Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Cátedra: Base de Datos I

Proyecto de estudio

Procedimientos y funciones almacenadas

Primera Entrega

INTEGRANTES:

- Andrada Soraire Francisco Sebastián. DNI:34.380.389
- Ayala Adrian de Jesus. DNI:41.843.499
- Resoagli Gaston Alejandro. DNI:40.047.945

Año 2023

Índice

1 INTRODUCCIÓN

En el mundo de la gestión de bases de datos y desarrollo de sistemas, los procedimientos y funciones almacenadas se erigen como pilares fundamentales. Estos componentes permiten la creación de bloques de código reutilizable que operan en el corazón mismo de las bases de datos relacionales. A medida que el volumen y la complejidad de los sistemas de información continúan creciendo, la comprensión y optimización de estos elementos se vuelve imperativa.

Este trabajo de investigación se adentra en el vasto universo de los procedimientos y funciones almacenadas, explorando sus características, ventajas, y aplicaciones en el contexto de sistemas de gestión de bases de datos modernos.

1.1 Propósito de la Investigación

El estudio detallado de los Procedimientos y Funciones Almacenadas tiene como objetivo principal comprender y optimizar uno de los pilares esenciales en el desarrollo de sistemas de gestión de bases de datos. Estos elementos proporcionan un medio eficaz para encapsular lógica de negocio compleja dentro de la base de datos misma, lo que resulta en mejoras significativas en la eficiencia, mantenibilidad y seguridad de los sistemas de información.

1.2 Razón de Realización

La realización de esta investigación se justifica por la creciente importancia de los procedimientos y funciones almacenadas en el panorama actual de la tecnología de bases de datos. A medida que las aplicaciones y sistemas se vuelven cada vez más complejos, la necesidad de optimizar el rendimiento y la gestión de datos se convierte en un imperativo. Los procedimientos y funciones almacenadas representan un recurso invaluable para lograr estos objetivos, al permitir la ejecución de operaciones sofisticadas directamente en la base de datos, reduciendo la necesidad de transmisión de datos entre el servidor y la aplicación.

Esta investigación busca proporcionar una comprensión profunda y práctica de estos elementos, mostrando cómo pueden aplicarse de manera efectiva en situaciones reales para mejorar la eficiencia operativa, la escalabilidad y la seguridad de los sistemas de gestión de bases de datos.

2 MARCO CONCEPTUAL

2.2 Definición y Naturaleza

Los Procedimientos y Funciones Almacenadas son componentes esenciales en el ámbito de la gestión de bases de datos relacionales. Se trata de conjuntos de instrucciones SQL encapsuladas que pueden ser invocadas desde aplicaciones o desde otras consultas SQL. Los procedimientos almacenan secuencias de comandos que pueden realizar operaciones complejas y devolver resultados, mientras que las funciones retornan un valor específico.

2.3 Ventajas y Beneficios

Reutilización de Código: Permiten la creación de bloques de código que pueden ser utilizados repetidamente en distintas partes de una aplicación o sistema, reduciendo la redundancia y mejorando la consistencia del código.

Optimización de Consultas y Transacciones: Al ejecutarse en el servidor de base de datos, minimizan la necesidad de transmitir grandes cantidades de datos entre la aplicación y la base de datos, lo que resulta en un rendimiento más eficiente.

Seguridad y Control de Acceso: Facilitan la implementación de políticas de seguridad al limitar el acceso directo a ciertas operaciones en la base de datos y permitir la ejecución controlada a través de procedimientos y funciones.

Mantenibilidad y Escalabilidad: Al organizar la lógica de negocio en bloques de código separados, facilitan la identificación y corrección de errores, así como la expansión y adaptación de la funcionalidad de la aplicación a medida que evoluciona.

2.3 Contexto de Aplicación

Los procedimientos y funciones almacenadas encuentran aplicaciones en una amplia gama de escenarios, desde sistemas de gestión de inventarios hasta sistemas de administración financiera. Son especialmente valiosos en entornos empresariales donde la eficiencia, la seguridad y el control de acceso a la información son críticos.

Es esencial tener en cuenta las buenas prácticas y estándares de codificación al desarrollar procedimientos y funciones almacenadas, así como considerar las peculiaridades y restricciones específicas de los sistemas de gestión de bases de datos utilizados (por ejemplo, MySQL, PostgreSQL, Oracle, etc.).

