Article
Kurro Lopez · Nov 25, 2019 11m read

Los Globals son espadas mágicas para almacenar datos. Parte 2. Árboles

Principiantes- ver Parte 1.

3. Variantes de estructuras cuando se usan globals



Una estructura, como un árbol ordenado, tiene varios casos especiales. Echemos un vistazo a aquellos que tienen un valor práctico para trabajar con globals.

3.1 Caso especial 1. Un nodo sin ramas

Los globals pueden usarse no solo como una matriz, sino como variables regulares. Por ejemplo, para crear un contador:

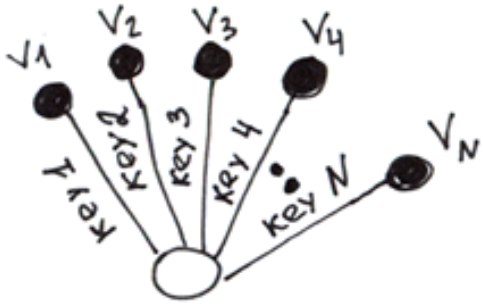



```
Set ^counter = 0  ; setting counter
Set id=$Increment(^counter) ;  atomic incrementation
```

Al mismo tiempo, un global puede tener ramas adicionales además de su valor. Uno no excluye al otro.

3.2 Caso especial 2. Un nodo y múltiples ramas

De hecho, es una base clásica de valor clave. Y si guardamos tuplas de valores en lugar de valores, obtendremos una tabla regular con una clave primaria.



Key-Value DB on the Global

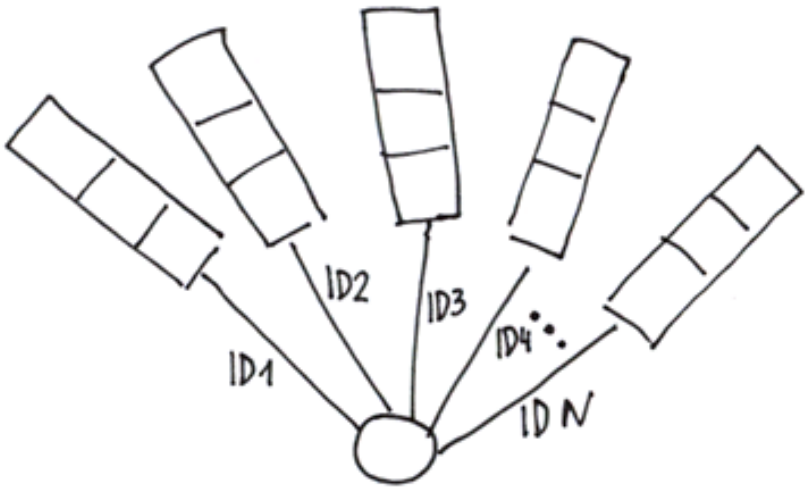


Table on the Global.
Storing row as Value.

Para implementar una tabla basada en globals, tendremos que formar cadenas de valores de columna, luego guardarlas en un global por la clave primaria. Para poder dividir la cadena en columnas durante la lectura, podemos usar lo siguiente:

1. Caracteres delimitadores

```
Set ^t(id1) = "col11/col21/col31"
Set ^t(id2) = "col12/col22/col32"
```

2. Un esquema fijo, por el cual cada campo ocupa un número particular de bytes. Así es como generalmente se hace en bases de datos relacionales.
3. Una función especial $LB (introducida en Caché) que compone una cadena de valores.

```
Set ^t(id1) = $LB("col11", "col21", "col31")
Set ^t(id2) = $LB("col12", "col22", "col32")
```

Lo interesante es que no es difícil hacer algo similar a las claves foráneas en bases de datos relacionales que usan globals. Llamemos a tales estructuras index globals. Un índice global es un árbol suplementario para la búsqueda rápida por campos que no son una parte integral de la clave primaria del global principal. Necesita escribir código adicional para llenarlo y usarlo.

