Hints en SQL Server

Los Hints en SQL Server (y en la mayoría de las bases de datos relaciones del mercado), son agregados a un comando SQL que indica que debe ejecutarse de manera diferente a la predeterminada por el motor. Existen 4 tipos de Hints diferentes:

- ✓ **Join Hints**: Especifican que tipo de Join (Mergue, Hash, Loop) vamos a usar en la query.
- ✓ **Index Hints**: Fuerzan el uso de uno o más índices en la ejecución de la query.
- ✓ **Lock Hints**: Especifican un tipo de *lockeo*.
- ✓ **Processing Hints**: Especifican una estrategia particular en la ejecución de la query.

En situaciones normales, no se debería utilizarlos, pero sin embargo, hay veces que es necesario. Por otra parte, en situaciones donde el volumen de información está en constante cambio, es importante recordar que un Hint puede funcionar bien el día de hoy, pero no tan bien la semana siguiente. Es conveniente retestear todos las queries que utilizan Hints periodicamente.

1. Join Hints:

- **HASH JOIN**: Se utiliza cuando se quiere forzar el uso del *HASH JOIN* entre dos tablas.

```
SELECT title_id,
       pub_name,
       title
FROM   Titles
       INNER HASH JOIN Publishers
    ON Titles.pub_id = Publishers.pub_id
```

- **MERGE JOIN**: Se utiliza cuando se quiere forzar el uso del *MERGE JOIN* entre dos tablas.

```
SELECT title_id,
       pub_name,
       title
FROM   Titles
       INNER MERGE JOIN Publishers
    ON Titles.pub_id = Publishers.pub_id
```

- **LOOP JOIN**: Se utiliza cuando se quiere forzar el uso del *LOOP JOIN* entre dos tablas.

```
SELECT title_id,
       pub_name,
       title
FROM   Titles
       INNER LOOP JOIN Publishers
    ON Titles.pub_id = Publishers.pub_id
```

Generalmente no deberíamos usar estos Hints, pero si por alguna razón el motor no está generando el plan de ejecución más óptimo, entonces es válida esta metodología. Pero hay que tener CUIDADO en usarlo.

2. Index Hints:

Este tipo de Hints, se utiliza cuando queremos forzar el uso de un índice en particular, para optimizar la consulta. El plan de ejecución generado por el SQL Server suele ser el más óptimo, pero en algunos casos excepcionales, no. Al igual que con los *Join Hints*, hay que tener cuidado a la hora de usarlos.

- Forzar un **Table Scan**: Poniendo como *ID de INDEX* el valor **0**

```
SELECT *
FROM   Authors WITH (INDEX(0))
```

- Forzar el uso del **Clustered Index**: Poniendo como *ID de INDEX* el valor **1**. En caso que no exista, tira error.

```
SELECT *
FROM   Authors WITH (INDEX(1))
```

- Forzar el uso de un **Non Clustered Index**: Poniendo como *ID* de *INDEX* el nombre del índice.

```
SELECT *
FROM   Authors WITH (INDEX(nombre_del_indice))
```

3. Lock Hints:

Por lejos, los más usados. Este tipo de Hints especifican que tipo de *lockeo* se debe efectuar en una operación. Existen varios Hints de este tipo, pero los más usados son *NOLOCK* y *ROWLOCK*.

- **NOLOCK**: (equivalente al *READUNCOMMITTED*): Se usa en la sentencia *SELECT*. Indica al motor que ignore los a *lockeos* exclusivos de datos y lea directamente de la tabla, lo que suele llamarse "lectura sucia". Con esto ganamos mayor performance y escalabilidad, pero al riesgo de leer datos de una transacción que todavía no finalizo, lo que significa una pérdida en la fiabilidad de los datos. Es un riesgo que tenemos que tener en cuenta. Por ejemplo, si queremos sacar un reporte de ventas entre dos periodos que ya terminaron, no tiene sentido realizar y verificar *lockeos* a la hora de leer, por lo cual un *NOLOCK* es totalmente válido. Pero si tenemos que leer datos entre periodos vigentes, donde pueden efectuarse transacciones, habría que evaluar el riesgo de leer datos que podrían llegar a ser inválidos.

  Algo importante que hay que aclarar, es que el *NOLOCK* no ignora todos los *lockeos*, de hecho, adquieren *lockeos* SCH-S (estabilidad del esquema). Por ejemplo, si se está corriendo un comando DDL que afecte a la tabla, esta adquiere un *lockeos* SCH-M (modificación del esquema), y por lo tanto, si se ejecuta una consulta aun teniendo el Hint *NOLOCK*, se bloqueara hasta que no termine la transacción anterior.

  En muchos sitios que requieren alta disponibilidad, se suele usar este Hint en prácticamente todas las sentencias SELECT, salvo en aquellas donde se quiere garantizar la integridad de los datos.

```
SELECT Count(*)
FROM   Usuarios WITH (NOLOCK)
INNER JOIN MenuUsuario WITH (NOLOCK)
    ON Usuarios.usuarioID = MenuUsuario.usuarioID
```

- **ROWLOCK**: Especifica que se apliquen bloqueos de fila cuando normalmente se aplicarían bloqueos de página o de tabla. Aplica solo a sentencias *UPDATE*, *DELETE* e *INSERT*. También, como en el caso del NOLOCK, este Hint sirve para ganar mayor performance en entornos muy concurrentes. Algo que hay que tener cuidado acá, es que si ocurren muchos *update* sobre la misma tabla, se puede saturar el servidor de tantos *lockeos* por fila

```
UPDATE Usuarios WITH (ROWLOCK)
SET    departamentoID = 20
WHERE  usuarioID = 1
```