**INDICES**

**(PPT-2)**

Un **índice** es una estructura que mejora la **velocidad de las consultas** al permitir buscar datos de forma más eficiente, similar al índice de un libro.

* Facilitan la obtención de información de una tabla. Una tabla se indexa por un campo (o varios).
* Posibilita el acceso directo y rápido haciendo más eficiente las búsquedas. Sin índice, SQL Server debe recorrer secuencialmente toda la tabla para encontrar un registro.
* Acelera la recuperación de información.
* Optimiza el acceso a los datos, mejora el rendimiento acelerando las consultas y otras operaciones.

SQL Server accede de 2 maneras a los datos **(PPT-3)**

* **recorriendo las tablas**: comenzando el principio y extrayendo los registros que cumplen las condiciones de la consulta.
* **empleando índices**: recorriendo la estructura de árbol del índice para localizar los registros y extrayendo los que cumplen las condiciones de la consulta.

**Tipos principales de índices: (PPT-5)**

1. **Índice clustered (agrupado)**:
   * Ordena físicamente las filas de la tabla según el índice.
   * Solo puede haber uno por tabla.
   * Ejemplo: si creas un índice clustered sobre la columna ID, los datos en disco se ordenan por ID.
2. **Índice non-clustered (no agrupado)**: (PPT-6)
   * Crea una estructura separada que apunta a las filas de la tabla.
   * Puedes tener varios en una tabla.
   * Es útil para búsquedas rápidas en columnas que no son clave primaria.
3. **Finalidad => Uso.**
   * Aceleran las consultas SELECT.
   * Mejoran filtros (WHERE), uniones (JOIN) y ordenamientos (ORDER BY).
   * Pero pueden ralentizar las operaciones de escritura (INSERT, UPDATE, DELETE) porque deben mantenerse actualizados.

***Ejemplo***:

CREATE NONCLUSTERED INDEX idx\_NombreON Clientes(Nombre);

**Forma de Acceso a los datos. (NO ESTA EN LAS PPTs)**

SQL Server accede a los datos de distintas formas, dependiendo de si existen **índices** o no, y del tipo de consulta que se realiza. Las formas principales de acceso a los datos son:

**🔹 1. Table Scan (Escaneo de tabla)**

* + SQL Server lee todas las filas de la tabla.
  + Ocurre cuando no hay índices útiles para la consulta.
  + Es costoso en rendimiento, especialmente en tablas grandes.
* Ejemplo típico:

SELECT \* FROM Clientes WHERE Ciudad = 'Madrid';

Si no hay índice sobre Ciudad, se revisan todas las filas.

**🔹 2. IndexScan (Escaneo de índice)**

* + SQL Server recorre todas las entradas de un índice (clustered o non-clustered).
  + Más rápido que el table scan, pero sigue leyendo muchas filas.
  + Se usa cuando hay un índice, pero el filtro no es muy selectivo (muchas coincidencias).

**🔹 3. IndexSeek (Búsqueda en índice) ✅**

* + SQL Server va directamente al lugar correcto en el índice y accede solo a las filas necesarias.
  + Es la forma más eficiente de acceso.
  + Requiere un índice útil y un filtro bien definido.
* Ejemplo:

SELECT \* FROM Clientes WHERE ID = 105;

Si hay un índice sobre ID, SQL Server hace un **seek** directamente a ese valor.

**4. BookmarkLookup (Key Lookup)**

* + Cuando se usa un índice non-clustered para encontrar una fila, pero se necesita leer columnas adicionales que no están en ese índice, SQL Server va a buscar esas columnas en la tabla base (si es heap) o al índice clustered.
  + Esto puede generar sobrecarga si ocurre muchas veces.

**🔹 5. RID Lookup**

* + Similar al Key Lookup, pero para tablas sin índice clustered (tablas "heap").
  + 📌 ¿Cómo saber qué tipo de acceso se está usando?
  + Puedes usar "Mostrar el plan de ejecución" en SQL Server Management Studio (SSMS):
  + Esto te muestra si SQL Server hace scan o seek, y te ayuda a optimizar la consulta=>

-- Activa el plan de ejecución

SET SHOWPLAN\_ALL ON;

-- O usa el botón "Incluir plan de ejecución estimado"

**Ventajas y desventajas de usar índices en SQL Server: (PPT-4)**

**✅ Ventajas de los índices**

1. **Aceleran las consultas**
   * Mejoran el rendimiento de búsquedas, filtros (WHERE), uniones (JOIN) y ordenamientos (ORDER BY).
2. **Reducen el uso de CPU y E/S**
   * Menos datos se leen desde disco o memoria, lo que mejora la eficiencia.
3. **Ayudan a mantener la integridad**
   * Índices únicos (como los de clave primaria) aseguran que no haya valores duplicados.
4. **Mejoran la experiencia del usuario**
   * Consultas más rápidas permiten aplicaciones más reactivas y eficientes.
5. **Soporte para consultas complejas**
   * Índices compuestos (varias columnas) ayudan a optimizar filtros múltiples.