Creemos un índice global basado en la primera columna.

```
Set ^i("col11", id1) = 1
Set ^i("col12", id2) = 1
```

Para buscar rápidamente por la primera columna, deberá buscar en ^i global y encontrar las claves principales (id) correspondientes al valor necesario en la primera columna.

Al insertar un valor, podemos crear valores e índices globals para los campos necesarios. Para mayor fiabilidad, envuélvala en una transacción.

```
TSTART
Set ^t(id1) = $LB("col11", "col21", "col31")
Set ^i("col11", id1) = 1
TCOMMIT
```

Más información sobre creando tablas en M usando globals y emulación de claves secundarias.

Estas tablas funcionarán tan rápido como en las bases de datos tradicionales (o incluso más rápido) si las funciones de inserción/actualización/eliminación se escriben en COS/M y se compilan.
Verifiqué esta declaración aplicando una gran cantidad de operaciones INSERT y SELECT a una sola tabla de dos columnas, también usando el comando TSTART y TCOMMIT (transacciones).

No he probado escenarios más complejos con acceso concurrente y transacciones paralelas.

Sin usar transacciones, la velocidad de inserción para un millón de valores fue de 778.361 inserciones/seg.
Para 300 millones de valores, la velocidad fue de 422.141 inserciones/segundo.

Cuando se utilizaron las transacciones, la velocidad alcanzó 572.082 inserciones/segundo para 50 millones de valores. Todas las operaciones se ejecutaron desde el código M compilado. Usé discos duros normales, no SSD. RAID5 con reescritura. Todo se ejecuta en una CPU Phenom II 1100T.

Para realizar la misma prueba para una base de datos SQL, necesitaríamos escribir un procedimiento almacenado que haga inserciones en un bucle. Al probar MySQL 5.5 (almacenamiento InnoDB) utilizando el mismo método, nunca obtuve más de 11K inserciones por segundo.

Correcto, la implementación de tablas con globals es más compleja que hacer lo mismo en bases de datos relacionales. Es por eso que los DB industriales basados en globales tienen acceso SQL para un trabajo simplificado con datos tabulares.

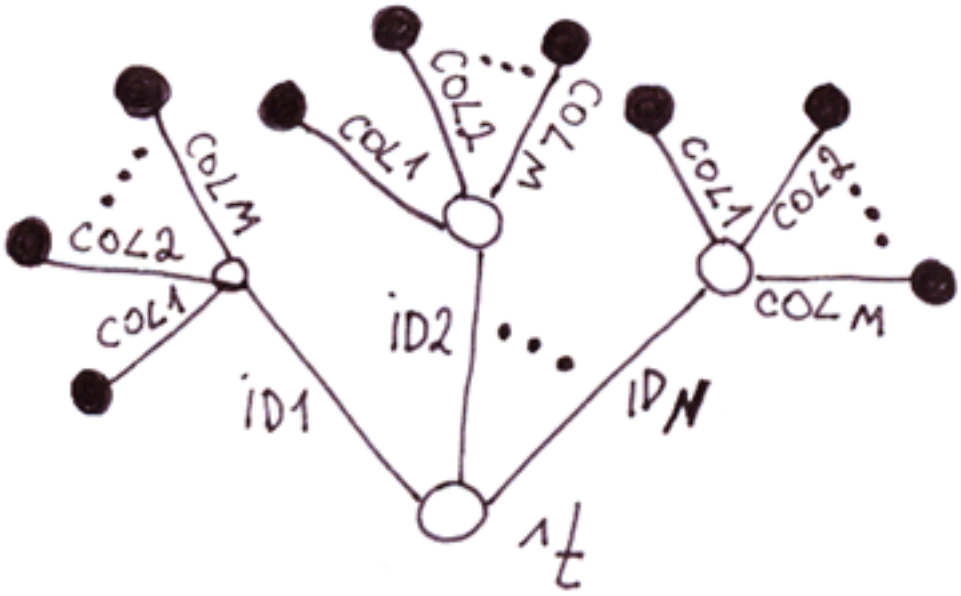

En general, si el esquema de datos no va a cambiar con frecuencia, la velocidad de inserción no es crítica y la base de datos completa se puede representar fácilmente con tablas normalizadas, es más fácil trabajar con SQL, ya que proporciona un mayor nivel de abstracción.


