

Re: Merging beans for composite search

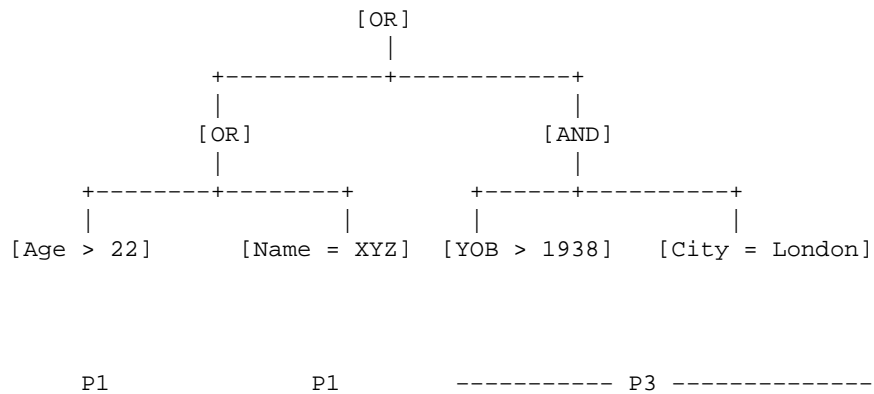
Source: <http://coding.derkeiler.com/Archive/General/comp.object/2005-07/msg00951.html>

- *From:* "H. S. Lahman" <h.lahman@xxxxxxxxxxx>
 - *Date:* Fri, 22 Jul 2005 17:51:06 GMT
-

Responding to Vivoli...

This sounds like you need to organize your compound criteria into a data structure where you can easily access (iterate over) the minimum sum of products. For example, a tree with the ORs all at the root:

Ok this structure is sound but I'm not able to figure how to use it (see below)



The minimum sum of products is {P1, P2, P3} and the compound criteria is satisfied if any one of the products evaluates to TRUE. So your bean "walks" the leaves of the criteria tree in order of boolean products and evaluates each product's criterion against the People in hand.

Re: Merging beans for composite search

Ehm...I've some problem here understandig what you mean with
"and the compound criteria is satisfied if any one of the products
evaluates to TRUE"

and

"and evaluates each product's criterion against the People in
hand."

I think I may have misunderstood your problem. I thought you had a collection of [Person] objects with some set of attributes and you needed to execute search to find some subset of those objects based on a compound set of /arbitrary/ criteria. IOW, the classic query problem for RAD systems where the user defines a query on a form with some arbitrary selection of attributes and values and wants all the tuples that satisfy that criteria. So the problem is to find some generic way evaluate any such compound criteria against the target instances.

A typical query might be: give me all people who are older than 22 or who are named XYZ or who were born after 1938 and live in London. This requires checks of the individual attributes {YOB, Name, City}. Since the criteria is compound, one has separate boolean products, P1, P2, and P3, that need to be evaluated. The given Person object is selected if at least one of the products evaluates as true. IOW, one needs to express the entire query selection criteria as a generic data structure that can be "walked" in the same manner for all queries (i.e., one manages complexity by separating the concerns of mapping criteria to the data structure and evaluation).

However, that doesn't seem to be your problem... B-)

I think (but maybe I just didn't understand your reply:P) that the misunderstandig arise since I didn't post the whole specs for this search, so I 'm going to try writing them more precisely.

-The search is composed of N Searches, to be executed in an ordered way
-Each Search S_i (i=1...N) is a compound of 1 or more subsearches (this

Re: Merging beans for composite search

was the case I was posting originally) $S_{\{i,j\}}$ $j=1\dots J_i$

```
-The execution of  $S_{\{i,j\}}$  and of the  $S_i$  follow the following rule:  
1 if  $S_{\{i,0\}}$  returned an empty set           => skip to  $S_{\{i+1\}}$   
2 if  $S_{\{i,0\}}$  returned a single record         => this is the result  
3 if  $S_{\{i,0\}}$  returned a result set  $S_0$  ( $|S_0|>1$ ) => execute  $S_{\{i,1\}}$  over  
   $S_0$   
4 for every  $S_{\{i,j\}}$  called  
5   if  $S_{\{i,j\}}$  returned an empty set         =>stop and accept the  
  previous                                     result(  $S_{\{j-1\}}$  )
```

If I understand this, you will accept a subset of tuples for a more general criteria if none of the members satisfy the more specific criteria, right?

