November 2026 : Amsterdam

SSJ: how to optimize network activity

PROGINOV

Ne pas divulguer ou diffuser. Propriété de Proginov - Reproduction interdite

/ Proginov

30years



/ Our business

/ A single point of contact for a responsible service

Publisher
since 1996

198
developers dedicated to Proginov's standard ERP (R&D) and to customers

Integrator
of Proginov ERP

55
Project managers

48
Engineers in charge of deployment and R&D

Hosting Provider
since 2001

750
hosted customers

+ 30
Cloud services

Assistance
on site

59
hotline operators (ERP, technical and telecom network support) 24/7

/ Who are we ?



Simon BERNARD
OpenEdge developer at Proginov since 2015

Maintain our desktop framework (C# WinForms + OpenEdge)

Build internal productivity tools,

Run OOP training sessions for the engineering team



Patrice PERROT
began his OpenEdge career as developer 25 years ago.

Monitors more than 16 000 DB in production, optimizes DB activity, analyzes source code and implements new tools for development and monitoring.

Playing on a "SPORTS" DB

/ Test done on this laptop

/ DB

/ Session

/ Like a sports2000 DB

/ OE 13.1 ESAP1

/ DB start with

proserve sports2013 -S 17503 -B 70000 -Mm 8192 -lruskips 10 -prefetchFactor 80 -prefetchNumRecs 1000

/ Tables

Table	Records	Size
Customer	201 120	31.1M
Order	727 285	63.9M
OrderLine	2 619 963	110.2M
Item	69	8.0K

Network performance 101

/ Message Buffer Size (-Mm)

/ Use Message Buffer Size (-Mm) to specify the standard message buffer size, in bytes.

/ Test several values

```
FOR EACH order NO-LOCK :
END.
```

-Mm	Elapsed Time (Ms)	Mean size message	Rec / Message
1 024	3 360	927	8,0
2 048	2 047	1 786	16,4
4 096	1 625	3 477	32,8
8 192	1 479	6 907	66,0
16 384	1 390	13 702	131,9
32 500	1 355	27 042	261,3

/ Prefetch Num Recs (-prefetchNumRecs)

/ the number of records placed in a network message before it is sent to a remote client.

/ Prefetch factor (-prefetchFactor)

/ The percent of a message buffer to fill before sending it to a waiting remote client.

/ Test of several values

```
FOR EACH order NO-LOCK :
END.
```

-prefetchNumRecs	Elpase time (ms)
10	2 095
20	1 741
40	1 569
80	1 411
160	1 361
320	1 314

/ Lock option

```
FOR EACH order NO-LOCK :
END.
```

```
FOR EACH order Exclusive-LOCK :
END.
```

	Elapse time (ms)	Message sent	Message rec.	Rec sent	Rec / mess
No-LOCK	4 996	23 104	23 105	727 285	31
EXCLUSIVE-LOCK	42 215	727 286	727 287	727 285	1

