Article
Ricardo Paiva · May 15, 2020 9m read

Gestión de índices

¡Hola desarrollador!

Si has leído la parte 1 de este artículo, ya tienes una buena idea del tipo de índices que necesitas para tus clases y cómo definirlos. Lo siguiente es saber cómo gestionarlos.

Plan de consultas

(**RECUERDA**: Al igual que cualquier modificación en una clase, añadir índices en un sistema en producción conlleva riesgos: si los usuarios están actualizando o accediendo a datos mientras se rellena un índice, podrían obtener resultados vacíos o incorrectos a sus consultas, o incluso dañar los índices que se están formando. Ten en cuenta que hay pasos adicionales para definir y usar índices en un sistema en producción. Estos pasos se analizarán en esta sección, y se detallan en nuestra documentación).

Cuando tengas un nuevo índice implementado, podemos ver si el optimizador de SQL decide que es el más eficiente para leer, al ejecutar la consulta. No hay que ejecutar la consulta para verificar el plan. Dada una consulta, puedes verificar el plan de forma programática:

```
Set query = 1

Set query(1) =  " SELECT SSN,Name FROM Sample.Person WHERE OfficeState = 'MA' "

        D $system.SQL.ShowPlan(.query)
```

O siguiendo la interfaz en el Portal de Administración del Sistema desde System Explorer -> SQL.

Desde aquí, se puede ver qué índices están siendo usados antes de cargar los datos de la tabla (o "mapa maestro"). Ten en cuenta lo siguiente: ¿tu nueva consulta está en el plan tal como se esperaba? ¿La lógica del plan tiene sentido?

Una vez sepas que el optimizador SQL está usando tus índices, puedes verificar que estos índices están funcionando correctamente.

Construcción de índices

(Si aún estás en las fases de planificación y todavía no tienes datos, los pasos detallados aquí no serán necesarios ahora.)

Definir un índice no lo poblará o "construirá" automáticamente con los datos de tu tabla. Si el plan de consulta usa un índice que aún no se ha construido, corres el riesgo de obtener resultados incorrectos o vacíos a la consulta. Para "desactivar" un índice antes de que esté listo para usar, pon su elegibilidad de mapa en 0 (esto básicamente le indica al optimizador SQL que no puede usar este índice para ejecutar consultas).

write $SYSTEM.SQL.SetMapSelectability("Sample.Person","QuickSearchIDX",0) ; Set selectability of index QuickSearchIDX false

Ten en cuenta que puedes usar la llamada de arriba incluso antes de que añadas un nuevo índice. El optimizador SQL reconocerá este nuevo nombre del índice, sabrá que está inactivo y no lo usará para ninguna consulta.

Puedes rellenar un índice usando el método %BuildIndices (##class(<class>).%BuildIndices($lb("MyIDX"))) o en la página SQL del Portal de Administración del Sistema (bajo el menú desplegable "Actions").

El tiempo necesario para construir índices depende de la cantidad de filas de la tabla y de los tipos de índices que estás construyendo. Los índices bitslices generalmente requieren de más tiempo.

Una vez completado el proceso, puedes usar el método SetMapSelectivity una vez más (esta vez ajuste en 1) para reactivar tu índice ahora poblado.

Ten en cuenta que construir índices significa esencialmente emitir comandos KILL y SET para poblarlos. Puedes considerar desactivar **journaling** durante este proceso para evitar llenar el espacio de disco.

Puedes encontrar instrucciones más detalladas sobre construir índices, en particular en sistemas en producción, en nuestra documentación:

https://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GSQLOPTindices#GSQLOPTindicesbuildreadwrite

Mantenimiento de índices

¡Ahora ya podemos usar nuestros índices! Algunos puntos a considerar son la eficiencia de ejecución de una consulta, la frecuencia de uso de un índice dado y los pasos a tomar si tus índices entran en un estado inconsistente.

Rendimiento

Primero, podemos evaluar cómo ha afectado este índice el rendimiento de la consulta. Si sabemos que el optimizador SQL está usando un nuevo índice para una consulta, podemos ejecutar la consulta para obtener sus estadísticas de desempeño: número de referencias globales, número de líneas leídas, tiempo para preparar y ejecutar la consulta y tiempo empleado en el disco.

Volvamos al ejemplo anterior:

```
"SELECT SSN,Name,DOB FROM Sample.Person WHERE Name %STARTSWITH 'Smith,J' "
```

Tenemos el siguiente índice para ayudar al desempeño de esta consulta:

```
Index QuickSearchIDX On Name [ Data = (SSN, DOB, Name) ];
```

Ya tenemos un índice NameIDX en la propiedad Name.

