Artículo
Estevan Martinez · Feb 11, 2020 Lectura de 7 min

Almacenamiento de datos - Información que se debe conocer para tomar buenas decisiones durante la programación

Esta publicación es el resultado directo de trabajar con un cliente de InterSystems que acudió a mí con el siguiente problema:

```
SELECT COUNT(*) FROM MyCustomTable
```

Esto tarda 0.005 segundos, con 2300 filas en total. Sin embargo:

```
SELECT * FROM MyCustomTable
```

Tardó algunos minutos. La razón de que esto sucediera es lo suficientemente sutil e interesante para mí como para escribir una publicación sobre ello. Esta publicación es extensa, pero si va hasta el final verá que escribí un resumen rápido. Por ello, si llegó hasta aquí y cree que ya leyó demasiado, desplácese hasta el final para leer únicamente las ideas principales. Busque la frase que se encuentra en **negrita**.

Cuando esté creando sus clases, hay un aspecto a tener cuenta relacionada con el almacenamiento. Como muchos ya saben, todos los datos en Caché se almacenan en *globals*.

<Paréntesis>

Si no sabe esto, creo que esta publicación va a ser un poco extensa. Recomiendo echar un vistazo a este excelente tutorial que se encuentra en nuestra documentación:

ObjectScript Tutorial

Si nunca ha utilizado Caché/Ensemble/HealthShare, el tutorial anterior es muy útil. E incluso si ya los ha utilizado, ¡ vale la pena revisarlo!

</Paréntesis>

Ahora, debido a que todos los datos se almacenan en *globals*, es importante que comprenda cómo las definiciones de su clase se mapean con los *globals*. ¡ Creemos una aplicación juntos! Repasaremos algunos de los errores más comunes y discutiremos cómo el desarrollo de su clase afecta a sus estrategias de almacenamiento, con atención especial al rendimiento de SQL.

Imaginemos que somos la Oficina del Censo de Estados Unidos y necesitamos una base de datos para almacenar la información de todas las personas que viven en el país. Entonces construiremos una clase como esta:

```
Class USA.Person extends %Persistent

{
Property Name as %String;
Property SSN as %String;
Property Address as %String;
Property DateOfBirth as %Date;
```

```
}
```

SSN significa "Social Security Number (Número de Seguridad Social)" que, aunque originariamente no estaba destinado a funcionar como un número para identificar a las personas, en la práctica es su número de identificación. Sin embargo, somos tradicionalistas, así que no lo utilizaremos para el ID. Dicho esto, a la vez queremos que se indexe, ya que es una excelente manera de buscar a alguien. Sabemos que algunas veces tendremos que buscar a las personas por su nombre, así que también queremos un índice de nombres. Y como a nuestro jefe le encantan los informes basados en grupos de edad, creemos que también sería bueno incluir un índice para la fecha de nacimiento (DOB). Así que vamos a añadir todo esto a nuestra clase

```
Class USA.Person extends %Persistent

{
Property Name as %String;
Property SSN as %String;
Property Address as %String;
Property DateOfBirth as %Date;

Index NameIDX On Name;
Index SSNIDX On SSN [Unique];
Index DOBIDX on DateOfBirth;
}
```

Muy bien. Agreguemos una fila y veamos cómo son nuestros *globals*. Nuestra sentencia INSERT se ve de la siguiente manera:

```
INSERT INTO USA.Person (Name,SSN,Address,DateOfBirth) VALUES
   ('Baxter, Kyle','111-11-1111','1 Memorial Drive, Cambridge, MA 02142','1985-07-20'
)
```

Y el *global*:

```
USER>zw ^USA.PersonD
^USA.PersonD=1
^USA.PersonD(1)=$lb(
"","Baxter, Kyle","111-11-1111","1 Memorial Drive, Cambridge, MA 02142",52796)
```

El almacenamiento predeterminado para una clase almacena sus datos en ^Package.ClassD. Si el nombre de la clase es demasiado extenso, puede descomponerse en otros más pequeños, y puede encontrarlo en la Definición de Almacenamiento que se encuentra en la parte inferior de su definición de clase. ¿ Cómo se ven los índices?

```
USER>zw ^USA.PersonI
^USA.PersonI("DOBIDX",52796,1)=""
^USA.PersonI("NameIDX"," BAXTER, KYLE",1)=""
^USA.PersonI("SSNIDX"," 111-11-1111",1)=""
```

¡ Genial! Nuestro almacenamiento se ve bastante bien hasta ahora. De modo que añadimos nuestros 320 millones de personas y podemos conseguir más personas muy rápidamente. Pero ahora tenemos un problema, porque queremos tratar al presidente y a todos los ex presidentes con algunas consideraciones especiales. Entonces, agregaremos una clase especial para el presidente:

```
Class USA.President extends USA.Person

{
```

```
Property PresNumber as %Integer;

Index PresNumberIDX on PresNumber;
}
```

Bien. Debido a la herencia, obtuvimos todas las propiedades de USA.Person, y agregamos una que nos permitiera saber qué número de presidente era. Ya que quiero ser lo menos político posible, colocaré INSERT para añadir a nuestro SIGUIENTE presidente. Esta es la sentencia:

```
INSERT INTO USA.President (Name,SSN,DateOfBirth,Address,PresNumber) VALUES ('McDonald
,Ronald','221-18-7518','01-01-1963','1600 Pennsylvania Ave NW, Washington, DC 20006',
45)
```

Nota: Su SSN significa 'burger'. Lo siento mucho si es su SSN.

