Artículo
Nancy Martínez · 6 mayo, 2020 Lectura de 8 min

El arte de mapear Globals para Clases (4 de 3)

¡Hola Comunidad!

Aquí está el cuarto artículo de la trilogía (¿hay algún otro fan de la "Guía del autoestopista galáctico"?).

Si estás buscando revivir una antigua aplicación MUMPS, sigue estos pasos para mapear tus globales para clases y exponer todos esos bonitos datos a Objects y SQL.

Si lo anterior no te suena familiar, puedes consultar los artículos anteriores:

El arte de mapear Globales para Clases (1 de 3)

El arte de mapear Globales para Clases (2 de 3)

El arte de mapear Globales para Clases (3 de 3)

¡Esta es para ti, Joel! Basados en la relación padre-hijo que definimos en el último ejemplo, vamos a crear una clase "nieto" para manejar la nueva información de estaciones (seasons) agregada al global ^ParentChild.

La misma aclaración: Si tus variables globales siguen sin tener ni pies ni cabeza después de leer estos artículos, contacta con el Centro de Soporte Internacional (WRC) y trataremos de ayudarte a resolver tu problema: Support@InterSystems.com.

Pasos para mapear un Global a una Clase:

1. Identificar un patrón que se repita en los global datos
2. Identificar lo que constituye una clave única.
3. Identificar las propiedades y sus tipos.
4. Definir las propiedades en la clase (no olvidar las propiedades de los subíndices de las variables).
5. Definir el índice IdKey.
6. Establecer la definición de almacenamiento:
    a. Definir los subíndices e incluir la IdKey.
    b. Establecer una sección para los datos.
    c. Ignorar la sección "Row ID" (Identificación de la fila). El 99% de las veces el valor que se quiere es el predeterminado, así que dejamos al sistema que lo rellene.
7. Compilar y probar tu clase / tabla.

^ParentChild(1)="Brendan^45956"

^ParentChild(1,"Hobbies",1)="Pit Crew"

^ParentChild(1,"Hobbies",1,"Seasons")="Fall*Winter"

^ParentChild(1,"Hobbies",2)="Kayaking"

^ParentChild(1,"Hobbies",2,"Seasons")="Spring*Summer*Fall"

^ParentChild(1,"Hobbies",3)="Skiing"

^ParentChild(1,"Hobbies",3,"Seasons")="Summer*Winter"

^ParentChild(2)="Sharon^46647"

^ParentChild(2,"Hobbies",1)="Yoga"

^ParentChild(2,"Hobbies",1,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(2,"Hobbies",2)="Scrap booking"

^ParentChild(2,"Hobbies",2,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(3)="Kaitlin^56009"

^ParentChild(3,"Hobbies",1)="Lighting Design"

^ParentChild(3,"Hobbies",1,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(3,"Hobbies",2)="pets"

^ParentChild(3,"Hobbies",2,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(4)="Melissa^56894"

^ParentChild(4,"Hobbies",1)="Marching Band"

^ParentChild(4,"Hobbies",1,"Seasons")="Fall"

^ParentChild(4,"Hobbies",2)="Pep Band"

^ParentChild(4,"Hobbies",2,"Seasons")="Winter"

^ParentChild(4,"Hobbies",3)="Concert Band"

^ParentChild(4,"Hobbies",3,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(5)="Robin^57079"

^ParentChild(5,"Hobbies",1)="Baking"

^ParentChild(5,"Hobbies",1,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(5,"Hobbies",2)="Reading"

^ParentChild(5,"Hobbies",2,"Seasons")="Spring*Summer*Fall*Winter"

^ParentChild(6)="Kieran^58210"

^ParentChild(6,"Hobbies",1)="SUBA"

^ParentChild(6,"Hobbies",1,"Seasons")="Summer"

^ParentChild(6,"Hobbies",2)="Marching Band"

^ParentChild(6,"Hobbies",2,"Seasons")="Fall"

^ParentChild(6,"Hobbies",3)="Rock Climbing"

^ParentChild(6,"Hobbies",3,"Seasons")="Spring*Summer*Fall"

^ParentChild(6,"Hobbies",4)="Ice Climbing"

^ParentChild(6,"Hobbies",4,"Seasons")="Winter"

Paso 1:

Bueno, no es tan difícil encontrar los datos que se repiten en nuestra nueva clase, es el subnodo Seasons (Estaciones). La parte difícil llega cuando no quiero que " Spring*Summer*Fall " estén todas en una fila, quiero 3 filas: " Spring ", " Summer ", " Fall ".

