Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Licenciatura en Sistemas

Base de Datos I

Vistas y Vistas Indexadas

Alumnos G5:	**Profesores:**
Lezcano ,Emanuel Ezequiel.	Lic. Walter O. VALLEJOS.
LU 54203	
Quintana, Facundo Nahuel.	Lic. Dario O. VILLEGAS.
LU 50248	
Dusicka, Marisol	Lic. Walter Vallejos
LU 36622	

Fecha 31/10/23

CAPÍTULO I: INTRODUCCIÓN

a) Tema:

El objetivo principal es explorar en profundidad el uso de vistas y vistas indexadas, así como su optimización en este sistema de gestión de bases de datos. Estas herramientas desempeñan un papel esencial en la administración eficiente de datos y la mejora del rendimiento de las consultas, lo que las convierte en un área de interés fundamental para los profesionales y estudiantes de bases de datos.

b) Definición o planteamiento del problema:

Las vistas de SQL Server son una forma poderosa de organizar datos y crear consultas personalizadas. Con las vistas, puede unir varias tablas, establecer condiciones e incluso controlar el acceso del usuario a los datos.

Comúnmente para simplificar la complejidad de una consulta SQL y para hacer que los datos sean más fáciles de leer y entender. También son útiles para la seguridad, ya que se pueden utilizar para restringir el acceso a ciertas partes de la base de datos.

c) Objetivo del Trabajo Práctico:

El objetivo principal de este Trabajo Práctico es investigar y analizar a fondo el uso, la optimización y la aplicación de vistas y vistas indexadas en bases de datos, con un enfoque específico en SQL Server 2019. Se busca comprender cómo estas herramientas pueden mejorar el rendimiento y la gestión de bases de datos relacionales.

Algunos objetivos específicos son:

- Comprender las vistas y vistas indexadas en SQL Server 2019.
- Explorar el Concepto de Vistas Indexadas
- Optimizar vistas indexadas para mejorar el rendimiento.
- Gestionar actualizaciones y mantenimiento de vistas.
- Evaluar el uso de vistas y vistas indexadas en casos prácticos.

CAPÍTULO II: MARCO CONCEPTUAL O REFERENCIAL

1. **Las Vistas en SQL Server:** en el contexto de la gestión de datos en un entorno tecnológico en constante cambio y en relación con conceptos clave que afectan a la toma de decisiones, la eficiencia operativa y el desarrollo sostenible en la era digital. Proporciona una estructura sólida para investigar, aplicar y evaluar vistas en situaciones específicas relacionadas con innovaciones tecnológicas, TICs, globalización, desarrollo sostenible y otros conceptos fundamentales construido de la siguiente manera:

1) **Innovaciones Tecnológicas:** Las innovaciones tecnológicas se refieren a la introducción de nuevas tecnologías o mejoras significativas en las tecnologías existentes. En el contexto de vistas en SQL Server, las innovaciones tecnológicas pueden incluir mejoras en el rendimiento de la base de datos, la seguridad de los datos y la eficiencia de acceso a través de consultas y vistas más avanzadas.

2) **Tecnologías de la Información y Comunicación (TICs):** Las TICs son fundamentales para la gestión y el acceso a datos en SQL Server. Estas tecnologías habilitan la recopilación, el procesamiento y la distribución de información a través de sistemas de bases de datos, incluyendo la creación y el uso de vistas.

3) **Globalización:** La globalización implica la interconexión de empresas y mercados a nivel mundial. En este contexto, las vistas en SQL Server pueden proporcionar una visión consolidada de datos globales para la toma de decisiones a nivel empresarial.

4) **Crecimiento:** _El crecimiento empresarial y el aumento de la cantidad de datos requieren una gestión eficiente de la información. Las vistas en SQL Server permiten a las organizaciones acceder a datos clave de manera estructurada y simplificada.

5) **Desarrollo Regional o Local:** Las vistas en SQL Server pueden ser fundamentales para la gestión de datos a nivel regional o local, lo que permite a las organizaciones adaptarse a las necesidades específicas de esas áreas y tomar decisiones informadas.

