Article
Robert Cemper · Aug 28, 2020 2m read

Open Exchange

Effective use of Collection Indexing and Querying Collections through SQL

Triggered by a question placed by @Kurro Lopez recently
I took a closer look at the indexing of collections.
My simple test setup is a serial class and a persistent class with a list of this serial.

```
Class rcc.IC.serItem Extends (%SerialObject, %Populate)
{ Property Subject As %String [ Required ];
  Property Change As %TimeStamp [ Required ];
  Property Color As %String(COLLATION = "EXACT",
     VALUELIST = ",red,white,blue,yellow,black,unknown") [ Required ];
}
```

```
Class rcc.IC.ItemList Extends (%Persistent, %Populate) [ Final ]
{ Property Company As %String [ Required ];
  Property Region As list Of %String(COLLATION = "EXACT", POPSPEC = ":4",
     VALUELIST = ",US,CD,MX,EU,JP,AU,ZA") [ Required ];
  Property Items As list Of rcc.IC.serItem(POPSPEC = ":4") [ Required ];

  Index xitm On Items(ELEMENTS);
  Index ycol On Items(ELEMENTS).Color;
}
```

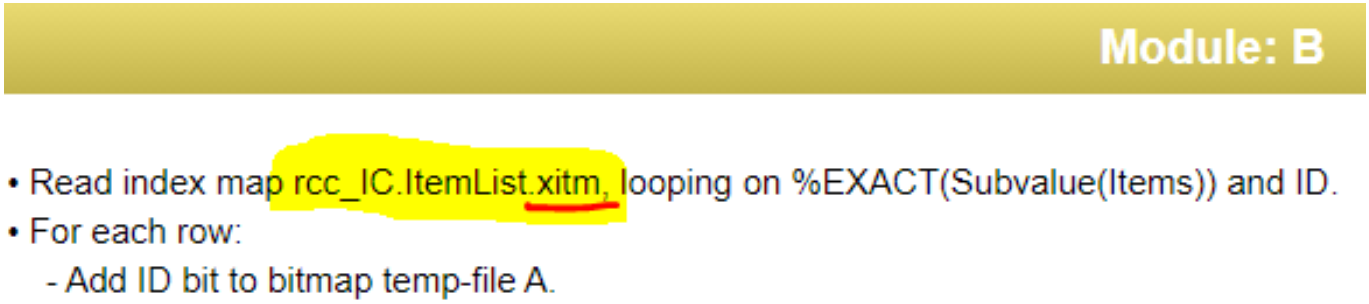
Related Docs
Index **xitm** holds the complete serial element. !!
With some records generated by %Populate utility I could place this query

```
Select ID,Company from rcc_IC.ItemList
Where FOR SOME %ELEMENT(rcc_IC.ItemList.Items) ($list(%Value,3) in ('blue','yellow'))
```

This works OK but disassembling every serial object wasn't very promising for my performance considerations.
So I followed a hit from @Dan Pasco recently seen in this forum a few days ago,
and expecting better performance I added

```
Index ycol On Items(ELEMENTS).Color;
```

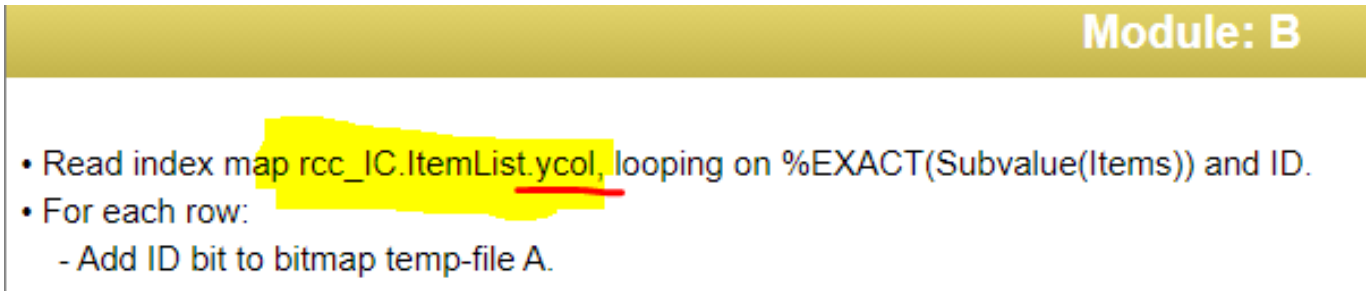
The result was rather disappointing.
No improvement.
Investigation of the query plan showed that the new index was just ignored.



After some trials, this query satisfied my needs

```
Select ID,Company
from %IGNOREINDEX xitm rcc_IC.ItemList
Where FOR SOME %ELEMENT(rcc_IC.ItemList.Items) ('blue,yellow' [ %Value )
```

with



During the investigation with many variations I found this rule:

IF you have more than one ELEMENT index on the same property the
query generator **always** takes the alphabetic first index it finds.
And you have to explicitly exclude a non-fitting index.

As there is no hint in the documentation **I would like to know:**

Is this observation correct or is it just an accidental effect in my case?

As ELEMENT index was designed for List of %String I understand that having
more than one index was just an unlikely case at the time of design.

GitHub

#Other
Check the related application on InterSystems Open Exchange