^ParentChild(1)="Brendan^45956"

^ParentChild(1,"Hobbies",1)="Pit Crew"

^ParentChild(1,"Hobbies",1,"Seasons")="Fall*Winter"

^ParentChild(1,"Hobbies",2)="Kayaking"

^ParentChild(1,"Hobbies",2,"Seasons")="Spring*Summer*Fall"

^ParentChild(1,"Hobbies",3)="Skiing"

^ParentChild(1,"Hobbies",3,"Seasons")="Summer*Winter"

Cada hobby puede tener de 1 a 4 estaciones. Supongo que podríamos crear otras 4 propiedades en la clase Example3Child, pero ¿qué haríamos si alguien inventa una nueva estación? Una solución más flexible sería crear una tabla "nietos", para que el número de estaciones pueda mantenerse dinámico.

Paso 2:

Todos sabemos buscar en los subíndices para obtener las partes de IdKey, así que sabemos que el Subíndice 1 y Subínidce 3 serán parte de la IdKey, necesitamos un valor adicional para identificar de forma única las distintas estaciones, ¡pero se nos acabaron los subíndices!

La información que estamos poniendo en el mapeo se utiliza para generar código para consultas. Quizás si pensamos en qué comandos COS son necesarios para obtener esta información, nos podría ayudar a definir el mapeo. Las 3 cosas principales que debemos hacer son:

```
SET sub1=$ORDER(^Parentchild(sub1))

SET sub2=$ORDER(^Parentchild(sub1, " Hobbies " ,sub2))

SET season=$PIECE(^ParentChild(sub1, " Hobbies " ,sub2 " , " Seasons " ), " * " ,PC)
```

Podemos hacer lo mismo en la sección Subíndices en el mapeo. El almacenamiento SQL de Caché soporta 4 tipos distintos de Subíndices: Piece (pieza), Global, Sub y Other (otro). El valor predeterminado es Sub y ese es el que hemos estado usando hasta ahora, lo que nos permite obtener esos hermosos bucles $ORDER() que necesitamos.

En este ejemplo añadiremos las opciones de Piece. La propiedad que se use a este nivel se usará como contador de piezas (PC, "Piece Counter" en el ejemplo anterior). El comportamiento predeterminado es aumentar esto en 1 hasta llegar al final de la cadena.

Paso 3:

3 propiedades: los datos es la fácil: Season (estación). Luego tenemos la propiedad de relación: HobbyRef. Y, finalmente, necesitamos un childsub: PieceCount (cantidad de piezas)

Paso 4:

Property Season As %String;

Property PieceCounter As %Integer;

Relationship HobbyRef As Mapping.Example3Child [Cardinality = parent, Inverse = Seasons];

Paso 5:

Cuando miramos el mapeo de los subscripts, tendremos 3 niveles de variables, pero para el índice IdKey, solo nos referimos a 2 propiedades: HobbyRef y PieceCounter