6) **Cadenas Productivas:** Las cadenas productivas involucran secuencias de producción y suministro de bienes y servicios. Las vistas en SQL Server pueden ayudar a rastrear y optimizar la eficiencia de estas cadenas al proporcionar información clave para la toma de decisiones.

7) **Clusters**: Los clusters son agrupaciones geográficas de empresas relacionadas. Las vistas en SQL Server pueden ayudar a las organizaciones a comprender y analizar el rendimiento de los clusters al proporcionar datos consolidados y procesables.

8) **Desarrollo Sustentable**: El desarrollo sostenible se basa en la gestión responsable de los recursos y la toma de decisiones informadas. Las vistas en SQL Server pueden ser una herramienta para rastrear y evaluar el impacto ambiental y social, lo que contribuye al desarrollo sostenible.

9) **Vistas en SQL Server**: Las vistas en SQL Server son consultas SQL almacenadas que permiten a los usuarios acceder a datos de manera más organizada y estructurada. Estas vistas son fundamentales para la presentación y el acceso a datos en un formato lógico.

2. El concepto de "**vistas indexadas"** en SQL Server se refiere a la capacidad de crear índices en vistas. Antes de profundizar en el concepto, es importante entender algunas definiciones clave:

1) **Vistas (Views)** : En SQL Server, una vista es una consulta almacenada que se puede tratar como una tabla virtual. Permite a los usuarios y las aplicaciones ver y acceder a los datos de una o más tablas como si fueran una sola tabla.

2) Í**ndices (Indexes)** : Los índices son estructuras de datos que mejoran el rendimiento de las consultas al acelerar la recuperación de registros de una tabla. Se utilizan para buscar y ordenar rápidamente los datos.

3) **Vistas Indexadas (Indexed Views)** : Las vistas indexadas son vistas que se han optimizado para un mejor rendimiento mediante la creación de índices en ellas. Estos índices permiten acelerar las consultas que involucran las vistas al recalcular los resultados de las consultas y almacenarlos en forma de índice.

El marco conceptual o referencial para vistas indexadas en SQL Server se puede resumir de la siguiente manera:

1. **Creación de Vistas Indexadas** Para crear una vista indexada, se deben cumplir ciertos requisitos:
   La vista debe estar escrita de una manera específica, como una vista agregada.
   Debe cumplir con ciertas restricciones, como no permitir UNION ni subconsultas correlacionadas.

Las tablas subyacentes a la vista deben tener índices clúster y seguir otras restricciones.

2. **Ventajas de las Vistas Indexadas**

    a. Mejora el rendimiento: Al crear índices en vistas, las consultas que involucran esas vistas se ejecutan más rápido, ya que los resultados se almacenan en el índice en lugar de calcularlos en tiempo real.

    b. Simplificación del acceso a datos: Las vistas indexadas pueden ocultar la complejidad de las consultas subyacentes y proporcionar a los usuarios una forma más sencilla de acceder a los datos.

3. **Actualización de Vistas Indexadas**: Las vistas indexadas pueden ser utilizadas para consultas de lectura eficientes, pero la actualización de datos puede ser más complicada. Las vistas indexadas son generalmente de solo lectura, y cualquier modificación de los datos subyacentes debe cumplir con ciertas restricciones.

4. **Mantenimiento de Índices**: Los índices en vistas deben ser actualizados cada vez que los datos subyacentes cambian. Esto puede afectar el rendimiento durante las operaciones de inserción, actualización o eliminación de datos en las tablas subyacentes.

5. **Limitaciones y Consideraciones**: Es importante comprender las limitaciones y restricciones asociadas con las vistas indexadas en SQL Server. Estas limitaciones incluyen el tipo de vista, las operaciones admitidas y los requisitos de índice en las tablas subyacentes.

En resumen, las vistas indexadas en SQL Server son una técnica avanzada para mejorar el rendimiento de las consultas al crear índices en vistas que cumplen con ciertos criterios. Sin embargo, su implementación y mantenimiento deben abordarse con cuidado, ya que pueden tener implicaciones en el rendimiento y la actualización de datos.

