Artículo
Kurro Lopez · 22 oct, 2019 Lectura de 8 min

Los Globals son espadas mágicas para administrar datos. Parte 1

¡Hola a tod@s!

En este artículo voy a hablar sobre los Globals, esas espadas mágicas para almacenar datos, que han estado con nosotros desde hace tiempo, pero no mucha gente las utiliza de forma eficiente o realmente conoce esta **súper herramienta**.

Si se utilizan globals para realizar las tareas en donde realmente brillan, los **resultados** pueden ser **sorprendentes**, ya sea en términos de un mayor rendimiento o en una simplificación drástica de la solución en general (1, 2).

Globals ofrecen una forma especial de almacenar y procesar datos, la cual es completamente diferente de las tablas SQL. Se introdujeron por primera vez en 1966 con el lenguaje de programación M(UMPS), donde inicialmente se utilizaron en las bases de datos médicas. Todavía se usan de la misma manera, pero también fueron adoptados por otras industrias donde la confiabilidad y el alto rendimiento son la máxima prioridad (como en las finanzas, las operaciones comerciales, etc.)

Posteriormente, el M(UMPS) evolucionó hasta convertirse en Caché ObjectScript (COS), el cual se desarrolló por InterSystems como un superconjunto de M. El lenguaje original aún es aceptado por la Comunidad de Desarrolladores y persiste en algunas implementaciones. Existen varias señales de que todavía tiene actividad en la web: Grupos de Google sobre MUMPS, Grupo de usuarios de Mumps, Normas ISO vigentes, etc.

Los globals que se basan en el moderno Sistema de Gestión de Bases de Datos (DBMS) son compatibles con transacciones, registros, replicaciones y particiones. Esto significa que pueden utilizarse para desarrollar sistemas de distribución que sean modernos, confiables y rápidos.

Los globals no lo restringirán a las limitaciones del modelo relacional. Más bien, le darán libertad para crear estructuras de datos optimizadas con el fin de realizar tareas particulares. Para muchas aplicaciones, el uso razonable de los globals puede ser una verdadera solución inmediata, la cual ofrece velocidades que los desarrolladores de aplicaciones relacionales convencionales solamente pueden soñar.

Los globals funcionan como un método para almacenar datos que puede utilizarse con muchos lenguajes de programación modernos, ya sean de nivel superior o inferior. Por lo tanto, este artículo se centrará específicamente en los globals y no en el lenguaje del que proceden.

2. Cómo funcionan los globals

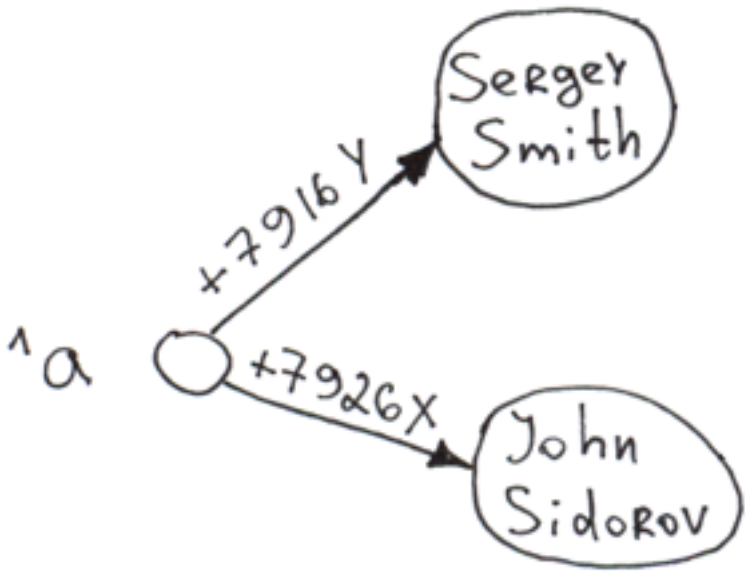
Primero, entenderemos cómo funcionan los globals y cuáles son las ventajas de utilizarlos. Los globals pueden verse desde diferentes perspectivas. En esta parte del artículo los analizaremos como árboles o como almacenes jerárquicos de datos.

Dicho de forma sencilla, un global es un **conjunto persistente**. Un conjunto que se guarda automáticamente en el disco.

Es difícil imaginar algo más sencillo para almacenar datos. En el código del programa (escrito con el lenguaje COS/M), la única diferencia que existe respecto a un conjunto asociativo normal es el símbolo ^ que se encuentra antes de sus nombres.