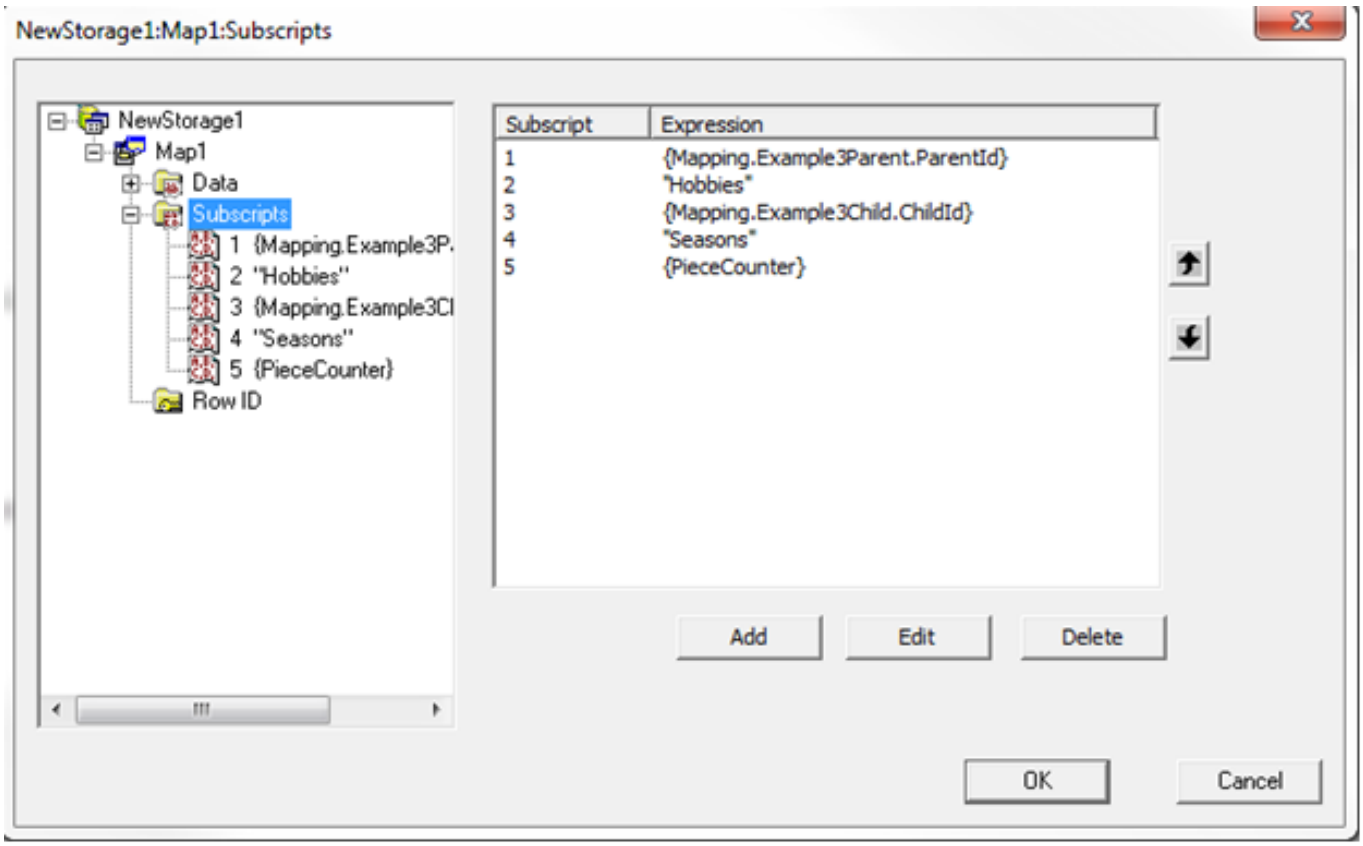
Index Master On (HobbyRef, PieceCounter) [IdKey];

Paso 6:

Las mismas tres secciones con las que hemos estado trabajado desde el principio. Aún estamos ignorando la Fila ID. Para esta, necesitamos entrar un poco más en detalle de los subíndices y definir el tipo de acceso. Esta clase usará "Sub" y "Piece". Si quieres ver ejemplos de "Global" y "Other", tendrás que descargar mi archivo zip de ejemplos.

Paso 6a:

La página principal de subíndices tiene el mismo aspecto que siempre, solo se han añadido dos niveles. Recuerda que para las dos partes de la referencia Parent necesitamos volver a referirnos a las clases que referenciamos: {Mapping.Example3Parent} and {Mapping.Example3Child}. La diferencia se ve cuando se hace clic en uno de los Niveles del Subíndice del lado izquierdo de la ventana.