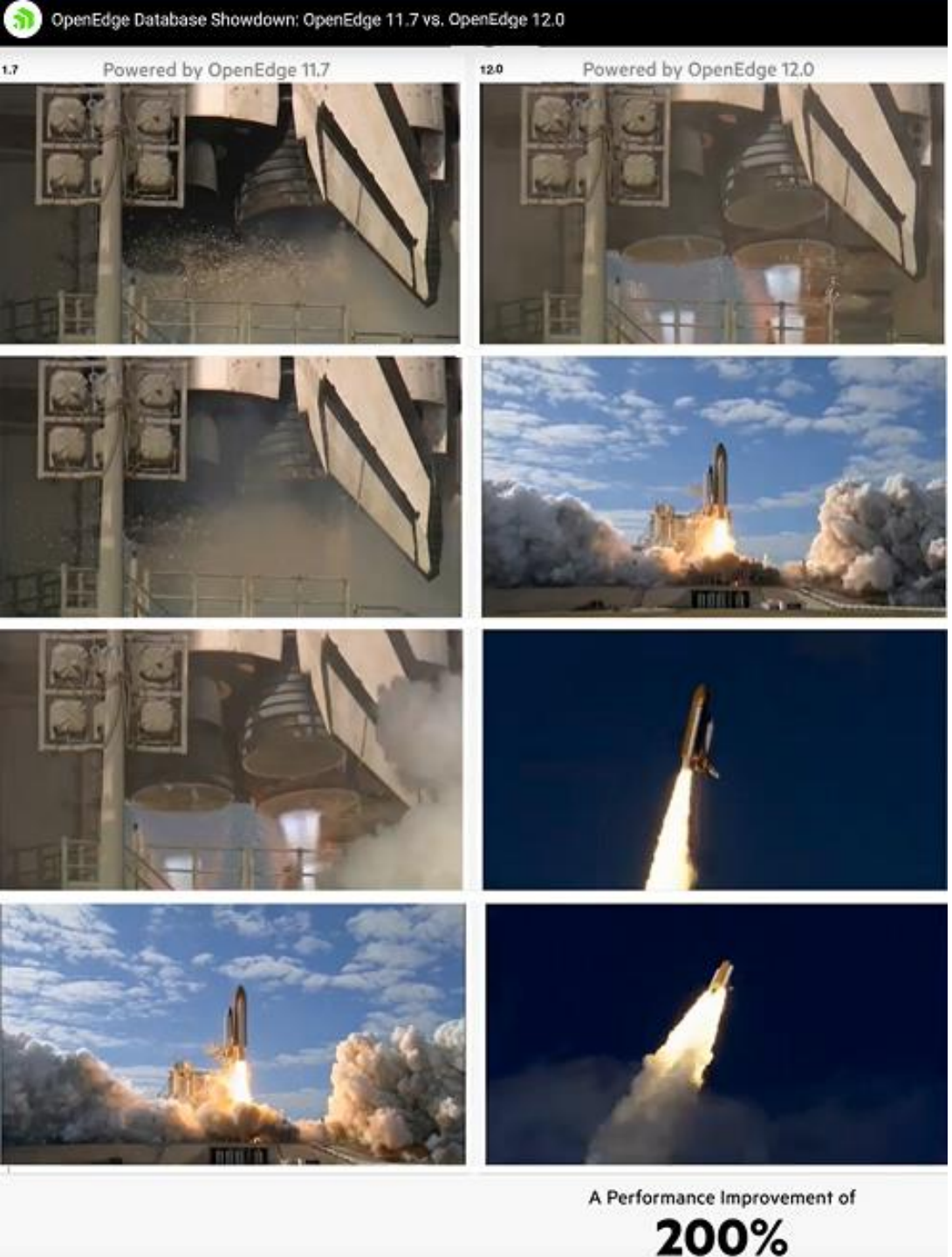
/ In case of Shared / Exclusive-lock , 1 record per message

- **-prefetchNumRecs 1**

What is this SSJ?







/ -SSJ : SERVER-SIDE-JOIN

/ Server-side join is a feature in which certain queries involving joins, execute on the database server, rather than the client.

/ Server-side join improves performance by reducing the amount of data transferred across the network and by reducing the computational cost on the client.

/ Prior to OpenEdge 12.0, when the AVM executed a join of multiple tables, some records which did not satisfy all levels of the join may have been returned to the AVM client, which needed to perform additional processing to eliminate these extra records from consideration.

/ HOW TO SET UP –SSJ

/ -threadedServer 1

/ -ssj 1

/ SSJ ENABLED WITH

/ REMC

/ NO-LOCK

/ Max 10 tables

/ In the same logical database

/ The ABL query construct must be one of the following:

- **FOR statement.**
- **Static or dynamic query that does not use the INDEXED-REPOSITION option or involve an outer join.**

/ SSJ : Requirements

- / All tables to be joined must be in the same logical database.

/ Restrictions

- / User-defined functions, methods, or properties with a GET implementation.
- / CAN-DO function.
- / Expressions with MEMPTR and LONGCHAR data types.
- / TENANT-WHERE clause.
- / The use of RECID and ROWID to join records disables server-side join.
- / “BY” if not based on an index of the most inner table of the join.
- / NO-LOCK must be used.