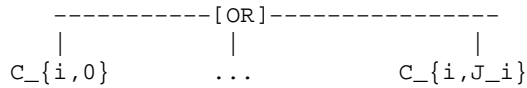
```
6   if  $S_{\{i,j\}}$  returned a single record       =>accept this one  
7   if  $S_{\{i,j\}}$  returned  $S(j)$ ,  $|S(j)|>1$        =>execute  $S_{\{i,j+1\}}$  over  
   $S(j)$ 
```

If I understand this correctly, the overall goal is to extract a single tuple based on a hierarchical suite of rules for pruning multiple members of subsets...

this is the procedural description that I'm trying to avoid since there are going to be changes/additions in the future.
The case I was posting earlier refers to line 7 where I have the search criteria of $S_{\{i,j\}}$ and I want to narrow the search using the one of $S_{\{i,j+1\}}$.

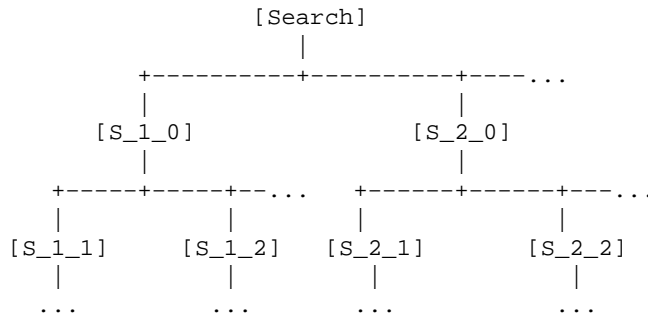
So the tree-like structure of each S_i would be flat that is (C is the criteria of the search S):

Re: Merging beans for composite search



wouldn't it?

I don't think of it that way. I see it as more like



Each $[S_{i_j}]$ node has an attribute matching criteria, C_{i_j} . At each node of the tree, one either: (a) descends (multiple tuples pass); (b) moves laterally (no passes and $J = 0$); or ascends ($J > 0$ and no passes). (Obviously, more levels would require more subscripts but the depth could be arbitrary.)

When the bean "walks" this tree it would evaluate the C_{i_j} associated with each node and decide where to go next.

The issue here is that this is the runtime structure, which uses a search resultset to determine which action to take next; nevertheless the other part is the instantiation of it: The exact criteria that are loaded depend on which attributes are set in the People object that we are searching on.

I am a bit confused about the phrasing here. This implies there is only one [People] object that one already has in hand. So what are you actually searching for (i.e., what is the set being searched)?

As I understand it now, you have a set of objects, $[X]$, from which you would like to select one. You have a hierarchical suite of rules for pruning the set. At each pruning node in the hierarchy, the specific values to compare to the $[X]$ attributes when checking are determined by attribute values in a People object.

Question: Are the specific attributes of $[X]$ to be checked at each node

Re: Merging beans for composite search

fixed for the node or are they also determined by the attribute values of the People object?

Is this a subclassing relation? If so I don't understand how PeopleSearchCriteria IS-A People. They seem like two entirely different things.

Well actually PeopleSearchCriteria IS a People in the sense that it only contains logic to map the class People to an Oracle datatype (is an O/R mapping done by hand); thus really queries are done using People's attributes.

FWIW, I think this is a real bad habit on the part of O/R mapping and layered model infrastructures. It makes life easier for the infrastructure implementer but it is not very OO.

A [People] is an abstraction of some problem space entity and it has some set of <knowledge> responsibilities. A [SearchCriteria] is an entirely different <conceptual> beast that just makes use of [People]'s knowledge in some special way necessary to the problem solution. So they should be related by association rather than subclassing.

There is nothing wrong with me that could not be cured by a capful of Drano.

H. S. Lahman
hsl@xxxxxxxxxxxxxxxxxxxxxx
Pathfinder Solutions -- Put MDA to Work
<http://www.pathfindermda.com>
blog: <http://pathfinderpeople.blogspot.com/hslahman>
(888)OOA-PATH