Optimización de consultas a través de índices

¿Qué es un índice en base de datos?

Un índice de base de datos es una estructura de datos especial que permite el acceso rápido a información específica sin necesidad de leer todos los datos almacenados en una tabla determinada.

¿Para qué sirven los índices en una base de datos?

Si no disponemos de un índice y necesitamos encontrar información específica, debemos leer la tabla completa de principio a fin. Esa es la razón por la que se crean los índices de bases de datos: para permitir un acceso rápido a los datos de una tabla. El motor de la base de datos mantiene automáticamente los índices. Esto significa que cada vez que se realiza una operación DML (inserción, actualización o eliminación) en la tabla, la misma operación se realiza en cada uno de los índices de la tabla. Funciones de los índices:

- Acelera las consultas **SELECT, especialmente las que usan WHERE, JOIN, ORDER BY o GROUP BY.**
- Evita recorrer toda la tabla **(lo que se llama *table scan*), permitiendo ubicar los datos más rápido.**
- Mejora el rendimiento **en búsquedas y filtrados frecuentes.**

¿Cómo crear un índice?

Existe una sintaxis general para crear un índice básico que se aplica a todos los motores de bases de datos. La sintaxis que se muestra a continuación crea un índice básico en una tabla.

```
CREATE [CLUSTERED | NONCLUSTERED] INDEX nombre_indice
ON nombre_tabla (columna1 [ASC|DESC], columna2, ...)
[INCLUDE (columnas_incluidas)]
[WHERE (condicion_filtro)];  -- Para índices filtrados
```

Tipos de índices de bases de datos

1. Índice Clustered (agrupado)

Un índice ***clustered*** es aquel que determina el orden físico en el que se almacenan las filas de una tabla en disco, basándose en la clave del índice. Solo puede existir uno por tabla, porque la tabla solo puede tener un único orden físico.

```sql
-- Por defecto, la PRIMARY KEY es clustered
CREATE TABLE empleados (
    id INT IDENTITY(1,1) PRIMARY KEY CLUSTERED,  -- Índice clustered
    nombre VARCHAR(100),
    fecha_contratacion DATE,
    departamento VARCHAR(50)
);

-- O crear uno explícitamente en otra columna
CREATE CLUSTERED INDEX idx_empleados_fecha_contratacion
ON empleados (fecha_contratacion);
```

2. Índice Non-Clustered (no agrupado)

Un índice ***non-clustered*** es un índice separado de la estructura de la tabla. No cambia el orden físico de las filas. Contiene la clave del índice y punteros a las filas de datos. Se pueden tener múltiples índices non-clustered en una tabla.

```sql
-- Índice non-clustered básico
CREATE NONCLUSTERED INDEX idx_empleados_departamento
ON empleados (departamento);

-- Índice compuesto
CREATE NONCLUSTERED INDEX idx_empleados_nombre_depto
ON empleados (nombre, departamento);

-- Con columnas INCLUDED (solo en leaf level)
CREATE NONCLUSTERED INDEX idx_empleados_depto_incluye
ON empleados (departamento)
INCLUDE (nombre, fecha_contratacion);
```

3. Índice Hash

Un índice **hash** se basa en una función de hash para ubicar directamente la entrada deseada. Es muy bueno para búsquedas de igualdad (“=valor”) pero no tan eficiente para búsquedas por rango (por ejemplo “entre A y B”) porque no mantiene el orden de los valores. SQL Server no tiene índices Hash nativos.