CAPÍTULO III: METODOLOGÍA SEGUIDA

Planificación y Distribución de Tareas

El plan seguido se centró en una distribución eficiente de tareas entre los miembros del equipo. Desde el principio, se acordó dividir el trabajo en bloques que permitieran abordar el tema de manera exhaustiva y organizada. Esta división de responsabilidades facilitó un enfoque más preciso en cada aspecto del tema. Además, se promovió la colaboración y el apoyo mutuo entre los integrantes del equipo, de modo que si alguien se encontraba con dificultades, otro miembro brindaba asistencia, asegurando un flujo constante de avance en el proyecto.

Superación de Dificultades

Aunque el tema de "Vistas y Vistas Indexadas en SQL Server" es relativamente corto, surgieron algunas dudas y desafíos que requirieron la atención del equipo. Sin embargo, gracias a la política de colaboración mencionada, las dificultades se superaron de manera efectiva. Se fomentó un ambiente abierto para plantear preguntas y discutir cualquier inquietud que surgiera, lo que permitió a los integrantes del equipo resolver las dudas de manera rápida y eficiente.

Herramientas Utilizadas

Para llevar a cabo este proyecto, se hicieron uso de las siguientes herramientas:

SQL Server Management Studio: Esta herramienta fue esencial para realizar pruebas y experimentos con vistas y vistas indexadas en SQL Server.

Chat-GPT: Se utilizó la asistencia de un modelo de lenguaje GPT-3.5 para aclarar dudas, obtener información adicional y recibir recomendaciones sobre cómo abordar ciertos aspectos del tema. Esto resultó útil para profundizar en el conocimiento y obtener explicaciones más detalladas cuando fue necesario.

En resumen, la metodología seguida para abordar el tema de "Vistas y Vistas Indexadas en SQL Server" se basó en la colaboración, la distribución de tareas, la resolución conjunta de dudas y el uso de herramientas específicas, lo que permitió un enfoque efectivo y una comprensión sólida de este aspecto de SQL Server.

CAPÍTULO IV: DESARROLLO DEL TEMA / PRESENTACIÓN DE RESULTADOS

¿ Qué es una vista en SQL SERVER ?

En SQL Server, una vista es un objeto de base de datos que actúa como una tabla virtual compuesta por los resultados de una consulta SQL. Las vistas permiten a los usuarios y aplicaciones acceder y consultar datos almacenados en una o varias tablas de la base de datos de una manera más conveniente. Las características clave de las vistas incluyen:

- **Virtualidad:** Las vistas no almacenan datos físicamente; en cambio, almacenan la definición de la consulta que se utiliza para obtener los datos de las tablas subyacentes.

- **Sencillez:** Las vistas pueden ocultar la complejidad de las relaciones y la estructura de la base de datos, proporcionando una interfaz más simple para los usuarios.

- **Seguridad:** Las vistas pueden utilizarse para restringir el acceso a ciertos datos, permitiendo a los administradores de la base de datos controlar quién puede ver y consultar qué datos.

Ejemplo de creación y utilización de vistas:

```
USE base_consorcio;

CREATE VIEW vistaAdministrador AS
SELECT apeynom, sexo, fechnac
FROM administrador;

SELECT * FROM vistaAdministrador;
```

En la vista creada anteriormente estamos consultando solo por los campos “apeynom”, “sexo” y “fechnac” de la tabla “administrador”. Luego se llama a la vista para obtener los datos.

Uso de Vistas para Mejorar la Seguridad y Simplificar Consultas:

Las vistas también son útiles para mejorar la seguridad y simplificar las consultas. Imagina que tienes una tabla con información de empleados, incluyendo datos sensibles como números de seguridad social. Puedes crear una vista que muestre solo información no sensible y oculte la información confidencial:

```
-- Creamos una vista con datos no sensibles de empleados
CREATE VIEW EmpleadosPublicos AS
SELECT EmployeeID, FirstName, LastName, Position
FROM Empleados;

-- Consultar la vista para obtener datos no sensibles de empleados
```

SELECT * FROM EmpleadosPublicos;

¿ Qué es una vista indexada en SQL SERVER ?
Las vistas indexadas en SQL Server son vistas que han sido optimizadas para mejorar el rendimiento de las consultas. Estas vistas son especialmente útiles cuando se trabaja con consultas complejas o consultas que acceden a grandes volúmenes de datos. Al crear una vista indexada, SQL Server crea un índice físico en la base de datos que almacena los resultados de la vista. Esto permite acelerar el proceso de recuperación de datos y optimizar consultas.