En la siguiente imagen se pueden ver todas las cosas peligrosas que se pueden hacer en cada nivel del subíndice. Para el Tipo de Acceso de "Sub", asumiremos que quieres comenzar en “ ” y $ORDER hasta llegar a “ ”. Si no es así, puedes ingresar un valor de inicio (Start Value) o un valor final (Stop Value).

‘ Data Access ’ te permite cambiar lo que estás viendo desde el nivel anterior. Por ejemplo, podrías cambiar el global que quieres recorrer.

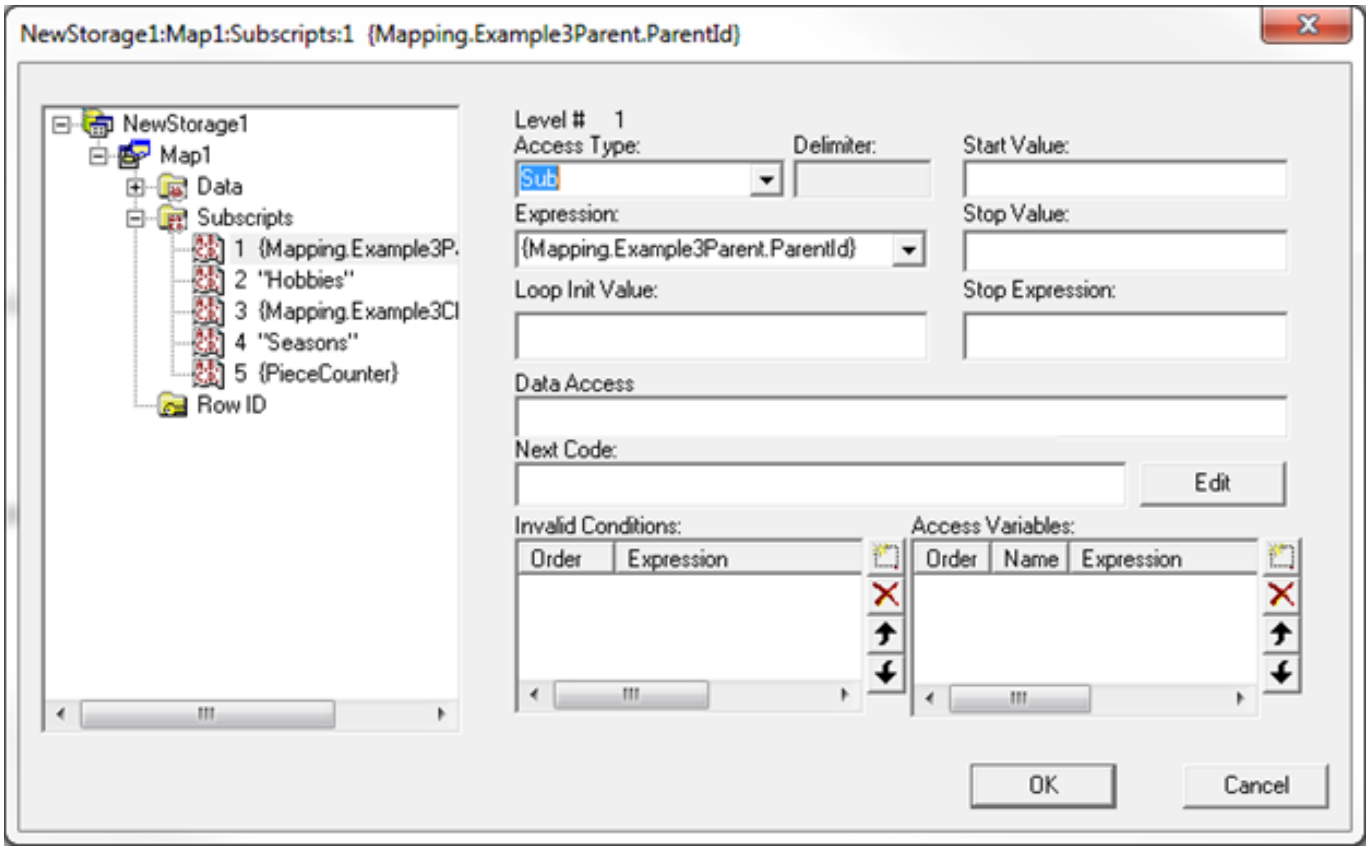
‘ Next Code ’ e ‘ Invalid Conditions ’ van juntas, y son las cosas más comunes que usarás en esta ventana. Si un sencillo $ORDER() no te llevara desde un valor válido al siguiente, puedes escribir tu propio código para ser usado en su lugar.