3 METODOLOGÍA

3.1 Definición de Objetivos y Preguntas de Investigación

Establecimos los objetivos de investigación, incluyendo los aspectos específicos que se abordaran, como el diseño, optimización y aplicaciones practicas de procedimientos y funciones almacenadas.

3.2 Revisión Bibliográfica

Realizamos una revisión preliminar de fuentes bibliográficas para familiarizar el tema y obtener una visión sobre los conceptos clave y tendencias en el campo de los procedimientos y funciones almacenadas.

3.3 Recopilación de datos y Análisis de la información

Recopilamos datos relevantes a partir de fuentes primarias (documentación oficial, manuales de usuario) y secundaria (libros, artículos, tutoriales) que aborden el tema dado.
Por otro lado Examinamos la información recopilada para identificar enfoques comunes en el diseño, implementación y optimización sobre el tema a estudiar.

3.4 Pruebas

Implementamos los procedimientos en el entorno proporcionado por la catedra de la materia para evaluar si cumple con la consigna dada en el trabajo.

3.5 Herramientas utilizadas

Sistema de Gestión de Bases de Datos (DBMS): SQL Server.

Entorno de Desarrollo Integrado (IDE) para SQL: Management Studio (SSMS), DBeaver.

Herramientas de Control de Versiones: GitHub.

Herramientas de Comunicación: Discord, WhatsApp.

Navegadores Web y Motores de Búsqueda: Utilizados para acceder a gran parte de la información utilizada, como documentación oficial, tutoriales y artículos académicos.

4 PRESENTACIÓN DE RESULTADOS

4.1 Desarrollo

Como adelantamos en la introduccion de esta investigacion, en el mundo de las bases de datos relacionales, la utilización de funciones y procedimientos almacenados desempeña un papel fundamental en la optimización y gestión eficiente de datos. SQL, como lenguaje de consulta estándar, ofrece estas dos herramientas que, aunque comparten ciertas similitudes, poseen características distintivas que las distinguen y hacen adecuadas para diferentes propósitos.

Diferencias entre funciones y procedimientos almacenados:

1. **Valor de retorno:**
    - **Funciones:** Devuelven un único valor como resultado. Pueden ser utilizadas en expresiones SQL.
    - **Procedimientos almacenados:** No tienen un valor de retorno específico, ya que su objetivo principal es realizar una serie de acciones.
2. **Uso en consultas:**
    - **Funciones:** Pueden ser empleadas en cualquier lugar donde se utilice una expresión SQL, como en la cláusula SELECT.
    - **Procedimientos almacenados:** Se ejecutan como unidades completas y no pueden ser utilizados directamente en una consulta.
3. **Parámetros de salida:**
    - **Funciones:** Pueden tener parámetros de salida, pero el valor devuelto es la principal forma de comunicación con el entorno externo.
    - **Procedimientos almacenados:** Pueden tener parámetros de salida, diseñados para transmitir información al entorno externo.

Uso de funciones en SQL:

Las funciones en SQL son esenciales para realizar cálculos y manipulaciones de datos de manera eficiente. Algunos casos de aplicación comunes de funciones integradas en SQL incluyen:

1. **Operaciones matemáticas:**

- SELECT SUM(columna) AS Total FROM tabla;

- **Manipulación de cadenas:**

- SELECT CONCAT(nombre, ' ', apellido) AS NombreCompleto FROM empleados;

- **Obtención de información de fecha y hora:**

3. SELECT YEAR(fecha_nacimiento) AS AnioNacimiento FROM personas;

Limitaciones:

Ambos, funciones y procedimientos almacenados, tienen limitaciones que deben ser consideradas:

1. **Reusabilidad:**
    - **Funciones:** Son más fácilmente reutilizables en consultas.
    - **Procedimientos almacenados:** Están diseñados para realizar acciones específicas y pueden ser menos flexibles en términos de reutilización.
2. **Manipulación de transacciones:**
    - **Funciones:** Limitadas en cuanto a la manipulación de transacciones.
    - **Procedimientos almacenados:** Pueden contener transacciones más complejas.
3. **Modificaciones de datos:**
    - **Funciones:** Restringidas en cuanto a la modificación de datos en la base de datos.
    - **Procedimientos almacenados:** Pueden realizar modificaciones de datos de manera más directa.

