
Universidad
Tecmilenio®

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Introducción a los índices

Semana 6

Introducción

El tema de los **índices** es clave para entender las bases de datos. Desde su diseño, hasta su relación con las tablas y las vistas, los índices dictan gran parte del buen desempeño de las bases de datos como un todo y la eficiencia que estas tendrán con otras aplicaciones en términos de su interoperabilidad.

Explicación

Conceptos básicos de indexación

Los índices ayudan a estructurar las filas de las tablas y de las vistas automáticamente para el uso eficiente de las aplicaciones o para la eficiente extracción y manipulación de datos.

De acuerdo con Microsoft (2022), un índice actúa como una estructura en el disco, la cual se relaciona con una tabla o vista, y permite que estas recuperen y organicen las filas de cada una de ellas.

CREATE UNIQUE INDEX Nombre ON Tabla (Columna)

Crear una columna única a través de índices.

Explicación

Índices compuestos y de una sola columna

La principal diferencia entre el **índice compuesto** y el **de una sola columna** es la cantidad de campos que se toman como referencia. Los índices de una sola columna (o índices sencillos) son aquellos que se definen con base en un solo campo de la tabla como referencia para la búsqueda.

```
CREATE INDEX nombre_índice ON tabla (Columna) USING BTREE;
```

Índice sencillo en T-SQL.

ID	Nombre_Producto	Código
1	Triciclo	1334340134
2	Bicicleta	1334823092
3	Patineta	1334283092

Tabla de origen.

Explicación

```
CREATE INDEX Nombre_Producto_Índice ON Productos (Nombre) USING BTREE;
```

Creación del índice para la tabla origen de datos.

Nombre_Producto	ID
Bicicleta	2
Patineta	3
Triciclo	1

Resultado de la creación del índice para la tabla origen de datos.

Explicación

Estructuras de tablas en SQL Server

- Los principales dos elementos de una tabla en SQL Server son las **columnas y las filas**, y son requisitos para que un elemento sea considerado tabla (Yaseen, 2018).
- Los índices hacen uso de estas dos propiedades para ordenar y almacenar los datos de diferentes maneras. Para esto, existe el término de **tablas agrupadas** o ***clustered tables***.

Tabla no agrupada	
Edades	Medicamentos
60-70	Sí
10-20	No
20-30	No
40-50	Depende
0-10	No
10-20	No
60-70	Sí
60-70	Sí

Tabla no agrupada	
Edades	Medicamentos
0-10	No
10-20	No
10-20	No
20-30	No
40-50	Depende
60-70	Sí
60-70	Sí
60-70	Sí

Actividad

Con base en lo descrito en el tema, reflexiona sobre las siguientes preguntas:

01 ¿En qué ayuda la implementación de índices en las tablas?

02 ¿En qué escenarios aplica indexar una tabla y por qué?

Universidad Tecmilenio

Cierre

La premisa fundamental de los **índices** es optimizar el funcionamiento de las tablas y las bases de datos.

Es importante entender que los **índices** están diseñados para no revisar todos los datos de una fila en la tabla, sino solo aquellos que se están buscando. Naturalmente, eso libera espacio para que el sistema operativo no entregue tanto recurso operacional al sistema y, por ende, no necesite tanta memoria para procesarse.

Bibliografía

• Microsoft. (2022). *SQL Server and Azure SQL index architecture and design guide*. Recuperado de https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver16

• Yaseen, A. (2018). *SQL Server table structure overview*. Recuperado de https://www.sqlshack.com/sql-server-table-structure-overview/

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Indexación avanzada

Semana 6

Introducción

Se revisarán los dos diferentes tipos de índices principales y cómo se comportan en las tablas.

Así como la sintaxis para crearlos y su funcionalidad dentro de SQL Server a través de la elaboración de ejemplos para su uso.

Se analizarán los planes de ejecución, verificando su desarrollo en función de los índices elaborados y su funcionamiento durante una consulta general en SQL.

Explicación

Índices agrupados

Un índice es un objeto que funciona para optimizar la lectura de una base de datos. Un factor clave de los índices es el rendimiento que estos dan en bases de datos relacionales y para tablas de muchos campos y registros.