‘ Next Code ’ se usaría para ir de un valor correcto al siguiente valor correcto, igual a como lo hace $ORDER().

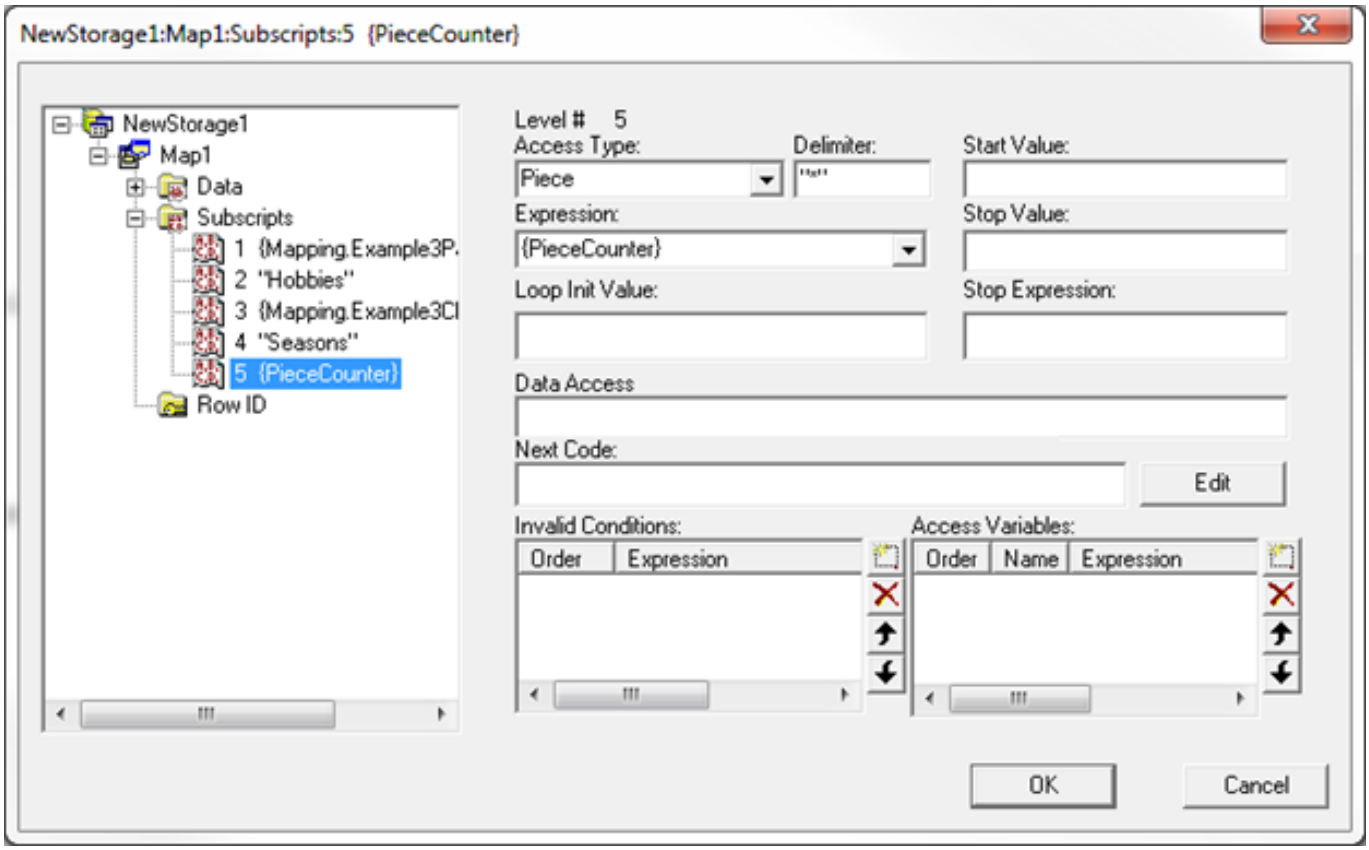
La ‘ Invalid Condition ’ se usa para probar un valor dado. Si se dio un valor a "subíndiceX", no es necesario llamar a Next para encontrarlo, ya lo has indicado. Lo que se necesita es algún código que permita saber si el valor es correcto o no.