/ Reducing the amount of data transferred :

```
FOR EACH  customer  NO-LOCK WHERE customer.country = "CA" OR customer.country = "FR" ,
   EACH  order NO-LOCK WHERE  order.custnum = customer.custnum AND order.orderdate >  09/18/2011  ,
   EACH orderline OF Order NO-LOCK WHERE orderline.qty > 5 ,
   FIRST ITEM  NO-LOCK WHERE item.itemnum = orderline.itemnum AND item.price >= 50  :
```

/ EVICTED RECORDS : Records sent to the session

Table	Without SSJ	With SSJ	Evicted By SSJ
Customer	13 651	8 118	5 533
Order	28 370	11 098	17 272
Orderline	96 180	13 890	82 290
Item	13 890	13 890	0

/ Time decreases from 4 230 ms to 1 132 ms with SSJ

/ Save 73,2 % on execution time

/ Where is it the most effective ?

```
FOR each customer No-Lock where country = 'CA' or country = 'FR'  ,
    EACH order     OF customer No-LOCK  WHERE order.orderdate > var-date,
    EACH orderline OF  order No-LOCK    WHERE orderline.qty > var-qty ,
    EACH item      OF orderline No-Lock WHERE item.price > var-price:
```

V-Date	V-Qty	Var-Price	Nb records	SSJ elapse	No SSj Elapse	Gain %	Eviction
01/01/2008	-5	-1	179 700	9 924	14 595	32	No eviction
09/06/2011	**-5**	**-1**	**124 022**	**6 797**	**10 209**	**33**	**1/3 on level 2**
11/05/2011	-5	-1	58 636	3 363	5 173	35	2/3 on level 2
01/01/2008	32	-1	118 452	7 120	11 065	36	1/3 on level 3
01/01/2008	67	-1	57 690	4 370	7 407	41	2/3 on level 3
01/01/2008	-5	15	116 001	7 164	14 318	50	1/3 on level 4
01/01/2008	-5	60	53 805	4 375	14 560	70	2/3 on level 4

/ More evicted records => Faster

/ Evicted records in a deeper clause => Higher gain

What Else?

/ Performance : What else

/ No eviction on this query

```
FOR EACH customer NO-LOCK
                    WHERE customer.country = "CA" OR customer.country = "FR" Or customer.country = "USA",
    EACH order NO-LOCK WHERE order.ordernum < NB-MAX :
```

SSJ	Time	Mess Rec.	Mess Sent	Mb sent
Disable	**2 870**	**59 178**	**29 669**	**29**
Enable	2 108	29 511	29 512	31
Diff %	**26,7**	**50,1**	**0,54**	**-6,4**

/ Multiple Sub-queries : Each record of parent tables run a sub-query for its children

/ LESS CHATTY

/ Performance : What else

```
FOR EACH customer NO-LOCK
                  WHERE customer.country = "CA" OR customer.country = "FR" Or customer.country = "USA",
    EACH order NO-LOCK WHERE order.ordernum < NB-MAX :
```

NB-MAX	Children	SSJ	Time	Mess Rec	Mess Sent	Mb sent	Block/child
10	**8**	**Disabled**	**1 231**	**59 633**	**30 141**	**30,52**	**1**
10	8	Enabled	972	29 494	29 493	32,44	1
10	**8**	**Diff %**	**21,0**	**50,5**	**2,1**	**-6,3**	**1**
100	**98**	**Disabled**	**5 726**	**89 124**	**59 632**	**285,86**	**2**
100	98	Enabled	5 001	58 985	58 984	299,59	2
100	**98**	**Diff %**	**12,7**	**33,8**	**1,1**	**-4,8**	**2**
1 000	**206**	**Disabled**	**11 090**	**118 615**	**89 123**	**597,72**	**3**
1 000	206	Enabled	10 121	88 476	88 474	611,46	3
1 000	**206**	**Diff %**	**8,7**	**25,4**	**0,7**	**-2,3**	**3**
1 000	**206 Field**	**Disabled**	**11 740**	**59 633**	**30 141**	**186,65**	**1**
1 000	206 Field	Enabled	9 328	29 494	29 493	223,60	1
1 000	**206 Field**	**Diff %**	**20,5**	**50,5**	**2,1**	**-19,8**	**1**

