

For each em profundidade

Cláusulas order e desempenho

GeneXus™

Vamos nos concentrar nas cláusulas order e sua relação com a otimização da consulta.

```
For each BaseTrn1, ... , BaseTrnn  
    skip expression1 count expression2  
    order att1, att2, ... , attn [when condition]  
    order att1, att2, ... , attn [when condition]  
    order none [when condition]  
    unique att1, att2, ... , attn  
    using DataSelector ( parm1, parm2, ... , parmn )  
    where condition [when condition]  
    where condition [when condition]  
    where att IN DataSelector ( parm1, parm2, ... , parmn )  
    blocking n  
        main_code  
    when duplicate  
        when_duplicate_code  
    when none  
        when_none_code  
endfor
```

Sabemos que para ordenar a informação a ser consultada e retornada, a sintaxe do for each nos permite especificar uma lista de cláusulas order condicionais e uma incondicional.

For each $BaseTrn_1, \dots, BaseTrn_n$

```

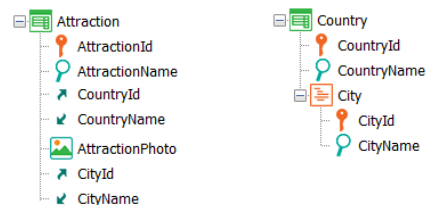
skip expression1 count expression2
order att1, att2, ... , attn [when condition]
order att1, att2, ... , attn [when condition]
order none [when condition]
unique att1, att2, ... , attn
using DataSelector ( parm1, parm2, ... , parmn )
where condition [when condition]
where condition [when condition]
where att IN DataSelector ( parm1, parm2, ... , parmn )
blocking n

```

```

main_code
for each Attraction
  order AttractionName
  where CountryId = &countryId
  where CityId = &cityId
  print attractionInfo //AttractionName
endfor
when_none_code
endifor

```



Vamos do mais simples ao mais complexo: vamos começar analisando o caso de uma única cláusula order sem condições, por exemplo esta. Estamos solicitando que seja percorrida a tabela de atrações turísticas, filtrando aquelas que correspondem a um país e cidade determinados, e que as apresente ordenadas por AttractionName, atributo secundário.

Na hora de especificar o que queremos, não deveria nos importar se para obter os dados necessários, primeiro os obtém e depois os classifica, ou se faz o contrário ou de alguma outra maneira. Nós desenvolvedores especificamos o que queremos, e de resolvê-lo se encarrega o especificador de GeneXus e, sobretudo, o DBMS.

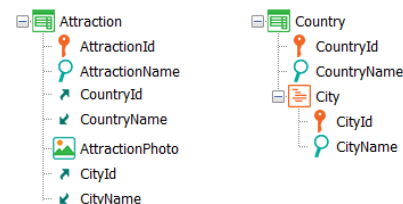
```

for each Attraction
  order AttractionName
  where CountryId = &countryId
  where CityId = &cityId
  print attractionInfo //AttractionName
endfor

```



Net - SQL Server
Java - Oracle



Warnings ^

▲ spc0038 There is no index for order [AttractionName](#); poor performance may be noticed in group starting at line 11.

LEVELS ^

For Each Attraction (Line: 11) ^

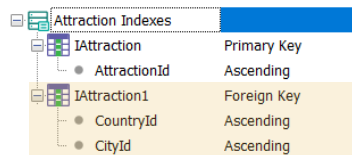
Order: [AttractionName](#)
No index!

Navigation Start from: FirstRecord

filters: Loop while: NotEndOfTable

Constraints: [CountryId](#) = &countryId
[CityId](#) = &cityId

[Attraction](#) ([AttractionId](#)) INTO [CountryId](#) [CityId](#) [AttractionName](#)



Quando pedimos ao GeneXus que especifique e gere o programa associado ao objeto que contém o for each, o fazemos para um environment específico, ou seja, em uma linguagem de programação específica, como Net, por exemplo, e para uma base de dados gerenciada por um determinado DBMS, como SQL Server, por exemplo. Ou poderia ser para um environment Java com Oracle, ou tantas outras alternativas.

Quando o desenvolvedor escreve seu For each, o faz em GeneXus, com certa independência de qual será o environment final, justamente para que com o mesmo código possamos obter a aplicação em diferentes environments.

Isto significa que da implementação específica se encarrega GeneXus, que conhece as particularidades de cada environment. No entanto, seu conhecimento tem um limite: conhece a estrutura da base de dados, mas não os dados, nem sua distribuição, quantidade etc. Estas informações o DBMS possui, que registra estatísticas, armazena dados e consultas no histórico das consultas que estão sendo realizadas, constrói planos de execução, mantém índices, etc. Quanto mais avançado e inteligente for o DBMS, menos necessitará que GeneXus lhe indique com precisão como realizar a consulta, porque GeneXus nunca saberá mais do que ele.

Assim, por exemplo, se pedimos que seja especificado o objeto que contém este For each, veremos esta lista de navegação, que nos avisa que não há um índice pelo atributo pelo qual queremos mostrar ordenada a consulta e que, portanto, poderíamos notar um baixo desempenho. “Poderíamos” não significa que realmente será assim. Por quê? Porque isso depende não só da quantidade de

dados da tabela, mas também do DBMS e suas estratégias. O que nos indica a lista de navegação é o pior cenário: neste, deverá ser percorrida toda a tabela ordenada por um atributo para o qual possivelmente não haja um índice (GeneXus não sabe se o DBMS o criou sozinho ou não, a verdade é que não tem notícias dele) então no pior cenário, que é o de arquiteturas centralizadas, poderia ter que criar temporariamente o índice para resolver a consulta por essa ordem e assim percorrer toda a tabela para avaliar registro por registro se cada um satisfaz ou não os filtros. Este seria o cenário de uma base de dados com um gerenciador pouco inteligente.

Pensemos, por exemplo, que neste caso talvez seja uma melhor estratégia utilizar o índice de chave estrangeira {CountryId, CityId} que sabemos que existe porque GeneXus obriga a criá-lo. Então, são obtidos de forma otimizada os registros que atendem aos filtros e só então, com esse resultado, é ordenado por AttractionName.

Qual é a melhor estratégia, dependerá em grande parte da distribuição dos dados. Se houver apenas 3 atrações desse país e cidade entre milhões, esta parece ser uma melhor estratégia porque o custo de ordenar 3 registros é insignificante. No entanto, se houver milhões de registros na tabela, sendo a grande maioria desse país e cidade, então usar o índice por país e cidade não reduzirá muito a consulta de toda a tabela e, em seguida, ordenar o resultado por AttractionName ser quase o mesmo que de entrada ordenar por AttractionName e depois avaliar um por um se também corresponde ao país/cidade ou não. GeneXus não conhece a distribuição de dados para tomar decisões deste tipo. Além disso, se esta mesma consulta já foi realizada antes, o DBMS possivelmente armazena em cache o resultado e não tenha que realizar a consulta novamente da mesma maneira. Isto, claro, se os filtros não mudaram e os dados também não.

