Indexovanie kolekčných atribútov

Nedávno som sa dostal k práci na projekte, ktorého cieľom bolo zostaviť register osôb. Jednou z podmienok bolo zabezpečiť čo najrýchlejšie vyhľadanie osoby podľa nejakého kritéria – textu, napr. rodného čísla, priezviska, dátumu narodenia alebo nejakého iného unikátneho alebo aspoň „takmer unikátneho" kľúča. Nechcelo sa mi ani trochu nad každou vlastnosťou, ktorá by mohla slúžiť ako vyhľadávacie kritérium, definovať index a potom tvoriť komplexnú vyhľadávaciu obrazovku s množstvom vstupných polí pre jednotlivé kritériá, takže som použil jednu peknú fintu, ktorú Caché nejaký čas ponúka, ale nanešťastie ju má veľmi zle zdokumentovanú.

Tou fintou je použitie jednej vlastnosti typu **zoznam** pre všetky možné vyhľadávacie aliasy a nad ňou definovanie indexu s uvedením špeciálneho kľúčového slova, definujúceho spôsob indexovania. Výhodou tohto prístupu je, že do takého poľa (zoznamu) môžete zadať ľubovoľný počet aliasov, ktoré môžu slúžiť na identifikáciu osoby a bez ohľadu na to, aký z nich použijete na vyhľadanie osoby, reťazec SQL, ktorý bude vyhľadávať osoby, sa nebude vôbec meniť.

Poďme si to ukázať na jednoduchej demonštračnej triede. Uvažujme o jednoduchej perzistentnej triede, nazvime ju **tt.Tip22**. Táto trieda bude mať pre jednoduchosť len jednu vlastnosť, názov typu% String, a ďalej potom zoznam aliasov, reprezentovaný opäť vlastnosťou nazvanou aliasy a definovanou ako zoznam - list of% String.

Ak budeme chcieť vyhľadať nejakú inštanciu, použijeme SQL. Vieme, že SQL je v Caché najrýchlejší a najjednoduchší spôsob vyhľadávania záznamov, takže nie je dôvod ho nepoužiť. Caché má na prácu SQL s kolekčnými atribútmi rozšírenú syntax oproti štandardu. My použijeme konštrukciu *FOR SOME% ELEMENTS*.

A teraz konkrétny príklad – ukážka definície triedy a metódy na hľadanie:

```
/// Indexy nad kolekcemi
Class tt.Tip22 Extends   (%Persistent,
%XML.Adaptor)
{
Property Nazev As %String;
Property Aliasy As list of %String (XMLITEM-
NAME = "Alias");
Index indexAliasy On Aliasy(ELEMENTS);
ClassMethod NajdiAlias (pAlias As %String =
"Alias 1 2")
    {
            #dim tResult as %SQL.StatementRe-
sult
            set tSQL=1
            set tSQL (1) ="SELECT ID, Nazev
FROM tt.Tip22 WHERE FOR SOME %ELEMENT
(Aliasy)   (%VALUE = ?)"
            set tStatement=##class(%SQL.State-
ment).%New(2)
            set tSC=tStatement.%Prepare(.tSQL)
            set
tResult=tStatement.%Execute(pAlias)
            do tResult.%Dislay ()
            quit
    }
```

Všimnite si definíciu indexu – obsahuje kľúčové slovo ELEMENTS v zátvorke. Tým hovoríme, že chceme indexovať každú hodnotu v zozname. Existuje ešte kľúčové slovo KEYS, ktorým, naopak, hovoríme, že chceme indexovať kľúč v asociatívnom poli (teda keby vlastnosť Aliasy bola definovaná ako *array of% String*).

Trieda obsahuje jednoduchú metódu používajúcu nový SQL Statement (pozri predchádzajúcu časť seriálu) a takisto konštrukciu FOR SOME% ELEMENT, ktorá slúži na vyhľadanie záznamov v poliach s viacerými hodnotami. Viac o tejto konštrukciu nájdete na *http://docs.intersystems.com/cache20102/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_forsomeelement*.

Na to, aby sme si mohli na niečom opisovanú funkčnosť vyskúšať, obsahuje trieda tt.Tip22 pomocnú metódu **Init ()**, ktorá naplní inštancie dátami z elementu XDATA. Obsah oboch je tu:

```
XData POPULATE
    {
<Data>
  <Tip22>
     <Nazev>Prvek 1</Nazev
    <Aliasy>
      <Alias>Prvek 1</Alias>
      <Alias>Alias 1 1</Alias>
      <Alias>Alias 1 2</Alias>
      <Alias>Alias 1 3</Alias>
    </Aliasy>
  </Tip22>
  <Tip22>
    <Nazev>Prvek 2</Nazev>
    <Aliasy>
     <Alias>Prvek 2</Alias>
      <Alias>Alias 2 1</Alias>
      <Alias>Alias 2 2</Alias>
    <Aliasy>
  </Tip22>
  <Tip22>
    <Nazev>Prvek 3</Nazev>
    <Aliasy>
    <Alias>Prvek 3</Alias>
     <!-- pozor, duplikace aliasu s prvkem
s prvkem 1! -->
      <Alias>Alias 1 3</Alias>
      <Alias>Alias 2 3</Alias>
      <Alias>Alias 3 3</Alias>
   </Aliasy>
 </Tip22>
</Data>
}
```

Všimnite si, že do zoznamu **Aliasy** dávam takisto hodnotu vlastnosti **Nazov**.

```
ClassMethod Init ()
    {
            #dim ex as %Exception.AbstractEx-
ception
            try  {
   set tSC=..%KillExtent ()
   set tXD=##class (%Dictionary.XDataDefini-
tion).%OpenId(..%ClassName (1) "||POPULATE")
   set tReader=##class(%XML.Reader).%New()
   $$$THROWONERROR(tSC,tReader.OpenStream
(tXD.Data))
   do tReader.Correlate("Tip22",..%ClassName
(1))
   while tReader.Next(.tInstance,.tSC)  {
          $$$THROWONERROR (tSC,
tInstance.%Save())
}
w !, "Import OK"!
          } catch  (ex)  {
w !, ex.DisplayString ()
          }
          Quit
}
```

Po naplnení triedy dátami pomocou volania d # # class (tt.Tip22). Init () už môžeme otestovať vyhľadávanie.

```
Cache TRM:5944 (MASTER)

SYM>d  ##CLASS (tt.,Tip22).Init()
Import OK!
SYM>d  ##CLASS (tt.,Tip22).NajdiAlias("Alias
1 3")
ID      Nazev
1       Prvek 1
3       Prvek 3

2 Rows (s)  Affected
SYM>d  ##CLASS (tt.Tip22) .NajdiAlias (" ")
ID      Nazev

0 Rows (s) Affected
SYM>d ## CLASS(tt.Tip22) .NajdiAlias ()
ID      Nazev
1       Prvek 1

1 Rows (s) Affected
SYM>d ## CLASS(tt.Tip22) .NajdiAlias ("Prvek
2")
ID      Nazev
2       Prvek 2

1 Rows (s) Affected
SYM>
```

Zvedavejší z vás si určite dajú tú prácu a pozrú sa jednak na fyzické úlo[illegible] jednak na Query plan použitého dopytu SQL.

DAN KUTÁČ, InterSystems

Ďalšie zaujímavé diely seriálu Tipy a triky s Caché, ktoré vychádzajú už od roku 2005, nájdete na stiahnutie tu: **http://www.intersystems.cz/education/university/serialy/tipyTrikyCache.html**