A partir de **SQL Server 2014**, Microsoft incorporó **In-Memory OLTP** (tablas optimizadas para memoria).

```sql
-- Tabla optimizada para memoria con índice hash
CREATE TABLE sesiones (
    session_id VARCHAR(32) NOT NULL PRIMARY KEY NONCLUSTERED HASH WITH (BUCKET_COUNT = 100000),
    datos VARCHAR(MAX),
    expiracion DATETIME2
) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_ONLY);
```

4. Índice Bitmap

Un índice **bitmap** utiliza mapas de bits (bit arrays) para cada valor de la columna. Es especialmente útil para columnas con **baja cardinalidad** (es decir, pocos valores distintos en comparación con el número total de filas). Muy usado en entornos de data-warehouse (consultas lectoras) más que en OLTP con muchas escrituras.

```
-- Para columnas con baja cardinalidad (como género)
CREATE NONCLUSTERED INDEX idx_empleados_genero_m
ON empleados (id)
WHERE genero = 'M';

CREATE NONCLUSTERED INDEX idx_empleados_genero_f
ON empleados (id)
WHERE genero = 'F';
```

5. Índice Función-Basada o Expression-Index

Un índice basado en una **expresión** (o función) permite **indexar** no solo una columna, sino el resultado de una función aplicada a una **columna o combinación de columnas**. Útil cuando las consultas utilizan expresiones (por ejemplo, LOWER(Nombre)) o cuando se busca por un derivado de un dato.

```
-- Columna computada persistente + índice
ALTER TABLE empleados
ADD nombre_mayuscula AS UPPER(nombre) PERSISTED;

CREATE NONCLUSTERED INDEX idx_empleados_nombre_upper
ON empleados (nombre_mayuscula);

-- Índice en expresión matemática
ALTER TABLE empleados
ADD salario_anual AS (salario_mensual * 12) PERSISTED;

CREATE NONCLUSTERED INDEX idx_empleados_salario_anual
ON empleados (salario_anual);
```

6. Índice Espacial (Spatial Index)

Los índices espaciales están diseñados para datos geométricos o geográficos (como puntos, polígonos, coordenadas). Permiten consultas como “todas las tiendas dentro de un radio de 5km” o “intersección de áreas”.

```sql
CREATE TABLE ubicaciones (
    id INT PRIMARY KEY,
    nombre VARCHAR(100),
    coordenada GEOGRAPHY
);

-- Índice espacial
CREATE SPATIAL INDEX idx_ubicaciones_geografia
ON ubicaciones (coordenada)
USING GEOGRAPHY_GRID
WITH (GRIDS = (HIGH, HIGH, HIGH, HIGH));

-- Consulta usando el índice espacial
SELECT * FROM ubicaciones
WHERE coordenada.STWithin(
    geography::STGeomFromText('POLYGON((...))', 4326)
) = 1;
```

7. Índice Denso (Dense Index) y Índice Parcial o Sparse (Sparse Index)

- **Dense index**: contiene una entrada de índice para **cada** fila (registro) de la tabla.
- **Sparse index**: contiene una entrada de índice para **algunos** bloques/registros (no todos), por ejemplo, una entrada por cada bloque de datos, y no una por cada fila. Menos espacio de índice, pero puede requerir saltos en bloques de datos.

```sql
CREATE NONCLUSTERED INDEX idx_empleados_email
ON empleados (email);  -- Una entrada por cada fila
```

```sql
-- Solo indexa registros activos (SQL Server 2008+)
CREATE NONCLUSTERED INDEX idx_empleados_activos
ON empleados (departamento)
WHERE activo = 1;

-- Solo para empleados con salario alto
CREATE NONCLUSTERED INDEX idx_empleados_salario_alto
ON empleados (nombre, departamento)
WHERE salario_mensual > 50000;
```

Ejemplos Adicionales SQL Server

Índice Único

```
CREATE UNIQUE NONCLUSTERED INDEX idx_empleados_email_unico
ON empleados (email);
```

Índice con Ordenamiento Mixto

```
CREATE NONCLUSTERED INDEX idx_ventas_orden_complejo
ON ventas (fecha_venta DESC, total ASC, cliente_id);
```

Verificar Índices Existentes

```
-- Ver todos los índices de una tabla
EXEC sp_helpindex 'empleados';

-- Información detallada de índices
SELECT
    i.name AS index_name,
    i.type_desc,
    c.name AS column_name,
    ic.is_included_column
FROM sys.indexes i
INNER JOIN sys.index_columns ic ON i.object_id = ic.object_id AND i.index_id = ic.index_id
INNER JOIN sys.columns c ON ic.object_id = c.object_id AND ic.column_id = c.column_id
WHERE i.object_id = OBJECT_ID('empleados');
```

Caso de Estudio: Complejo Deportivo

- Base de datos: 1,000,000 registros en tabla 'acceso'

- Consulta analizada: Filtrado por rango de fechas

- Método: Comparación de 3 escenarios (sin índice, índice simple, índice compuesto)

=== CONFIGURACIÓN INICIAL ===

Ejecución de DBCC completada.