Em resumo, então, a lista de navegação nos informa com base no cenário mais conservador, com o pior DBMS. Mas pode ser que o DBMS melhore muito o cenário mais pessimista e a consulta seja otimizada.

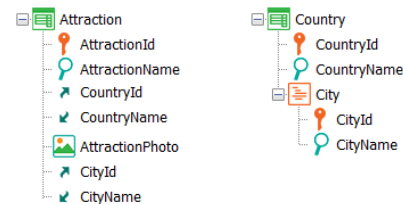
```

for each Attraction
order AttractionName
where CountryId = &countryId
where CityId = &cityId
print attractionInfo //AttractionName
endfor

```



Net - SQL Server
Java - Oracle



| LEVELS | |
|--------------------------------|---|
| For Each Attraction (Line: 24) | |
| Order: | AttractionName |
| | Index: UATTRACTION |
| Navigation filters: | Start from: FirstRecord |
| | Loop while: NotEndOfTable |
| Constraints: | CountryId = &countryId |
| | CityId = &cityId |
| | =Attraction (AttractionId) INTO CityId CountryId AttractionName |

| Attraction X | | |
|--------------------|-------------|-----------------|
| Structure | | |
| Indexes | | |
| Attribute | Order | Description |
| Attraction Indexes | | |
| IAttraction | Primary Key | Attraction |
| ● AttractionId | Ascending | Automatic Index |
| IAttraction1 | Foreign Key | Attraction Id |
| ● CountryId | Ascending | Automatic Index |
| ● CityId | Ascending | Country Id |
| IAttraction2 | Foreign Key | City Id |
| ● CategoryId | Ascending | Automatic Index |
| ● CategoryId | Ascending | Category Id |
| UAttraction | Duplicate | User Index |
| ● AttractionName | Ascending | Attraction Name |

Poderíamos ser tentados a pensar que se sabemos que haverá milhões de registros na tabela Attraction, o melhor será solicitar a partir do GeneXus para o DBMS a criação do índice de usuário por AttractionName. Neste caso, será reorganizada a base de dados, e ao especificar novamente o objeto, agora GeneXus nos indicará que utilizará o novo índice para resolver a consulta e não será mais exibido o aviso de desempenho.

No entanto, esta pode ser uma solução muito pior. Justamente, se o DBMS for inteligente, não precisará em absoluto que o forcemos a criar um índice. Ele mesmo fará isso se precisar. Tanto que nem mesmo neste caso, em que o próprio GeneXus solicitou a criação do índice, é enviado ao DBMS quando realiza a consulta a partir do environment com DBMS inteligente.

```
listcountries.cs - Visual Studio Code
File Edit Selection View Go Run Terminal Help
listcountries.cs x .redAtt{ Untitled-1
C:\> Models > GX175tableForu9 > TravelAgency_forExpert > NetModel > Web > listcountries.cs
294 public class listcountries_default : DataStoreHelperBase, IDataStoreHelper
295
296 public ICursor[] getCursors( )
297 {
298     cursorDefinitions();
299     return new Cursor[] {
300         new ForEachCursor(def[0])
301     };
302
303
304 private static CursorDef[] def;
305 private void cursorDefinitions( )
306
307 if ( def == null )
308 {
309     object[] prmP000Q2;
310     prmP000Q2 = new Obj
311     new ParDef("@AV14cou
312     new ParDef("@AV15cit
313     };
314     def= new CursorDef[] {
315         new CursorDef("P000Q2", "SELECT [CityId], [CountryId], [AttractionName], [AttractionId] FROM [Attraction] WHERE ([CountryId] = @AV14countryId) AND ([CityId] = @A
316     };
317 }
318
319
320 public void getResults( int cursor ,
321     IFieldGetter rslt ,
322     Object[] buf )
323
324 switch ( cursor )
325 {
326     case 0 :
327         ((short[]) buf[0])[0] = rslt.getShort(1);
328         ((short[]) buf[1])[0] = rslt.getShort(2);
329         ((string[]) buf[2])[0] = rslt.getString(3, 50);
330         ((short[]) buf[3])[0] = rslt.getShort(4);
331         return;
332 }
333
Ln 318, Col 6 Spaces: 3 UTF-8 CRLF C#
```

Basta observar o fonte gerado para o environment Net com SQL Server. Na instrução Select não está sendo enviada nenhuma informação sobre o índice a ser utilizado. Por que vai enviá-la se o DBMS sabe de sua existência? Se precisar, vai utilizá-lo. E se não o faz, é porque vai utilizar uma estratégia melhor.

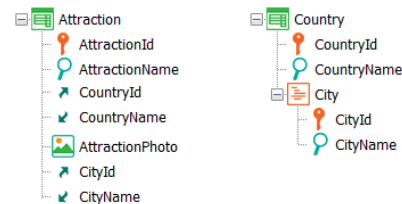
Podemos supor que seria diferente o caso se o DBMS fosse um sem inteligência.

```

for each Attraction
  order AttractionName
  where CountryId = &countryId
  where CityId = &cityId
  print attractionInfo //AttractionName
endfor
    
```



Net - SQL Server
Java - Oracle



LEVELS

For Each Attraction (Line: 24)

Order: [AttractionName](#)
Index: UATTRACTION

Navigation filters: Start from: [FirstRecord](#)
Loop while: [NotEndOfTable](#)

Constraints: [CountryId](#) = &countryId
[CityId](#) = &cityId

= [Attraction \(AttractionId \)](#) INTO [CityId](#) [CountryId](#) [AttractionName](#)

Attraction X

Structure **Indexes**

| Attribute | Order | Description |
|--------------------|-------------|-----------------|
| Attraction Indexes | | |
| IAttraction | Primary Key | Automatic Index |
| ● AttractionId | Ascending | Attraction Id |
| IAttraction1 | Foreign Key | Automatic Index |
| ● CountryId | Ascending | Country Id |
| ● CityId | Ascending | City Id |
| IAttraction2 | Foreign Key | Automatic Index |
| ● CategoryId | Ascending | Category Id |
| UAttraction | Duplicate | User Index |
| ● AttractionName | Ascending | Attraction Name |

Devemos levar em conta que a criação de um índice é algo dispendioso, e que tem implicações não só no momento de sua criação, mas também depois no momento de sua manutenção, durante todo o ciclo de vida da tabela. Cada vez que a tabela é atualizada, é pago um pequeno preço para mantê-lo.