¡ Genial! Veamos el *global* de su presidente:

```
USER>zw ^USA.PresidentD
```

¡ No hay datos! Y aquí es donde llegamos al motivo de esta publicación. Debido a que PRIMERO decidimos heredarlo desde USA.Person, no solo heredamos sus propiedades e índices, ¡ también obtuvimos su almacenamiento! Entonces, para localizar al presidente McDonald tenemos que buscarlo en ^USA.PersonD. Podemos ver lo siguiente:

```
^USA.PersonD(2)=$lb(
"~USA.President~","McDonald,Ronald","221-18-7518","1600 Pennsylvania Ave NW, Washingt
on, DC 20006",44560)
^USA.PersonD(2,"President")=$lb(45)
```

Aquí destacamos dos cosas importantes. La primera es que podemos ver que el nodo (2) tiene toda la información que ya se almacenó en USA.Person. Mientras que el nodo (2, "President") solamente tiene la información específica para la clase USA.President.

¿ Qué significa esto en la práctica? Bueno, si queremos realizar: SELECT * FROM USA.President entonces NECESITAREMOS revisar toda la tabla de personas. Si esperamos que la tabla de personas tenga 320,000,000 filas, y la tabla de presidentes 45, ¡ entonces debemos hacer más de 320,000,045 referencias globales para extraer 45 filas! De hecho, si miramos el plan de la consulta:

- `Read master map USA.President.IDKEY, looping on ID.`
- `For each row:`
- `Output the row.`

Vemos exactamente lo que esperábamos. Sin embargo, ya vimos que esto significa realizar búsquedas en todo el *global* ^USA.PersonD. Por tanto, esta será una referencia global de más de 320,000,000 ya que necesitamos probar CADA ^USA.PersonD para comprobar si existen datos en ^USA.PersonD(i, "Presidente"), pues no sabemos cuáles de entre todas las personas serán los presidentes. Esto no está muy bien. ¡ No es lo que queríamos hacer! ¡ ¿ Qué podemos hacer ahora?! Bueno, tenemos 2 opciones:

Opción 1

Agregar un índice de extensión. Si lo hacemos, obtendremos una lista de los ID, de modo que sabremos qué personas son presidentes y podemos utilizar esa información para leer nodos específicos del global ^USA.Person. Debido a que tengo un almacenamiento predeterminado, puedo utilizar un índice para mapas de bits, el cual hará que este proceso se realice aún más rápidamente. Agregamos el índice de esta manera:

```
Index Extent [Type=Bitmap, Extent];
```

Y cuando buscamos SELECT * FROM USA.President en nuestro plan de consulta, podemos observar:

- `Read extent bitmap USA.President.Extent, looping on ID.`
- `For each row:`
- `Read master map USA.President.IDKEY, using the given idkey value.`
- `Output the row.`

Muy bien, ahora esto va a ser rápido. Una referencia global para revisar la extensión, y después 45 más para los presidentes. Eso es bastante eficiente.

¿ Cuáles son los inconvenientes? Bueno, unir esta tabla lo vuelve un poco más problemático y puede involucrar más tablas temporales de las que le gustaría tener.

Opción 2

Cambiar la definición de clase por:

```
Class USA.President extends (%Persistent, USA.Person)
```

Hacer que %Persistent sea la primera clase extendida significaría que USA.President obtendrá su propia definición de almacenamiento. Entonces, los presidentes se almacenarán de la siguiente manera:

```
USER>zw ^USA.PresidentD
^USA.PresidentD=1
^USA.PresidentD(1)=$lb(
"","McDonald,Ronald","221-18-7518","1600 Pennsylvania Ave NW, Washington, DC 20006",4
4560,45)
```

Esto está bien, porque seleccionarlo desde USA.President. significa que podrá leer los 45 miembros de este *global*. Es agradable, fácil y tiene un diseño limpio.

¿ Cuáles son los inconvenientes? Bueno, ahora los Presidentes NO están en la tabla de personas (Person). De modo que si quiere la información acerca de los presidentes y las personas que no son presidentes, es necesario que haga lo siguiente *SELECT ... FROM USA.Person UNION ALL SELECT ... FROM USA.President*

Si dejó de leer al principio, siga leyendo aquí:

Cuando creamos herencias, tenemos dos opciones

Opción 1: Heredar desde la primera superclase. Esto almacena los datos en el mismo *global* que la superclase. Hacer esto es útil si quiere tener toda la información junta, y puede mitigar los problemas de rendimiento en la subclase mediante un índice de extensión.

Opción 2: Heredar primero desde %Persistent. Esto almacena los datos en un nuevo *global*. Hacer esto es útil si va a consultar mucho la subclase. Sin embargo, si quiere ver los datos tanto de la superclase como de la subclase, necesita utilizar una consulta de tipo UNION.

¿ Cuál de estas opciones es mejor? Bueno, eso depende de cómo va a utilizar su aplicación. Si va a querer hacer muchas consultas sobre todos los datos juntos, entonces probablemente prefiera el primer método. Sin embargo, si cree que nunca tendrá que consultar todos los datos juntos, entonces probablemente prefiera utilizar el segundo método. Ambos son bastante buenos, siempre y cuando recuerde el índice de extensión en la Opción 1.

¿ Preguntas? ¿ Comentarios? ¿ Se siente confundido? Puede escribir en la parte de abajo.

¡ Muchas gracias!

#Consejos y trucos #Globals #Modelo de datos #Modelo de datos de objetos #ObjectScript #SQL #Tutorial #Caché #InterSystems IRIS

URL de fuente: https://es.community.intersystems.com/post/almacenamiento-de-datos-informaci%C3%B3n-que-se-debe-conocer-para-tomar-buenas-decisiones-durante-la