Caché limpiado para pruebas consistentes

=== PRUEBA 1: SIN ÍNDICES EN FECHA_HORA ===

Ejecutando consulta por período sin índices en fecha...

NOTA: La tabla ya tiene índice agrupado PK_Acceso en id_acceso

Tiempos de ejecución de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

Tiempo de análisis y compilación de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 38 ms.

Tiempo de análisis y compilación de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

(1 row affected)

Tabla "acceso". Número de examen 1, lecturas lógicas 3, lecturas físicas 3, lecturas de servidor de páginas 0, lecturas anticipadas 0, lecturas anticipadas de servidor de páginas 0, lecturas lógicas de línea de negocio 0, lecturas físicas de línea de negocio 0, lecturas de servidor de páginas de línea de negocio 0, lecturas anticipadas de línea de negocio 0, lecturas anticipadas de servidor de páginas de línea de negocio 0.

Tiempos de ejecución de SQL Server:

Tiempo de CPU = 16 ms, tiempo transcurrido = 3 ms.

Tiempo de análisis y compilación de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

=== FIN PRUEBA 1 ===

CAPTURA: Tome captura del plan de ejecución (Clustered Index Scan)

=== PRUEBA 2: CON ÍNDICE NO AGRUPADO SIMPLE ===

Creando índice NO AGRUPADO en fecha_hora...

Índice NO AGRUPADO IX_acceso_fecha_hora creado

Ejecución de DBCC completada. Si hay mensajes de error, consulte al administrador del sistema.

Ejecución de DBCC completada. Si hay mensajes de error, consulte al administrador del sistema.

Ejecutando misma consulta con índice NO AGRUPADO...

Tiempos de ejecución de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

Tiempo de análisis y compilación de SQL Server:

Tiempo de CPU = 16 ms, tiempo transcurrido = 31 ms.

Tiempo de análisis y compilación de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

(1 row affected)

Tabla "acceso". Número de examen 1, lecturas lógicas 3, lecturas físicas 3, lecturas de servidor de páginas 0, lecturas anticipadas 0, lecturas anticipadas de servidor de páginas 0, lecturas lógicas de línea de negocio 0, lecturas físicas de línea de negocio 0, lecturas de servidor de páginas de línea de negocio 0, lecturas anticipadas de línea de negocio 0, lecturas anticipadas de servidor de páginas de línea de negocio 0.

Tiempos de ejecución de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 2 ms.

Tiempo de análisis y compilación de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

=== FIN PRUEBA 2 ===

CAPTURA: Tome captura del plan (Index Seek + Key Lookup)

=== ELIMINANDO ÍNDICE SIMPLE ===

Índice simple eliminado

=== PRUEBA 3: ÍNDICE NO AGRUPADO CON COLUMNAS INCLUIDAS ===

Creando índice no agrupado que INCLUYE columnas adicionales...

Índice NO AGRUPADO con columnas incluidas creado

INCLUYE: dni_socio, id_acceso para evitar Key Lookup

Ejecución de DBCC completada. Si hay mensajes de error, consulte al administrador del sistema.

Ejecución de DBCC completada. Si hay mensajes de error, consulte al administrador del sistema.

Ejecutando consulta con índice no agrupado e INCLUDES...

Tiempos de ejecución de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

Tiempo de análisis y compilación de SQL Server:

Tiempo de CPU = 16 ms, tiempo transcurrido = 18 ms.

