Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Carrera: Licenciatura en Sistemas de Información

Cátedra: Base de Datos I

Profesor: Vallejos, Walter Oscar

Tema: Procedimientos y funciones almacenada

Año: 2023

Grupo 8

Integrantes:

- Alcaraz, Mauricio
- Cruz, Julian Luis
- Peloso, Nicolás Anibal

ÍNDICE:

CAPÍTULO I: INTRODUCCIÓN

Tema

El tema de este proyecto de estudio son los procedimientos y funciones en bases de datos relacionales, en donde se buscará comprender y aplicar de forma práctica estas herramientas dado un problema y modelo de datos específicos. El problema a investigarse trata sobre la base de datos de un consorcio en donde intervienen varias entidades y relaciones.

Planteamiento del problema

Al iniciar con este tema se nos dio como tarea realizar 3 procedimientos almacenados para 3 tablas distintas de entidades relacionadas con el dominio del problema y la creación de una función, que permita calcular la edad en base a la fecha de nacimiento de los administradores.
A raíz de eso nos surgió la pregunta: ¿Que es un procedimiento almacenado y que es una función y cómo funcionan en sql ?
para responder a esa pregunta tuvimos que buscar cual era el alcance de cada una y en qué se diferenciaban, para qué servían y cómo podríamos aplicarlas en nuestro trabajo, esperamos que estas funcionaran de forma similar a como son en otros lenguajes de programacion, asi como tambien ver las limitantes que tiene cada una y que es lo que pueden hacer las funciones que los procedimientos no y viceversa, saber si era posible utilizar una función dentro de un procedimiento almacenado o cómo influyen estos en el rendimiento de un sistema y que tan seguros pueden llegar a ser así como las posibilidades que tienen para poder reutilizarse.

Objetivo del Trabajo Práctico

El objetivo de este proyecto de estudio es poder comprender y aplicar de forma práctica los conceptos teóricos adquiridos que esperamos sean de gran utilidad tanto para nosotros como para los lectores del documento, para ello nos enfocamos en lo siguiente:

Objetivos Generales

- Brindar una visión general de los procedimientos y funciones en bases de datos SQL.
- Explorar cómo los procedimientos y funciones pueden mejorar la eficiencia y reutilización del código.
- Demostrar cómo diseñar, crear y ejecutar procedimientos y funciones en SQL.
- Comprender el uso de parámetros y variables en procedimientos y funciones.

- Identificar escenarios comunes en los que los procedimientos y funciones son especialmente útiles.

Objetivos Específicos

- Explicar la diferencia entre procedimientos y funciones en SQL.
- Detallar la sintaxis para crear procedimientos almacenados en diferentes sistemas de gestión de bases de datos.
- Mostrar ejemplos de procedimientos que realicen tareas comunes, como la actualización de registros o la generación de informes.
- Ilustrar cómo las funciones pueden utilizarse para realizar cálculos y devolver resultados.
- Abordar la seguridad y buenas prácticas en el uso de procedimientos y funciones en bases de datos.

CAPÍTULO II: MARCO CONCEPTUAL O REFERENCIAL

Procedimientos Almacenados:

Un procedimiento almacenado de SQL Server es un conjunto de instrucciones Transact-SQL a las que se les da un nombre, que se almacena en el servidor. Los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden:

- Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada.

- Contener instrucciones de programación que realicen operaciones en la base de datos. Entre otras, pueden contener llamadas a otros procedimientos.

- Devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos.

Ventajas de usar procedimientos almacenados

Tráfico de red reducido entre el cliente y el servidor

Los comandos de un procedimiento se ejecutan en un único lote de código. Esto puede reducir significativamente el tráfico de red entre el servidor y el cliente porque únicamente se envía a través de la red la llamada que va a ejecutar el procedimiento. Sin la encapsulación de código que proporciona un procedimiento, cada una de las líneas de código tendría que enviarse a través de la red.

Mayor seguridad

Varios usuarios y programas cliente pueden realizar operaciones en los objetos de base de datos subyacentes a través de un procedimiento, aunque los usuarios y los programas no tengan permisos directos sobre esos objetos subyacentes. El procedimiento controla qué procesos y actividades se llevan a cabo y protege los objetos de base de datos subyacentes. Esto elimina la necesidad de conceder permisos en cada nivel de objetos y simplifica los niveles de seguridad.

Al llamar a un procedimiento a través de la red, solo está visible la llamada que va a ejecutar el procedimiento. Por tanto, los usuarios malintencionados no pueden ver los nombres de los objetos de la base de datos y las tablas, insertar sus propias instrucciones Transact-SQL ni buscar datos críticos.