No es necesario tener conocimientos previos de SQL para guardar datos en un global, ya que todos los comandos necesarios son realmente sencillos y pueden aprenderse en una hora.

Comencemos con el ejemplo más sencillo, un árbol de un solo nivel que tiene dos ramas. Los ejemplos están escritos en COS.



```
Set ^a("+7926X") = "John Sidorov"
Set ^a("+7916Y") = "Sergey Smith"
```

Cuando los datos se insertan en un global (mediante el comando Set), automáticamente suceden 3 cosas:

1. **Se guardan** los datos en el disco.
2. **Se crean índices**. Lo que está entre paréntesis es un subíndice, y lo que está a la derecha del signo igual es el valor del nodo "a".
3. **Se ordenan**. Los datos se ordenan mediante un elemento. El próximo recorrido colocará a «Sergey Smith» en la primera posición, seguido por «John Sidorov». Cuando se obtiene una lista de usuarios desde un global, la base de datos no dedica tiempo a ordenarla. De hecho, puede solicitar que la lista comience a ordenarse desde cualquier elemento, incluso cuando no exista uno (la salida comenzará a partir del primer elemento verdadero y seguirá después de este).

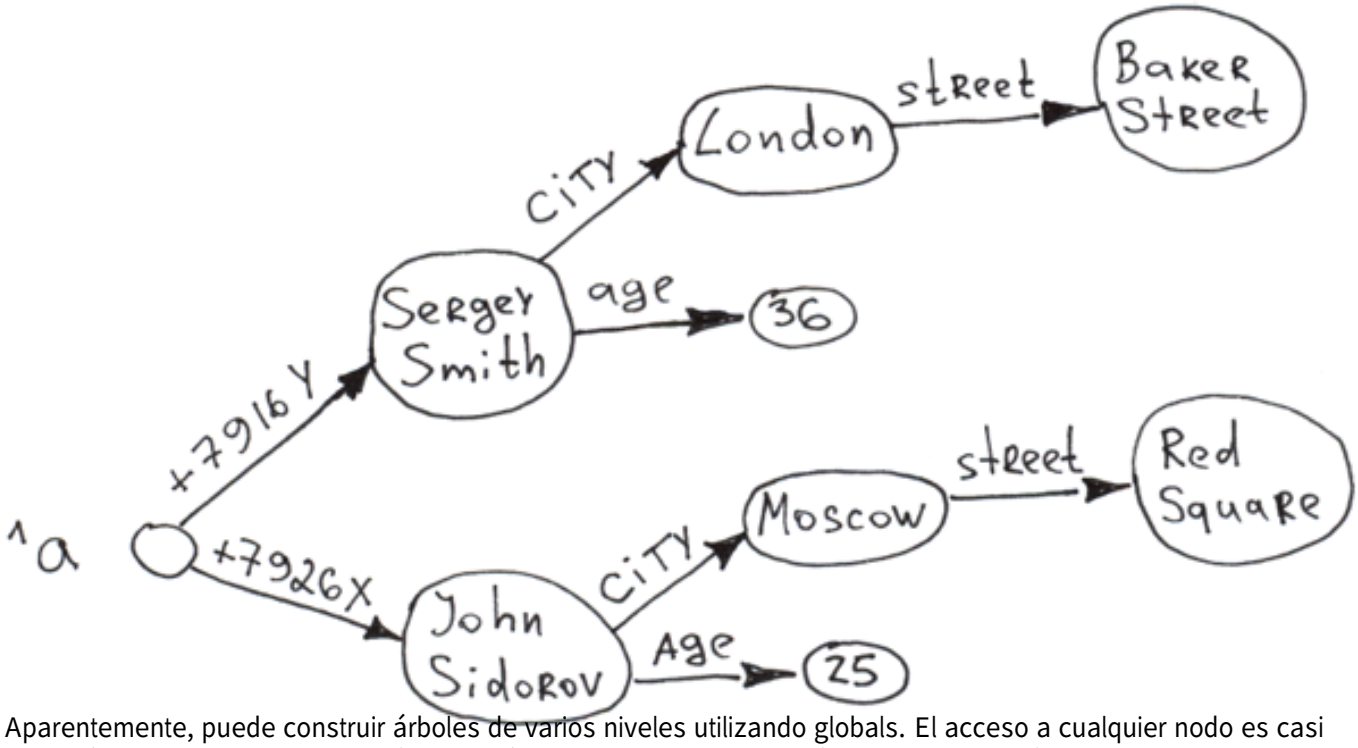
Todas estas operaciones se realizan a una velocidad increíble. En mi equipo doméstico (que cuenta con las siguientes especificaciones: i5-3340, 16GB, HDD WD 1TB Blue) logré alcanzar las 1,050,000 inserciones/segundo en un solo proceso. En equipos con procesadores multinúcleo, las velocidades pueden alcanzar docenas de milliones de inserciones/segundo.

