

# For each en profundidad

Cláusulas order y performance

*GeneXus™*

Vamos a concentrarnos en las cláusulas order y su relación con la optimización de la 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 la información a ser consultada y devuelta, la sintaxis del for each nos permite especificar una lista de cláusulas order condicionales y una 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)

```

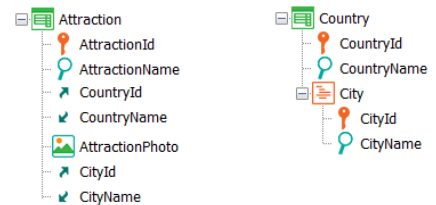
blocking n

```

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

```

endfor



Vayamos de lo más simple a lo más complejo: empecemos por analizar el caso de una única cláusula order sin condiciones, por ejemplo esta. Estamos pidiendo que se recorra la tabla de atracciones turísticas, filtrando las que correspondan a un país y ciudad determinados, y que las presente ordenadas por AttractionName, atributo secundario.

A la hora de especificar lo que queremos no debería importarnos si para obtener los datos requeridos primero los obtiene y luego los ordena, o si lo hace al revés o de alguna otra manera. Los desarrolladores especificamos lo que queremos y de resolverlo se encarga el especificador de GeneXus y, sobre todo, el DBMS.

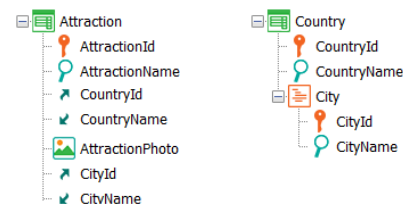
```

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](#)

Attraction Indexes	
IAttraction	Primary Key
● AttractionId	Ascending
IAttraction1	Foreign Key
● CountryId	Ascending
● CityId	Ascending

Cuando pedimos a GeneXus que especifique y genere el programa asociado al objeto que contiene el for each, lo hacemos para un environment determinado, es decir, en un lenguaje de programación determinado, como Net por ejemplo, y para una base de datos manejada por un DBMS determinado, como SQL Server, por ejemplo. O podría ser para un environment Java contra Oracle, o tantas otras alternativas.

Cuando el desarrollador escribe su For each lo hace en GeneXus, con cierta independencia de cuál será el environment final, justamente para que con el mismo código podamos obtener la aplicación en diferentes environments.

Esto significa que de la implementación concreta se encarga GeneXus que conoce las particularidades de cada environment. Sin embargo su conocimiento tiene un límite: conoce la estructura de la base de datos, pero no los datos, ni su distribución, cantidad, etcétera. Esta información la tiene el DBMS, que registra estadísticas, cachea datos y consultas en la historia de las consultas que se vienen realizando, arma planes de ejecución, mantiene índices, etcétera. Cuanto más evolucionado e inteligente el DBMS menos necesitará que GeneXus le indique con precisión cómo realizar la consulta, porque GeneXus nunca sabrá más que él.

Así, por ejemplo, si pedimos que se especifique el objeto que contiene este For each veremos este listado de navegación, que nos advierte que no existe un índice por el atributo por el que queremos mostrar ordenada

la consulta y que por lo tanto podríamos notar una baja performance. "Podríamos" no significa que efectivamente será así. ¿Por qué? Porque eso depende no solo de la cantidad de datos de la tabla, sino también del DBMS y sus estrategias. Lo que nos indica el listado de navegación es el peor escenario: en este deberá recorrerse toda la tabla ordenada por un atributo para el que posiblemente no exista índice (GeneXus no sabe si el DBMS lo creó por las suyas o no, lo cierto es que no tiene noticias de él) por lo que en el peor escenario que es el de arquitecturas centralizadas podría tener que crear temporalmente el índice para resolver la consulta por ese orden y así recorrer toda la tabla para evaluar registro a registro si cada uno satisface o no los filtros. Este sería el escenario de una base de datos con un manejador poco inteligente.

Pensemos, por ejemplo, que en este caso tal vez sea una mejor estrategia utilizar el índice por clave foránea {CountryId, CityId} que sabemos que existe porque GeneXus obliga a crearlo. Entonces se obtienen de forma óptima los registros que cumplen los filtros y recién allí, con ese resultado, se lo ordena por AttractionName.