El uso de parámetros de procedimientos ayuda a protegerse contra ataques por inyección de código SQL. Dado que la entrada de parámetros se trata como un valor literal y no como código ejecutable, resulta más difícil para un atacante insertar un comando en las instrucciones Transact-SQL del procedimiento y poner en peligro la seguridad, los procedimientos pueden cifrarse, lo que ayuda a ofuscar el código fuente.

Reutilización del código

El código de cualquier operación de base de datos redundante resulta un candidato perfecto para la encapsulación de procedimientos. De este modo, se elimina la necesidad de escribir de nuevo el mismo código, se reducen las inconsistencias de código y se permite que cualquier usuario o aplicación que cuente con los permisos necesarios pueda acceder al código y ejecutarlo.

Mantenimiento más sencillo

Cuando las aplicaciones cliente llaman a procedimientos y mantienen las operaciones de base de datos en la capa de datos, solo deben actualizarse los cambios de los procesos en la base de datos subyacente. El nivel de aplicación permanece independiente y no tiene que tener conocimiento sobre los cambios realizados en los diseños, las relaciones o los procesos de la base de datos.

Rendimiento mejorado.

De forma predeterminada, un procedimiento se compila la primera vez que se ejecuta y crea un plan de ejecución que vuelve a usarse en posteriores ejecuciones.

Como el procesador de consultas no tiene que crear un nuevo plan, normalmente necesita menos tiempo para procesar el procedimiento.

Tipos de procedimientos almacenados

Definidos por el usuario

Un procedimiento definido por el usuario se puede crear en una base de datos definida por el usuario o en todas las bases de datos del sistema excepto en la base de datos Resource .

Temporales

Los procedimientos temporales son una forma de procedimientos definidos por el usuario. Los procedimientos temporales son iguales que los procedimientos permanentes salvo porque se almacenan en tempdb. Hay dos tipos de procedimientos temporales: locales y globales. Se diferencian entre sí por los nombres, la visibilidad y la disponibilidad. Los procedimientos temporales locales tienen como primer carácter de sus nombres un solo signo de número (#); solo son visibles en la conexión actual del usuario y se eliminan cuando se cierra la conexión. Los procedimientos temporales globales presentan dos signos de número (##) antes del nombre; son visibles para cualquier usuario después de su creación y se eliminan al final de la última sesión en la que se usa el procedimiento.

Sistema

Los procedimientos del sistema se incluyen con SQL Server. Están almacenados físicamente en la base de datos interna y oculta Resource y se muestran de forma lógica en el esquema sys de cada base de datos definida por el sistema y por el usuario. Dado que los procedimientos del sistema empiezan con el prefijo sp_ , es recomendable que no se utilice este prefijo cuando se asigne un nombre a los procedimientos definidos por el usuario.

Extendidos definidos por el usuario

Los procedimientos extendidos permiten crear rutinas externas en un lenguaje de programación, como C. Estos procedimientos son bibliotecas DLL que una instancia de SQL Server puede cargar y ejecutar de forma dinámica.

Estos se quitarán en una versión futura de SQL Server. No es recomendable utilizar esta característica en nuevos trabajos de desarrollo y modificar lo antes posible las aplicaciones que actualmente la utilizan.

Funciones:

Una función es un conjunto de sentencias que operan como una unidad lógica, al igual que las funciones de los lenguajes de programación, las funciones definidas

por el usuario de SQL Server son rutinas que aceptan parámetros, realizan una acción, como un cálculo complejo, y devuelven el resultado de esa acción como un valor. El valor devuelto puede ser un valor escalar único o un conjunto de resultados.

Ventajas de las funciones definidas por el usuario

Programación modular.
Puede crear la función una vez, almacenarla en la base de datos y llamarla desde el programa tantas veces como desee. Las funciones definidas por el usuario se pueden modificar, independientemente del código de origen del programa.

Ejecución más rápida.
Al igual que los procedimientos almacenados, las funciones definidas por el usuario de Transact-SQL reducen el costo de compilación del código de Transact-SQL almacenando los planes en la caché y reutilizándolos para ejecuciones repetidas. Esto significa que no es necesario volver a analizar y optimizar la función definida por el usuario con cada uso, lo que permite obtener tiempos de ejecución mucho más rápidos.