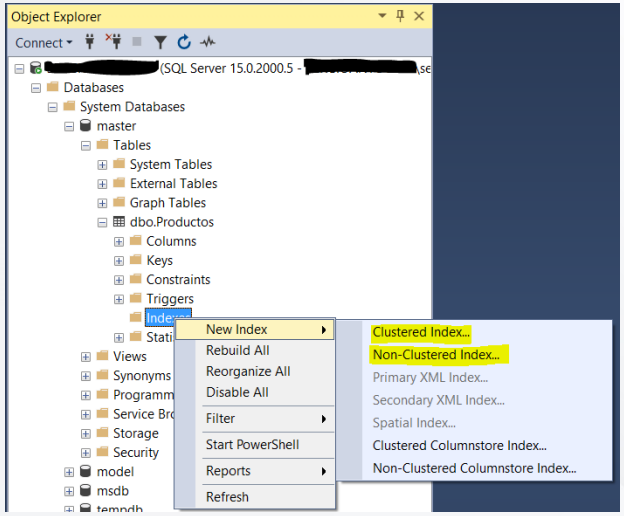
Los diseñadores de las bases de datos deben saber dónde poner los índices, por ejemplo, en las llaves primarias, en algún grupo de campos, en campos que son regularmente usados, entre otros escenarios (Tejada, 2020).

Índices no agrupados

Estos son regularmente creados cuando se requiere que una columna sea buscada o consultada de manera frecuente (a parte de la ya definida por la llave primaria).

Universidad Tecmilenio

Explicación



Crear índices a través de la aplicación.

Crear un índice automáticamente.

```
CREATE TABLE Ordenes
(OrdenId INT NOT NULL PRIMARY KEY,
 Monto INT);
```

Crear un índice no agrupado.

```
CREATE TABLE Empleados
(EmpleadoID VARCHAR(15) NOT NULL PRIMARY KEY,
 EdadEmpleado VARCHAR(100),
 INDEX EdadEmpleado_IND(EdadEmpleado)
);
```

Explicación

Planes de ejecución

De acuerdo con Microsoft (2022), cuando una consulta se corre, pasa por un proceso que consiste en evaluar cuál es el requerimiento del usuario, además de ver cuál es la dirección y forma de leer los datos solicitados.

El optimizador de consultas considera los índices en función y decide si conviene tomarlos como referencia para mostrar los datos o simplemente leer la tabla con las restricciones que tenga.

Si la consulta selecciona todos los datos de todos los campos de una tabla, por ejemplo, no tendría sentido aplicar algún mecanismo o leer algún índice, ya que simplemente el servidor tendrá que explorar la tabla completa.

Actividad

Con base en lo descrito en el tema, reflexiona sobre las siguientes preguntas:

01 ¿Qué diferencias existen entre un índice agrupado y uno no agrupado?

02 ¿Cuándo se necesita aplicar un índice no agrupado y por qué?

Universidad Tecmilenio

Cierre

Los **índices** son subobjetos de las tablas que, así como pueden resultar beneficiosos para la búsqueda de los datos dentro de las bases de datos, también pueden resultar contraproducentes si no se les brindan actualizaciones y no resultan ser consistentes con las tablas en función.

Entender que SQL Server ya viene con un software integrado, que se conoce como optimizador de consultas que, como su nombre lo indica, optimiza el proceso de búsqueda de datos y puede llegar a usar los índices como auxiliar de búsqueda.

Bibliografía

• Microsoft. (2022). *Execution Plans*. Recuperado de https://docs.microsoft.com/en-us/sql/relational-databases/performance/execution-plans?view=sql-server-ver16

• Tejada, R. (2020). *Bases de Datos Relacional, principios básicos de cómo diseñar una*. Recuperado de https://academy.aviada.mx/2020/03/30/bases-de-datos-relacional-principios-basicos/

Universidad Tecmilenio

Consulta en Microsoft SQL Server®

Diseño e implementación de vistas

Semana 6

El lenguaje T-SQL soporta la creación de objetos de tipo vistas que permitirán contener el acceso a una o a “n” número de tablas, de manera que estén relacionadas para retornar datos exactamente como se necesitan.