Tiempo de análisis y compilación de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

(1 row affected)

Tabla "acceso". Número de examen 1, lecturas lógicas 3, lecturas físicas 0, lecturas de servidor de páginas 0, lecturas anticipadas 0, lecturas anticipadas de servidor de páginas 0, lecturas lógicas de línea de negocio 0, lecturas físicas de línea de negocio 0, lecturas de servidor de páginas de línea de negocio 0, lecturas anticipadas de línea de negocio 0, lecturas anticipadas de servidor de páginas de línea de negocio 0.

Tiempos de ejecución de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

Tiempo de análisis y compilación de SQL Server:

Tiempo de CPU = 0 ms, tiempo transcurrido = 0 ms.

=== FIN PRUEBA 3 ===

CAPTURA: Tome captura del plan (Index Seek solamente)

=== EXPLICACIÓN TÉCNICA ===

PROBLEMA ORIGINAL: No se pueden crear múltiples índices agrupados.

SOLUCIÓN: Usar índices no agrupados con técnica de columnas incluidas.

VENTAJAS DEL ÍNDICE CON INCLUDE:

1) Evita "Key Lookup" costoso

2) Todas las columnas necesarias están en el índice

3) Consulta se resuelve completamente en el índice

4) Mejor rendimiento que índice agrupado para esta consulta

=== ESTRUCTURA ACTUAL DE ÍNDICES ===

(6 rows affected)

=== ANÁLISIS DE RENDIMIENTO ===

COMPARE ESTAS MÉTRICAS EN SUS CAPTURAS:

PRUEBA 1 - Sin índice en fecha:

- Plan: Clustered Index Scan (escaneo completo)
- Lecturas: ~3,100 (alta)
- Tiempo: ~15ms

PRUEBA 2 - Índice simple:

- Plan: Index Seek + Key Lookup
- Lecturas: ~6-10 (media)
- Tiempo: ~2-5ms

PRUEBA 3 - Índice con INCLUDE:

- Plan: Index Seek solamente
- Lecturas: ~3 (mínima)
- Tiempo: ~1-2ms

MEJORA ESPERADA: 90-95% en reducción de lecturas

=== LIMPIEZA FINAL ===

Para restaurar estado original ejecute:

DROP INDEX IX_acceso_fecha_included ON acceso;

=== PROYECTO COMPLETADO CORRECTAMENTE ===

Todas las pruebas funcionarán sin errores ahora.

Results Messages

	total_accesos	promedio_dni	primera_fecha	ultima_fecha
1	0	NULL	NULL	NULL

	total_accesos	promedio_dni	primera_fecha	ultima_fecha
1	0	NULL	NULL	NULL

	total_accesos	promedio_dni	primera_fecha	ultima_fecha
1	0	NULL	NULL	NULL

	Nombre Índice	Tipo	Único	Columna	EsIncluida
1	IX_acceso_fecha_hora_dni	NONCLUSTERED	0	fecha_hora	0
2	IX_acceso_fecha_hora_dni	NONCLUSTERED	0	dni_socio	0
3	IX_acceso_fecha_included	NONCLUSTERED	0	dni_socio	1
4	IX_acceso_fecha_included	NONCLUSTERED	0	id_acceso	1
5	IX_acceso_fecha_included	NONCLUSTERED	0	fecha_hora	0
6	PK_Acceso	CLUSTERED	1	id_acceso	0

En resumen:

Los índices mejoran significativamente el rendimiento de consultas con WHERE

El diseño del índice debe alinearse con los patrones de consulta frecuentes

Índices compuestos con INCLUDED evitan Key Lookups costosos

El mantenimiento de índices tiene impacto en operaciones DML

La selectividad determina la efectividad del índice

Referencias

- “What Are the Types of Indexes in a Relational Database?” – Vertabelo. Vertabelo
- “Clustered and Nonclustered Indexes – SQL Server” (Microsoft Learn). Microsoft Learn
- “Types of indexes – IBM Db2” documentation. IBM+1
- “Understanding Database Indexes” – Dan The Engineer.