En este caso, quería mostrar que los globals pueden usarse como constructores para crear otras bases de datos. Al igual que el lenguaje ensamblador que se puede usar para crear otros idiomas. Y aquí hay

algunos ejemplos de uso de globals para crear contrapartes de valores-clave, listas, conjuntos, tablas, bases de datos orientadas a documentos.

Si necesita crear una base de datos no estándar con un esfuerzo mínimo, debe considerar el uso de globals.

3.3 Caso especial 3. Un árbol de dos niveles con cada nodo de segundo nivel que tiene un número fijo de ramas



Probablemente lo hayas adivinado: es una implementación alternativa de tablas usando globales. Comparémoslo con el anterior.

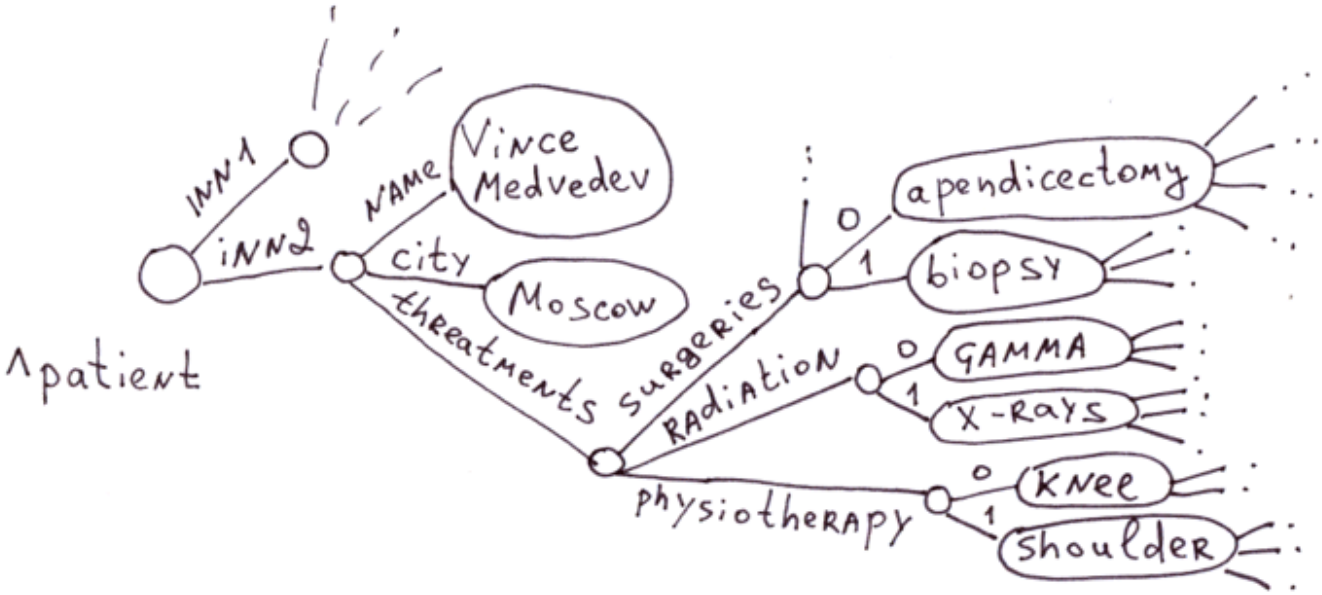
Tablas en un árbol de dos niveles vs. Tablas en un árbol de un nivel	
Contras	Pros
1. Inserciones más lentas, ya que el número de nodos debe establecerse igual al número de columnas 2. Mayor consumo de espacio en el disco duro, ya que los índices globals (como los índices de matriz) con nombres de columna ocupan espacio en el disco duro y se duplican para cada fila	1. Acceso más rápido a los valores de columnas particulares, ya que no necesita analizar la cadena. Según mis pruebas, es un 11,5% más rápido para 2 columnas e incluso más rápido para más columnas. 2. Más fácil cambiar el esquema de datos 3. Código más fácil de leer

Conclusión: No hay nada que destacar. Dado que el rendimiento es una de las ventajas clave de los globals, prácticamente no tiene sentido utilizar este enfoque, ya que es poco probable que funcione más rápido que las tablas normales en bases de datos relacionales.