Reducción del tráfico de red.
Una operación que filtra datos basándose en restricciones complejas que no se puede expresar en una sola expresión escalar se puede expresar como una función. La función se puede invocar luego en la cláusula WHERE para reducir el número de filas que se envían al cliente.

Tipos de funciones

Funciones escalares
Las funciones escalares definidas por el usuario devuelven un único valor de datos del tipo definido en la cláusula RETURNS. En una función escalar insertada, el valor escalar es el resultado de una sola instrucción. Para una función escalar de varias instrucciones, el cuerpo de la función puede contener una serie de instrucciones de Transact-SQL que devuelven el único valor. El tipo devuelto puede ser de cualquier tipo de datos excepto text, ntext, image, cursor y timestamp.

Funciones con valores de tabla
Las funciones con valores de tabla definidas por el usuario devuelven un tipo de datos table. Las funciones insertada con valores de tabla no tienen cuerpo; la tabla es el conjunto de resultados de una sola instrucción SELECT.

Funciones del sistema
SQL Server proporciona numerosas funciones del sistema que se pueden usar para realizar diversas operaciones. No se pueden modificar.

Instrucciones válidas en una función

No todas las sentencias SQL son válidas dentro de una función. Entre los tipos de instrucciones válidos en una función se incluyen:

- Las instrucciones DECLARE se pueden usar para definir variables y cursores de datos locales de la función.
- La asignación de valores a objetos locales de la función, como el uso de SET para asignar valores a variables locales escalares y de tabla.
- Las operaciones de cursores que hacen referencia a cursores locales que están declarados, abiertos, cerrados y no asignados en la función. No se permiten las instrucciones FETCH que devuelven datos al cliente. Solo se permiten las instrucciones FETCH que asignan valores a variables locales mediante la cláusula INTO.
- Instrucciones de control de flujo excepto instrucciones TRY...CATCH.
- Instrucciones SELECT que contienen listas de selección con expresiones que asignan valores a las variables locales para la función.
- Instrucciones UPDATE, INSERT y DELETE que modifican las variables de tabla locales de la función.
- Instrucciones EXECUTE que llaman a un procedimiento almacenado extendido.

Los parámetros de entrada pueden ser de cualquier tipo, excepto timestamp, cursor y table. Las funciones definidas por el usuario no permiten parámetros de salida. NO es posible emplear en ellas funciones no deterministas (como getdate()) ni sentencias de modificación o 14 actualización de tablas o vistas. Si podemos emplear sentencias de asignación, de control de flujo (if), de modificación y eliminación de variables locales. SQL Server admite. Las funciones definidas por el usuario se crean con la instrucción "create function" y se eliminan con "drop function". Una función escalar retorna un único valor. Como todas las funciones, se crean con la instrucción "create function". Las funciones que retornan una tabla pueden emplearse en lugar de un "from" de una consulta.

Diferencias entre Funciones y Procedimientos almacenados

Las diferencias más generales entre procedimientos y funciones es que cada una es llamada de modo diferente y para propósitos distintos:

1. Un procedimiento no nos regresa un valor. En su lugar, es llamado con una declaración CALL para realizar una operación como modificar una tabla o procesar la recuperación de registros.
2. Una función es llamada dentro de una expresión y nos regresa un valor único directamente al que lo llama para ser utilizado en la expresión.
3. No se puede invocar una función con una instrucción CALL, ni puedes invocar un procedimiento en una expresión.
4. Normalmente, las funciones se utilizan para cálculos en los que normalmente se utilizan procedimientos para ejecutar la lógica de negocio.
5. Dentro de una función, no está permitido modificar una tabla que ya está siendo utilizada (para leer o escribir) por la sentencia que invocó la función o disparador.

La sintaxis para la creación de rutinas es algo diferente para procedimientos y funciones:

1) Los parámetros de procedimiento se pueden definir como sólo de entrada, sólo de salida o ambos. Esto significa que un procedimiento puede devolver valores al que los llama usando parámetros de salida. Se pueden acceder a estos valores en sentencias que siguen a la instrucción CALL. Las funciones sólo tienen parámetros de entrada. Como resultado, aunque ambos procedimientos y funciones pueden tener parámetros, la declaración de parámetros de procedimiento difiere de la de funciones.
2) Las funciones devuelven valores, por lo que debe haber una cláusula RETURNS en la definición de función para indicar el tipo de datos del valor que se regresara. Además, debe haber al menos una declaración RETURN dentro del cuerpo de la función para devolver un valor al que llama. RETURNS y RETURN no aparecen en las definiciones de los procedimientos.
    a) Para invocar un procedimiento almacenado, utilice la instrucción CALL. Para invocar una función almacenada, refiérase a ella en una expresión. La función devuelve un valor durante la evaluación de la expresión.
    b) Un procedimiento se invoca mediante una instrucción CALL y sólo puede devolver valores utilizando variables de salida. Una función puede ser llamada desde dentro de una sentencia como cualquier otra función (es decir, invocando el nombre de la función), y puede devolver un valor escalar.
    c) Especificar un parámetro como IN, OUT o INOUT sólo es válido para PROCEDIMIENTO. Para una FUNCIÓN, los parámetros siempre se consideran parámetros IN. Si no se da ninguna palabra clave antes de un nombre de parámetro, es un parámetro IN por defecto. Los parámetros para las funciones almacenadas no están precedidos por

IN, OUT o INOUT. Todos los parámetros de función se tratan como parámetros IN.

Para definir un procedimiento o una función almacenada, utilice CREATE PROCEDURE o CREATE FUNCTION respectivamente:

CREATE PROCEDURE proc_name ([parameters])
[characteristics]
routine_body

CREATE FUNCTION func_name ([parameters])
RETURNS data_type //diferente
[characteristics]
routine_body

Una extensión de MySQL para el procedimiento almacenado (no para las funciones) es que un procedimiento puede generar un conjunto de resultados, o incluso varios conjuntos de resultados, que el que llama procesa de la misma manera que el resultado de una sentencia SELECT. Sin embargo, el contenido de tales conjuntos de resultados no se puede utilizar directamente en la expresión.

Los procedimientos y funciones almacenados no comparten el mismo espacio de nombres. Es posible tener un procedimiento y una función con el mismo nombre en una base de datos.
En Procedimientos almacenados se puede utilizar SQL dinámico(instrucciones preparadas de SQL PREPARE, EXECUTE, DEALLOCATE **) pero no en funciones o disparadores.**

CAPÍTULO III: METODOLOGÍA SEGUIDA

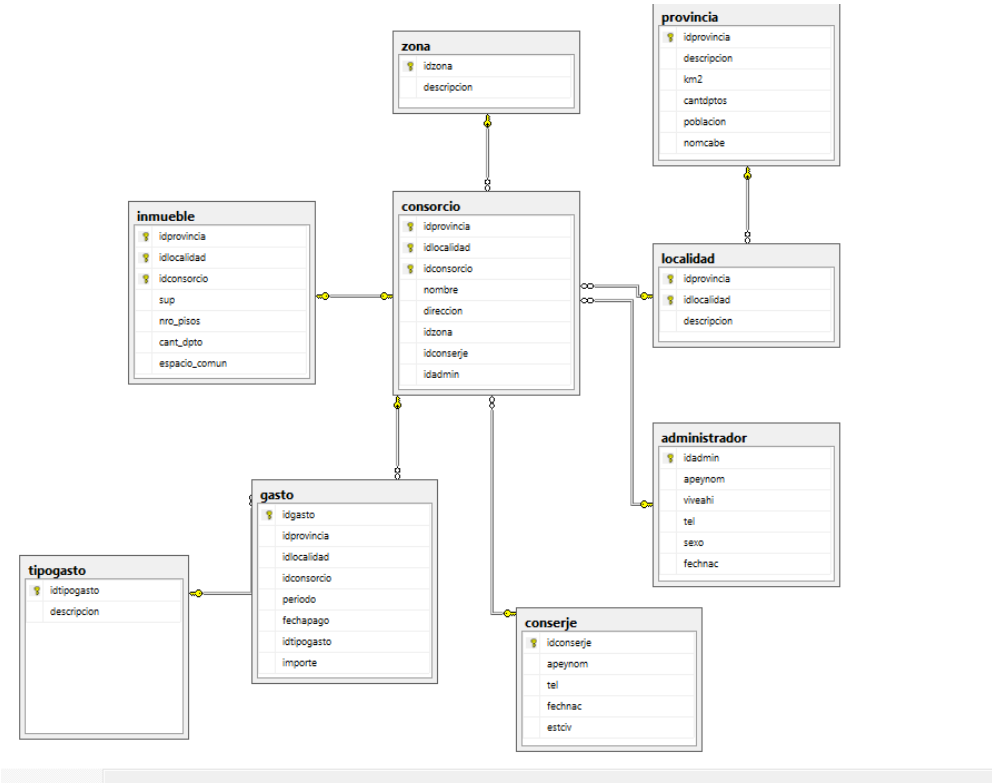
Herramientas utilizadas

Las herramientas que utilizamos para la realización del proyecto fueron las detalladas a continuación