/ Sub-queries : Effects decrease with children / Block

PROG/NOV

/ « EACH » are not « FIRST »

```
FOR EACH orderline NO-LOCK WHERE orderline.itemnum = 8 ,
   FIRST item NO-LOCK WHERE item.itemnum = orderline.itemnum ,
   FIRST order NO-LOCK WHERE order.ordernum = orderline.ordernum ,
   FIRST customer NO-LOCK WHERE customer.custnum = order.custnum :
```

```
FOR EACH orderline NO-LOCK WHERE orderline.itemnum = 8 ,
   EACH item NO-LOCK WHERE item.itemnum = orderline.itemnum ,
   EACH order NO-LOCK WHERE order.ordernum = orderline.ordernum ,
   EACH customer NO-LOCK WHERE customer.custnum = order.custnum :
```

	Time	Loop	Mess received	Mess sent	Mb sent
For each FIRST	1 639	44 198	44 203	44 202	24,9
For each EACH	5 264	44 198	132 599	132 598	34,8
% diff	- 212	=	-200	-200	-40

/ the 3 « FIRST » are sent in the same message

/ Message VS Byte

```
FOR EACH employee  NO-LOCK WHERE employee.Empnum >= 10001 AND Empnum <= 20001 ,
    first Family  NO-LOCK // Family.relation      = Fill('X' ,  300  )
                  WHERE Family.empnum = Employee.empnum ,
    first department NO-LOCK // department.deptname = Fill('X' , 3000  ).
                    WHERE department.deptcode = STRING ( employee.empnum, "99999"):


FOR EACH employee  NO-LOCK WHERE employee.Empnum >= 10001 AND Empnum <= 20001,
    EACH Family  NO-LOCK  // Family.relation      = Fill('X' ,   30  )
                 WHERE Family.empnum = Employee.empnum ,
    EACH department NO-LOCK //department.deptname = Fill('X' ,  300  )
                   WHERE department.deptcode = STRING ( employee.empnum, "99999"):
```

/ Time and Network activity

	Time	Loop	Mess rec.	Mess sent	Mb sent	
For each FIRST	935	10 001	10 004	10 004	34	Few large messages sent
For each EACH	1 661	10 001	20 006	20 005	7	More small messages sent
% diff	- 43	=	- 50	- 50	414	

/ The way to send the records has more effect than their size

What should we modify to (re-)engage SSJ ?

/ What should we modify ?

```
FOR EACH orderline   NO-LOCK WHERE orderline.itemnum = 8 :
    FIND FIRST ITEM  NO-LOCK WHERE item.itemnum = orderline.itemnum NO-ERROR.
    FIND FIRST order NO-LOCK WHERE order.ordernum = orderline.ordernum  NO-ERROR.
    FIND FIRST customer  NO-LOCK WHERE customer.custnum = order.custnum  NO-ERROR.
    IF NOT ( AVAIL ITEM AND AVAIL order AND AVAIL customer) THEN NEXT .
    i1 = i1 + 1 .
```

/ For each : Find First , Find First

```
FOR EACH orderl
    FIND FIRST
    FIND FIRST
    FIND FIRST
    IF NOT ( AV
    i1 = i1 + 1 .
```



```
FOR EACH orderline   NO-LOCK WHERE orderline.itemnum = 8 ,
    FIRST ITEM  NO-LOCK WHERE item.itemnum = orderline.itemnum ,
    FIRST order NO-LOCK WHERE order.ordernum = orderline.ordernum  ,
    FIRST customer  NO-LOCK WHERE customer.custnum = order.custnum  :
    IF NOT ( AVAIL ITEM AND AVAIL order AND AVAIL customer) THEN NEXT .
```

