTIDO DE FRUTAS	PAQUETE 450 GR	300	50
3 GOLOSINAS	3	CARAMELOS FRUTAS SURTIDA ARCOR	PAQUETE 520 GR	250	50
4 GOLOSINAS	4	CARAMELOS FRUTAS MASTICABLES	PAQUETE 454 GR	250	50
5 GOLOSINAS	5	CHUPETES LOLY AMBROSOLI	KILOGRAMO	150	100
6 GOLOSINAS	6	FRUNA SURTIDA DONOFRIO	PAQUETE X 24 UNIDADES	500	100
7 GOLOSINAS	7	CHOCOLATE DOÑA PEPA FIELD	PAQUETE X 6 UNIDADES	500	100
8 GOLOSINAS	8	CHOCOLATE CUA CUA FIELD	PAQUETE X 6 UNIDADES	500	100
9 GOLOSINAS	9	MELLOWS FAMILIAR FIELD	PAQUETE 454 GR	100	50
10 GOLOSINAS	10	WAFER CHOCOLATE FIELD	PAQUETE X 9 UNIDADES	900	100

Ahora, trataremos de averiguar cómo se ha definido la vista.

```
-- Revisando la definición de la vista
sp_helptext v_Inventario
GO
```

Se obtiene el siguiente mensaje:

El texto para el objeto v_inventario está cifrado.

19.4. Modificación de datos a través de vistas

Desde el punto de vista del usuario, una vista es una "tabla" ya que él puede ejecutar sobre la vista las sentencias SELECT, INSERT, UPDATE y DELETE. Estas sentencias ejecutadas sobre la vista afectan a las tablas dependientes de ella.

Deben tenerse en cuenta las siguientes consideraciones al momento de crear las vistas que luego se utilizarán para modificar los datos a través de ellas:

- La modificación no puede afectar a más de una de las tablas dependientes.
- Las vistas con columnas computadas deben ser de solo lectura ya que producen error cuando se trata de ejecutar modificaciones a través de ellas.
- Las columnas no nulas que no son referenciadas en la vista pueden producir error cuando se ejecuta una modificación a través de la vista.
- Si el SELECT asociado a la vista ha sido definido con una cláusula WHERE, el uso de la opción WITH CHECK OPTION al momento de crear la vista hará que las modificaciones que se hagan a través de ella respeten el criterio del WHERE.



Teniendo en cuenta todo lo anterior, mi recomendación es que las vistas deben utilizarse solo como medios de consulta (solo para ejecutarlos SELECT). Para modificar los datos es mejor utilizar procedimientos almacenados.

20. EJERCICIOS PROPUESTOS

Para los siguientes ejercicios debe utilizar las bases de datos RH y EDUCA. Los scripts para crearlas en su servidor los encontrará en el CD que acompaña al libro.

14. En la base de datos RH, crear una vista que muestre los siguientes datos:

- Código del empleado
- Nombre del empleado
- Departamento
- CarGO
- Sueldo
- Comisión, cero si es nulo
- Sueldo total (sueldo + comisión)

15. En la base de datos EDUCA, crear una vista que muestre los siguientes datos:

- Código del curso
- Nombre del curso
- Código del alumno
- Nombre del alumno
- Precio del curso
- PaGO a cuenta
- PaGO pendiente