Cuál sea la mejor estrategia dependerá en buena medida de la distribución de los datos. Si solo hay 3 atracciones de ese país y ciudad entre millones, esta parece una mejor estrategia porque el costo de ordenar 3 registros es insignificante. Sin embargo si hay millones de registros en la tabla siendo la inmensa mayoría de ese país y ciudad, entonces usar el índice por país y ciudad no reducirá demasiado la consulta de toda la tabla, y ordenar luego el resultado por AttractionName será casi lo mismo que de entrada ordenar por AttractionName y después ir evaluando uno por uno si corresponde además al país/ciudad o no. GeneXus desconoce la distribución de datos como para tomar decisiones de este tipo. Además si esta misma consulta ya se realizó con anterioridad, el DBMS posiblemente guarde en caché el resultado y no tenga que volver a realizar la consulta de la misma manera. Esto, por supuesto, si los filtros no cambian y los datos tampoco.

En definitiva, entonces, el listado de navegación nos informa sobre la base del escenario más conservador, con el peor DBMS. Pero puede que el DBMS mejore muchísimo el escenario más pesimista y la consulta resulte optimizada.

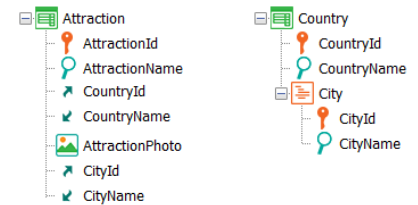
```

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:	<a href="#">AttractionName</a>
Index:	UATTRACTION
Navigation filters:	Start from: <a href="#">FirstRecord</a>
	Loop while: <a href="#">NotEndOfTable</a>
Constraints:	<a href="#">CountryId</a> = &countryId
	<a href="#">CityId</a> = &cityId
	=Attraction ( <a href="#">AttractionId</a> ) INTO <a href="#">CityId</a> <a href="#">CountryId</a> <a href="#">AttractionName</a>

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
UAttraction	Duplicate	User Index
● AttractionName	Ascending	Attraction Name

Podríamos vernos tentados a pensar que si sabemos que habrá millones de registros en la tabla Attraction lo mejor será ordenar desde GeneXus al DBMS la creación de índice de usuario por AttractionName. En ese caso se reorganizará la base de datos, y al especificarse nuevamente el objeto ahora GeneXus nos indicará que utilizará el nuevo índice para resolver la consulta y ya no se nos muestra la advertencia de performance.

Sin embargo esta puede ser una muy peor solución. Justamente, si el DBMS es inteligente no necesitará en absoluto que lo obliguemos a crear un índice. Él mismo lo hará si lo necesita. Tanto es así que ni siquiera en este caso en el que el propio GeneXus pidió la creación del índice se lo envía al DBMS cuando realiza la consulta desde environment con DBMS inteligente.

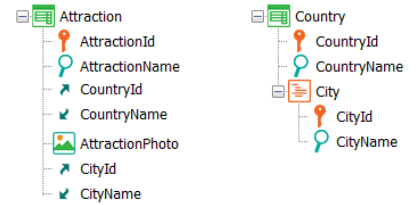
```
File Edit Selection View Go Run Terminal Help listcountries.cs - Visual Studio Code
listcountries.cs x .redAtt[ Untitled-1
C: > Models > GX175tableForu9 > TravelAgency_forExpert > NetModel > Web > listcountries.cs
294 public class listcountries_default : DataStoreHelperBase, IDataStoreHelper
295
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
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
283
```

```

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

Debemos tener en cuenta que la creación de un índice es algo costoso, y que tiene implicancias no solo al momento de crearlo, sino después a la hora de mantenerlo, durante todo el ciclo de vida de la tabla. Cada vez que se actualiza la tabla se paga un pequeño precio por mantenerlo.

Es por eso que no parece una buena práctica la creación de índices de usuario, a menos que se los necesite para controlar unicidad, es decir, para definir claves candidatas, como índices unique.



```

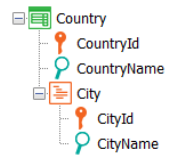
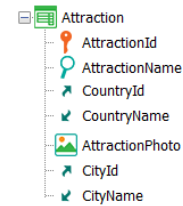
for each Attraction
order AttractionName
where CountryId = &countryId
where CityId = &cityId
print attractionInfo //AttractionName
endfor

```



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](#)

Entonces, decíamos, el especificador de GeneXus hace lo mejor que puede con la información que tiene y con su inteligencia actual (es esperable que esa inteligencia se vaya incrementando conforme GeneXus evoluciona), y envía la consulta lo más optimizada que puede al DBMS sin indicarle obviedades, sabiendo que éste en el peor de los casos la tomará pero en general la mejorará.