/ 1 Query

/ Time and Network activity

	Time	Loop	Mess Rec.	Mess sent	Mb sent
FOR EACH + 3 FIND FIRST	7 034	44 198	265 266	132 671	32,0
1 Query	1 625	44 198	44 203	44 202	24,9
% Benefits	76,9	=	83,3	66,7	22,3

/ What should we modify ?

```
FOR EACH orderline   NO-LOCK WHERE orderline.itemnum = 8 ,
    EACH ITEM    NO-LOCK WHERE item.itemnum = orderline.itemnum ,
    EACH order   NO-LOCK WHERE order.ordernum = orderline.ordernum ,
    EACH customer        WHERE customer.custnum = order.custnum  :
```

/ FORGETTING A NO-LOCK

```
FOR EACH orderline    NO-LOCK WHERE orderline.itemnum = 8 ,
    EACH ITEM    NO-LOCK WHERE item.itemnum = orderline.itemnum ,
    EACH order   NO-LOCK WHERE order.ordernum = orderline.ordernum ,
    EACH customer        WHERE customer.custnum = order.custnum  :

 FOR EACH orderline    NO-LOCK WHERE orderline.itemnum = 8 ,
    EACH ITEM  NO-LOCK WHERE item.itemnum = orderline.itemnum ,
    EACH order NO-LOCK WHERE order.ordernum = orderline.ordernum ,
    EACH customer NO-LOCK WHERE customer.custnum = order.custnum  :
```

/ ADD the NO-LOCK

/ Time and Network activity

	Time	Loop	Mess Rec.	Mess sent	Mb sent
Without Lock	9 715	44 198	353 662	176 869	37
With NO-LOCK	5 309	44 198	132 599	132 598	35
% Benefits	45	=	63	25	5

/ FORGETTING A NO-LOCK

```
FOR EACH orderline   NO-LOCK WHERE orderline.itemnum = 8 ,
    EACH ITEM    NO-LOCK WHERE item.itemnum = orderline.itemnum ,
    EACH order   NO-LOCK WHERE order.ordernum = orderline.ordernum ,
    EACH customer        WHERE customer.custnum = order.custnum  :
```



```
 FOR EACH orderline   NO-LOCK WHERE orderline.itemnum = 8 ,
     EACH ITEM  NO-LOCK WHERE item.itemnum = orderline.itemnum ,
     EACH order NO-LOCK WHERE order.ordernum = orderline.ordernum ,
     EACH customer NO-LOCK WHERE customer.custnum = order.custnum :
     FIND CURRENT customer EXCLUSIVE-LOCK .
```

/ Set the LOCK Outside the query

/ Time and Network activity

	Time	Loop	Mess Rec.	Mess sent	Mb sent
Without Lock	9 715	44 198	353 662	176 869	37
LOCK outside the query	8 737	44 198	220 995	220 994	50
% Gain	10	=	38	25	-36

/ What should we modify?

```
FOR EACH customer NO-LOCK WHERE country = 'CA' or country = 'FR'   ,
    EACH order NO-LOCK WHERE oreder.custnum = customer.custnum ,
    EACH orderline OF order NO-LOCK ,
    FIRST ttitem1 NO-LOCK WHERE ttitem1.itemnum = orderline.itemnum :
```


PROG/NOV

/ USE A TEMP-TABLE

```
FOR EACH customer NO-LOCK WHERE country = 'CA' or country = 'FR'   ,
    EACH order NO-LOCK WHERE oreder.custnum = customer.custnum ,
    EACH orderline OF order NO-LOCK ,
    FIRST ttitem1 NO-LOCK WHERE ttitem1.itemnum = orderline.itemnum :
```

/ => Put the TT Outside the query

```
FOR EACH customer No-Lock WHERE country = 'CA' or country = 'FR'   ,
    EACH order  No-LOCK WHERE order.custnum = customer.custnum ,
    EACH orderline OF order NO-LOCK ,
FOR FIRST ttitem1 NO-LOCK WHERE ttitem1.itemnum = orderline.itemnum :
```

