Artículo
Ricardo Paiva · 9 sep, 2019 Lectura de 3 min

# Cómo mejorar el rendimiento de SQL en las consultas sobre el rango de fechas

¡Hola desarroladores!

¿Os parece que las consultas sobre el rango de fechas son demasiado lentas? ¿Os parece que el rendimiento de SQL es bajo? ¡Tengo un curioso truco que podría ayudaros a solucionar estos problemas! (¡Los desarrolladores de SQL odian que sepáis estas cosas!)*

Si tenéis una clase que guarda los registros de hora cuando se añaden datos, entonces esos datos se ordenarán con vuestros valores IDKEY, *es decir,* $TimeStamp_1 < TimeStamp_2$ *si y solo si la condición* $ID_1 < ID_2$ *se cumple para todos los valores ID y TimeStamp en la tabla* - entonces podéis utilizar esta información para aumentar el rendimiento de las consultas en relación con los rangos de TimeStamp. Echad un vistazo a la siguiente tabla:

```
Class User.TSOrder extends %Persistent
{

Property TS as %TimeStamp;

Property Data as %String (MAXLEN=100, MINLEN=200);

Index TSIdx on TS;

Index Extent [type=bitmap, extent];

}
```

Si añadimos 30 000 000 de filas aleatorias con las fechas de los últimos 30 días, se obtendrán 1 000 000 de filas por día. Ahora, si queremos consultar la información de un día específico, hay que escribir lo siguiente:

```
SELECT ID, TS, Data
FROM TSOrder
WHERE
     TS >= '2016-07-01 00:00:00.00000' AND
     TS <= '2016-07-01 23:59:59.999999'
```

Es una consulta razonable. Sin embargo, en mi sistema tomó 2 171 792 referencias globales y 7,2 segundos. Pero si sabemos que los IDs y los TimeStamps están en el mismo orden, podemos utilizar los TimeStamps para obtener un rango de los ID. Mirad la siguiente consulta:

```
SELECT ID, TS, Data
FROM TSOrder
WHERE
     ID >= (SELECT TOP 1 ID FROM TSOrder WHERE TS >='2016-07-01 00:00:00.00000' ORDER
 BY TS ASC) AND
     ID <= (SELECT TOP 1 ID FROM TSOrder WHERE TS <='2016-07-01 23:59:59.999999' ORDE
R BY TS DESC)
```

La nueva consulta se completa en 5,1 segundos, ¡y solo necesita 999 985 referencias globales**!

Esta técnica puede aplicarse de una manera más práctica a las tablas con más campos indexados y a las consultas que tengan varias condicionales WHERE. El rango del ID que se genera a partir de las subconsultas puede ponerse en un formato de mapa de bits, el cual genera una velocidad increíble cuando se obtiene una solución con varios índices. La tabla Ens.MessageHeader es un excelente ejemplo en la que se podría aplicar este truco.

Este es el resultado de un *EJEMPLO*. Si se tienen muchas sentencias del condicional WHERE en la misma tabla (y están indexadas, ¡obvio!), ¡entonces esta técnica puede ofrecer resultados mucho **MEJORES**! ¡Probadla en vuestras consultas!

* En realidad, los desarrolladores de SQL no odian que sepáis estas cosas, pero si algo nos ha enseñado Internet es que las frases llamativas atraen más tráfico.

** Cuando se prueban consultas que devuelven tantas filas, el SMP no puede gestionarlas, y la mayor parte del tiempo se dedica a mostrar los datos. La manera correcta de probarlas es con los métodos Embedded o Dynamic SQL, comprobando los resultados, pero sin generarlos en función del tiempo, y usando SQL Shell para los recuentos globales. También se puede utilizar las estadísticas de SQL para hacerlo.

#Code Snippet #Mejores prácticas #SQL #Caché

---

**URL de fuente:**https://es.community.intersystems.com/post/c%C3%B3mo-mejorar-el-rendimiento-de-sql-en-las-consultas-sobre-el-rango-de-fechas