En definitiva nunca podemos asegurar que lo que indica el listado de navegación será lo que finalmente hará el DBMS cuando éste es inteligente. Lo que sí sabemos es que será eso o algo mejor.


Veamos ejemplos de la inteligencia actual del especificador de GeneXus, más allá de lo que después termine haciendo el DBMS.

Sabemos que el orden en el que se pedirá que los registros resultantes sean retornados se determina en base a la especificación del desarrollador en la cláusula order, pero también a algoritmos internos de optimización.

```
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 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>

Por ejemplo, si no nos importara cómo saldrán ordenados los nombres de atracción podríamos escribir el For each sin cláusula order. En general sabíamos que si hacíamos esto la consulta se ordenaba por clave primaria. Es decir que se enviaba al DBMS un Select con order AttractionId. Sin embargo en este caso sucederá algo distinto, como vemos en el listado de navegación. Sabemos que existe en la base de datos un índice por CountryId y CityId, los dos filtros por igualdad. Lo sabemos porque conforman una clave foránea. Claramente, entonces, será preferible que se utilice ese índice y se devuelva la consulta ordenada por esos valores. Es por eso que si observamos la consulta que envía el fuente al DBMS, encontramos este ORDER BY. Si nuestra intención al no especificar cláusula order era que salieran ordenados por clave primaria, en este caso tendremos que explicitarlo.

De igual manera, si ahora la consulta es esta otra, donde estamos pidiendo que las atracciones salgan ordenadas por identificador de ciudad, y solo estamos filtrando por identificador de país, veremos que el listado de navegación no está mostrando que pedirá ordenar por CityId, sino por el par, debido a que así optimiza la consulta, porque sabe de la existencia del índice compuesto (por ser clave foránea, justamente), sin dejar de cumplir con el resultado del ordenamiento pedido por el desarrollador.

En definitiva, con la cláusula order el desarrollador indica el orden en que desea que los registros sean devueltos, pero para ello el especificador podría alterar esa cláusula, suplementándola con información contextual (si existen condiciones implícitas o explícitas por igualdad y existe índice

que las contiene, además de los atributos del order) de manera tal de optimizar la consulta, aunque el DBMS será quien tenga al final del día la última palabra.

Lo importante es entender que los datos serán devueltos en el orden explicitado por el desarrollador, aunque para resolver la consulta se utilicen otros criterios.

```

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>

Si nuestra intención al no especificar cláusula order era que salieran ordenados por clave primaria, en este caso tendremos que explicitarlo.


```
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>

De igual manera, si ahora la consulta es esta otra, donde estamos pidiendo que las atracciones salgan ordenadas por identificador de ciudad, y solo estamos filtrando por identificador de país, veremos que el listado de navegación no está mostrando que pedirá ordenar por CityId, sino por el par, debido a que así optimiza la consulta, porque sabe de la existencia del índice compuesto (por ser clave foránea, justamente), sin dejar de cumplir con el resultado del ordenamiento pedido por el desarrollador.

En definitiva, con la cláusula order el desarrollador indica el orden en que desea que los registros sean devueltos, pero para ello el especificador podría alterar esa cláusula, suplementándola con información contextual (si existen condiciones implícitas o explícitas por igualdad y existe índice que las contiene, además de los atributos del order) de manera tal de optimizar la consulta, aunque el DBMS será quien tenga al final del día la última palabra.

Lo importante es entender que los datos serán devueltos en el orden explicitado por el desarrollador, aunque para resolver la consulta se utilicen otros criterios.

```
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>

En este caso donde no se especificó orden, como existe índice que permite optimizar esas condiciones, es el elegido, en lugar del índice por clave primaria.