En resumen, mientras que las funciones están diseñadas principalmente para devolver valores y ser utilizadas en consultas, los procedimientos almacenados están orientados a realizar acciones y manipular datos de manera más extensa.

4.2Diccionario de datos

Tabla Provincia				
Nombre del campo	Tipo de Dato	Formato del dato	Tamaño del campo	Descripcion
Idprovincia	INT		8	Clave primaria que identifica el tipo de provincia
descripcion	Varchar		50	Contiene el nombre de la Provincia
km2	INT		8	Contiene el numero de kilometros cuadrados total del terreno de la provincia
cantdptos	INT		8	Contiene la cantidad de departamentos de la provincia
poblacion	INT		8	Contiene el numero de la poblacion total de la provincia
nomcabe	Varchar		50	Nombre de la capital de la provincia

Tabla Localidad				
Nombre del campo	Tipo de Dato	Formato del dato	Tamaño del campo	Descripcion
Idprovincia	INT		8	Clave foranea cuyo valor Identifica el tipo de provincia que corresponde
idlocalidad	INT		8	Clave primeria que contiene el valor para identificar el departamento
descripcion	Varchar		50	Contiene el nombre del departamento

Nombre del campo	Tipo de Dato	Formato del dato	Tamaño del campo	Descripcion
idzona	INT		8	Clave primaria que identifica una zona especifica del departamento
descripcion	Varchar		50 b	Contiene el nombre de la zona que corresponde al campo "idzona"

Tabla Conserje				
Nombre del campo	Tipo de Dato	Formato del dato	Tamaño del campo	Descripcion
idconserje	Int		8	Clave primaria que identifica al conserje correspondiente
apeynom	Varchar		50	Contiene el apellido y nombre del conserje
tel	Varchar		20	Contiene el numero de telefono del conserje
fechanac	Datetime	YYYY-MM-DD hh:mm:ss[.nnn]	8	Contiene la fecha de nacimiento del conserje
estciv	Varchar		1	Contiene el caracter que define el estado civil del conserje

Tabla Administrador				
Nombre del campo	Tipo de Dato	Formato del dato	Tamaño del campo	Descripcion
idadmin	INT		8	Clave primaria que identifica al administrador/ra
apeynom	Varchar		50	Contiene el apellido y nombre de dicho administrador/ra
viveahi	Varchar		1	Contiene el caracter que afirma si el administrador/ra vive en esa localidad o no
tel	Varchar		20	Contiene el numero de contacto del administrador/ra
sexo	Varchar		1	Contiene el caracter que define si es del sexo masculino o femenino

Tabla Tipogasto				
Nombre del campo	Tipo de Dato	Formato del dato	Tamaño del campo	Descripcion
idtipogasto	INT		8	Clave primaria que identifica que tipo de gasto
descripcion	Varchar		50	Define el nombre del tipo de gasto

Tabla Consorcio				
Nombre del campo	Tipo de Dato	Formato del dato	Tamaño del campo	Descripcion

idprovincia	INT		8	Clave foranea cuyo valor identifica la provincia que corresponde
idlocalidad	INT		8	Clave foranea cuyo valor identifica la localidad que corresponde
idconsorcio	INT		8	Clave primaria cuyo valor identifica al consorcio
nombre	Varchar		50	Contiene el nombre del consorcio
direccion	Varchar		250	Contiene la direccion completa de donde esta ubicado el consorcio
idzona	INT		8	Clave foranea cuyo valor identifica a la zona que corresponde
idconserje	INT		8	Clave foranea cuyo valor identifica al conserje que trabaja en el consorcio
idadmin	INT		8	Clave foranea cuyo valor identifica al administrador del consorcio

Tabla Gasto				
Nombre del campo	Tipo de Dato	Formato del dato	Tamaño del campo	Descripcion
idgasto	INT		8	Clave primaria cuyo valor identifica el gasto
idprovincia	INT		8	Clave foranea cuyo valor identifica a la provincia que corresponde
idlocalidad	INT		8	Clave foranea cuyo valor identifica la localidad que corresponde
idconsorcio	INT		8	Clave foranea cuyo valor identifica al consorcio
periodo	INT		8	Contiene el mes donde se realizo dicho gasto
fechapago	Datetime	YYYY-MM-DD hh:mm:ss[.nnn]	8	Contiene la fecha en la cual se realizo el pago
idtipogasto	INT		8	Clave foranea cuyo valor identifica el tipo de gasto correspondiente
importe	Decimal	(8 , 2)	5-17	Contiene el monto con decimales del importe del gasto