- *WhatsApp*
- *GitHub*
- *SQL Server Management Studio*
- *Google Docs*

CAPÍTULO IV: DESARROLLO DEL TEMA / PRESENTACIÓN DE RESULTADOS

Imagen del diagrama de entidad relación de la base de datos utilizada para realizar el caso de estudio



Diccionario de Datos:

Descripción de las todas las entidades y sus relaciones

provincia: esta tabla contiene datos sobre las provincias en las que se pueden encontrar los consorcios

localidad: esta tabla contiene el nombre de las localidades de las distintas provincias, se relaciona con la tabla: provincia

zona: esta tabla contiene los nombres de las zonas donde pueden encontrarse los consorcios

conserje: esta tabla contiene los datos personales de los conserjes que trabajan en los distintos consorcios.

administrador: esta tabla contiene los datos personales de los distintos administradores de los distintos consorcios.

tipogasto: esta tabla almacena todos los tipos de gastos que se pueden dar en los consorcios

consorcio: esta tabla almacena los datos de todos los consorcios, se relaciona con las tablas: provincia, localidad, zona, conserje y admin

gasto: esta tabla contiene el importe de todos los gastos realizados por los distintos consorcios, se relaciona con las tablas: provincia, localidad, consorcio y tipogasto

Provincia			
Nombre	Tipo(longitud)	Acepta Valores Null	Descripcion
idprovincia	INT	NO	Clave primaria para identificar la provincia
descripcion	VARCHAR(50)	SI	Nombre de la provincia
km2	INT	SI	km2 que tiene la provincia
cantdptos	INT	SI	Cantidad de departamentos que tiene la provincia
poblacion	INT	SI	Numero de avitantes que tiene la poblacion
nomcabe	VARCHAR(50)	SI	Nombre de la capital de la provincia

Localidad			
Nombre	Tipo(longitud)	Acepta Valores Null	Descripcion
idprovincia	INT	NO	Clave foranea para identificar la provincia
idlocalidad	INT	NO	Clave primaria para identificar la localidad
descripcion	VARCHAR(50)	SI	Nombre de la localidad

Zona			
Nombre	Tipo(longitud)	Acepta Valores Null	Descripcion
idzona	INT	NO	Clave primaria para identificar la zona
descripcion	VARCHAR(50)	SI	Nombre de la zona

Conserje			
Nombre	Tipo(longitud)	Acepta Valores Null	Descripcion
idconserje	INT	NO	Clave primaria para identificar al conserje
apeynom	VARCHAR(50)	SI	Apellido y nombre del conserje
tel	VARCHAR(20)	SI	Telefono del conserje
fechnac	DATETIME	SI	fecha de nacimiento del conserje
estciv	VARCHAR(1)	NO	estado civil del conserje

Administrador			
Nombre	Tipo(longitud)	Acepta Valores Null	Descripcion
idadmin	INT	NO	Clave primaria para identificar al administrador
apeynom	VARCHAR(50)	SI	Apellido y Nombre del administrador
viveahi	VARCHAR(1)	NO	Indica si el administrador vive o no en la localidad
tel	VARCHAR(20)	SI	Telefono del adminstrador
sexo	VARCHAR(1)	NO	Sexo del administrador
fechnac	DATETIME	SI	fecha de nacimiento del administrador

TipoGasto			
Nombre	Tipo(longitud)	Acepta Valores Null	Descripcion
idtipogasto	INT	NO	Clave primaria para identificar el tipo de gasto
descripcion	VARCHAR(50)	SI	Nombre del tipo de gasto

Consorcio			
Nombre	Tipo(longitud)	Acepta Valores Null	Descripcion
idprovincia	INT	NO	Clave foranea para identificar la provincia
idlocalidad	INT	NO	Clave foranea para identificar la localidad
idconsorcio	INT	NO	Clave primaria para identificar el consorcio
nombre	VARCHAR(50)	SI	Nombre del consorcio
direccion	VARCHAR(250)	SI	Direccion del consorcio
idzona	INT	NO	Clave foranea para identificar la zona
idconserje	INT	NO	Clave foranea para identificar al conserje
idadmin	INT	NO	Clave foranea para identificar al admin

