Artigo
ROBSON SERPA DA ROSA · Jun. 28, 2021 5min de leitura

Views materializadas

Uma VIEW em SQL é basicamente uma instrução SQL preparada.
Deve ser executado e montado como qualquer outra consulta SQL.
VIEW MATERIALIZADA significa que o conteúdo é coletado antes das mãos e pode ser recuperado com bastante rapidez.
Eu vi o conceito primeiro com meu concorrente favorito chamado O * e eles fizeram muito barulho sobre isso.

```
{ favorite: because I could win every benchmark against them 😈 }
```

Dependendo da construção individual dessas views materializadas, atualizações e manutenção podem ser necessárias.

No Caché / IRIS isso existe quase desde sempre e consideramos isso normal e um fato dado.
A maioria dos desenvolvedores simplesmente não está ciente disso e, com um pouco de polimento, pode ser apresentado como um excelente recurso.
Além disso, qualquer atualização e manutenção acontecem como uma funcionalidade integrada sem nenhum esforço extra.

Veja este exemplo:
Em nossa famosa classe Sample.Person no namespace SAMPLES, estendi e reconstruí seu índice.

```
/// Define an index for <property>Name</property>.
Index NameIDX On Name [ Data = (Name, Home.State, SSN) ];
```

E sendo experiente e confortável com o gerador de consultas, você sabe que

```
SELECT ID, Name, Home_State, SSN  from Sample.Person
```

será executado rapidamente usando apenas o índice global ^ Sample.PersonI ("NameIDX")
e nunca toque em seus dados globais.
Essa é basicamente a funcionalidade de uma View materializada e a atualização está implícita.

Definido como VIEW (de MgmtPortal, já que Studio não é tão prático) você obtém

esta classe.

```
Class Sample.Person.NameView [ ClassType = view
                             , CompileAfter = Sample.Person
                             , DdlAllowed
                             ,  Not ProcedureBlock
                             , SqlTableName = NameView
                             , ViewQuery = { SELECT ID
                                                  , Name

, Home_State
                                                  , SSN
   from Sample.Person  } ]
  { Parameter READONLY = 1; }
```

Mas se você quiser um pouco mais de conforto, como um backlink para seus dados, pode mapear o próprio índice global.
Portanto, você pode aplicar a sintaxe JOIN implícita e ter uma tabela totalmente funcional como aqui

```
SELECT Name, BaseClass->DOB, HomeState,
SSN,"%CLASSNAME",BaseClass FROM Sample_Person.NameIDX
```

Name	DOB	HomeState	SSN	%CLASSNAME	BaseClass
Bachman,George E.	06/12/1977	AK	232-30-3200		67
Bachman,Hannah A.	05/22/1955	UT	537-37-2290		83
Bachman,Laura C.	11/24/2014	IN	899-59-6879	~Sample.Employee~	151
Basile,Lawrence N.	06/15/2008	NC	400-29-2516	~Sample.Employee~	171
Basile,Mark C.	12/18/1966	CT	275-41-4866		18

e aqui está a definição da classe e você deve projetá-la manualmente

```
/// mapped index
/// Index NameIDX On Name [ Data = (Name, Home.State, SSN)
];
Class Sample.Person.NameIDX Extends %Persistent [ Final ]
{
Property IndexName As %String [
InitialExpression = "NameIDX", ReadOnly ];
Property SQLUPPERname As %String [ ReadOnly ];
Property BaseClass As Sample.Person [ ReadOnly ];

Index min On (IndexName, SQLUPPERname, BaseClass) [ IdKey ];
```

```
/// Classname of Index Source
Property %CLASSNAME As %String [ ReadOnly ];
/// Person's name.
Property Name As %String [ ReadOnly ];
/// Person's home address. This uses an embedded object.
Property HomeState As %String [ ReadOnly ];
/// Person's Social Security number. This is validated using
pattern match.
Property SSN As %String(PATTERN = "3N1""-""2N1""-""4N")
[ ReadOnly ];

Parameter READONLY = 1;
Parameter MANAGEDEXTENT As INTEGER = 0;

Storage Default
{
 <Data name="NameIDXDefaultData">
  <Value name="1">
  <Value>%CLASSNAME</Value>
  </Value>
  <Value name="2">
  <Value>Name</Value>
  </Value>
  <Value name="3">
  <Value>HomeState</Value>
  </Value>
  <Value name="4">
  <Value>SSN</Value>
  </Value>
 </Data>
 <DataLocation>^Sample.PersonI</DataLocation>
 <DefaultData>NameIDXDefaultData</DefaultData>
 <IdLocation>^Sample.Person.NameIDXD</IdLocation>
 <IndexLocation>^Sample.Person.NameIDXI</IndexLocation>
 <StreamLocation>^Sample.Person.NameIDXS</StreamLocation>
 <Type>%Library.CacheStorage</Type>
} }
```

Este é um exemplo de codificação funcionando no Caché 2018.1.3 e IRIS 2020.2
Não será sincronizado com as novas versões

```
Também NÃO é atendido pelo Suporte da InterSystems!
```

#Indexando #ObjectScript #SQL #Caché #Ensemble #InterSystems IRIS

URL de origem:https://pt.community.intersystems.com/post/views-materializadas