É por isso que não parece ser uma boa prática a criação de índices de usuário, a menos que sejam necessários para controlar unicidade, ou seja, para definir chaves candidatas, como índices unique.


```

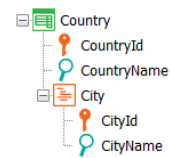
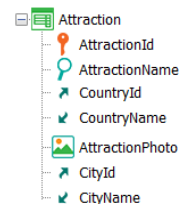
for each Attraction
  order AttractionName
  where CountryId = &countryId
  where CityId = &cityId
  print attractionInfo //AttractionName
endfor

```



Net - SQL Server

Java - Oracle



Warnings ⌵

▲ spc0038 There is no index for order [AttractionName](#); poor performance may be noticed in group starting at line 11.

LEVELS ⌵

For Each Attraction (Line: 11) ⌵

Order: [AttractionName](#)
No index!

Navigation Start from: FirstRecord

filters: Loop while: NotEndOfTable

Constraints: [CountryId](#) = &countryId
[CityId](#) = &cityId

[Attraction](#) ([AttractionId](#)) INTO [CountryId](#) [CityId](#) [AttractionName](#)

Então, como dissemos, o especificador de GeneXus faz o melhor que pode com a informação que tem e com sua inteligência atual (espera-se que essa inteligência aumente conforme GeneXus evolui), e envia a consulta mais otimizada possível ao DBMS sem lhe indicar o óbvio, sabendo que este, no pior dos casos a aceitará, mas em geral vai melhorá-la.

Em resumo, nunca podemos garantir que o que indica a lista de navegação será o que finalmente fará o DBMS quando este é inteligente. O que sabemos é que será isso ou algo melhor.


Vejamos exemplos da inteligência atual do especificador de GeneXus, além do que depois acaba fazendo o DBMS.

Sabemos que a ordem em que será solicitado que os registros resultantes sejam retornados é determinada com base na especificação do desenvolvedor na cláusula order, mas também em algoritmos internos de otimização.

```
for each Attraction
```

```
  where CountryId = &countryId
  where CityId = &cityId
  print attractionInfo //AttractionName
endfor
```

```
"SELECT [CityId], [CountryId],
[AttractionName], [AttractionId] FROM
[Attraction] WHERE [CountryId] =
@AV14countryId and [CityId] = @AV15cityId
ORDER BY [CountryId], [CityId]"
```

| LEVELS | |
|--------------------------------|---|
| For Each Attraction (Line: 11) | |
| Order: | <u>CountryId</u> , <u>CityId</u> Index: IATTRACTION1 |
| Navigation | Start from: <u>CountryId</u> = &countryId |
| filters: | <u>CityId</u> = &cityId Loop while: <u>CountryId</u> = &countryId <u>CityId</u> = &cityId |
| |  =Attraction (<u>AttractionId</u>) INTO <u>CityId</u> <u>CountryId</u> <u>AttractionName</u> |

Por exemplo, se não nos importasse como serão ordenados os nomes de atração, poderíamos escrever o For each sem cláusula order. Em geral, sabíamos que, se fizessemos isto, a consulta seria classificada por chave primária. Ou seja, que se enviava ao DBMS um Select com order AttractionId. No entanto, neste caso, acontecerá algo diferente, como vemos na lista de navegação. Sabemos que existe na base de dados um índice por CountryId e CityId, os dois filtros por igualdade. Sabemos disso porque formam uma chave estrangeira. Claramente, então, seria preferível utilizar esse índice e retornar a consulta classificada por esses valores. É por isso que se observamos a consulta que envia o fonte ao DBMS, encontramos este ORDER BY. Se nossa intenção ao não especificar cláusula order era que fossem ordenados por chave primária, neste caso teremos que torná-la explícita.

Da mesma forma, se agora a consulta for esta outra, onde estamos pedindo que as atrações sejam ordenadas por identificador de cidade, e só estamos filtrando por identificador de país, veremos que a lista de navegação não está mostrando que pedirá para ordenar por CityId, mas pelo par, pois assim otimiza a consulta, pois sabe da existência do índice composto (por ser chave estrangeira, justamente), sem deixar de cumprir o resultado de ordenação solicitado pelo desenvolvedor.

Em resumo, com a cláusula order o desenvolvedor indica a ordem em que deseja que os registros sejam retornados, mas para isso o especificador pode alterar essa cláusula, complementando-a com informação contextual (se houver condições implícitas ou explícitas por igualdade e houver índice que as contém, além dos atributos da order) de forma a otimizar a consulta, embora o DBMS será que tem no final das contas a última palavra.

O importante é entender que os dados serão retornados na ordem explicitada pelo desenvolvedor, mesmo que para resolver a consulta sejam utilizados outros critérios.

```

for each Attraction
order AttractionId
where CountryId = &countryId
where CityId = &cityId
print attractionInfo //AttractionName
endfor


```

```

"SELECT [CityId], [CountryId],
[AttractionName], [AttractionId] FROM
[Attraction] WHERE [CountryId] =
@AV14countryId and [CityId] = @AV15cityId
ORDER BY [CountryId], [CityId]"

```

[AttractionId]

| LEVELS | |
|---|---|
| For Each Attraction (Line: 11) | |
| Order: | <u>CountryId</u> , <u>CityId</u> Index: IATTRACTION1 |
| Navigation | Start from: <u>CountryId</u> = &countryId |
| filters: | <u>CityId</u> = &cityId Loop while: <u>CountryId</u> = &countryId <u>CityId</u> = &cityId |
|  =Attraction (<u>AttractionId</u>) INTO <u>CityId</u> <u>CountryId</u> <u>AttractionName</u> | |

Se nossa intenção ao não especificar cláusula order era que fossem ordenados por chave primária, neste caso teremos que torná-la explícita.


```
for each Attraction
```


```
  where CountryId = &countryId
  where CityId = &cityId
  print attractionInfo //AttractionName
endfor
```

```
"SELECT [CityId], [CountryId],
[AttractionName], [AttractionId] FROM
[Attraction] WHERE [CountryId] =
@AV14countryId and [CityId] = @AV15cityId
ORDER BY [CountryId], [CityId]"
```

```
for each Attraction
```

```
  order CityId
  where CountryId = &countryId
  print attractionInfo //AttractionName
endfor
```

| LEVELS | |
|--------------------------------|---|
| For Each Attraction (Line: 11) | |
| Order: | <u>CountryId</u> , <u>CityId</u> |
| | Index: IATTRACTION1 |
| Navigation | Start from: <u>CountryId</u> = &countryId |
| filters: | <u>CityId</u> = &cityId |
| | Loop while: <u>CountryId</u> = &countryId |
| | <u>CityId</u> = &cityId |
| |  =Attraction (<u>AttractionId</u>) INTO <u>CityId</u> <u>CountryId</u> <u>AttractionName</u> |

| LEVELS | |
|--------------------------------|---|
| For Each Attraction (Line: 11) | |
| Order: | <u>CountryId</u> , <u>CityId</u> |
| | Index: IATTRACTION1 |
| Navigation | Start from: <u>CountryId</u> = &countryId |
| filters: | Loop while: <u>CountryId</u> = &countryId |
| |  =Attraction (<u>AttractionId</u>) INTO <u>CountryId</u> <u>AttractionName</u> <u>CityId</u> |