/ Time and Network activity

	Time	Loop	Mess Rec.	Mess sent	Mb sent
TT in the query	4 460	179 700	126 519	63 297	22,6
TT outside the query	3 060	179 700	62 434	62 433	24,4
% Gain	31	=	50	1	9

PROG/NOV

/ OUTER-JOIN

```
qh:QUERY-PREPARE(
     'FOR EACH customer NO-LOCK WHERE country = "CA" or country = "FR",'+
     ' EACH order NO-LOCK WHERE order.custnum = customer.custnum , ' +
     ' EACH orderline OF order NO-LOCK ,  ' +
     ' FIRST ITEM NO-LOCK WHERE item.itemnum = orderline.itemnum ' +
     '                      And item.price > 17 OUTER-JOIN '  ).
```

/ Vs Without Outer-Join

	Time	Loop	Avail Item	Mess Rec.	Mess sent	Mb sent
OUTER-JOIN	17 218	179 700	109 896	485 937	243 011	56
No OUTER-JOIN	7 448	109 896	109 896	166 787	166 782	46

/ OUTER-JOIN : Use a dummy record

```
CREATE ITEM.
ASSIGN item.itemnum = -10000000
item.itemname = "DUMMY-RECORD".
```

```
qh:QUERY-PREPARE(
    'FOR EACH customer NO-LOCK WHERE country = "CA" or country = "FR",'+
    ' EACH order NO-LOCK WHERE order.custnum = customer.custnum , ' +
    ' EACH orderline OF order NO-LOCK ,  ' +
    ' FIRST ITEM NO-LOCK WHERE (     item.itemnum = orderline.itemnum ' +
    '                           And item.price > 17 )'+
    '                       Or (     item.itemnum = -10000000 ' +
    '                           AND item.itemname = "DUMMY-RECORD" )'  ).
```

/ Time and Network activity

	Time	Loop	Avail Item or Not Dummy	Mess Rec.	Mess sent	Mb sent
OUTER-JOIN	17 218	179 700	109 896	485 937	243 011	56
DUMMY RECORD	12 183	179 700	109 896	242 155	242 150	61
% Gain	29	=	=	50	,3	-9

/ SORTED RECORD - What should we modify?

```
FOR EACH orderline  NO-LOCK WHERE orderline.itemnum = 8 ,
    first ITEM  NO-LOCK WHERE item.itemnum = orderline.itemnum ,
    first order NO-LOCK WHERE order.ordernum = orderline.ordernum ,
    first customer  NO-LOCK WHERE customer.custnum = order.custnum
    BY order.orderdate:
  LEAVE.
```


PROG/NOV

/ SORTED RECORD

/ Put the By on the « last Child »

/ Use an extra buffer, sort on this one

```
FOR EACH orderline  NO-LOCK WHERE orderline.itemnum = 8 ,
    first ITEM  NO-LOCK WHERE item.itemnum = orderline.itemnum ,
    first order NO-LOCK WHERE order.ordernum = orderline.ordernum ,
    first customer  NO-LOCK WHERE customer.custnum = order.custnum ,
    FIRST Buf-order NO-LOCK WHERE Buf-order.ordernum = order.ordernum
    BY Buf-order.orderdate:
```

/ Time and Network activity

	Time	Loop	Mess Rec.	Mess sent	Mb sent
Sorted on a middle value	6 195	1	176 845	88 447	12,1
Sorted on the last child	2 003	1	44 208	44 207	11,8
% Benefits	38	=	76	50	2

/ SORTED RECORD

/ « Dynamic queries » do not work like « For Statement »

```
qh:SET-BUFFERS( bh_orderline , bh_item , bh_order , bh_customer ) NO-ERROR .
qh:QUERY-PREPARE
    ('   FOR EACH orderline  NO-LOCK WHERE orderline.itemnum = 8 , ' +
     '       first ITEM  NO-LOCK WHERE item.itemnum = orderline.itemnum ,' +
     '       first order NO-LOCK WHERE order.ordernum = orderline.ordernum ,' +
     '       first customer  NO-LOCK WHERE customer.custnum = order.custnum ' +
     '       BY order.orderdate: ' ).
```



/ Time and Network activity

	Time	Loop	Mess Rec.	Mess sent	Mb sent
Sorted on a middle value	6 195	1	176 845	88 447	12,1
Dynamic query	2 008	1	44 207	44 206	10,0
% Gains	68	=	75	50	17

PROG/NOV

/ SORTED RECORD : Summary

/ No SSJ on static queries if sorted on a middle child

/ SSJ on dynamic queries even when sorted on a middle child

/ HUGE optimisation on static queries with OE12.8.?