Tabla Inmueble				
Nombre del campo	Tipo de Dato	Formato del dato	Tamaño del campo	Descripcion
idprovincia	INT		8	Clave foranea cuyo valor identifica a la provincia que corresponde
idlocalidad	INT		8	Clave foranea cuyo valor identifica la localidad que corresponde
idconsorcio	INT		8	Clave foranea cuyo valor identifica al consorcio
sup	Decimal	(6 , 2)	5-17	Contiene la superficie del terreno del inmueble
nro_pisos	INT		8	Contiene el numero de pisos del inmueble
cant_dpto	INT		8	Contiene el numero de departamentos del inmueble
espacio_comun	INT		8	Define si el inmueble contiene un espacio en comun para los departamentos

4.3 Scripts de SQL

4.4 Funciones de la tabla administrador

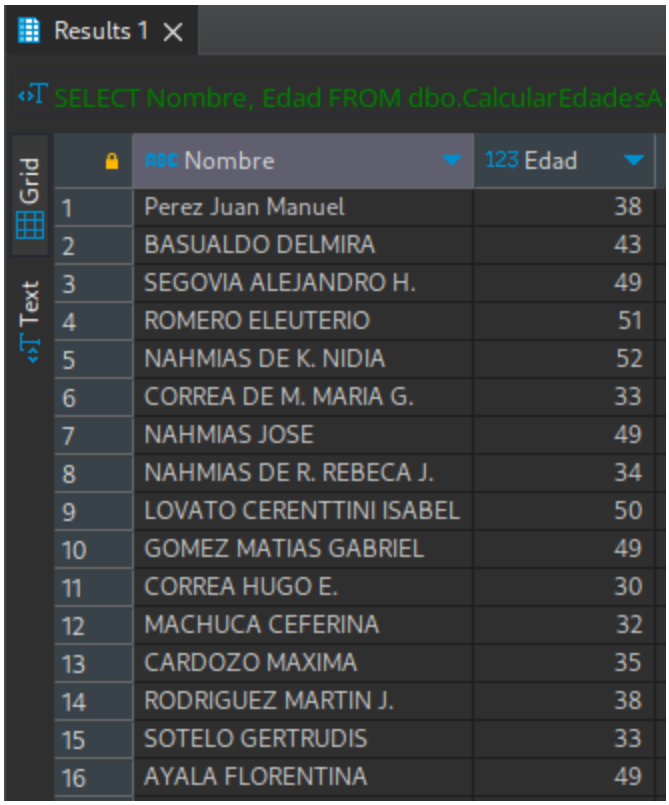
```sql
--Funcion para insertar un nuevo registro a la tabla administrador
CREATE PROCEDURE InsertarAdministrador
(
    @apeynom varchar(50),
    @viveahi varchar(1),
    @tel varchar(20),
    @sexo varchar(1),
    @fechnac datetime
)
AS
BEGIN
    INSERT INTO base_consorcio.dbo.administrador (apeynom, viveahi, tel, sexo, fechnac)
    VALUES (@apeynom, @viveahi, @tel, @sexo, @fechnac);
END;
```

```sql
--Funcion para eliminar un registro de la tabla administrador
CREATE PROCEDURE EliminarAdministrador
(
    @idadmin int
)
AS
BEGIN
    DELETE FROM base_consorcio.dbo.administrador
    WHERE idadmin = @idadmin;
END;
```

```sql
--Funcion para modificar un registro en la tabla administrador
CREATE PROCEDURE ModificarAdministrador
(
    @idadmin int,
    @apeynom varchar(50),
    @viveahi varchar(1),
    @tel varchar(20),
    @sexo varchar(1),
    @fechnac datetime
)
AS
BEGIN
    UPDATE base_consorcio.dbo.administrador
    SET
        apeynom = @apeynom,
        viveahi = @viveahi,
        tel = @tel,
        sexo = @sexo,
        fechnac = @fechnac
    WHERE idadmin = @idadmin;
END;
```

```sql
--Funcion para calcular la edad de los administradores.
CREATE FUNCTION CalcularEdadesAdministradores()
RETURNS TABLE
AS
RETURN (
    SELECT apeynom AS Nombre,
           DATEDIFF(YEAR, fechnac, GETDATE()) AS Edad
    FROM base_consorcio.dbo.administrador
);
```