3.4 Caso general. Árboles y llaves ordenadas

Cualquier estructura de datos que se pueda representar como un árbol se ajusta a los globals de una manera perfecta.

3.4.1 Objetos con subobjetos



Esta es el área donde se usan tradicionalmente los globals. Existen numerosas enfermedades, medicamentos, síntomas y métodos de tratamiento en el área médica. Es irracional crear una tabla con un millón de campos para cada paciente, especialmente porque el 99% de ellos estará en blanco.

Imagine una base de datos SQL compuesta de las siguientes tablas: "Paciente" ~100.000 campos, "Medicación" 100.000 campos, "Terapia" 100.000 campos, "Complicaciones" 100.000 campos y así sucesivamente. Como alternativa, puede crear una base de datos con miles de tablas, cada una para un tipo de paciente en particular (¡y también pueden superponerse!), tratamiento, medicamentos y miles de tablas para las relaciones entre estas tablas.

Los globals se ajustan a la atención médica como un guante, ya que hacen posible que cada paciente tenga un registro completo de casos, una lista de terapias, medicamentos administrados y sus efectos, todo en forma de árbol, sin desperdiciar demasiado espacio en el disco en columnas vacías, como sería el caso con las bases de datos relacionales.


Globals funciona bien para bases de datos con detalles personales, cuando la tarea es acumular y sistematizar el máximo de varios datos personales sobre un cliente. Esto es especialmente importante para la salud, la banca, el marketing, el archivo y otras áreas.

No hace falta decir que SQL también le permite emular un árbol usando solo varias tablas (EAV, 1,2,3,4,5,6, 7,8), pero es mucho más complejo y funciona más lento. En esencia, tendríamos que escribir un global basado en tablas y ocultar todas las rutinas relacionadas con tablas bajo una capa de abstracción. No es correcto emular una tecnología de nivel inferior (globales) con la ayuda de una de nivel superior (SQL). Es simplemente injustificado.

No es un secreto que cambiar un esquema de datos en tablas gigantes (ALTER TABLE) puede llevar una cantidad considerable de tiempo. MySQL, por ejemplo, realiza la operación ALTER TABLE ADD¦DROP COLUMN copiando todos los datos de la tabla anterior a la nueva (lo probé en MyISAM e InnoDB). Que puede colgar una base de datos de producción con miles de millones de registros durante días, si no semanas.



Si estamos usando globals, cambiar la estructura de datos no tiene ningún coste para nosotros. Podemos agregar cualquier propiedad nueva a cualquier objeto en cualquier nivel de la jerarquía en cualquier momento dado. Los cambios que requieren el cambio de nombre de las ramas se pueden aplicar en modo de fondo con la base de datos en funcionamiento.

Por lo tanto, cuando se trata de almacenar objetos con una gran cantidad de propiedades opcionales, los globals funcionan perfectamente bien.

Permítame recordarle que el acceso a cualquiera de las propiedades es instantáneo, ya que en un global, todas las rutas son un árbol B.

En el caso general, las bases de datos basadas en globals son un tipo de bases de datos orientadas a documentos que admiten el almacenamiento de información jerárquica. Por lo tanto, las bases de datos orientadas a documentos pueden competir eficientemente con los globals en el campo del almacenamiento de tarjetas médicas.