Lo ideal es que una vista encapsule las relaciones que frecuentemente se realizan en la base de datos para evitar tener código repetido con la ventaja de ahorrar tiempo al momento de diseñar, ejecutar o crear procesos en la base de datos.

Universidad Tecmilenio

Explicación

Introducción a las vistas

¿Qué es una vista en **T-SQL**?

- De acuerdo con Microsoft (2021), es una tabla virtual cuyo contenido se define mediante una consulta. Al igual que una tabla, una vista consta de un conjunto de columnas y filas de datos con nombre.

- **Las vistas** son objetos programados en T-SQL, considerados como “tablas virtuales”; su estructura es creada a base de una consulta “SELECT” y el resultado se genera en su ejecución. El uso de vistas es muy común para el tema de reportes, monitoreo, extracción e intercambio de información.

Explicación

Creación y gestión de vistas

CREATE VIEW será el comando que se utilizará para la creación de un objeto tipo vista en base de datos, y en el siguiente *script* se muestra la sintaxis completa:

```
CREATE VIEW view_name AS
SELECT column1, column2...
FROM table_name
WHERE condition;
```

Creación de una Vista en T-SQL.

Fuente: Blog código fuente. (2018). *Crear Vistas (tablas virtuales) en SQL*. Recuperado de https://www.codigofuente.org/crear-vistas-tablas-virtuales-sql/

Para modificar una vista se utilizará el comando ALTER VIEW más los cambios que se aplicarán en la vista.

```
ALTER VIEW EmpleadosCumpleaños AS
SELECT
            [EmpleadoID]                    AS [Codigo]
            ,[EmpleadoNombre]               AS [Nombre]
            ,[EmpleadoFechaNacimiento]      AS [Cumpleaños]
FROM [dbo].[Empleados]
WHERE [EmpleadoStatus] = 1
```

Modificación de una vista en T-SQL.

Universidad Tecmilenio

Explicación

- La eliminación de una vista en base de datos se lograría con DROP VIEW y el nombre de la tabla. Asimismo, se puede utilizar el comando IF EXISTS con el que la estructura quedaría de la siguiente manera:

```
DROP VIEW [IF EXISTS] schema_name.view_name;
```

Eliminar una vista en T-SQL.
Fuente: SQLServer Tutorial.Net. (2022). *SQL Server DROP VIEW*. Recuperado de https://www.sqlservertutorial.net/sql-server-views/sql-server-drop-view/

Universidad Tecmilenio

Explicación

Consideraciones para el rendimiento óptimo de las vistasx

1. Evitar, en la medida de lo posible, el uso de **OR, LIKE, IN, GROUP BY.**
2. Si se usan tablas con mucha información, hay que validar si es factible la creación de índices.
3. Evaluar el orden de las relaciones entre tablas y campos. Se recomienda comenzar con **INNER JOIN** y dejar al final **LEFT JOIN, RIGHT JOIN**.
4. Relacionar los campos entre llaves primarias y foráneas.
5. Usar **WITH(NOLOCK)** para evitar bloqueos en la base de datos.
6. No usar el comando **ORDER BY**.
7. No hacer uso excesivo de **CAST y CONVERT**.
8. Ejecutar la consulta con plan de ejecución (Execution plan) para conocer la sección que más está teniendo costo y optimizarla.
9. Si una tabla tiene demasiados datos, considerar la creación de una tabla temporal que se llene mediante un Job y que posteriormente la vista haga uso de esa tabla temporal.
10. Evitar **SELECT**.

Actividad

Con base en lo descrito en el tema, reflexiona sobre las siguientes preguntas:

01 Al exponer información sensible, ¿es ideal usar vistas?, ¿por qué?

02 ¿Es buena práctica hacer vistas para no realizar la consulta directamente en la tabla?

Universidad Tecmilenio

Cierre

Una **vista** es una simplificación de una consulta con varias relaciones y condiciones para resguardar la integridad y seguridad de la estructura interna del origen de los datos, pero también la simplicidad al momento de implementar proyectos o procesos nuevos, pues se evitarán contratiempos en las nuevas consultas.

Con el uso considerado de **vistas** se evitará exponer datos sensibles, mostrando exclusivamente información que requiere el proceso.