A continuación se muestra una captura de pantalla con algunos de los resultados obtenidos con la ejecución de la función: CalcularEdadesAdministradores



Results 1

SELECT Nombre, Edad FROM dbo.CalcularEdadesA

Grid | Text

	Nombre	Edad
1	Perez Juan Manuel	38
2	BASUALDO DELMIRA	43
3	SEGOVIA ALEJANDRO H.	49
4	ROMERO ELEUTERIO	51
5	NAHMIAS DE K. NIDIA	52
6	CORREA DE M. MARIA G.	33
7	NAHMIAS JOSE	49
8	NAHMIAS DE R. REBECA J.	34
9	LOVATO CERENTTINI ISABEL	50
10	GOMEZ MATIAS GABRIEL	49
11	CORREA HUGO E.	30
12	MACHUCA CEFERINA	32
13	CARDOZO MAXIMA	35
14	RODRIGUEZ MARTIN J.	38
15	SOTELO GERTRUDIS	33
16	AYALA FLORENTINA	49

```sql
--Pruebas
--Insertamos administradores usando el procedimiento almacenado
EXEC InsertarAdministrador 'LOPEZ JUAN CARLOS', 'S', 3794222222, 'M', '19920828'
EXEC InsertarAdministrador 'RAMIREZ JULIA', 'N', 3794448899, 'F', '19910521'
EXEC InsertarAdministrador 'TUCKSON FACUNDO', 'S', 3794123222, 'F', '19900121'

--Insertamos usando sentencias INSERT
Insert into administrador(apeynom,viveahi,tel,sexo,fechnac) values ('PEREZ EZEQUIEL', 'N', '3794112233', 'M', '19850218')
Insert into administrador(apeynom,viveahi,tel,sexo,fechnac) values ('GIORNO LUCIA', 'N', '3794128888', 'F', '19881009')

--Modificamos el primer registro insertado en las pruebas
EXEC ModificarAdministrador 349, 'LOPEZ JUAN CARLOS', 'S', 3794420111, 'M', '19920828'

--Eliminamos el registro de RAMIREZ JULIA
EXEC EliminarAdministrador 355

--Probamos la funcion del calculo de edades
SELECT Nombre, Edad
FROM dbo.CalcularEdadesAdministradores();
```

4.5 Funciones de tabla consorcio

```sql
--Funcion para insertar un nuevo registro a la tabla consorcio
CREATE PROCEDURE InsertarConsorcio
(
    @idprovincia int,
    @idlocalidad int,
    @idconsorcio int,
    @nombre varchar(50),
    @direccion varchar(250),
    @idzona int,
    @idconserje int,
    @idadmin int
)
AS
BEGIN
    INSERT INTO base_consorcio.dbo.consorcio (idprovincia, idlocalidad, idconsorcio, nombre, direccion, idzona, idconserje, idadmin)
    VALUES (@idprovincia, @idlocalidad, @idconsorcio, @nombre, @direccion, @idzona, @idconserje, @idadmin);
END;
```

```sql
--Funcion para eliminar un registro de la tabla consorcio
CREATE PROCEDURE EliminarConsorcio
(
    @idprovincia int,
    @idlocalidad int,
    @idconsorcio int
)
AS
BEGIN
    DELETE FROM base_consorcio.dbo.consorcio
    WHERE idprovincia = @idprovincia
    AND idlocalidad = @idlocalidad
    AND idconsorcio = @idconsorcio;
END;
```

```sql
--Funcion para modificar un registro en la tabla consorcio
CREATE PROCEDURE ModificarConsorcio
(
    @idprovincia int,
    @idlocalidad int,
    @idconsorcio int,
    @nombre varchar(50),
    @direccion varchar(250),
    @idzona int,
    @idconserje int,
    @idadmin int
)
AS
BEGIN
    UPDATE base_consorcio.dbo.consorcio
    SET nombre = @nombre,
        direccion = @direccion,
        idzona = @idzona,
        idconserje = @idconserje,
        idadmin = @idadmin
    WHERE idprovincia = @idprovincia
    AND idlocalidad = @idlocalidad
    AND idconsorcio = @idconsorcio;
END;
```

```
--Pruebas.
--Insertamos 2 registros usando los procedimientos almacenados
EXEC InsertarConsorcio 7, 7, 500, 'Edificio Pepega 1', 'Av. Poggers 1522', 1, 1, 349
EXEC InsertarConsorcio 7, 7, 501, 'Edificio Jupiter 1', '9 de Julio 2002', 1, 1, 349

--Aqui insertamos usando sentencias insert
INSERT INTO consorcio(idprovincia,idlocalidad,idconsorcio, Nombre,direccion,idzona,idconserje,idadmin)
    VALUES (7, 7, 502, 'Edificio Urano I', 'Jujuy 998', 1, 2, 356)
INSERT INTO consorcio(idprovincia,idlocalidad,idconsorcio, Nombre,direccion,idzona,idconserje,idadmin)
    VALUES (7, 7, 503, 'Edificio Marte II', '25 de Mayo 2233', 1, 2, 356)

--Modificamos el registro el nombre y direccion del primer registro insertado
EXEC ModificarConsorcio 7, 7, 500, 'Edificio Neptuno 1', 'Av Poggers 1522', 1, 1, 349
--Acontinuacion eliminamos el Edificio Jupiter 1
EXEC EliminarConsorcio 7, 7, 501
```