Desde luego, la velocidad de inserción del registro no proporciona mucha información. Por ejemplo, podemos escribir los datos en archivos de texto, según los rumores, así es como funciona el procesamiento de los datos en Visa. Sin embargo, con los globals, obtenemos un almacenamiento estructurado e indexado que permite trabajar mientras se disfruta de su alta velocidad y facilidad de uso.

- **La mayor fortaleza de los globals es la velocidad con que es posible insertar nuevos nodos dentro de ellos**
- **Los datos siempre se indexan en un global. Los recorridos en los árboles con un solo nivel y profundidad siempre son muy rápidos**

Vamos a añadir algunas ramas de segundo y tercer nivel en el global.

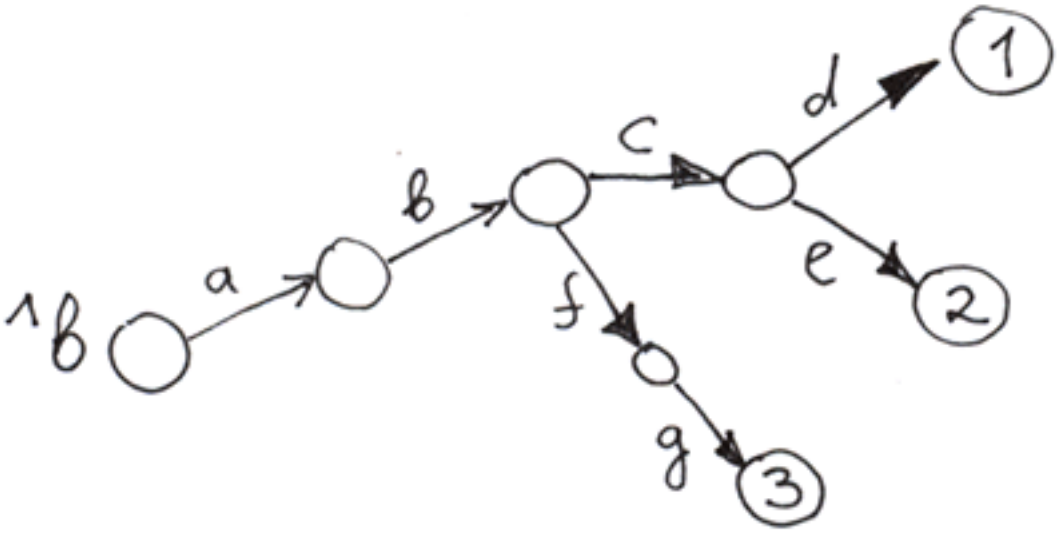
```
Set ^a("+7926X", "city") = "Moscow"
Set ^a("+7926X", "city", "street") = "Req Square"
Set ^a("+7926X", "age") = 25
Set ^a("+7916Y", "city") = "London"
Set ^a("+7916Y", "city", "street") = "Baker Street"
Set ^a("+7916Y", "age") = 36
```



Aparentemente, puede construir árboles de varios niveles utilizando globals. El acceso a cualquier nodo es casi instantáneo gracias a la indexación automática que se realiza después de cada inserción. Las ramas de los árboles de cualquier nivel se ordenan mediante un elemento.

Como puede ver, tanto los datos como los valores pueden almacenarse en elementos. La longitud combinada de un elemento (la suma de las longitudes de todos los índices) puede alcanza 511 bytes, y los valores pueden alcanzar hasta 3.6 MB de tamaño en el Caché. El número de niveles de un árbol (número de dimensiones) se limita a 31.

Una última cosa que también es genial: puede construir un árbol sin definir los valores de los nodos de nivel superior.