Pero todavía no lo es.
Tomemos MongoDB, por ejemplo. **En este campo**, pierde frente a los globales por las siguientes razones:

1. **Tamaño del documento.** La unidad de almacenamiento es un texto en formato JSON (BSON, para ser exactos) con un tamaño máximo de alrededor de 16 MB. La limitación se introdujo a propósito para asegurarse de que la base de datos JSON no se vuelva demasiado lenta durante el análisis, cuando se guarda un gran documento JSON y se abordan valores de campo particulares. Se supone que este documento tiene información completa sobre un paciente. Todos sabemos cuán gruesas pueden ser las tarjetas de pacientes. Si el tamaño máximo de la tarjeta tiene un límite de 16 MB, filtra inmediatamente a los pacientes cuyas tarjetas contienen imágenes de resonancia magnética, rayos X y otros materiales. Una sola rama de un global puede tener gigabytes y petabytes de terabytes de datos. De alguna manera lo dice todo, pero déjame contarte más.
2. **El tiempo requerido para crear/cambiar/eliminar nuevas propiedades de la tarjeta del paciente.** Dicha base de datos necesitaría copiar toda la tarjeta en la memoria (¡muchos datos!), Analizar los datos de BSON, agregar/cambiar/eliminar el nuevo nodo, actualizar índices, empaquetarlo todo nuevamente en BSON y guardarlo en el disco. Un global solo necesitaría abordar la propiedad necesaria y realizar la operación necesaria.
3. **Velocidad de acceso a propiedades particulares.** Si el documento tiene muchas propiedades y una estructura de varios niveles, el acceso a propiedades particulares será más rápido porque cada ruta en el global es un árbol B. En BSON, deberá analizar linealmente el documento para encontrar la propiedad necesaria.

3.3.2 Matrices asociativas

Las matrices asociativas (incluso con matrices anidadas) funcionan perfectamente con globals. Por ejemplo, esta matriz PHP se verá como la primera ilustración en 3.3.1.

```
$a = array(
  "name" => "Vince Medvedev",
  "city" => "Moscow",
  "threatments" => array(
    "surgeries" => array("apedicectomy", "biopsy"),
    "radiation" => array("gamma", "x-rays"),
```

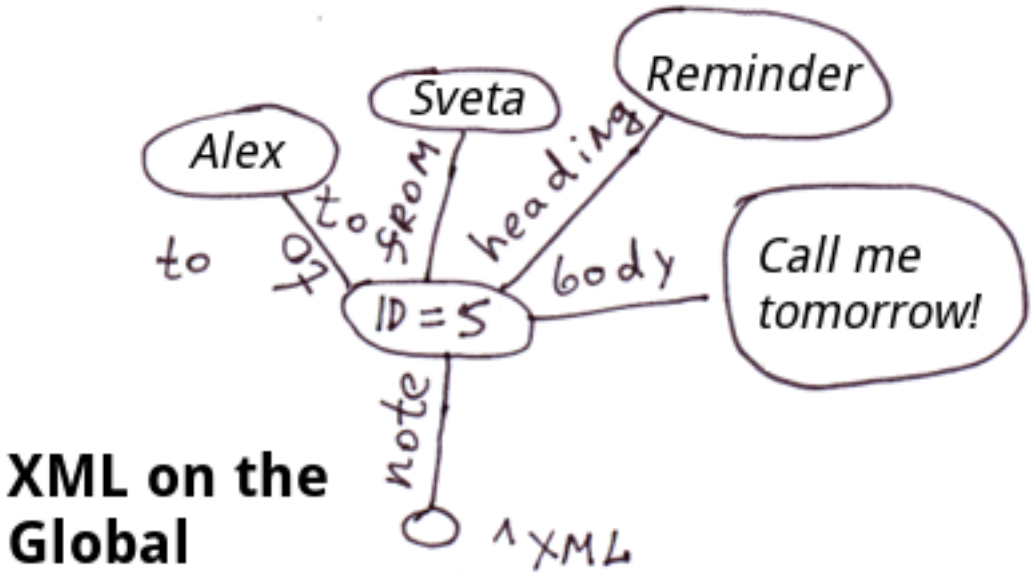
```
    "physiotherapy" => array("knee", "shoulder")
  )
);
```

3.3.3 Documentos jerárquicos: XML, JSON

También se puede almacenar fácilmente en globals y descomponerse de diferentes maneras.

XML

El método más fácil de descomponer XML en globals es almacenar atributos de etiqueta en nodos. Y si necesita acceso rápido a los atributos de etiqueta, podemos colocarlos en ramas separadas.