Da mesma forma, se agora a consulta for esta outra, onde estamos pedindo que as atrações sejam ordenadas por identificador de cidade, e só estamos filtrando por identificador de país, veremos que a lista de navegação não está mostrando que pedirá para ordenar por CityId, mas pelo par, pois assim otimiza a consulta, pois sabe da existência do índice composto (por ser chave estrangeira, justamente), sem deixar de cumprir o resultado de ordenação solicitado pelo desenvolvedor.

Em resumo, com a cláusula order o desenvolvedor indica a ordem em que deseja que os registros sejam retornados, mas para isso o especificador pode alterar essa cláusula, complementando-a com informação contextual (se houver condições implícitas ou explícitas por igualdade e houver índice que as contém, além dos atributos da order) de forma a otimizar a consulta, embora o DBMS será que tem no final das contas a última palavra.

O importante é entender que os dados serão retornados na ordem explicitada pelo desenvolvedor, mesmo que para resolver a consulta sejam utilizados outros critérios.

```
for each Attraction
```


```
  where CountryId = &countryId
  where CityId = &cityId
  print attractionInfo //AttractionName
endfor
```


```
"SELECT [CityId], [CountryId],
[AttractionName], [AttractionId] FROM
[Attraction] WHERE [CountryId] =
@AV14countryId and [CityId] = @AV15cityId
ORDER BY [CountryId], [CityId]"
```

```
for each Attraction
```

```
  where AttractionName = &attractionName
  print attractionInfo //AttractionName
endfor
```

```
"SELECT [AttractionName], [AttractionId]
FROM [Attraction] WHERE [AttractionName] =
@AV8AttractionName ORDER BY [AttractionId]"
```

| LEVELS | |
|--------------------------------|---|
| For Each Attraction (Line: 11) | |
| Order: | <u>CountryId</u> , <u>CityId</u> |
| | Index: IATTRACTION1 |
| Navigation filters: | Start from: <u>CountryId</u> = &countryId |
| | <u>CityId</u> = &cityId |
| | Loop while: <u>CountryId</u> = &countryId |
| | <u>CityId</u> = &cityId |
| |  =Attraction (<u>AttractionId</u>) INTO <u>CityId</u> <u>CountryId</u> <u>AttractionName</u> |

| LEVELS | |
|--------------------------------|--|
| For Each Attraction (Line: 24) | |
| Order: | <u>AttractionId</u> |
| | Index: IATTRACTION |
| Navigation filters: | Start from: FirstRecord |
| | Loop while: NotEndOfTable |
| Constraints: | <u>AttractionName</u> = &AttractionName |
| |  =Attraction (<u>AttractionId</u>) INTO <u>AttractionName</u> |

Neste caso, onde não foi especificada order, como existe índice que permite otimizar essas condições, é o escolhido, em vez do índice por chave primária.

Quando não há índice, então escolhe a chave primária. Aqui vemos a instrução SQL que constrói GeneXus.

```
for each Attraction
```


```
  where CountryId = &countryId
  where CityId = &cityId
  print attractionInfo //AttractionName
endfor
```

```
"SELECT [CityId], [CountryId],
[AttractionName], [AttractionId] FROM
[Attraction] WHERE [CountryId] =
@AV14countryId and [CityId] = @AV15cityId
ORDER BY [CountryId], [CityId]"
```

```
for each Attraction
```

```
  order NONE
  where AttractionName = &attractionName
  print attractionInfo //AttractionName
endfor
```

```
"SELECT [AttractionName], [AttractionId]
FROM [Attraction] WHERE [AttractionName] =
@AV8AttractionName-ORDER BY [AttractionId]"
```

| LEVELS | |
|--------------------------------|---|
| For Each Attraction (Line: 11) | |
| Order: | CountryId , CityId |
| | Index: IATTRACTION1 |
| Navigation filters: | Start from: CountryId = &countryId |
| | CityId = &cityId |
| | Loop while: CountryId = &countryId |
| | CityId = &cityId |
| |  =Attraction (AttractionId) INTO CityId CountryId AttractionName |

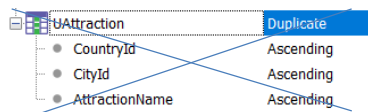
| LEVELS | |
|--------------------------------|---|
| For Each Attraction (Line: 24) | |
| Order: | NONE |
| Navigation filters: | Start from: FirstRecord |
| | Loop while: NotEndOfTable |
| Constraints: | CountryId = &countryId |
| | CityId = &cityId |
| |  =Attraction (AttractionId) INTO CountryId CityId AttractionName |

A menos que especifiquemos cláusula Order none, caso em que delegamos ao DBMS a escolha da ordem. Não se adiciona ORDER BY à instrução SQL construída por GeneXus.

```

for each Attraction
  order AttractionName
  where CountryId = &countryId
  where CityId = &cityId
  print attractionInfo
endfor

```



```

"SELECT [CityId], [CountryId],
[AttractionName], [AttractionId] FROM
[Attraction] WHERE ([CountryId] =
@AV14countryId) AND ([CityId] = @AV15cityId)
ORDER BY [AttractionName] "

```

LEVELS ^

For Each Attraction (Line: 11) ^

Order: [CountryId](#) , [CityId](#) , [AttractionName](#)
Index: UATTRACTION

Navigation filters: Start from: [CountryId](#) = &countryId
[CityId](#) = &cityId
Loop while: [CountryId](#) = &countryId
[CityId](#) = &cityId

=Attraction ([AttractionId](#)) INTO [CityId](#) [CountryId](#) [AttractionName](#)

▲ spc0038 There is no index for order [AttractionName](#); poor performance may be noticed in group starting at line 11.

LEVELS ^

For Each Attraction (Line: 11) ^

Order: [AttractionName](#)
No index!

Navigation filters: Start from: FirstRecord
Loop while: NotEndOfTable

Constraints: [CountryId](#) = &countryId
[CityId](#) = &cityId

=Attraction ([AttractionId](#)) INTO [CountryId](#) [CityId](#) [AttractionName](#)

Vamos voltar a este exemplo. Se GeneXus pediu para criar este índice de usuário composto, então o proporá na lista de navegação (mesmo que finalmente não o envie ao SQL Server, porque este já sabe de sua existência e para que dizer algo que já sabe). O benefício é que nos permite saber que pelo menos esta otimização será realizada pelo DBMS. Será assim ou melhor.