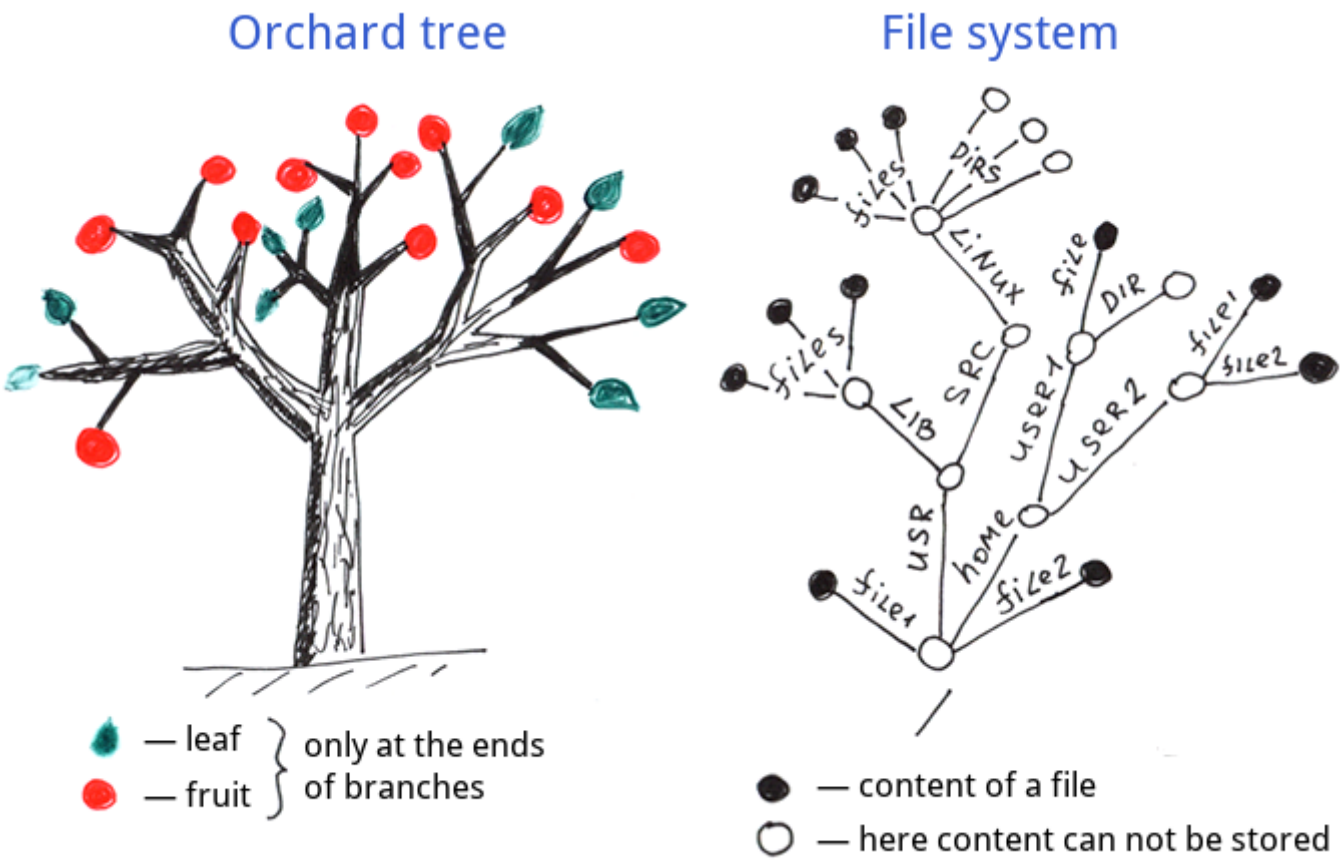



```
Set ^b("a", "b", "c", "d") = 1
Set ^b("a", "b", "c", "e") = 2
Set ^b("a", "b", "f", "g") = 3
```

Los círculos vacíos son nodos que no tienen ningún valor.