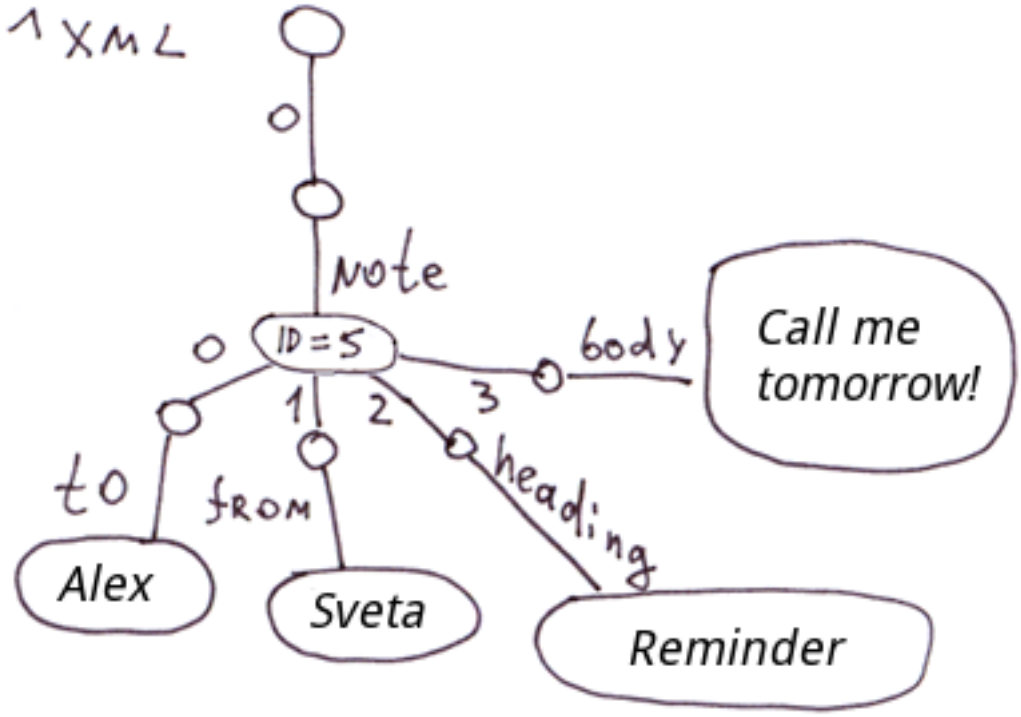

```
<note id=5>
<to>Alex</to>
<from>Sveta</from>
<heading>Reminder</heading>
<body>Call me tomorrow!</body>
</note>
```

En COS, el código se verá así:

```
Set ^xml("note")="id=5"
Set ^xml("note","to")="Alex"
Set ^xml("note","from")="Sveta"
Set ^xml("note","heading")="Reminder"
Set ^xml("note","body")="Call me tomorrow!"
```

Nota: Para XML, JSON y matrices asociativas, puede encontrar una serie de métodos para mostrarlos en globales. En este caso particular, no reflejamos el orden de las etiquetas anidadas en la etiqueta "note". En el **^xml** global, las etiquetas anidadas se mostrarán en orden alfabético. Para una visualización precisa del orden, puede usar el

siguiente modelo, por ejemplo:



JSON.