Por outro lado, se o índice de usuário não existe, então a lista de navegação nos mostrará este outro, mesmo que o fonte seja exatamente o mesmo.

Então? Vamos repetir mais uma vez: a lista de navegação nos oferece o pior cenário. Se o índice existe, sabemos que o pior cenário será bastante bom. Isto, se o índice foi criado antes por algum outro motivo e já que estamos o aproveitando. Criar o índice apenas para garantir que esta navegação seja otimizada não parece uma boa ideia se estivermos usando um DBMS inteligente. E se o DBMS não for inteligente mas a tabela tiver poucos registros, também não é. Em resumo, a sugestão é criar índices somente após verificar problemas de desempenho e avaliar prós e contras.


```

for each Attraction
  order cityName
  unique CountryId, CityId
  print relevantInfo //CityName
endfor

```

```

"SELECT DISTINCT T1.[CityId], T1.[CountryId],
T2.[CityName] FROM ([Attraction] T1 INNER JOIN
[CountryCity] T2 ON T2.[CountryId] = T1.[CountryId] AND
T2.[CityId] = T1.[CityId])
ORDER BY T2.[CityName] "

```

```

for each Attraction
  //order cityName
  unique CountryId, CityId
  print relevantInfo //CityName
endfor

```

```

"SELECT DISTINCT T2.[CityName], T1.[CityId],
T1.[CountryId] FROM ([Attraction] T1 INNER JOIN
[CountryCity] T2 ON T2.[CountryId] = T1.[CountryId] AND
T2.[CityId] = T1.[CityId])
ORDER BY T1.[CountryId], T1.[CityId] "

```

For Each Attraction (Line: 18)

Order: [CityName](#)
 No index!

Unique: [CountryId](#), [CityId](#)

Navigation Start from: FirstRecord

filters: Loop while: NotEndOfTable

Join location: Server

=Attraction ([AttractionId](#)) INTO [CityId](#) [CountryId](#)

=CountryCity ([CountryId](#), [CityId](#)) INTO [CityName](#)

For Each Attraction (Line: 18)

Order: [CountryId](#), [CityId](#)
 Index: IATTRACTION1

Unique: [CountryId](#), [CityId](#)

Navigation Start from: FirstRecord

filters: Loop while: NotEndOfTable

Join location: Server

=Attraction ([AttractionId](#)) INTO [CityId](#) [CountryId](#)

=CountryCity ([CountryId](#), [CityId](#)) INTO [CityName](#)

Outro exemplo que mostra como GeneXus tenta melhorar as coisas:

Se quisermos obter todos os nomes de cidade para as quais existem atrações turísticas, utilizamos a cláusula unique por CountryId, CityId, para que de todas as atrações que compartilham país e cidade, fique apenas com uma, para poder listar seu nome de cidade na saída. Se quisermos que essa saída seja exibida ordenada pelo nome de cidade, colocamos a cláusula order e vemos que na lista de navegação o especificador escreve exatamente essa order, para a qual não conhece nenhum índice. Ficará a cargo do DBMS a otimização desta consulta.

Em vez disso, se não nos importa a ordem em que são mostradas na saída essas cidades, então vamos ver que, ao não escrevê-la, o especificador está escolhendo ordenar pelos atributos que estamos pedindo que sejam únicos, pois tem um índice por eles.

```
For each BaseTrn1, ... , BaseTrnn  
    skip expression1 count expression2  
    order att1, att2, ... , attn [when condition]  
    order att1, att2, ... , attn [when condition]  
    order none [when condition]  
    unique att1, att2, ... , attn  
    using DataSelector ( parm1, parm2, ... , parmn )  
    where condition [when condition]  
    where condition [when condition]  
    where att IN DataSelector ( parm1, parm2, ... , parmn )  
    blocking n  
        main_code  
    when duplicate  
        when_duplicate_code  
    when none  
        when_none_code  
endfor
```

Agora vamos revisar as cláusulas order condicionais.

Event 'Print Attractions'
ListAttractions(&CategoryId, &CountryId, &AttractionName)
Endevent

Source * | Layout | Rules | Conditions | Variables | Help | Documentation

Subroutines

```

1 printAttractionTitles
2
3 for each Attraction
4   order AttractionName
5   where CategoryId = &CategoryId when not &CategoryId.IsEmpty()
6   where CountryId = &CountryId when not &CountryId.IsEmpty()
7   where AttractionName >= &AttractionName when not &AttractionName.IsEmpty()
8     print attractionInfo
9   endfor

```

- Attraction
 - AttractionId
 - AttractionName
 - AttractionDescription
 - CountryId
 - CountryName
 - CategoryId
 - CategoryName
 - AttractionPhoto
 - CityId
 - CityName
 - AttractionAddress

Queremos a partir deste Web Panel listar as atrações turísticas, mas permitindo ao usuário filtrar aquelas que são de uma categoria determinada, como Tourist Site, de um país determinado, e cujo nome seja posterior a um valor dado. Assim, ao pressionar o botão chamamos este procedimento, passando as três variáveis para ele.

Se o usuário não inserir valor em uma das variáveis de filtro, não queremos que esse filtro seja aplicado, e por isso condicionamos as três cláusulas Where.

Se quiséssemos que independentemente dos filtros aplicados, as atrações fossem exibidas ordenadas pelo nome de atração, então escreveríamos uma única cláusula order incondicional.

Procedure ListAttractions Navigation Report

Name: ListAttractions
Description: List Attractions
Output Devices: File

Environment: Default (.NET)
Spec. Version: 17_0_10-159906
Form Class: Graphic
Program Name: ListAttractions
Parameters: in: &categoryId, in: &countryId, in: &AttractionName

Warnings

▲ spc0038 There is no index for order **AttractionName**; poor performance may be noticed in group starting at line 3.

LEVELS

For Each Attraction (Line: 4)

Order: AttractionName
 No index!

Navigation filters: Start from: FirstRecord
 Loop while: NotEndOfTable

Constraints: CategoryId = &categoryId WHEN not &categoryId.isempty()
CountryId = &countryId WHEN not &countryId.isempty()
AttractionName >= &AttractionName WHEN not &AttractionName.isempty()

Join location: Server

Attraction (AttractionId) INTO AttractionName CountryId CategoryId
 Country (CountryId) INTO CountryName
 Category (CategoryId) INTO CategoryName

| Category | Country | Attraction |
|--------------|---------------|-----------------------|
| Monument | Brazil | Christ the Redemmer |
| Tourist site | Italy | Cinque Terre |
| Monument | France | Eiffel Tower |
| Tourist site | China | Forbidden city |
| Tourist site | Scotland | Glenfinnan Viaduct |
| Tourist site | United States | London Bridges |
| Monument | England | London Towers |
| Museum | France | Louvre Museum |
| Museum | France | Matisse Museum |
| Tourist site | China | Meet the Emperor |
| Tourist site | Italy | Rifugio Nuvolau |
| Museum | United States | Smithsonian Institute |
| Tourist site | China | The Great Wall |

Vamos executar o web Panel, que definimos como main para facilitar a execução.

