Article
Gevorg Arutiunian · Nov 8, 2018 4m read

Routine that converts indices to bitmap indices

(Originally posted on Intersystems CODE by @Iain Bray) The following code snippet converts all indices in a package to bitmap indices. The subroutine "test" runs the code:

```
ROUTINE iainbray.indexToBitmap
/// f.k.a. bitmapIndices
test(package,logfile) public { // Resets indices to "type = bitmap" where appropriate
    // Iain Bray - InterSystems Corporation
    //
    // Accepts (optional) arguments of a package and a logfile and changes all the in
dex types to bitmap
    // where the index is not a system type, is not unique and the class does not hav
e an IDKey based upon properties
    // (default IDKEY only!)
    //
    // usage: do ^bitmapIndices(myPackage,myLog)
    // where myPackage = the name of the package (case sensitive)
    //       myLog is the name of a log file
    // examples:
    //       do ^bitmapIndices()
    //       do ^bitmapIndices("User")
    //       do ^bitmapIndices("User","c:\bitmaps.txt")
    //       do ^bitmapIndices(,"c:\bitmaps.txt")

    set package=$get(package)
    set logfile=$get(logfile)

    if logfile'="" close logfile open logfile:"WNS":0 if '$test write !!,"Could not o
pen logfile!" quit

    set msg = "Change indices to bitmaps"
    if package'="" {
        set msg=msg_" for package '"_package_"'"
    }
    else {
        set msg=msg_" for all packages"
    }
    set msg=msg_" in namespace '"_$znspace_"'."
    do log(logfile,msg,1,1)

    // Resultset for classes
    set rsClass=##class(%ResultSet).%New("%Dictionary.CompiledClassQuery:Summary")
    // Resultset for indices
    set rsIndex=##class(%ResultSet).%New("%Dictionary.CompiledIndexQuery:Summary")
    if rsClass.Execute() {
        while rsClass.Next() {
            // Ignore system and non-persistent classes
            if (rsClass.Get("System")=0)&&(rsClass.Get("Persistent")=1) {
                // Check that we have the correct package - if specified
```

```
                set className=rsClass.GetData(1)
                if (package="")||($piece(className,".",1,$length(package,."))=packag
e) {
                    do log(logfile,"Class: "_className,1,1)
                    if rsIndex.Execute(className) {
                        set ok = 1
                        set indices=""
                        while rsIndex.Next() {
                            set indexName=rsIndex.GetData(1)
                            set objIndex = ##class(%Dictionary.CompiledIndex).%OpenId
(className_"||"_indexName,0)
                            // The next check looks for IDKeys with attributes - can'
t yet do bitmap indices on these
                            if (objIndex.IdKey=1)&&(objIndex.SystemAssigned=0)&&(objI
ndex.Properties'="") {
                                set ok = 0
                                quit
                            }
                            // Build up an array of indices that are ok for type = bi
tmap
                            if (objIndex.SystemAssigned=0)&&(objIndex.Unique = 0)&&(o
bjIndex.Type'="bitmap") set indices=indices_$listbuild(indexName)
                            kill objIndex
                        }
                    }
                    do rsIndex.Close()
                    if 'ok {
                        do log(logfile,"Skipped due to attributes in IDKey",0,1)
                    }
                    else {
                        if indices'="" {
                            do log(logfile,"Purging indices",0,1)
                            do $zobjclassmethod(className,"%PurgeIndices",indices)
                            do log(logfile,"Compiling class",0,1)
                            set indexLength = $listlength(indices)
                            set ok = 1
                            for index=1:1:indexLength {
                                set indexName=$list(indices,index)
                                set objIndex = ##class(%Dictionary.IndexDefinition).%
OpenId(className_"||"_indexName)
                                set objIndex.Type = "bitmap"
                                set ok = objIndex.%Save()
                                if 'ok quit
                            }
                            if 'ok {
                                do log(logfile,"FATAL ERROR editing "_indexName,0,1)
                            }
                            else {
                                set sc = $system.OBJ.Compile(className,"-d")
                                if 'sc {
                                    do log(logfile,"FATAL ERROR compiling "_className
,0,1)
                                }
                                else {
                                    for index=1:1:indexLength do log(logfile,"Changed
 Index: "_$list(indices,index),0,1)
                                    do log(logfile,"Re-Building indices",0,1)
                                    do $zobjclassmethod(className,"%BuildIndices",ind
ices)
```

```
                                }
                            }
                        }
                        else {
                            do log(logfile,"No indices to re-build",0,1)
                        }
                    }
                }
            }
        }
    }
    do rsClass.Close()
    if logfile'="" close logfile
    quit
}

log(logfile,text,time,newline) private {
    use $principal do write(text,time,newline)
    if logfile'="" use logfile do write(text,time,newline)
    use $principal
}
write(text,time,newline) private {
    set time=$get(time,0)
    set newline=$get(newline,0)
    if time set newline=1
    if newline write !
    if time write $zdatetime($horolog,3)
    if newline write ?25
    write text
}
```

Here's a link to the code on GitHub: https://github.com/intersystems-community/code-snippets/blob/master/src/...

#Code Snippet #InterSystems IRIS #ObjectScript #Caché