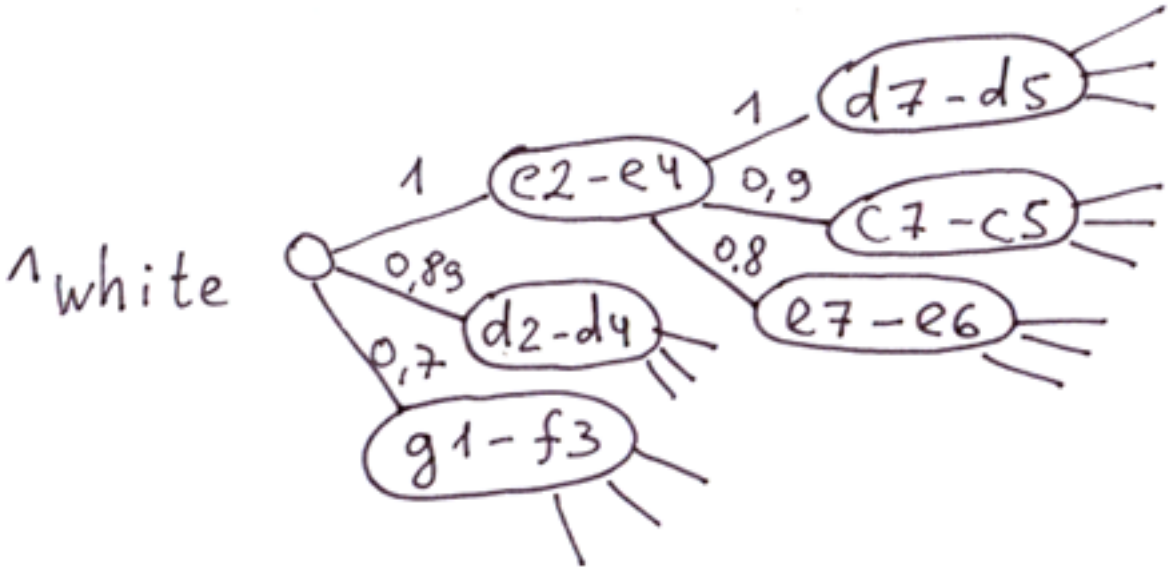
El contenido de este documento JSON se muestra en la primera ilustración en la Sección 3.3.1:

```
var document = {
  "name": "Vince Medvedev",
  "city": "Moscow",
  "threatments": {
    "surgeries": ["apedicectomy", "biopsy"],
    "radiation": ["gamma", "x-rays"],
    "physiotherapy": ["knee", "shoulder"]
  },
};
```

3.3.4 Estructuras idénticas unidas por relaciones jerárquicas.

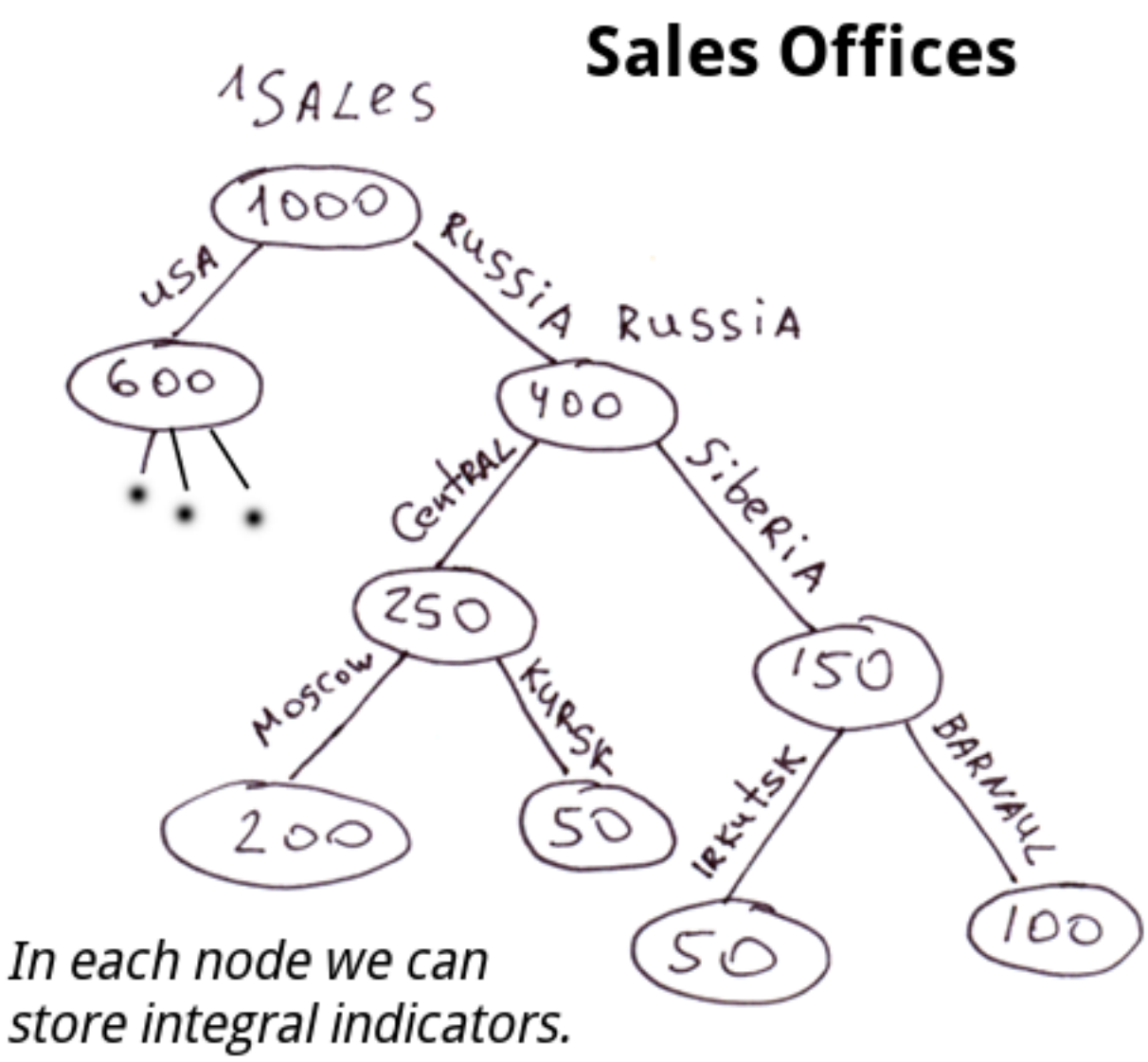
Ejemplos: estructura de oficinas de ventas, puestos de personas en una estructura MLM.