/ To sort the records, ABL gathers the needed records

	Time	Mess Rec.	Mess sent	Mb sent	Record sent
Sorted on a middle value	6 195	176 845	88 447	12,1	132 598
Sorted on the last child	2 003	44 208	44 207	11,8	220 998
Dynamic query	2 008	44 207	44 206	10,0	176 799

/ Those tricks are faster because we are using FIRST

/ 1 message to sent 3 « children level »

/ SORTED RECORD : with EACH and a lower level

```
FOR EACH customer NO-LOCK WHERE country = 'CA' or country = 'FR',
    EACH order NO-LOCK WHERE order.custnum = customer.custnum ,
    EACH orderline OF order NO-LOCK ,
    FIRST ITEM NO-LOCK WHERE item.itemnum = orderline.itemnum
    BY order.orderdate:
```

	Time	Mess Rec.	Mess sent	Mb sent	Record sent
Sorted on a middle value	1 096	27 290	13 652	3,3	63 225
Sorted on the last child	12 735	277 298	277 297	49,6	636 696
Dynamic query	11 438	242 157	242 152	36,4	421 845

/ Static query gathers records on « customer » and « order » for sorting

/ Faster than SSJ

/ Dynamic queries do not have this feature

/ Sorted …

/ What is the best ??

/ Dynamic queries

- / SSJ
- / Evicted records

/ Static queries

- / SSJ if sorted on the most inner child
- / Gather only what needed (without SSJ)

/ Probably gathering only what needed (No SSJ)

- / Mostly

The blocks still « poorly » fill ?

/ Main reasons of poor messaging

/ A lot of FIND …

- / Each "Find" requires request/response exchanges between the client and the server

/ A LOT OF FIND

/ VOTE FOR IDEA N° 1576 - Improve the performance of code using FIND statements

Home / OPENEDGE-I-1576 / New idea

61
VOTED

Improve the performance of code using FIND statements

Improve the performance of code using FIND statements by consolidating multiple FINDs into fewer messages.

Progress continues to urge people to migrate from shared memory connections to client/server architectures. Customers are hesitant to do so because the performance penalty is substantial. Over the years Progress has made huge strides in improving the performance of FOR EACH ... NO-LOCK "queries" but the humble FIND statement has not been improved. The FIND statement is, however, one of the things that is special about the 4gl and which provides numerous advantages over SQL tooling and, as a result, is very heavily used in both legacy code and new code.

Consider code such as:

find job-act where job-act.job-Id = jobIdNum no-lock no-error.

find job where job.job-id = job-act.job-id no-lock no-error.

Currently such a snippet requires 2 sets of request/response exchanges between the client and the server. Many such snippets contain 5 or 10 or more consecutive FINDs that could be easily consolidated.

The idea: create something along the lines of a "type 2" network message protocol that allows multiple records from mixed tables to be requested and returned together. The compiler would see that the statements above are an opportunity to make one request and receive one response resulting in half the network traffic in this simple example. Far greater reductions would occur in longer examples.