**❌ Desventajas de los índices**

1. **Impactan el rendimiento de escritura**
   * Las operaciones INSERT, UPDATE y DELETE se vuelven más lentas porque los índices deben actualizarse.
2. **Consumen espacio en disco**
   * Cada índice ocupa espacio adicional, especialmente los índices compuestos o con columnas incluidas.
3. **Requieren mantenimiento**
   * Con el tiempo, los índices se fragmentan y deben ser reorganizados o reconstruidos (REBUILD o REORGANIZE).
4. **Exceso de índices puede perjudicar**
   * Demasiados índices pueden hacer que el optimizador elija un plan de ejecución subóptimo y aumenten el costo de mantenimiento.
5. **No siempre se usan**
   * Si no se diseñan bien o no coinciden con las consultas reales, simplemente no se aprovechan.

**🧠 Recomendación general:**

* + Analiza el uso real de las tablas (consultas más frecuentes).
  + Crea índices selectivos sobre columnas que se usan comúnmente en filtros o uniones.
  + Revisa y ajusta periódicamente los índices con base en estadísticas de uso.

**📌 Índice Agrupado (ClusteredIndex) (PPT-5)**

* + Ordena físicamente los datos de la tabla según el campo indexado.
  + Solo puede haber uno por tabla, ya que solo puede haber un orden físico.
  + Muy eficiente para búsquedas por rango o para ordenar resultados.
  + Suele estar en la clave primaria por defecto.

CREATE CLUSTERED INDEX idx\_IDON Clientes(ID);

**📌 Índice No Agrupado (Non-ClusteredIndex) (PPT-6)**

* + Crea una estructura separada que apunta a las filas de la tabla.
  + Puede haber varios en una misma tabla.
  + Útil para acelerar búsquedas sobre columnas no clave.
  + No afecta el orden físico de los datos.

CREATE NONCLUSTERED INDEX idx\_NombreON Clientes(Nombre);

📋**Cuadro comparativo: Clustered vs Non-Clustered (PPT-7)**

|  |  |  |
| --- | --- | --- |
| **Característica** | **Índice Agrupado (Clustered)** | **Índice No Agrupado (Non-Clustered)** |
| Orden físico de los datos | Sí (datos ordenados en disco) | No (estructura separada) |
| Cantidad por tabla | Solo uno | Varios |
| Estructura | Árbol B con datos en hojas | Árbol B con punteros a las filas |
| Velocidad de búsqueda | Muy alta (especialmente por rangos) | Alta (dependiendo del diseño) |
| Espacio adicional | No (usa la tabla misma) | Sí (estructura aparte) |
| Uso común | Clave primaria, ordenamientos | Columnas de búsqueda o filtro |

**Casi cualquier campo (columna)** se puede indexar, pero no todas las columnas se **deben** indexar. A continuación te explico qué **campos se pueden** y cuáles son los **más recomendados** para indexar.

**✅Campos que se pueden indexar**

Puedes indexar columnas de tipos comunes como:

* INT, BIGINT, VARCHAR, NVARCHAR, CHAR, DATETIME, DATE, DECIMAL, FLOAT, etc.
* Columnas que **no sean tipos complejos o grandes**, como:
  + ❌ TEXT, NTEXT, IMAGE, XML, VARBINARY(MAX), VARCHAR(MAX), etc.

⚠️ Estos tipos grandes **no se pueden indexar directamente**, aunque sí pueden estar como **columnas incluidas** (INCLUDE) en un índice no agrupado.

**⭐Campos más recomendados para indexar**

Se recomienda indexar columnas que:

1. **Se usan frecuentemente en filtros (WHERE)**
   * Ejemplo: UsuarioID, Estado, Fecha, Categoria.
2. **Participan en cláusulas JOIN**
   * Las columnas usadas para unir tablas se benefician mucho de índices.
3. **Se usan en ORDER BY o GROUP BY**
   * Un índice puede ayudar a evitar ordenamientos adicionales.
4. **Son altamente selectivas**
   * Es decir, tienen muchos valores distintos.
   * Ejemplo: CorreoElectronico o ID\_Producto.
5. **Están en columnas clave primaria o clave foránea**
   * SQL Server ya crea índices automáticamente en claves primarias y foráneas, pero es bueno saber que son esenciales.

**❌Columnas que NO conviene indexar**

Evita indexar columnas que:

* Tienen **muy pocos valores distintos** (poca selectividad), como Sexo, Activo, Sí/No.
* Se modifican con mucha frecuencia.
* Son **columnas grandes** o de tipo texto largo.
* No se usan en filtros ni uniones.

**📌Ejemplo recomendado:**

Supón que tienes esta tabla:

**CREATE TABLE Ventas (**

**VentaID INT PRIMARY KEY,**

**ClienteID INT,**

**Fecha DATE,**

**Monto DECIMAL(10,2),**

**Estado VARCHAR(20)**

**);**

**Índices útiles podrían ser**:

-- Ya hay índice clustered por ser PRIMARY KEY

-- Índice para buscar por cliente

**CREATE NONCLUSTERED INDEX idx\_ClienteID ON Ventas(ClienteID);**

-- Índice para búsquedas por fecha

**CREATE NONCLUSTERED INDEX idx\_Fecha ON Ventas(Fecha);**

-- Índice para búsquedas por estado

**CREATE NONCLUSTERED INDEX idx\_Estado ON Ventas(Estado);**