Intuitivamente, nos damos cuenta de que la consulta se ejecutará usando QuickSearchIDX. NameIDX seguramente sería una segunda opción, ya que está basado en la propiedad Name, pero no contiene ninguno de los valores de datos para SSN o DOB.

Para comprobar el rendimiento de esta consulta con QuickSearchIDX, basta con ejecutarla.

(Por otra parte: he depurado las consultas en caché antes de ejecutar estas consultas para mostrar mejor las diferencias de rendimiento. Cuando se ejecuta una consulta SQL, almacenamos el plan usado para ejecutarla, lo que mejora el rendimiento en su próxima ejecución. Los detalles de esto están fuera del alcance de este artículo, pero al final de este artículo incluiré recursos adicionales para otras consideraciones de desempeño de SQL.)

Query Plan

Relative cost = 1856

- Read index map Sample.Person.QuickSearchIDX, looping on %SQLUPPER(Name) (with a %STARTSWITH range condition) and ID.
- For each row:

  Output the row.

Cantidad de filas: **31** Rendimiento: **0.003** segundos **154** referencias globales **3264** líneas ejecutadas **1** latencia de lectura de disco (ms)

Simple: solo necesitamos QuickSearchIDX para ejecutar la consulta.

Vamos a comparar el desempeño de usar NameIDX en lugar de QuickSearchIDX. Para hacer esto podemos añadir una palabra clave %IGNOREINDEX para la consulta, que evitará que el optimizador SQL elija ciertos índices.

Reescribimos la consulta de la siguiente forma:

“SELECT SSN,Name,DOB FROM %IGNOREINDEX QuickSearchIDX Sample.Person WHERE Name %STARTSWITH 'Smith,J' ”

El plan de consulta ahora usa NameIDX, y vemos que debemos leer desde el global de datos (“ mapa maestro ”) de Sample.Person usando los IDs de filas relevantes, encontrados mediante el índice.

Query Plan

Relative cost = 24763

- Call module B, which populates bitmap temp-file A.
- Read bitmap temp-file A, looping on ID.
- For each row:

  Read master map Sample.Person.IDKEY, using the given idkey value.
  Output the row.

module B

- Read index map Sample.Person.NameIDX, looping on %SQLUPPER(Name) (with a %STARTSWITH range condition) and ID.
- For each row:

  Add ID bit to bitmap temp-file A.

Cantidad de filas: **31** Rendimiento: **0.020** segundos **137** referencias globales **3792** líneas ejecutadas **17** latencia de lectura de disco (ms)

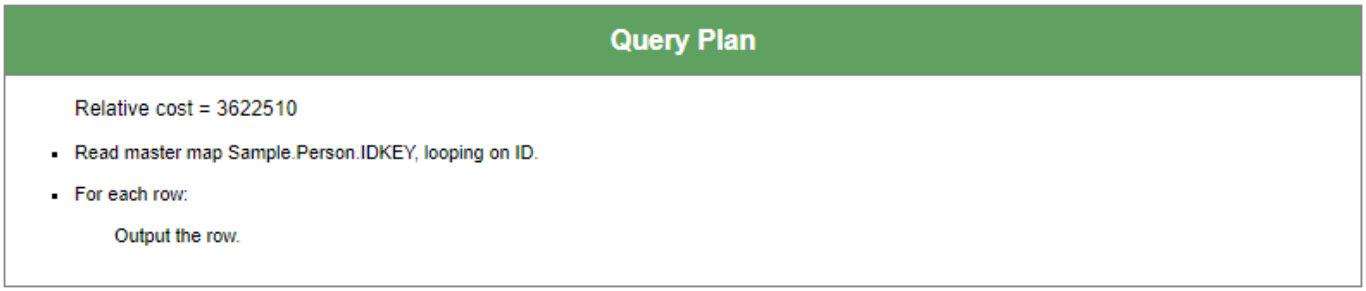
Vemos que es necesario más tiempo para la ejecución, más líneas de código ejecutadas y una mayor latencia de disco.

A continuación, considera ejecutar esta consulta sin ningún índice. Ajustamos nuestra consulta de la siguiente forma:

“SELECT SSN,Name,DOB FROM %IGNOREINDEX * Sample.Person WHERE Name %STARTSWITH

'Smith,J' ”

Sin usar índices, debemos comprobar nuestra condición en las filas de datos.



Cantidad de filas: **31** Rendimiento: **0.765** seconds **149999** referencias globales **1202681** líneas ejecutadas **517** latencia de lectura de disco (ms)

Esto muestra que un índice especializado, QuickSearchIDX, permite a nuestra consulta ejecutarse más de 100 veces más rápido que sin índice y casi 10 veces más rápido que con el más general NameIDX, y usar NameIDX permite una ejecución al menos 30 veces más rápido que sin usar índices.