Base de datos de inicio. Puede usar una evaluación de fuerza de movimiento como el valor del índice de nodo global. En este caso, deberá seleccionar una rama con el mayor peso para determinar el mejor movimiento. En el global, todas las ramas en cada nivel se ordenarán por la fuerza del movimiento.



An example of debuts DB on the Global. Select the branch with the highest weight and make a move.

La estructura de las oficinas de ventas, las personas en una empresa de MLM. Los nodos pueden almacenar algunos valores de almacenamiento en caché que reflejan las características de todo el subárbol. Por ejemplo, las ventas de este subárbol en particular. Podemos obtener información exacta sobre los logros de cualquier sucursal en cualquier momento.



4. Situaciones en las que merece la pena usar globals

La primera columna contiene una lista de casos en los que el uso de globals le dará una ventaja considerable en términos de rendimiento; y la segunda, una lista de situaciones en las que simplificarán el desarrollo o el modelo de datos.

Velocidad	Conveniencia de procesamiento/presentación de datos
1. Inserción [con clasificación automática en cada nivel], [indexación por la clave primaria] 2. Eliminación de subárbol 3. Objetos con muchas propiedades anidadas a las que necesita acceso individual 4. Una estructura jerárquica con la posibilidad de atravesar ramas secundarias a partir de cualquier rama, incluso una no existente 5. Transversal de árboles en profundidad	1. Objetos/instancias con una gran cantidad de propiedades/instancias no requeridas [y/o anidadas] 2. Datos sin esquema: a menudo se pueden agregar nuevas propiedades y eliminar las antiguas 3. Necesita crear una base de datos no estándar 4. Bases de datos de ruta y árboles de soluciones. Cuando los caminos se pueden representar convenientemente como un árbol 5. Eliminación de estructuras jerárquicas sin usar recursividad

Descargo de responsabilidad: *Este artículo y mis comentarios reflejan solo mi opinión y no tienen nada que ver con la posición oficial de InterSystems Corporation.*

#Globals #Data Model #Node.js #Beginner #Performance #Relational Tables #Caché #InterSystems IRIS

Source URL:https://community.intersystems.com/node/468946