Algunas características clave de las vistas indexadas incluyen:

Mejora de rendimiento: Las vistas indexadas mejoran la velocidad de recuperación de datos al precomputar y almacenar los resultados de la vista. Esto es especialmente útil cuando se realizan consultas frecuentes a través de la vista.

Mantenimiento automático: SQL Server se encarga del mantenimiento del índice de la vista indexada, lo que significa que se actualiza automáticamente cuando se realizan cambios en las tablas subyacentes.

Restricciones: Las vistas indexadas tienen ciertas restricciones, como la incapacidad de contener subconsultas o funciones definidas por el usuario. Esto es importante tenerlo en cuenta al diseñar vistas indexadas.

Ejemplo práctico:

```
-- Crear una vista indexada que muestra los productos más vendidos en un rango de fechas
CREATE VIEW ProductosMasVendidosConIndice WITH SCHEMABINDING AS
SELECT ProductID, ProductName, SUM(Quantity) AS TotalVendido
FROM dbo.Productos
JOIN dbo.DetallesPedido ON Productos.ProductID = DetallesPedido.ProductID
WHERE OrderDate BETWEEN '2023-01-01' AND '2023-12-31'
GROUP BY ProductID, ProductName;

-- Crear un índice clustered en la vista
CREATE UNIQUE CLUSTERED INDEX IX_ProductosMasVendidos
ON ProductosMasVendidosConIndice (ProductID);

-- Consultar la vista indexada
SELECT * FROM ProductosMasVendidosConIndice
WHERE TotalVendido > 100;
```

Ventajas de las Vistas Indexadas en Términos de Rendimiento:

Mejora en el rendimiento de consulta: Una de las principales ventajas de las vistas indexadas es que pueden acelerar el rendimiento de las consultas. Al precalcular y almacenar los resultados de la vista en un índice físico, las consultas que acceden a la vista pueden obtener datos de manera más eficiente, especialmente en casos de consultas complejas o de gran volumen de datos.

- Optimización de consultas frecuentes: Las vistas indexadas son especialmente útiles cuando se realizan consultas frecuentes a través de una vista. Al reducir la necesidad de cálculos repetitivos, se acelera el tiempo de respuesta de las consultas y se mejora la experiencia del usuario.

- Mantenimiento automático: SQL Server se encarga del mantenimiento del índice de la vista indexada. Esto significa que, a medida que se realizan cambios en las tablas subyacentes, como inserciones, actualizaciones o eliminaciones, el índice de la vista se actualiza automáticamente, lo que facilita la gestión de los datos y garantiza la integridad.

Desventajas de las Vistas Indexadas en Términos de Rendimiento:

- **Restricciones en el diseño:** Las vistas indexadas tienen ciertas restricciones, como la prohibición de contener subconsultas, funciones definidas por el usuario o UNION. Esto puede limitar la flexibilidad en el diseño de la vista y requerir modificaciones en la consulta original para que sea apta para la indexación.

- **Consumo de espacio:** Los índices ocupan espacio en disco, y las vistas indexadas no son una excepción. Si se crean múltiples vistas indexadas en una base de datos, pueden ocupar una cantidad significativa de espacio de almacenamiento. Esto es especialmente relevante si la base de datos tiene limitaciones de espacio.

Ventajas de las Vistas Indexadas en Términos de Mantenimiento:

- **Actualización automática:** Como se mencionó previamente, las vistas indexadas se actualizan automáticamente cuando se realizan cambios en las tablas subyacentes. Esto reduce la carga de trabajo para los administradores de la base de datos y garantiza que los datos reflejen con precisión los cambios realizados en las tablas.

- **Simplificación de la lógica de consulta:** Las vistas indexadas pueden simplificar la lógica de consulta al proporcionar una vista precalculada y optimizada de los datos. Esto facilita la escritura de consultas más simples y comprensibles.