Gasto			
Nombre	Tipo(longitud)	Acepta Valores Null	Descripcion
idgasto	INT	NO	Clave primaria para identificar el gasto
idprovincia	INT	NO	Clave foranea para identificar la provincia
idlocalidad	INT	NO	Clave foranea para identificar la localidad
idconsorcio	INT	NO	Clave foranea para identificar el consorcio
periodo	INT	NO	Numero del perioro en que se realizo el gasto
fechapago	DATETIME	SI	fecha en la que se realizo el gasto
idtipogasto	INT	NO	Clave foranea para identificar el tipogasto
importe	DECIMAL(8,2)	SI	importe total del gasto

Restricciones

Nombre Entidad	Tipo de restriccion	Nombre de restriccion	Columna
provincia	PRIMARY KEY	PK__provinci__1AF96EE85C28B	idprovincia
localidad	PRIMARY KEY FOREIGN KEY	PK_localidad	idlocalidad, idprovincia
localidad	FOREIGN KEY	FK_localidad_pcia	idprovincia
zona	PRIMARY KEY	PK__zona__D75A19E2E6BBAE23	idzona
conserje	PRIMARY KEY	PK__conserje__095979F6436082	idconserje
conserje	CHECK	CK_estadocivil	estciv
administrador	PRIMARY KEY	PK__administ__5B93BD6E7A630	idadmin
administrador	CHECK	CK_habitante_viveahi	viveahi
administrador	CHECK	CK_sexo	sexo
tipogasto	PRIMARY KEY	PK__tipogast__8CBCB42109CAD6	idtipogasto
consorcio	PRIMARY KEY FOREIGN KEY	PK_consorcio	idlocalidad, idprovincia, idconsorcio
consorcio	FOREIGN KEY	FK_consorcio_pcia	idlocalidad, idprovincia
consorcio	FOREIGN KEY	FK_consorcio_zona	idzona
consorcio	FOREIGN KEY	FK_consorcio_conserje	idconserje
consorcio	FOREIGN KEY	FK_consorcio_admin	idadmin
gasto	PRIMARY KEY	PK_gasto	idgasto
gasto	FOREIGN KEY	FK_gasto_consorcio	idlocalidad, idprovincia, idconsorcio
gasto	FOREIGN KEY	FK_gasto_tipo	idtipogasto

CAPÍTULO V: CONCLUSIONES

A lo largo de este proyecto de estudio hemos aprendido que los procedimientos y funciones ofrecen ventajas significativas en la administración de bases de datos como son la capacidad de mejorar la eficiencia y la reutilización del código.
Los procedimientos almacenados permiten agrupar tareas comunes en un solo bloque de código, lo que facilita la ejecución y el mantenimiento de las operaciones en la base de datos.
Además, hemos comprendido que la utilización de parámetros y variables en procedimientos y funciones aumenta la flexibilidad y adaptabilidad de estas herramientas.
Por lo tanto, concluímos que el trabajo con los procedimientos y funciones que implica como diseñar, crear y ejecutar estos elementos son unas herramientas esenciales para cualquier profesional de bases de datos. Este proyecto nos ha permitido adquirir conocimientos valiosos en esta área y aplicarlos de manera práctica al problema dado.

Otras observaciones: Aunque puede parecer que los Procedimientos Almacenados pueden hacer lo mismo o hasta más que una Función, cada una tiene sus propias limitaciones y usos.
Los usos típicos para Procedimientos Almacenados es la validación de datos, integrados dentro de la estructura de la base de datos , la "encapsulación" de un API para un proceso complejo que podría requerir la ejecución de varias consultas SQL, tales como la manipulación de un conjunto de datos enormes para producir un resultado resumido y también pueden ser usados para el control de gestión de operaciones.
Mientras que para las Funciones se usan cuando se necesita recibir un único valor simple que se obtiene de una operación recurrente.

VI. BIBLIOGRAFÍA.

- Softwero. MySQL: Procedimientos Almacenados VS Funciones http://www.softwero.com/2017/07/mysql-procedimientos-almacenados-vs-funciones.html#:~:text=b

- Microsoft Learn. Crear procedimientos almacenados y funciones definidas por el usuario https://learn.microsoft.com/es-es/training/modules/create-stored-procedures-table-valued-functions/

- Microsoft Learn. Procedimientos almacenados (motor de base de datos) https://learn.microsoft.com/es-es/sql/relational-databases/stored-procedures/stored-procedures-database-engine?view=sql-server-ver16

- Microsoft Learn. Funciones definidas por el usuario https://learn.microsoft.com/es-es/sql/relational-databases/user-defined-functions/user-defined-functions?view=sql-server-ver16