Para este ejemplo específico, la diferencia de rendimiento entre usar QuickSearchIDX y NameIDX puede ser despreciable, pero para consultas que se ejecutan cientos de veces por día con millones de filas a consultar, veríamos que se ahorra un tiempo muy valioso cada día.

Análisis de índices existentes usando SQLUtilites

%SYS.PTools.SQLUtilities incluye procedimientos como IndexUsage, JoinIndices, TablesScans y TempIndices. Estos analizan las consultas existentes en cualquier namespace dado y ofrecen información sobre la frecuencia de uso de cualquier índice dado, qué consultas eligen iterar sobre cada fila de una tabla y qué consultas generan archivos temporales para simular índices.

Puedes usar estos procedimientos para determinar "gaps" (falta de datos) que un índice podría solucionar. De igual manera puedes usar estos procedimientos para eliminar algún indice que no se utilice o que no sea eficiente.

Puedes encontrar detalles de estos procedimientos y ejemplos de su uso en nuestra documentación:

https://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GSQLOPToptqueryindexanalysis

¿Problemas de índices?

Validar índices confirma si un índice existe y está definido correctamente para cada fila de una clase. Ninguna clase debería entrar en un estado en el que los índices estén dañados, pero si descubres que las consultas están devolviendo conjuntos de resultados vacíos o incorrectos, podrías considerar verificar si los índices de las clases existentes son válidos actualmente.

Puedes validar índices de forma programática así:

```
Set status = ##class(<class>).%ValidateIndices(indices,autoCorrect,lockOption,multiProcess)
```

Aquí, el parámetro "índices" es una cadena vacía de forma predeterminada, lo que significa que validamos todos los índices o un objeto $listbuild que contenga los nombres de índices.

Ten en cuenta que autoCorrect tiene como valor predeterminado 0. Si es 1, cualquier error encontrado durante el proceso de validación será corregido. Si bien esta funcionalidad es la misma que reconstruir los índices, el rendimiento de ValidateIndices es más lento comparativamente.

Puedes consultar la documentación de la clase %Library.Storage para más información:

https://docs.intersystems.com/latest/csp/documatic/%25CSP.Documatic.cls?PAGE=CLASS&LIBRARY=%25SYS&CLASSNAME=%25Library.Storage#%ValidateIndices

Eliminación de índices

Si ya no necesitas un índice (o si estás planeando hacer una gran cantidad de modificaciones a una tabla y quieres guardar para más tarde el impacto sobre el rendimiento de construir los índices relevantes) puedes simplemente eliminar la definición del índice de la clase en Studio y eliminar el nodo global de índice apropiado. O puedes ejecutar un comando DROP INDEX vía DDL, lo que también limpiará la definición y los datos del índice. Desde ahí, puedes depurar las consultas en caché para asegurarte de que ningún plan existente use el índice ya eliminado.

¿Y ahora qué?

Los índices son solo una parte del desempeño de SQL. En este mismo sentido, existen otras opciones para monitorizar el rendimiento y el uso de tus índices. Para entender el desempeño de SQL, también puedes aprender:

Tune Tables – una herramienta que se ejecuta una vez poblada una tabla con datos representativos o si la distribución de los datos cambia radicalmente. Esto llena la definición de tu clase con metadatos. Por ejemplo, qué longitud esperamos que tenga un campo o cuántos valores únicos hay en un campo, lo que ayuda al optimizador SQL a elegir un plan de consulta que permita una ejecución eficiente.

Kyle Baxter escribió un artículo sobre esto: https://community.intersystems.com/post/one-query-performance-trick-you-need-know-tune-table

Query Plans – la representación lógica de cómo nuestro código subyacente ejecuta consultas SQL. Si tienes una consulta lenta, podemos analizar qué plan de consulta se está generando, si tiene sentido para tu consulta y qué se puede hacer para optimizar más este plan.

Cached queries – Declaraciones SQL dinámicas preparadas – las consultas guardadas en caché son esencialmente el código subyacente de los planes de consulta.
cached queries are essentially the code underneath query plans.

Para saber más

Parte 1 de este artículo.

Documentación sobre definición y construcción de índices. Incluye pasos adicionales a considerar en sistemas de lectura-escritura en producción. https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=GSQLOPTindices

Comandos SQL ISC: vea CREATE INDEX y DROP INDEX por referencias sintácticas de manejo de índices vía DDL. Incluye permisos de usuario adecuados para ejecutar estos comandos. https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQLCOMMANDS

Detalles sobre Collation SQL en clases ISC. De forma predeterminada, los valores de cadenas se almacenan en globales de índices como SQLUPPER (" STRING "):
https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=GSQLcollation

[EDIT 05/06/2020: Corrections on index build performance and the DROP INDEX command.]

#Indexing #Performance #SQL #Caché #InterSystems IRIS