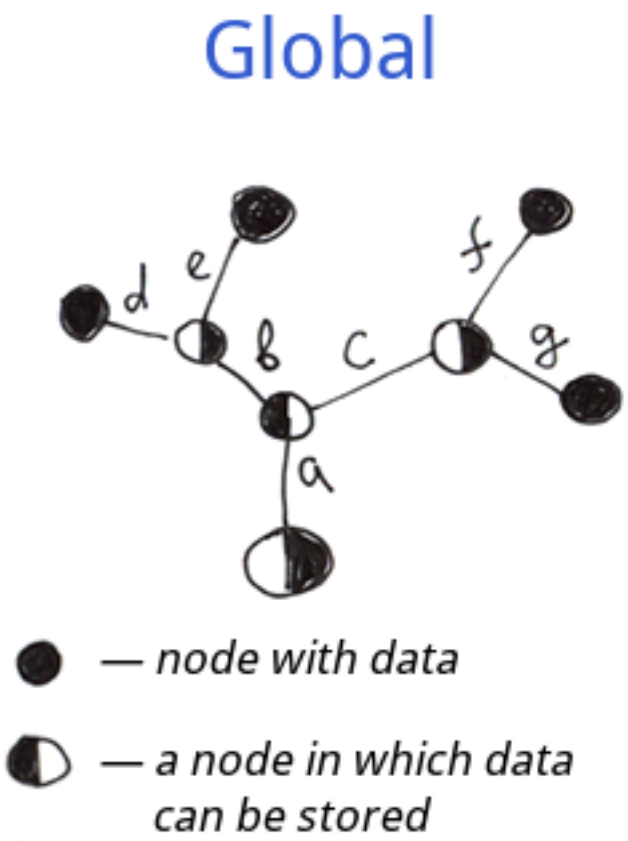
Para entender mejor los globals, los compararemos con otros árboles: los árboles en los jardines y los árboles de nombres en el sistema de archivos.

Compararemos los globales con las estructuras jerárquicas más comunes: los árboles que generalmente crecen en los jardines y campos, al igual que los sistemas de archivos.



Como podemos ver, las hojas y los frutos solamente crecen en los extremos de las ramas de los árboles comunes. Sistemas de archivos – La información también se almacena en los extremos de las ramas, también conocida como nombres completos de los archivos.

Aquí puede ver la estructura de los datos en un global.



Las diferencias son:

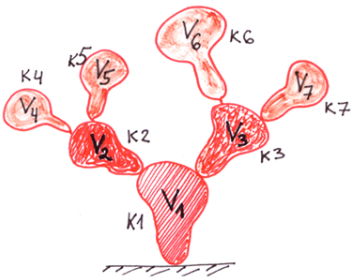
1. **Nodos internos:** en un global la información puede almacenarse en cada nodo, no solo en los extremos de las ramas
2. **Nodos externos:** los globals deben tener extremos definidos en sus ramas (extremos con valores), aunque esto no es obligatorio para el sistema de archivos y los árboles en los jardines

En cuanto a los nodos internos, podemos referirnos a la estructura del global como un superconjunto que contiene los árboles con nombres de los sistemas de archivos y las estructuras que conforman los árboles en un jardín. De modo que la estructura del global es más flexible.

En general, un global es un **árbol estructurado que es compatible con el almacenamiento de datos en cada nodo.**

Para entender mejor cómo funcionan los globals, imaginemos: ¿qué pasaría si los creadores de un sistema de archivos utilizaran un enfoque idéntico al de los globals para almacenar información?

1. Si el último archivo de una carpeta se eliminara, también se eliminaría la misma carpeta, así como todas las carpetas de nivel superior que solamente contenían la carpeta que se eliminó.
2. No habría necesidad de utilizar carpetas. Tendríamos archivos con subarchivos y archivos sin subarchivos. Si compara esto con un árbol normal, cada rama se convertiría en una fruta.