Desventajas de las Vistas Indexadas en Términos de Mantenimiento:

- **Mantenimiento adicional:** Si bien las vistas indexadas pueden simplificar la lógica de consulta, también introducen un nivel adicional de mantenimiento. Los administradores de la base de datos deben supervisar el rendimiento de las vistas indexadas y asegurarse de que estén actualizadas y optimizadas.

- **Posible sobrecarga de actualización:** Si las tablas subyacentes se actualizan con frecuencia, las vistas indexadas pueden experimentar una sobrecarga de actualización, lo que podría afectar negativamente el rendimiento. En tales casos, se requiere un equilibrio cuidadoso entre el rendimiento y la sobrecarga de actualización.

Resultados de ejercicios propuestos

2) Crear una vista sobre la tabla administrador que solo muestre los campos apynom, sexo y fecha de nacimiento.

```
use base_consorcio;

CREATE VIEW vistaAdministrador AS
SELECT apeynom, sexo, fechnac
FROM administrador;

SELECT * FROM vistaAdministrador;
```

	apeynom	sexo	fechnac
1	Perez Juan Manuel	M	1985-02-18 00:00:00.000
2	BASUALDO DELMIRA	F	1980-10-09 00:00:00.000
3	SEGOVIA ALEJANDRO H.	M	1974-06-02 00:00:00.000
4	ROMERO ELEUTERIO	M	1972-08-19 00:00:00.000
5	NAHMIAS DE K. NIDIA	F	1971-11-28 00:00:00.000
6	CORREA DE M. MARIA G.	F	1990-01-16 00:00:00.000
7	NAHMIAS JOSE	M	1974-09-02 00:00:00.000
8	NAHMIAS DE R. REBECA J.	F	1989-03-07 00:00:00.000
9	LOVATO CERENTTINI ISABEL	F	1973-10-15 00:00:00.000
10	GOMEZ MATIAS GABRIEL	M	1974-03-20 00:00:00.000
11	CORREA HUGO E.	M	1993-08-11 00:00:00.000
12	MACHUCA CEFERINA	F	1991-09-16 00:00:00.000
13	CARDOZO MAXIMA	F	1988-11-07 00:00:00.000
14	RODRIGUEZ MARTIN J.	M	1985-12-09 00:00:00.000

3) Realizar insert de un lote de datos sobre la vista recién creada. Verificar el resultado en la tabla administrador.

```
Insert into vistaAdministrador(apeynom,sexo,fechnac) values ('USUARIO PRUEBA', 'F', '19801030');
Insert into vistaAdministrador(apeynom,sexo,fechnac) values ('USUARIO PRUEBA 1', 'M', '19740622');
Insert into vistaAdministrador(apeynom,sexo,fechnac) values ('USUARIO PRUEBA 2', 'M', '19720331');
```

SELECT * FROM vistaAdministrador ;

	apeynom	sexo	fechnac
1	USUARIO PRUEBA	F	1980-10-30 00:00:00.000
2	USUARIO PRUEBA 1	M	1974-06-22 00:00:00.000
3	USUARIO PRUEBA 2	M	1972-03-31 00:00:00.000

4) Realizar update sobre algunos de los registros creados y volver a verificar el resultado en la tabla.

Update vistaAdministrador set fechnac = '19990916' where apeynom = 'USUARIO PRUEBA';

SELECT * FROM vistaAdministrador ;

	apeynom	sexo	fechnac
1	USUARIO PRUEBA	F	1999-09-16 00:00:00.000

5) Borrar todos los registros insertados a través de la vista.

DELETE FROM administrador
WHERE tel is NULL;

Con esta sentencia me quedan todas las filas que tenía ya cargada en la tabla “administrador”.

6) Crear una vista que muestre los datos de las columnas de las siguientes tablas: (Administrador->Apeynom, consorcio->Nombre, gasto->periodo, gasto->fechaPago, tipoGasto->descripcion) .