Mi prometido archivo zip de ejemplos tiene montones de clases que usan ‘ Next Code ’ e ‘ Invalid Conditions ’.

Las ‘ Access Variables ’ son lo último de la página, y rara vez se usan. Básicamente, puedes definir una variable aquí, asignarle un valor en un nivel de subíndice y luego usarla en los Niveles de subíndices superiores. El código de tabla generada se encargará de definir el alcance por ti.

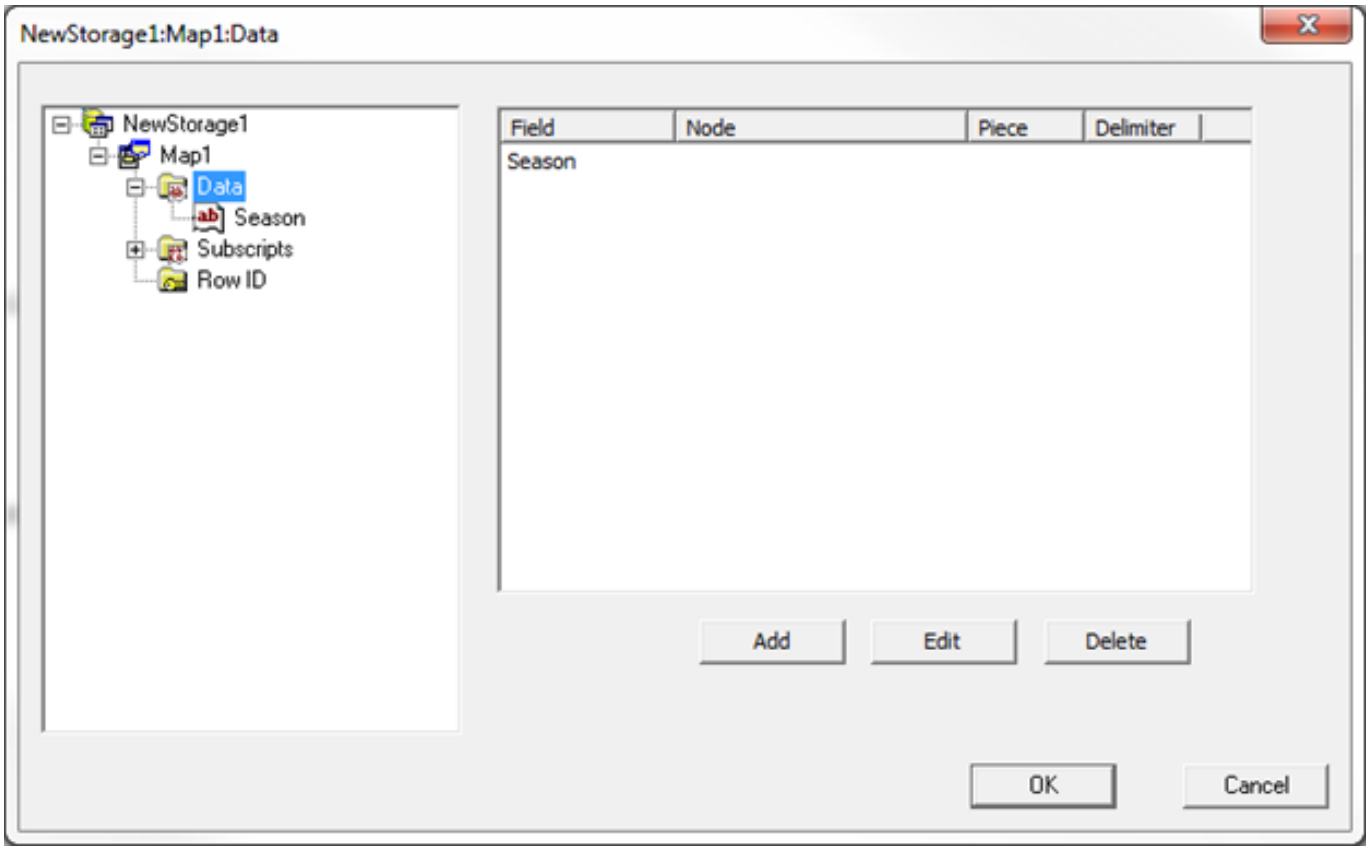


Para el subíndice Nivel 5, el "Access Type" es "Piece" y el "Delimiter" (delimitador) es "*". El código generado comenzará en Piece 1 y aumenta en 1 hasta que se acaben los valores de $PIECE. Nuevamente, podemos poner valores de inicio (Start Values) y/o valores finales (Stop Values) para controlar esto.



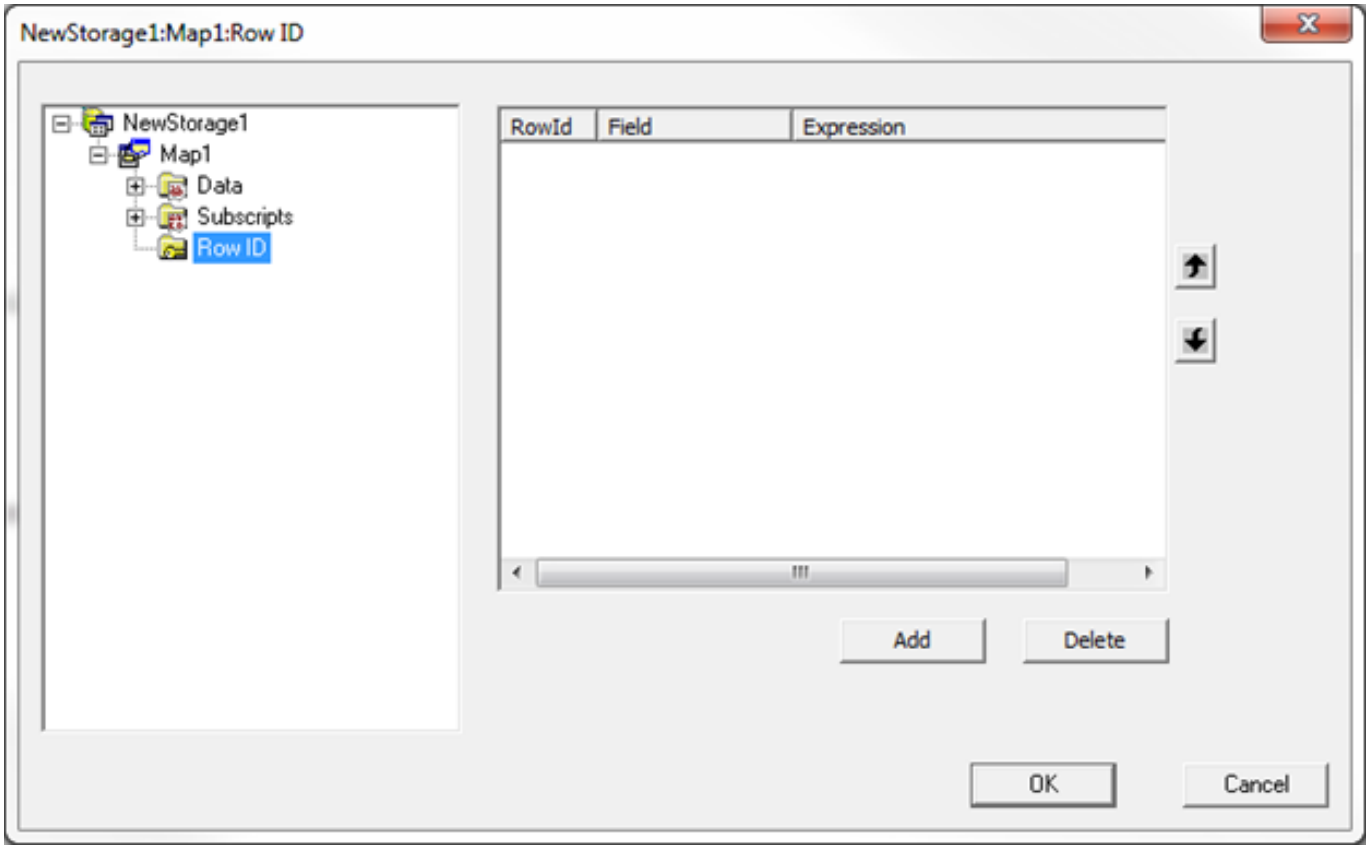
Paso 6b:

La sección de datos es solo una propiedad; no se necesita un "Piece" o "Delimiter". Si esta tabla tuviera más campos, lo mas probable es que sería necesario proporcionar un "Piece" y "Delimiter" y eso estaría bien.



Paso 6c:

Seguimos dejando esto vacío.



Paso 7:

Todo se compila correctamente.

```
Compilation started on 11/30/2016 08:17:42 with qualifiers 'uk/importselectivity=1 /checkuptodate=expandedonly'
Compiling 2 classes, using 2 worker jobs
Compiling class Mapping.Example3Child
Compiling class Mapping.Example3GrandChild
Compiling table Mapping.Example3GrandChild
Compiling table Mapping.Example3Child
Compiling routine Mapping.Example3Child.1
Compiling routine Mapping.Example3GrandChild.1
Compilation finished successfully in 1.021s.
```

Una combinación de las tres tablas:

```
SELECT P.ID, P.Name, P.DateOfBirth,

C.ID, C.Hobby, G.ID, G.Season

FROM Mapping.Example3Parent P

JOIN Mapping.Example3Child C ON P.ID = C.ParentRef

JOIN Mapping.Example3Grandchild G ON C.ID = G.HobbyRef

WHERE P.Name = 'Kieran'
```

Nos da:

ID	Name	DateOfBirth	ID	Hobby	ID	Season
6	Kieran	05/16/2000	6¦¦1	SUBA	6¦¦1¦¦1	Summer
6	Kieran	05/16/2000	6¦¦2	Marching Band	6¦¦2¦¦1	Fall
6	Kieran	05/16/2000	6¦¦3	Rock Climbing	6¦¦3¦¦1	Spring
6	Kieran	05/16/2000	6¦¦3	Rock Climbing	6¦¦3¦¦2	Summer
6	Kieran	05/16/2000	6¦¦3	Rock Climbing	6¦¦3¦¦3	Fall
6	Kieran	05/16/2000	6¦¦4	Ice Climbing	6¦¦4¦¦1	Winter

Recuerda que dije que el IdKey de la tabla hijo siempre está formado por 2 partes: la referencia Parent y el Childsub.

En la primera fila, el ID de Example3Child es 6¦¦1: Parent Reference = 6 y Childsub = 1.

Para Example3GrandChild la IdKey está formada por tres partes 6¦¦1¦¦1, pero sigue siendo solo la Parent Reference y el Childsub. La Parent reference se vuelve un poco más compleja: Parent Reference = 6¦¦1 y Childsub = 1.

En Example3Child, la cantidad de propiedades en los subscripts coincide con la cantidad de propiedades del IdKey. A medida que anidamos mas profundamente en una estructura padre-hijo, el IdKey se vuelve compuesto y aumentará el número de subíndices.

Aquí puede ver una exportación de las 3 clases usadas en este ejemplo: MappingExample4.zip

#Globals #Mapeo #Modelo de datos de objetos #SQL #Caché