Se observamos a lista de navegação, vemos que os filtros aparecem na seção de Constraints. Isto não se deve apenas à inexistência de um índice que permita a otimização, mas porque contém as condições When. Todo Where condicional aparecerá na seção de Constraints, mas isso não significa que a consulta não será otimizada. Voltaremos a isso.

Aqui vemos a lista de todas as atrações, pois nenhum dos 3 filtros terá sido aplicado. Observemos que são listadas ordenadas pelo nome de atração, conforme solicitamos.

Se agora pedimos para listar as atrações da categoria 3, que é Tourist Site, também é exibido o resultado ordenado por AttractionName e não ordenado por país.

The screenshot displays the GeneXus IDE interface. On the left, a window titled 'ListAttractions * X' shows the source code of a subroutine. The code is as follows:

```

1 print AttractionTitles
2
3 for each Attraction
4   order CategoryName when not &categoryId.IsEmpty()
5   order CountryName when not &countryId.IsEmpty()
6   order AttractionName
7   where CategoryId = &categoryId when not &categoryId.IsEmpty()
8   where CountryId = &countryId when not &countryId.IsEmpty()
9   where AttractionName >= &AttractionName when not &AttractionName.IsEmpty()
10  print attractionInfo
11 -endfor
12

```

On the right, a 'stAttractions Navigation Report' window provides details about the subroutine:

- Environment:** Default (.NET)
- Spec. Version:** 17_0_10-159906
- Form Class:** Graphic
- Program Name:** ListAttractions
- Parameters:** in: &categoryId, in: &countryId, in: &AttractionName

Below the report, a detailed view for 'For Each Attraction (Line: 4)' is shown:

Order: CategoryName WHEN &categoryId.isEmpty()
 No index!
 CountryName WHEN &countryId.isEmpty()
 No index!
 AttractionName OTHERWISE
 No index!

Navigation filters: Start from: FirstRecord
 Loop while: NotEndOfTable

Constraints: CategoryId = &categoryId WHEN not &categoryId.isEmpty()
 CountryId = &countryId WHEN not &countryId.isEmpty()
 AttractionName >= &AttractionName WHEN not &AttractionName.isEmpty()

Join location: Server

The join locations are defined as:

- Attraction (AttractionId) INTO AttractionName CountryId CategoryId
- Country (CountryId) INTO CountryName
- Category (CategoryId) INTO CategoryName

Mas suponhamos que no caso de não filtrar por categoria, ou seja, que esta variável está vazia, queremos classificadas justamente por nome de categoria e, em vez disso, se foi selecionado um valor para &categoryId (por exemplo, o de Tourist Site) queremos que saia ordenado por nome de país (se não foi selecionado país, ou seja, foi deixada em branco esta variável) e somente caso contrário (ou seja, caso seja filtrado por categoria e país), queremos ordenar por nome de atração.

Vemos que na lista de navegação aparecem as cláusulas order condicionais, onde a última é incondicional. Ao contrário das cláusulas where que fazem um AND entre si, somente uma das cláusulas order será aplicada. Para isso, a primeira condição que for True fará com que sua cláusula order seja a escolhida. Só será ordenado pela incondicional se nenhuma das condições das cláusulas order anteriores for satisfeita. Claro que poderíamos não colocar cláusula order incondicional, e aí a ordem ficará indefinida se não for satisfeita nenhuma das condições.

| Category | Country | Attraction | Category | Country | Attraction |
|--------------|---------------|-----------------------|--------------|---------------|--------------------|
| Monument | France | Eiffel Tower | Tourist site | China | Meet the Emperor |
| Monument | Brazil | Christ the Redemmer | Tourist site | China | The Great Wall |
| Monument | England | London Towers | Tourist site | China | Forbidden city |
| Museum | France | Louvre Museum | Tourist site | Italy | Rifugio Nuvolau |
| Museum | United States | Smithsonian Institute | Tourist site | Italy | Cinque Terre |
| Museum | France | Matisse Museum | Tourist site | Scotland | Glenfinnan Viaduct |
| Tourist site | China | Forbidden city | Tourist site | United States | London Bridges |
| Tourist site | Scotland | Glenfinnan Viaduct | | | |
| Tourist site | China | Meet the Emperor | | | |
| Tourist site | Italy | Rifugio Nuvolau | | | |
| Tourist site | China | The Great Wall | | | |
| Tourist site | Italy | Cinque Terre | | | |
| Tourist site | United States | London Bridges | | | |

Vemos rapidamente que, se não selecionamos categoria, é exibida ordenada por ela.

Por outro lado, se selecionarmos categoria e deixarmos o país sem selecionar, então sairá ordenada por país e sem ordenar pelo restante.

E se agora selecionarmos categoria e país, sai ordenado, agora sim, por nome de atração.

```
listattractions.cs - Visual Studio Code
listattractions.cs x .redAtt[ Untitled-1
C: > Models > GX175tableForu9 > TravelAgency_forExpert > NetModel > Web > listattractions.cs
350     short A5CategoryId ,
351     short A3CountryId ,
352     string A20AttractionName )
353 {
354     System.Text.StringBuilder sWhereString = new System.Text.StringBuilder();
355     string scmdbuf;
356     short[] GXV_int1 = new short[3];
357     Object[] GXV_Object2 = new Object[2];
358     scmdbuf = "SELECT T1.[AttractionName], T1.[CountryId], T1.[CategoryId], T2.[CountryName], T3.[CategoryId], T1.[AttractionId] FROM (([Attraction] T1 INNER JOIN
359     if ( ! (0==AV16categoryId) )
360     {
361         AddWhere(sWhereString, "(T1.[CategoryId] = @AV16categoryId)");
362     }
363     else
364     {
365         GXV_int1[0] = 1;
366     }
367     if ( ! (0==AV14countryId) )
368     {
369         AddWhere(sWhereString, "(T1.[CountryId] = @AV14countryId)");
370     }
371     else
372     {
373         GXV_int1[1] = 1;
374     }
375     if ( ! String.IsNullOrEmpty(StringUtil.RTrim( AV8AttractionName)) )
376     {
377         AddWhere(sWhereString, "(T1.[AttractionName] >= @AV8AttractionName)");
378     }
379     else
380     {
381         GXV_int1[2] = 1;
382     }
383     scmdbuf += sWhereString;
384     if ( (0==AV16categoryId) )
385     {
386         scmdbuf += " ORDER BY T3.[CategoryName]";
387     }
388     else if ( (0==AV14countryId) )
389     {
```