Cuando no hay índice, entonces sí elige la clave primaria. Aquí vemos la sentencia SQL que construye 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 CountryId CountryId AttractionName

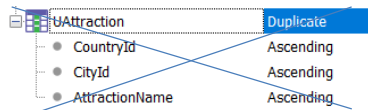
LEVELS	
For Each Attraction (Line: 24)	
Order:	NONE
Navigation filters:	Start from: FirstRecord Loop while: NotEndOfTable
Constraints:	CountryId = &countryId CityId = &cityId
	 =Attraction (AttractionId) INTO CountryId CityId AttractionName

A menos que especifiquemos cláusula Order none, y en ese caso le delegamos al DBMS la elección del orden. No se agrega ORDER BY a la sentencia SQL armada 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](#) ^

Volvamos a este ejemplo. Si GeneXus pidió crear este índice de usuario compuesto entonces lo propondrá en el listado de navegación (aunque finalmente no lo envíe al SQL Server, porque éste ya sabe de su existencia y para qué decirle algo que ya sabe). La gracia es que nos hace saber que al menos esta optimización será realizada por el DBMS. Será así o mejor.

En cambio si el índice de usuario no existe, entonces el listado de navegación nos mostrará esto otro, aunque el fuente sea exactamente el mismo.

¿Entonces? Repitémoslo una vez más: el listado de navegación nos propone el peor escenario. Si el índice existe, sabemos que el peor escenario será bastante bueno. Eso si el índice se creó antes por alguna otra razón y ya que estamos lo aprovechamos.

Crear el índice solo para asegurarnos de que esta navegación esté optimizada no parece una buena idea si estamos utilizando un DBMS inteligente. Y si el DBMS no fuera inteligente pero la tabla tiene pocos registros, tampoco. En definitiva la sugerencia es crear índices solo tras verificar problemas de performance y evaluar pros y 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](#)

Otro ejemplo que muestra cómo GeneXus intenta mejorar las cosas:

Si queremos obtener todos los nombres de ciudad para las que hay atracciones turísticas, utilizamos la cláusula unique por CountryId, CityId, para que de todas las atracciones que comparten país y ciudad solo se quede con una, para poder listar su nombre de ciudad en la salida. Si queremos que esa salida se muestre ordenada por nombre de ciudad, colocamos la cláusula order y vemos que en el listado de navegación el especificador escribe exactamente ese order, para el que no conoce ningún índice. Quedará en manos del DBMS la optimización de esta consulta.

En cambio, si no nos importa el orden en que se muestren en la salida esas ciudades, entonces veamos que al no escribirlo, el especificador está eligiendo ordenar por los atributos que estamos pidiendo que sean únicos, ya que tiene un índice por ellos.

```
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
```

Ahora repasemos las cláusulas order condicionales.

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

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

Subroutines

```

1 print AttractionTitles
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 desde este Web Panel listar las atracciones turísticas pero permitiéndole al usuario filtrar las que son de una categoría determinada, como Tourist Site, de un país determinado, y cuyo nombre sea posterior a un valor dado. Así que al presionar el botón llamamos a este procedimiento, pasándole las tres variables.

Si el usuario no ingresa valor en una de las variables de filtro, no queremos que ese filtro se aplique, y por eso condicionamos las tres cláusulas Where.

Si quisiéramos que independientemente de los filtros que se apliquen las atracciones se muestren ordenadas por nombre de atracción, entonces escribiríamos una 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** Start from: FirstRecord

**filters:** 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

Ejecutemos el web Panel, que definimos como main para facilitar la ejecución.

Si observamos el listado de navegación, vemos que los filtros aparecen en la sección de Constraints. Esto no es únicamente por la inexistencia de un índice que permita optimizar, sino porque contienen las condiciones When. Todo Where condicional se mostrará en la sección de Constraints, pero eso no significa que la consulta no vaya a optimizarse. Volveremos sobre esto.

Aquí vemos el listado de todas las atracciones, ya que ninguno de los 3 filtros se habrá aplicado. Observemos que se listan ordenadas por nombre de atracción, tal como pedimos.

Si ahora pedimos listar las atracciones de la categoría 3 que es Tourist Site, también se muestra el resultado ordenado por AttractionName y desordenado 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 'ListAttractions 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 location section includes the following table definitions:

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

Pero supongamos que en caso de no filtrar por categoría, es decir, que esta variable esté vacía, los queremos ordenados justamente por nombre de categoría, y en cambio si sí se seleccionó un valor para &categoryId (por ejemplo el de Tourist Site) allí queremos que salga ordenado por nombre de país (si es que no se seleccionó país, es decir, se dejó vacía esta variable) y solo en caso contrario (es decir, en caso en que se filtre por categoría y país), queremos ordenar por nombre de atracción.

Vemos que en el listado de navegación nos aparecen las cláusulas order condicionales, donde la última es incondicional. A diferencia de las cláusulas where que hacen un AND entre ellas, solamente una de las cláusulas order se aplicará. Para ello la primera condición que sea True hará que su cláusula order sea la elegida. Solo se ordenará por la incondicional si ninguna de las condiciones de las cláusulas order anteriores se satisfizo. Por supuesto, podríamos no colocar cláusula order incondicional, y allí el orden quedará indefinido si no se satisface ninguna de las condiciones.

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			

Category	Country	Attraction
Tourist site	China	Forbidden city
Tourist site	China	Meet the Emperor
Tourist site	China	The Great Wall

Vemos rápidamente que si no seleccionamos categoría se muestra ordenado por ella.

En cambio si seleccionamos categoría y dejamos el país sin seleccionar, entonces saldrá ordenado por país y desordenado por lo demás.

Y si ahora seleccionamos categoría y país, sale ordenado, ahora sí, por nombre de atracción.