4.6 Funciones tabla Gasto

```
--Funcion para insertar un nuevo registro a la tabla gasto
CREATE PROCEDURE InsertarGasto
(
    @idprovincia int,
    @idlocalidad int,
    @idconsorcio int,
    @periodo int,
    @fechapago datetime,
    @idtipogasto int,
    @importe decimal(8,2)
)
AS
BEGIN
    INSERT INTO base_consorcio.dbo.gasto (idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (@idprovincia, @idlocalidad, @idconsorcio, @periodo, @fechapago, @idtipogasto, @importe);
END;
```

```
--Funcion para eliminar un registro de la tabla consorcio
CREATE PROCEDURE EliminarGasto
(
    @idgasto int
)
AS
BEGIN
    DELETE FROM base_consorcio.dbo.gasto
    WHERE idgasto = @idgasto;
END;
```

```sql
--Funcion para modificar un registro en la tabla consorcio
CREATE PROCEDURE ModificarGasto
(
    @idgasto int,
    @idprovincia int,
    @idlocalidad int,
    @idconsorcio int,
    @periodo int,
    @fechapago datetime,
    @idtipogasto int,
    @importe decimal(8,2)
)
AS
BEGIN
    UPDATE base_consorcio.dbo.gasto
    SET
        idprovincia = @idprovincia,
        idlocalidad = @idlocalidad,
        idconsorcio = @idconsorcio,
        periodo = @periodo,
        fechapago = @fechapago,
        idtipogasto = @idtipogasto,
        importe = @importe
    WHERE idgasto = @idgasto;
END;
```

```
--Pruebas
--Insertamos gasto usando un procedimiento almacenado
EXEC InsertarGasto 7, 7,500, 1,'20230101', 5, 500.50;
EXEC InsertarGasto 7, 7,500, 2,'20230211', 3, 48522.99;
EXEC InsertarGasto 7, 7,500, 1,'20230310', 3, 62573.65;
EXEC InsertarGasto 7, 7,500, 1,'20230405', 2, 91137.50;
EXEC InsertarGasto 7, 7,500, 1,'20230509', 2, 3031.15;

--Aqui insertamos nuevos registros usando sentencias INSERT
INSERT INTO gasto (idprovincia,idlocalidad,idconsorcio,periodo,fechapago,idtipogasto,importe)
    VALUES (7,7,502,6,'20230616',5,608.97)
INSERT INTO gasto (idprovincia,idlocalidad,idconsorcio,periodo,fechapago,idtipogasto,importe)
    VALUES (7,7,503,3,'20230311',3,48026.65)

--Modificamos el ultimo registro gasto insertado
EXEC ModificarGasto 16011, 7, 7, 500, 1, '20230511', 2, 4031.15

--Eliminamos el primer registro insertado en las pruebas
EXEC EliminarGasto 16006
```