Se formos investigar o fonte, para ver como constrói a instrução SQL que envia ao gerenciador... vemos que primeiro constrói a primeira parte fixa do select (a dos atributos a serem selecionados e de quais tabelas com os joins para acessar a estendida... mas então complementa dinamicamente a partir da avaliação das variáveis a parte do Where (adicionando Wheres quando as variáveis não estão vazias).

```
File Edit Selection View Go Run Terminal Help listattractions.cs - Visual Studio Code
listattractions.cs x .redAtt( Untitled-1
C:\> Models > GX17StableForu9 > TravelAgency_forExpert > NetModel > Web > listattractions.cs
364
365 {
366     GXv_int1[0] = 1;
367 }
368 if ( ! (0==AV14countryId) )
369 {
370     AddWhere(swhereString, "(T1.[CountryId] = @AV14countryId)");
371 }
372 else
373 {
374     GXv_int1[1] = 1;
375 }
376 if ( ! String.IsNullOrEmpty(StringUtil.RTrim( AV8AttractionName) ) )
377 {
378     AddWhere(swhereString, "(T1.[AttractionName] >= @AV8AttractionName)");
379 }
380 else
381 {
382     GXv_int1[2] = 1;
383 }
384 scmdbuf += swhereString;
385 if ( (0==AV16categoryId) )
386 {
387     scmdbuf += " ORDER BY T3.[categoryName]";
388 }
389 else if ( (0==AV14countryId) )
390 {
391     scmdbuf += " ORDER BY T2.[countryName]";
392 }
393 else if ( true )
394 {
395     scmdbuf += " ORDER BY T1.[AttractionName]";
396 }
397 GXv_Object2[0] = scmdbuf;
398 GXv_Object2[1] = GXv_int1;
399 return GXv_Object2 ;
400 }
401 public override Object [] getDynamicStatement( int cursor ,
402     IGxContext context ,
```

E para o ORDER BY da instrução SQL, faz algo semelhante, só que com if aninhados, para refletir justamente o que dissemos antes, que apenas uma cláusula order será adicionada ao Select.

Estas avaliações para obter a instrução SQL final que é enviada ao DBMS para resolver a consulta são realizadas dinamicamente, em tempo de execução. Cada vez que é executada esta lista, deverá executar esta seção de código para compor a consulta final.

For each

For each **Attraction**

```

order CategoryName when &categoryId.IsEmpty()
order CountryName when &countryName.IsEmpty()
order AttractionName

where CategoryId = &categoryId when not &categoryId.IsEmpty()
where CountryId = &countryId when not &countryId.IsEmpty()
where AttractionName >= &attractionName when not &attractionName.IsEmpty()

```

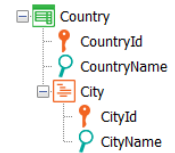
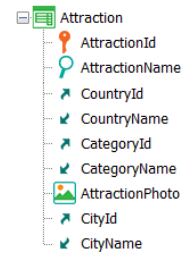
```

print info // CategoryName, CountryName, AttractionName

```

endfor

endfor



Portanto, se este for each fosse incluído em outra estrutura repetitiva que é executada para milhões de registros, o custo da montagem dinâmica da consulta poderia ser significativo.

For each **Attraction**

order CategoryName, CountryName, AttractionName

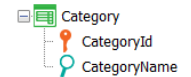
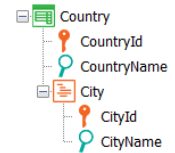
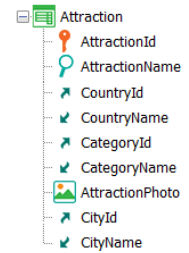
where CategoryId = &categoryId **when** not &categoryId.IsEmpty()

where CountryId = &countryId **when** not &countryId.IsEmpty()

where AttractionName >= &attractionName **when** not &attractionName.IsEmpty()

print info // CategoryName, CountryName, AttractionName

endfor



No exemplo que vimos, utilizamos ordens condicionais porque nos interessava exibir a informação ordenada de maneira diferente com base nas condições. Ou seja, as cláusulas order atenderam a um requisito lógico da consulta. Eram parte do tópico do problema, digamos. Embora não fossem necessárias neste caso. Pensemos que bastava escolher esta ordem incondicional para satisfazer o requisito.

For each **Attraction**

```

order CategoryId when not &categoryId.IsEmpty()
order CountryId when not &countryId.IsEmpty()
order AttractionName when not &attractionName.IsEmpty()

where CategoryId = &categoryId when not &categoryId.IsEmpty()
where CountryId = &countryId when not &countryId.IsEmpty()
where AttractionName >= &attractionName when not &attractionName.IsEmpty()

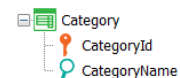
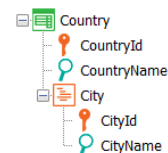
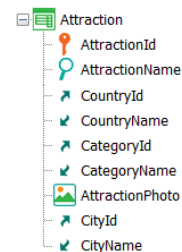
```

```

print info // CategoryName, CountryName, AttractionName

```

endfor



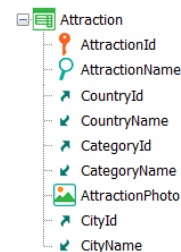
Mas muitas vezes, assim como vimos para o caso de uma única cláusula order incondicional, isso é especificado com o objetivo de otimização e não é um requisito. Nesses casos, escolher ordens compatíveis com os filtros geralmente é uma boa prática, especialmente no caso de DBMSs pouco inteligentes.

Então, por exemplo, se não importasse em qual ordem seria listada a informação, poderíamos colocar essas outras cláusulas order. Isto será traduzido dinamicamente da seguinte maneira: se &categoryId não estiver vazia, então sabemos que a consulta será semelhante a...

```

For each Attraction
  order CategoryId
  where CategoryId = &categoryId
  where CountryId = &countryId when not &countryId.IsEmpty()
  where AttractionName >= &attractionName when not &attractionName.IsEmpty()
  print info // CategoryName, CountryName, AttractionName
endfor

```



```

For each Attraction
  order CategoryId
  where CategoryId = &categoryId
  where CountryId = &countryId
  where AttractionName >= &attractionName
  print info // CategoryName, CountryName, AttractionName
endfor

```

```

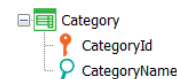
For each Attraction
  order CategoryId
  where CategoryId = &categoryId
  where AttractionName >= &attractionName
  print info // CategoryName, CountryName, AttractionName
endfor

```

```

For each Attraction
  order CategoryId
  where CategoryId = &categoryId
  where CountryId = &countryId
  print info // CategoryName, CountryName, AttractionName
endfor

```



...esta, onde dependendo se &countryId estiver vazia ou não, e se &attractionName estiver vazia ou não, teremos a consulta final assim, assim ou assim.