```
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.[CategoryName], 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 {
```

Si vamos a investigar el fuente, para ver cómo construye la sentencia SQL que envía al manejador... vemos que primero arma la primera parte fija del select (la de los atributos a ser seleccionados y de qué tablas con los joins para acceder a la extendida...pero luego complementa dinámicamente a partir de la evaluación de las variables la parte del Where (agregando Wheres cuando las variables no estén vacías).

```
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 else
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 ,
```

Y para el ORDER BY de la sentencia SQL hace algo similar, solo que con if anidados, para reflejar justamente lo que decíamos antes, que solo una cláusula order será agregada al Select.

Estas evaluaciones para obtener la sentencia SQL final que se envía al DBMS para que resuelva la consulta se realizan dinámicamente, en tiempo de ejecución. Cada vez que se ejecuta este listado deberá ejecutar esta sección de código para componer la consulta final.



## For each

For each **Attraction**

```

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

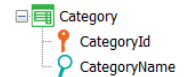
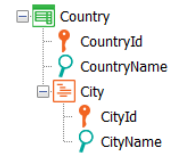
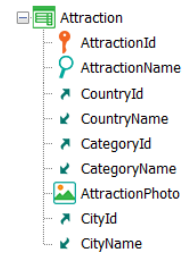
where CategoryId = &categoryId when not &categoryId.IsEmpty()
where CountryId = &countryId when not &countryId.IsEmpty()
where AttractionName >= &attractionName when not &attractionName.IsEmpty()

  print info // CategoryName, CountryName, AttractionName

```

endfor

endfor



Por lo que si este for each se incluyera dentro de otra estructura repetitiva que se ejecutara para millones de registros el costo del armado dinámico de la consulta podría ser importante.

For each **Attraction**

```
order CategoryName, CountryName, AttractionName
```

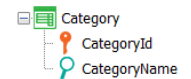
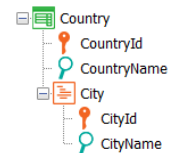
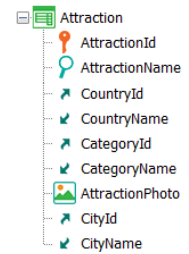
```
where CategoryId = &categoryId when not &categoryId.IsEmpty()
```

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

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

```
print info // CategoryName, CountryName, AttractionName
```

```
endfor
```



En el ejemplo que vimos utilizamos órdenes condicionales porque nos interesaba desplegar la información ordenada en forma distinta en base a condiciones. Es decir, las cláusulas order cumplieron un requerimiento lógico de la consulta. Eran parte de la letra del problema, digamos. Aunque no eran necesarias en este caso. Pensemos que bastaba con elegir este orden incondicional para satisfacer el requerimiento.

For each **Attraction**

```

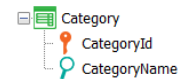
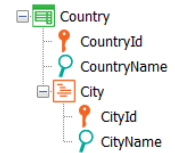
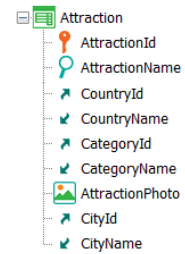
order CategoryId when not &categoryId.IsEmpty()
order CountryId when not &countryId.IsEmpty()
order AttractionName when not &attractionName.IsEmpty()