Una **vista** es susceptible a que marque error cuando se utilizan conversiones de datos, aplicación de fórmulas matemáticas o cuando el nombre de una tabla o columna cambia.

Bibliografía

• Blog código fuente. (2018). *Crear Vistas (tablas virtuales) en SQL*. Recuperado de https://www.codigofuente.org/crear-vistas-tablas-virtuales-sql/

• Microsoft. (2021). *Views SQL Docs*. Recuperado de https://docs.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver15

• SQLServer Tutorial.Net. (2022). *SQL Server DROP VIEW*. Recuperado de https://www.sqlservertutorial.net/sql-server-views/sql-server-drop-view/

Universidad Tecmilenio®

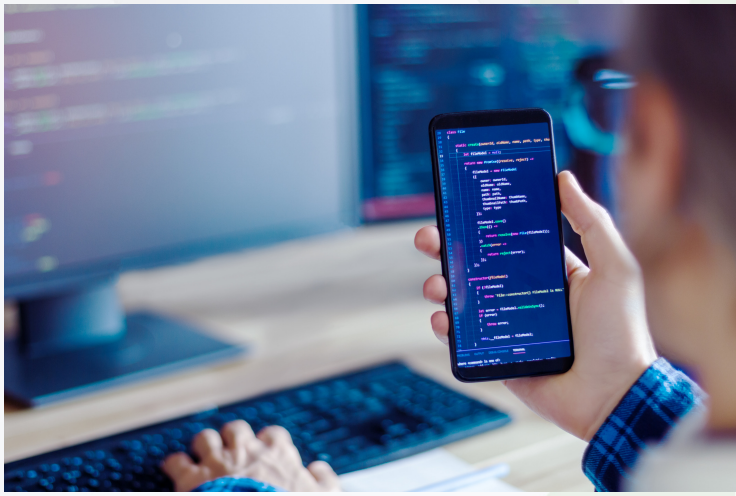
Consulta en Microsoft SQL Server®

Diseño e implementación de los procedimientos almacenados

Semana 6

Introducción

El diseño, la creación y la ejecución de un procedimiento almacenado son los puntos que se verán en este tema para identificar lo potente que es este tipo de objeto en el lenguaje T-SQL, de tal manera que se logrará comprimir un flujo de programa tan básico o complejo como se requiera.



Explicación

Introducción a los procedimientos almacenados con parámetros complejos

¿Qué es un **procedimiento almacenado** en **T-SQL**?

- Sierra (2022) menciona que **los procedimientos almacenados de SQL** pueden ser instrucciones de tipo Transact-SQL o con un Common Runtime Language (CLR) de .NET. Dichas instrucciones se encuentran almacenadas de forma física con un nombre dentro de la base de datos.

- Un procedimiento almacenado se implementa cuando un proceso es reutilizable y su ejecución sucede en el servidor, por lo cual, necesita ser optimo en cuanto a sus validaciones, operaciones y en las relaciones entre tablas.

Explicación

Creación de los procedimientos almacenados con parámetros complejos

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Estructura de un procedimiento almacenado en T-SQL.

En la estructura de un procedimiento almacenado no se incluyen parámetros de entrada, salida y mucho menos variables, debido a que es opcional su uso, pero lo más común es que se utilice, al menos, un parámetro de entrada.

¿Qué es un parámetro?

Los **parámetros** se utilizan para intercambiar datos entre procedimientos y funciones que se almacenan y la aplicación o herramienta que llamó al procedimiento o función que se almacenan (Microsoft, 2022).

Explicación

```
CREATE PROCEDURE CalcularAreaTriangulo
AS
(
        --Cuerpo del procedimiento
        SELECT 100 * 2 / 2 AS [Area]
)
```

Creación del procedimiento almacenado “CalcularAreaTriangulo” en T-SQL.
Fuente: W3schools. (2022). *SQL Stored Procedures for SQL Server.*
Recuperado de https://www.w3schools.com/sql/sql_stored_procedures.asp

```
ALTER PROCEDURE [dbo].[CalcularAreaTriangulo]
        @base           AS INT
        ,@altura    AS INT
AS
BEGIN
        DECLARE @division AS INT = 2;

        --Cuerpo del procedimiento
        SELECT @base * @altura / @division AS [Area]
END
```

Implementación de parámetros de entrada y variables en un procedimiento almacenado en T-SQL.

```
1  EXEC [dbo].[CalcularAreaTriangulo] @base=25, @altura=20
```

106 %

Results | Messages

	Area
1	250

Ejecución de un procedimiento almacenado con parámetros de entrada en T-SQL.