CREATE VIEW vistaGeneral
WITH SCHEMABINDING
AS SELECT [dbo].[administrador].apeynom, [dbo].[consorcio].nombre, [dbo].[gasto].periodo, [dbo].[gasto].fechaPago, [dbo].[tipogasto].descripcion

```
FROM [dbo].[consorcio]
JOIN [dbo].[administrador] ON [dbo].[administrador].idadmin = [dbo].[consorcio].idadmin
JOIN [dbo].[gasto] ON [dbo].[gasto].idconsorcio = [dbo].[consorcio].idconsorcio
JOIN [dbo].[tipogasto] ON [dbo].[tipogasto].idtipogasto = [dbo].[gasto].idtipogasto;

select * from vistaGeneral ;
```

	apeynom	nombre	periodo	fechaPago	descripcion
1	Perez Juan Manuel	EDIFICIO-111	6	2013-06-16 00:00:00.000	OTROS
2	SEGOVIA ALEJANDRO H.	EDIFICIO-2481	6	2013-06-16 00:00:00.000	OTROS
3	NAHMIAS DE K. NIDIA	EDIFICIO-3161	6	2013-06-16 00:00:00.000	OTROS
4	GOMEZ MATIAS GABRIEL	EDIFICIO-4211	6	2013-06-16 00:00:00.000	OTROS
5	CARDOZO MAXIMA	EDIFICIO-531	6	2013-06-16 00:00:00.000	OTROS
6	RATTI JUAN E.	EDIFICIO-6331	6	2013-06-16 00:00:00.000	OTROS
7	MARTINEZ EDUARDO.	EDIFICIO-2011	6	2013-06-16 00:00:00.000	OTROS
8	BENITEZ ANSELMA B.	EDIFICIO-21181	6	2013-06-16 00:00:00.000	OTROS
9	DEL VALLE ANDRES	EDIFICIO-2231	6	2013-06-16 00:00:00.000	OTROS
10	TORRES LUIS.	EDIFICIO-2311	6	2013-06-16 00:00:00.000	OTROS
11	TORRES ELADIA	EDIFICIO-2441	6	2013-06-16 00:00:00.000	OTROS
12	VALLEJOS CYNTHIA ELIZABET	EDIFICIO-1481	6	2013-06-16 00:00:00.000	OTROS
13	GARCIA MARIA F.	EDIFICIO-1551	6	2013-06-16 00:00:00.000	OTROS
14	PARRAS DE ESQUIVEL MARIA	EDIFICIO-1641	6	2013-06-16 00:00:00.000	OTROS
15	AYALA JUAN S.	EDIFICIO-1771	6	2013-06-16 00:00:00.000	OTROS
16	CENTIN MIGUEL A. (H)	EDIFICIO-1831	6	2013-06-16 00:00:00.000	OTROS
17	GONZALEZ DANIEL PABLO	EDIFICIO-1921	6	2013-06-16 00:00:00.000	OTROS
18	PALACIOS DE F. OLGA	EDIFICIO-851	6	2013-06-16 00:00:00.000	OTROS
19	ALCARAZ MARIA CRISTINA	EDIFICIO-9141	6	2013-06-16 00:00:00.000	OTROS
20	GOIRIZ DIEGO	EDIFICIO-10111	6	2013-06-16 00:00:00.000	OTROS
21	GONZALEZ JOSE S.	EDIFICIO-1131	6	2013-06-16 00:00:00.000	OTROS
22	ESCOBAR GERONIMO DE LA C.	EDIFICIO-1221	6	2013-06-16 00:00:00.000	OTROS

7) Crear un indice sobre la columna fechaPago sobre la vista recien creada.

Para poder cumplir con la consigna, tuve que colocar el campo “fechaPago” como único ya que es una de las restricciones para crear un índice sobre una columna en una tabla de base de datos. Luego cree el índice sobre dicha columna sin inconvenientes.

```
CREATE UNIQUE CLUSTERED INDEX IX_vistaGeneral_FechaPago
ON [dbo].[vistaGeneral] (fechaPago);

select * from vistaGeneral order by nombre asc ;
```

CAPÍTULO V: INCLUSIÓN DE TEMAS:

1) Implementación de backup y restore:

El informe habla sobre la importancia de realizar copias de seguridad (backup) y restauraciones (restore) en SQL Server para proteger los datos críticos almacenados en las bases de datos.
Se menciona que una estrategia bien diseñada de copia de seguridad y restauración ayuda a proteger las bases de datos de posibles pérdidas de datos causadas por errores. Se destaca que la estrategia de copia de seguridad y restauración debe personalizarse según el entorno y los recursos disponibles. Se deben tener en cuenta factores como los objetivos de la organización, la naturaleza de las bases de datos y las restricciones de los recursos. Además, se enfatiza la importancia de probar las copias de seguridad para garantizar su eficacia.
Se recomienda realizar pruebas de restauración en un sistema de prueba y verificar la coherencia de la base de datos restaurada. Por último, se menciona la restauración en línea, que permite restaurar datos mientras la base de datos está en línea. Se describen los pasos básicos de la restauración en línea y se concluye resaltando la importancia de realizar backups regularmente para proteger los datos importantes.

2) Implementación de transacciones

Una transacción en base de datos es una secuencia de operaciones que se realizan de manera indivisible, operaciones agrupadas como una unidad. Es como un paquete que contiene múltiples acciones que deben completarse en su totalidad o deshacerse por completo. Las operaciones que contiene una transacción se van almacenando temporalmente, no a nivel de disco. Es hasta que termina la transacción que se tienen efecto de manera permanente o no. Esto asegura que los datos se mantengan consistentes y evita problemas en caso de errores o fallos en el sistema.
Una transacción aplica a datos recuperables, puede estar formada por operaciones simples o compuestas y su intención es que sea atómica. Una transacción siempre termina, aun en la presencia de fallas. Si una transacción termina de manera exitosa se dice que la transacción hace un commit (consumación).
Si la transacción se detiene sin terminar su tarea, se dice que la transacción aborta. Cuando la transacción es abortada, su ejecución se detiene y todas las acciones ejecutadas hasta el momento se deshacen (undone) regresando a la base de datos al estado antes de su ejecución. A esta operación también se le conoce como rollback.

Estados de una transacción

- Transacción Activa: se encuentra en este estado justo después de iniciar su ejecución.
- Transacción Parcialmente Confirmada: en este punto, se han realizado las operaciones de la transacción pero no han sido almacenados de manera permanente.

- Transacción Confirmada: Ha concluido su ejecución con éxito y se almacenan de manera permanente.
- Transacción Fallida: En este caso, es posible que la transacción deba ser cancelada.
- Transacción Terminada: indica que la transacción ha abandonado el sistema.
Tipos de Transacciones:
- Transacciones planas: Estas transacciones tienen un punto de partida simple (Begin_transaction) y un punto simple de terminación (End_transaction).
- Transacciones anidadas: las operaciones de una transacción anidada pueden incluir otras transacciones.

Existen restricciones para una transacción anidada:

- Debe empezar después que su padre y debe terminar antes que él.
- El commit de una transacción padre está condicionada al commit de sus transacciones hijas.
- Si alguna transacción hija aborta (rollback), la transacción padre también será abortada (rollback).

3) Procedimientos y Funciones Almacenadas

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.

- 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.

CAPÍTULO VI: CONCLUSIONES

Esto permitió profundizar en la importancia de definir restricciones en los campos para garantizar la integridad de los datos. También ha mostrado la necesidad de una planificación rigurosa en el diseño de bases de datos para evitar problemas de redundancia, inconsistencia o pérdida de información.

Además, ha desarrollado la habilidad para seleccionar entre vistas comunes y vistas indexadas según las características y requerimientos del sistema, buscando optimizar el rendimiento de las consultas.
De esta manera, se han cumplido los objetivos del trabajo práctico, que consistían en adquirir un conocimiento más sólido sobre la gestión de datos en SQL Server y las estrategias para mejorar la eficiencia y la calidad de las consultas.

CAPÍTULO VI. BIBLIOGRAFÍA.

https://www.youtube.com/watch?v=awD4nqp_5w0&ab_channel=RenzoCaceresRossi

https://desktop.arcgis.com/es/arcmap/latest/manage-data/using-sql-with-gdbs/example-creating-a-view-in-sql-server-with-sql.htm

https://juapri.freehostia.com/2007/07/07/vistas-indexadas-en-sql-server/

▷ Que es una vista en SQL SERVER (codigosql.top)