La consulta de los script están disponibles en el repositorio: https://github.com/GastonResoagli/Data-base

4.7 Aplicación de otros proyectos

4.8 Manejo de permisos a nivel de usuarios

El manejo de permisos a nivel de usuarios en bases de datos es un componente esencial en la gestión y seguridad de la información almacenada. En el entorno dinámico y cada vez más interconectado de las bases de datos, es crucial garantizar que solo usuarios autorizados tengan acceso a la información relevante, mientras se protege contra posibles amenazas y se cumple con regulaciones de privacidad. La asignación de permisos a usuarios en bases de datos SQL no se trata simplemente de conceder o negar acceso; implica una cuidadosa planificación y administración de privilegios para garantizar la integridad, confidencialidad y disponibilidad de los datos. Los sistemas de gestión de bases de datos (DBMS), como MySQL, PostgreSQL, SQL Server y Oracle, ofrecen una variedad de herramientas y mecanismos para implementar un control granular sobre las acciones que los usuarios pueden realizar en la base de datos.

4.9 Vistas y vistas indexadas

Las vistas son consultas predefinidas almacenadas en la base de datos que actúan como tablas virtuales. Estas consultas, definidas por el usuario, pueden combinar datos de varias tablas, aplicar funciones de agregación o filtrar información específica. Las vistas no almacenan datos físicamente, sino que ofrecen una perspectiva lógica y simplificada de los datos subyacentes.

Ventajas de Utilizar Vistas:

- Abstracción de Complejidad: Permite a los usuarios trabajar con conjuntos de datos de manera más comprensible y sin tener que preocuparse por la complejidad de las relaciones entre tablas.
- Seguridad de Datos: Controla el acceso a datos sensibles, ya que los usuarios pueden acceder solo a las columnas específicas que se han incluido en la vista.
- Reutilización de Consultas: Facilita la reutilización de consultas comunes, reduciendo así la redundancia y mejorando la eficiencia del desarrollo.

Beneficios de las Vistas Indexadas:

- Mejora de Rendimiento: Al aplicar índices a los resultados de las vistas, se acelera la ejecución de consultas, especialmente aquellas que involucran operaciones de filtrado y ordenación.
- Optimización de Consultas Complejas: Las vistas indexadas son particularmente útiles cuando se trabajan con consultas complejas que involucran un conjunto extenso de datos.
- Eficiencia en Aplicaciones de BI (Business Intelligence): Al agilizar la recuperación de datos, las vistas indexadas son valiosas en entornos donde la velocidad de consulta es esencial, como en aplicaciones de Business Intelligence.

5 CONCLUSIÓN

En el transcurso de esta investigación, se han explorado los Procedimientos y Funciones almacenadas, elementos fundamentales en el ámbito de la gestión de bases de datos. Estos componentes, que encapsulan lógica de negocio compleja directamente al núcleo de la base de datos, han demostrado ser pilares indispensables para la eficiencia, mantenibilidad y seguridad de los sistemas de información modernos.

A lo largo del trabajo hemos notado la serie de ventajas notables que presentan los Procedimientos y funciones almacenadas. Desde la reutilización de código hasta la optimización de consultas y transacciones, estas herramientas se han erigido como catalizadores para la eficiencia operativa y la escalabilidad de sistemas complejos.
Su capacidad para mejorar la seguridad y el control de acceso a la información, así como su papel vital en la reducción de la carga de comunicación entre la aplicación y la base de datos, han destacado su relevancia en el panorama actual de la tecnología de bases de datos.

En un mundo donde la gestión eficaz de la información es crucial, los Procedimientos y Funciones Almacenadas emergen como herramientas esenciales, con un potencial transformador en la forma en que abordamos el diseño y desarrollo de sistemas de gestión de bases de datos. Su comprensión y aplicación adecuadas representan un paso adelante significativo hacia la creación de soluciones tecnológicas más eficientes y seguras.

6 BIBLIOGRAFÍA

"SQL in 10 Minutes, Sams Teach Yourself" por Ben Forta.

- Editorial: Sams Publishing
- Año de Publicación: 2012

"MySQL 8 Cookbook" por Karthik Appigatla, Alex Kuznetsov.

- Editorial: Packt Publishing
- Año de Publicación: 2020