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

```



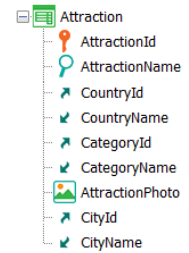
Pero muchas veces, así como vimos para el caso de una única cláusula order incondicional, ésta se especifica con objetivo de optimización y no es un requerimiento. En esos casos elegir órdenes compatibles con los filtros suele ser una buena práctica, sobre todo en el caso de DBMSs poco inteligentes.

Así, por ejemplo, si no nos importara en qué orden saldrá listada la información, podríamos colocar estas otras cláusulas order. Esto se traducirá dinámicamente del siguiente modo: si &categoryId no está vacía, entonces sabemos que la consulta se parecerá 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

```

```

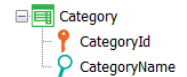
For each Attraction
  order CategoryId
  where CategoryId = &categoryId
  where AttractionName >= &attractionName
  print info // CategoryName, CountryName, AttractionName
endfor

```

```

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

```



...esta, donde dependiendo de si &countryId está o no vacía, y si &attractionName está o no vacía, tendremos la consulta final así, así o así.

Observemos que en cualquier caso como existe índice por CategoryId, al menos la primera cláusula Where estará optimizada.

For each **Attraction**

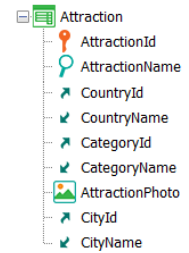
```

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

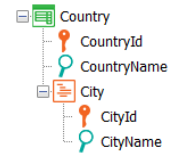
```

For each **Attraction**

```

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

```



Si en cambio &categoryId está vacía, entonces si &countryId no, la consulta quedará así o así, dependiendo de si &attractionName no está vacía o sí lo está.

En cualquiera de los dos casos, estará optimizada en relación al filtro por CountryId, dado que tiene un índice, por ser clave foránea.

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

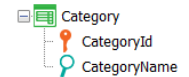
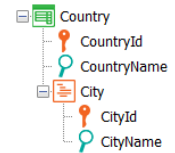
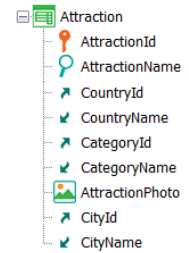
```

For each **Attraction**

```

print info // CategoryName, CountryName, AttractionName
Endfor

```



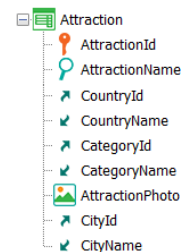
Si en cambio también &countryId está vacía, entonces si &attractionName no lo está la consulta quedará así. Y si sí lo está, quedará de este otro modo pero el order será indefinido. Eso significa que puede variar de DBMS en DBMS e incluso entre ejecuciones sucesivas.

En el primer caso como no sabemos de la existencia de un índice por AttractionName, no sabemos qué tan optimizada estará la 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)

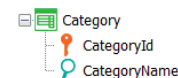
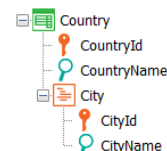
**Order:**            CategoryId WHEN not &categoryId. isempty()  
                           Index: IATTRACTION2  
                           CountryId WHEN not &countryId. isempty()  
                           Index: IATTRACTION1  
                           AttractionName WHEN not &AttractionName. 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



Si observamos el listado de navegación veremos que los filtros siguen mostrándose en la sección de Constraints, aunque sepamos que dependiendo de los valores de las variables algunos deberían mostrarse en los Navigation filters. Es que el listado de navegación no realiza la descomposición que hicimos antes. Debemos comprender, entonces, que el escenario será mejor que este que puede parecer a primera vista si no se tiene en cuenta todo lo 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

```

¿Qué más decir de los órdenes condicionales?

No se soportan en cortes de control.

No aplican a generadores legacy Cobol y RPG.

Si las condiciones tienen atributos, son considerados como instanciados, es decir son evaluados antes de comenzar la navegación y no cambian durante ella.

Con esto terminamos de explorar el tema de los orders de las 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
```

Por supuesto esto que vimos para el for each vale para grupos de Data Providers y grids con tabla base, así como para consultas con In en Data Selectors.