Explicación

Uso de los procedimientos almacenados con parámetros complejos

Un punto relevante a relucir es que no solo se pueden realizar operaciones matemáticas dentro de este objeto, sino que también se pueden implementar comandos DML (**INSERT, UPDATE, DELETE**), y a menor medida, pero posible, los comandos DDL (**CREATE, DROP, TRUNCATE**).

```
ALTER PROCEDURE [dbo].[CalcularAreaTriangulo]
	@xml		VARCHAR(MAX)	= NULL
	,@area		AS INT		= 0 OUTPUT
AS
BEGIN
	--Validar si el XML no está NULL
	IF @xml IS NOT NULL
	BEGIN
		DECLARE @division	AS INT = 2
			,@xmlDocID	AS INT
			,@base		AS INT
			,@altura	AS INT

		EXEC sp_ xml_ preparedocument @xmlDocID
OUTPUT, @xml

		--Leer XML y guardar valores en variables
		SELECT
			@base = [xml].[Base]
			,@altura = [xml].[Altura]
		FROM
			OPENXML(@xmlDocID, 'Area/ Triangulo', 2)
WITH
			(
				Base	INT
				,Altura	INT
			) AS [xml]
```

Implementación de INSERT en un procedimiento almacenado en T-SQL.

Universidad Tecmilenio

Explicación

```
DECLARE @o_area AS INT
EXEC [dbo].[CalcularAreaTriangulo]
    @xml=
        '<Area>
            <Triangulo>
                <Base>200</Base><Altura>3</Altura>
            </Triangulo>
        </Area>'
    ,@area = @o_area OUTPUT
SELECT @o_area AS [Area]
SELECT [base], [altura], [area] FROM [dbo].[Triangulos]
```

106 %

Results | Messages

	Area
1	300

	base	altura	area
1	200	3	300

Ejecución del procedimiento almacenado aplicando el ALTER del script de la tabla anterior.

```
DECLARE @o_area AS INT
EXEC [dbo].[CalcularAreaTriangulo]
    @area = @o_area OUTPUT
SELECT @o_area AS [Area]
SELECT [base], [altura], [area] FROM [dbo].[Triangulos]
```

106 %

Results | Messages

	Area
1	NULL

	base	altura	area
1	200	3	300

Ejecución del procedimiento almacenado sin enviar parámetro de entrada en T-SQL.

Actividad

Con base en lo descrito en el tema, reflexiona sobre las siguientes preguntas:

01 Al diseñar un procedimiento almacenado, ¿qué aspectos se deben considerar para que el objeto sea óptimo?

02 En un sistema de compras, ¿qué parámetros de entrada solicitarías para la consulta de un proveedor activo en la base de datos?

Universidad Tecmilenio

Cierre

La creación de **procedimientos almacenados** implica diseñar los parámetros de entrada y salida que tendrá el objeto, realizar las validaciones adecuadas para que no existan errores en la ejecución y usar correctamente los comandos para mantener optimizados los recursos de la base de datos.

Reutilizar un código, modificaciones en menor tiempo, optimización de recursos, código estructurado y centralizado desde un mismo objeto son algunas ventajas de usar procedimientos almacenados, también conocidos como **“Stored Procedures”.**

Bibliografía

• Microsoft. (2022). *ALTER PROCEDURE (Transact-SQL)*. Recuperado de https://docs.microsoft.com/es-es/sql/t-sql/statements/alter-procedure-transact-sql?view=sql-server-ver16

• Sierra, Y. (2018). *Procedimientos almacenados SQL Server: tipos y funcionalidad*. Recuperado de https://blog.mdcloud.es/procedimientos-almacenados-sql-server

• W3schools. (2022). *SQL Stored Procedures for SQL Server*. Recuperado de https://www.w3schools.com/sql/sql_stored_procedures.asp