Observemos que, em qualquer caso, como existe índice por CategoryId, pelo menos a primeira cláusula Where estará otimizada.

For each **Attraction**

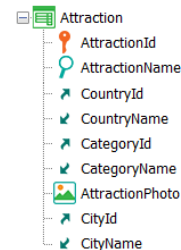
```

order CategoryId when not &categoryId.IsEmpty()
order CountryId when not &countryId.IsEmpty()
order AttractionName when not &attractionName.IsEmpty()
where CategoryId = &categoryId when not &categoryId.IsEmpty()
where CountryId = &countryId when not &countryId.IsEmpty()
where AttractionName >= &attractionName when not &attractionName.IsEmpty()

print info // CategoryName, CountryName, AttractionName

```

endfor



For each **Attraction**

```

order CountryId
where CountryId = &countryId
where AttractionName >= &attractionName
print info // CategoryName, CountryName, AttractionName
endfor

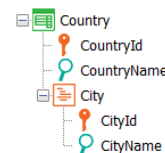
```

For each **Attraction**

```

order CountryId
where CountryId = &countryId
print info // CategoryName, CountryName, AttractionName
endfor

```



Se, em vez disso, &categoryId estiver vazia, então se &countryId não estiver, a consulta ficará assim ou assim, dependendo se &attractionName não estiver vazia ou estiver.

Em qualquer dos dois casos, estará otimizada em relação ao filtro por CountryId, pois possui um índice, por ser chave estrangeira.

For each **Attraction**

```

order CategoryId when not &categoryId.IsEmpty()
order CountryId when not &countryId.IsEmpty()
order AttractionName when not &attractionName.IsEmpty()
where CategoryId = &categoryId when not &categoryId.IsEmpty()
where CountryId = &countryId when not &countryId.IsEmpty()
where AttractionName >= &attractionName when not &attractionName.IsEmpty()

print info // CategoryName, CountryName, AttractionName

```

endfor

For each **Attraction**

```

order AttractionName
where AttractionName >= &attractionName
print info // CategoryName, CountryName, AttractionName
endfor

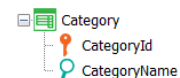
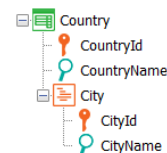
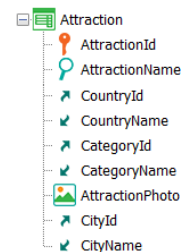
```

For each **Attraction**

```

print info // CategoryName, CountryName, AttractionName
Endfor

```



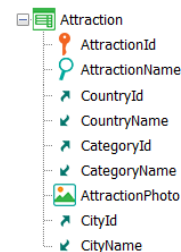
Se, por outro lado, &countryId também estiver vazia, então se &attractionName não estiver, a consulta ficará assim. E se estiver, ficará desta outra forma, mas a order será indefinida. Isto significa que pode variar de DBMS para DBMS e até mesmo entre execuções sucessivas.

No primeiro caso, como não sabemos da existência de um índice por AttractionName, não sabemos quão otimizada estará a consulta.

```

for each Attraction
  order CategoryId when not &categoryId.IsEmpty()
  order CountryId when not &CountryId.IsEmpty()
  order AttractionName when not &AttractionName.IsEmpty()
  where CategoryId = &categoryId when not &categoryId.IsEmpty()
  where CountryId = &countryId when not &countryId.IsEmpty()
  where AttractionName >= &AttractionName when not &AttractionName.IsEmpty()
    print attractionInfo
endfor

```



For Each Attraction (Line: 4)

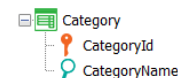
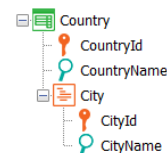
Order: [CategoryId](#) WHEN not &categoryId. isempty()
Index: IATTRACTION2
[CountryId](#) WHEN not &countryId. isempty()
Index: IATTRACTION1
[AttractionName](#) WHEN not &AttractionName. isempty()
No index!

Navigation filters: Start from: FirstRecord
Loop while: NotEndOfTable

Constraints: [CategoryId](#) = &categoryId WHEN not &categoryId. isempty()
[CountryId](#) = &countryId WHEN not &countryId. isempty()
[AttractionName](#) >= &AttractionName WHEN not &AttractionName. isempty()

Join location: Server

=Attraction ([AttractionId](#)) INTO [AttractionName](#) [CountryId](#) [CategoryId](#)
=Country ([CountryId](#)) INTO [CountryName](#)
=Category ([CategoryId](#)) INTO [CategoryName](#)



Se observamos a lista de navegação veremos que os filtros continuam sendo mostrados na seção de Constraints, embora saibamos que dependendo dos valores das variáveis alguns deveriam ser mostrados nos Navigation filters. É que a lista de navegação não realiza a decomposição que fizemos antes. Devemos entender, então, que o cenário será melhor do que este que pode parecer à primeira vista se não é levado em conta todo o anterior.

```

For each  BaseTrn1, ... , BaseTrnn
    skip expression1 count expression2
    order att1, att2, ... , attn [when condition]
    order att1, att2, ... , attn [when condition]
    order none [when condition]
    unique att1, att2, ... , attn
    using DataSelector ( parm1, parm2, ... , parmn )
    where condition [when condition]
    where condition [when condition]
    where att IN DataSelector ( parm1, parm2, ... , parmn )
    blocking n
        main_code
    when duplicate
        when_duplicate_code
    when none
        when_none_code
endfor

```

O que mais dizer sobre as ordens condicionais?

Não são suportadas em cortes de controle.

Não se aplicam a geradores legados Cobol e RPG.

Se as condições possuem atributos, são considerados como instanciados, ou seja, são avaliados antes de iniciar a navegação e não mudam durante ela.

Com isto, terminamos de explorar o tema das orders das consultas.



For each



BaseTrn

skip *exp count exp*
order *att...*
unique *att...*
using *DataSelector(parm...)*
where *condition when condition*
blocking *n*

Navigation
groups

DP Group



BaseTrn

skip *exp count exp*
order *att...*
unique *att...*
using *DataSelector(parm...)*
where *condition when condition*

Grids



Base Trn property
Order property
Conditions property
Unique property
Data Selector property

Claro, isto que vimos para o for each é válido para grupos de Data Providers e grids com tabela base, bem como para consultas com In em Data Selectors.