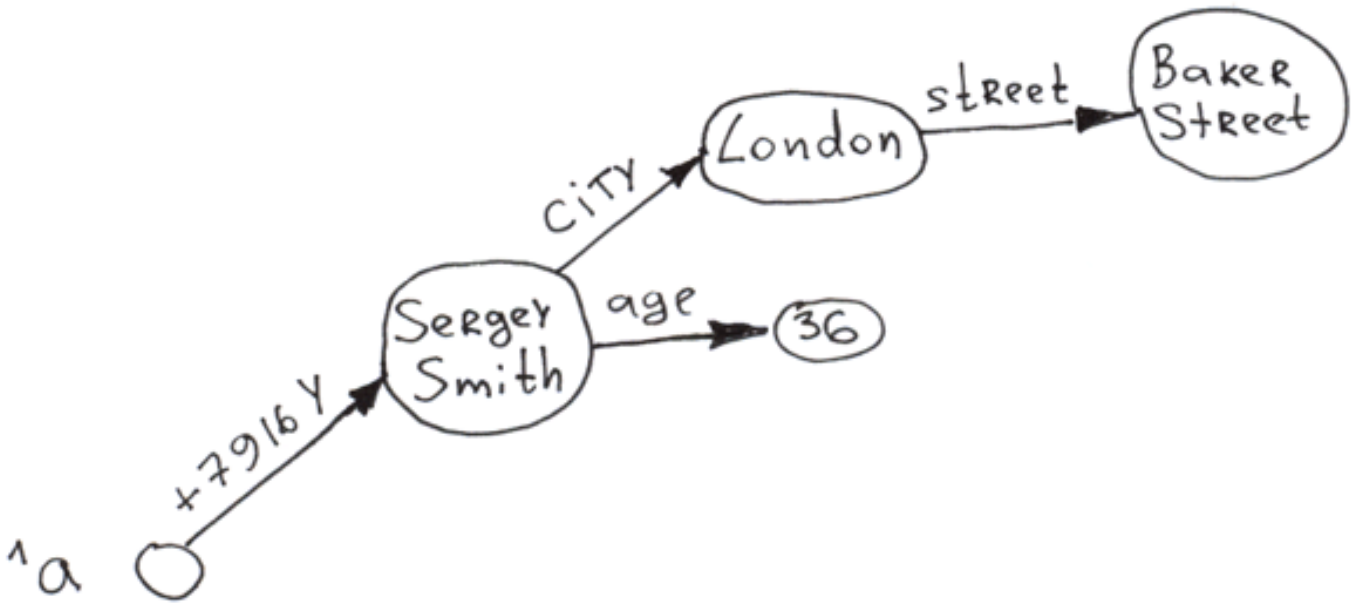
If the orchard tree was a global ...

3. Probablemente ya no se necesitarían cosas como README.txt. Todo lo que quisiera decir sobre el contenido de una carpeta podría escribirse en el propio archivo dentro la carpeta. Generalmente, los nombres de los archivos no se distinguen de los nombres de las carpetas (por ejemplo, /etc/readme puede ser cualquiera de los dos, el archivo o la carpeta), lo cual significa que no tendríamos ningún problema si solo manipulamos los archivos.
4. Las carpetas con subcarpetas y archivos podrían eliminarse mucho más rápido. Existen artículos en la red que relatan historias sobre lo difícil y laborioso que es eliminar millones de archivos pequeños (1, 2, 3). Sin embargo, si crea un sistema de pseudo archivos basado en un global, esto solo llevará segundos o incluso fracciones de segundo. Cuando probé la eliminación de subárboles en el equipo PC que tengo en mi casa, logré eliminar desde 96 hasta 341 millones de nodos de un árbol con dos niveles en un HDD (no en un SDD). Es importante mencionar que nos referimos a la eliminación de una sección en un árbol global, no a la eliminación de un archivo completo que contenga un global.

La eliminación de subárboles es otra de las ventajas de los globals: no necesita la recurrencia para realizar esto. Es increíblemente rápido.

En nuestro árbol, esto puede realizarse mediante el comando **Kill.**

```
Kill ^a("+7926X")
```



A continuación se muestra una pequeña tabla que le permitirá comprender mejor las acciones que puede realizar

en un global.

Comandos y funciones clave relacionados con los globales en COS	
Set	Configura (inicializa) las ramificaciones hasta uno de los nodos (si no se definieron) y el valor del nodo
Merge	Realiza la copia de un subárbol
Kill	Elimina un subárbol
ZKill	Elimina el valor de un nodo específico. No afecta al subárbol que se deriva del nodo
$Query	Muestra el recorrido y la profundidad completa del árbol
$Order	Devuelve el siguiente subíndice que está en el mismo nivel
$Data	Comprueba si un nodo está definido
$Increment	Muestra el incremento atómico del valor de un nodo para evitar que ACAD (ACID, por sus siglas en inglés) lo lea y escriba. Por último, se recomienda utilizar $Sequence en su lugar.

Gracias por leer mi artículo, estaré encantado de responder vuestras preguntas.

Descargo de responsabilidad: *Este artículo refleja la opinión personal del autor y no tiene ninguna relación con la opinión oficial de InterSystems.*

#Globals #Node.js #Principiante #Rendimiento #Tablas relacionales #Caché #InterSystems IRIS

URL de fuente:https://es.community.intersystems.com/post/los-globals-son-espadas-m%C3%A1gicas-para-administrar-datos-parte-1