/ Main reasons of poor messaging

/ A lot of FIND …

- / Each "Find" requires request/response exchanges between the client and the server
- / IDEA :1576 - Improve the performance of code using FIND statements

/ The messages sent/received are smaller than –Mm

- / Especially when parent records have few children or field selection shrinks the row size

/ Could be better with a better filling for message

```
FOR EACH  customer  NO-LOCK WHERE customer.custnum < 1500 ,
   EACH  order NO-LOCK WHERE  order.custNum = customer.custNum ,
   EACH Orderline NO-LOCK WHERE orderline.ordernum = order.ordernum:
```

/ SSJ is limited by the number of children of a parent

/ Probably less then –Mm

- **Write a "Beta" to gather the children of 50 parents at a time**

	Elapse time	Loop	Mess received	Mess sent	Kb sent	Rec sent	Msg Size
SSJ	62	6 588	2 377	2 376	935,82	8 963	393
Better filling (No SSJ)	40	6 588	219	112	634,91	8 989	5 668
SELF	18	6 588	No Activity for mtprosrv				

/ IDEA1607 - Optimize the performance by having a better filling of the message sent with SSJ enable

/ Main reasons of poor messaging

/ A lot of FIND …

- / Each "Find" requires request/response exchanges between the client and the server
- / IDEA :1576 - Improve the performance of code using FIND statements

/ The messages sent/received are smaller than –Mm

- / Especially when parent records have few children or field selection shrinks the row size
- / IDEA1607 - Optimize the performance by having a better filling of the message sent with SSJ enable

/ Sort : Gather the records to sort

- / Could be improved to use SSJ

/ Sort : make a better use of capabilities of dynamic and static queries

/ Gather the needed records to sort them

/ Using SSJ to avoid gathering unneeded records

- Do not match on an inner table of the join

/ Idea 1822 : on sort, use the best between SSJ and sort on static query for dynamic and static query

/ Main reasons for poor messaging

/ A lot of FIND …

- / Each "Find" requires request/response exchanges between the client and the server
- / IDEA :1576 - Improve the performance of code using FIND statements

/ The messages sent/received are smaller than –Mm

- / Especially when parent records have few children or field selection shrinks the row size
- / IDEA1607 - Optimize the performance by having a better filling of the message sent with SSJ enabled

/ Sort : Gather the records to sort

- / Could be improve to use SSJ
- / Idea 1822 : on sort, use the best between SSJ and sort on static query for dynamic and static query

/ Sort bring back the records records

- / use 1 message per record/loop

/ By with a sort of the result

/ Focus on "bring back" the sorted records

```
FOR each customer NO-LOCK  ,
    EACH order     OF customer No-LOCK  ,
    EACH orderline OF  order No-LOCK     ,
    EACH item      OF orderline NO-LOCK
    BY orderline.linenum
    BY order.ordernum
    BY customer.custnum :
```

/ Beta : bringing back the records per 50 rows

/ From the first result and the last result

	Elapse time	Mess Rec.	Kb Rec	Mess Sent	Kb Sent	Mess Sent size
BY	2 001	34 075	3 593	34 075	6 176	186
Beta BY	1 066	2 064	6 175	1 032	3 137	3 113
Gain %	47	94	-72	-97	49	- 1 574

/ VOTE FOR IDEA 1664 - Optimize performance for REMC on BY

/ TO SUM UP THINGS ON IDEAS

/ PERFORMANCE

- / 1576 - Improve the performance of code using FIND statements
- / 1607 - Optimize the performance by having a better filling of the message sent with SSJ enable
- / 1664 - Optimize performance for REMC on BY
- / 1730 - Optimize performance on dynamic queries using “break by” for REMC
- / 1822 - on sort, use the best between SSJ and sort on static query for dynamic and static query

/ To Analyze

- / 1573 - Add client server traffic metrics to _connect

/ Seen on CVP

2026.4 Program Increment (PI) Objectives for Nirvana Team

Dear OpenEdge Community Users:

As part of this 2026.4 PI (Program Increment), Nirvana will be working on two features - **SSJ Performance improvements** and **Primitive Datatypes as Classes**.

Enhanced SSJ Message Packing (OCTA-97885):

OpenEdge's Server-Side Join already cuts network traffic by doing join processing on the database server, but today it can send many network messages that are only partially filled - especially when parent records have few children or field selection shrinks the row size - adding unnecessary round trips and slowing queries. This enhancement improves how SSJ packs and fills those messages, combining results into fewer, fuller messages so queries run faster - with no application changes, no change to query results, and benefits that scale across local, remote, and distributed/cloud deployments.

Merci de votre écoute

PROGINOV