# GREAT PERFORMANCE OUT OF THE BOX WITH SSJ IN OPENEDGE 12

****

**OpenEdge 12.1 was released. Did you know that it extends the server-side join support to dynamic queries?**

**OpenEdge 12 introduced the server-side join functionality with the FOR statement. In OpenEdge 12.1, you can now use the server-side join functionality with FORWARD-ONLY dynamic queries. In this post, we take a look at the enhancement.**

Contents

**[About Server-Side Join](#_Toc23247019)**

**[Server-Side Join Processing with a FOR Statement](#_Toc23247020)**

**[Testing Performance by Turning Off Server-Side Join](#_Toc23247021)**

**[Sample Code Using FORWARD-ONLY Dynamic Queries](#_Toc23247022)**

**[QryInfo Logging](#_Toc23247023)**

**[PROSERVE Parameters](#_Toc23247024)**

**[Videos](#_Toc23247025)**

**[Resources](#_Toc23247026)**

**[Conclusion](#_Toc23247027)**

## **About Server-Side Join**

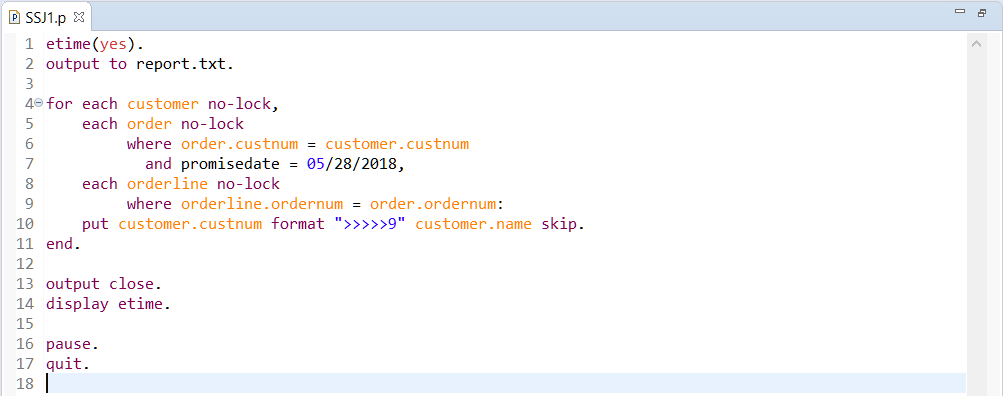
The server-side join (SSJ) processing improves performance by resolving queries on the server and reducing the data sent over the network.

SSJ is enabled, by default, and is used to resolve joins that meet the following conditions:

* Joins in a client/server environment
* Joins using NO-LOCK with the FOR statement or FORWARD-ONLY dynamic query
* Joins with up to 10 tables on the same logical database

## **Server-Side Join Processing with a FOR Statement**

The following program shows a join using the FOR statement. The connection to the database is done using network parameters so that SSJ can be used.

****

To run the sample program using SSJ on a system with OpenEdge 12.1 installed, follow these steps:

1. prodb sports2020 sports2020
2. proserve sports2020 -S 20000
3. mpro sports2020 -S 20000 -p SSJ1.p

The connection to the database is done using the network parameter “-S 20000”. A client/server connection is established. The program SSJ1.p runs and displays the elapsed time for the query.

You can run the program on the same machine or on a remote machine.

When you run your program with SSJ, the performance is much faster. (In case you need to compare the performance with and without SSJ, the next section provides the steps you need to run without SSJ.)

You may ask, “How fast is the performance?” And, you might simply answer, “great performance.”

**Notes:**

* Your performance metrics may vary. In actuality, the performance depends on the data and the reduction of data sent over the network.
* There are no changes to the index selection rules when you run the server-side join functionality.
* If needed, you can specify USE-INDEX to ensure that a certain index is used for optimal query resolution.

## **Testing Performance by Turning Off Server-Side Join**

To turn off SSJ so that you can compare the performance, follow these steps:

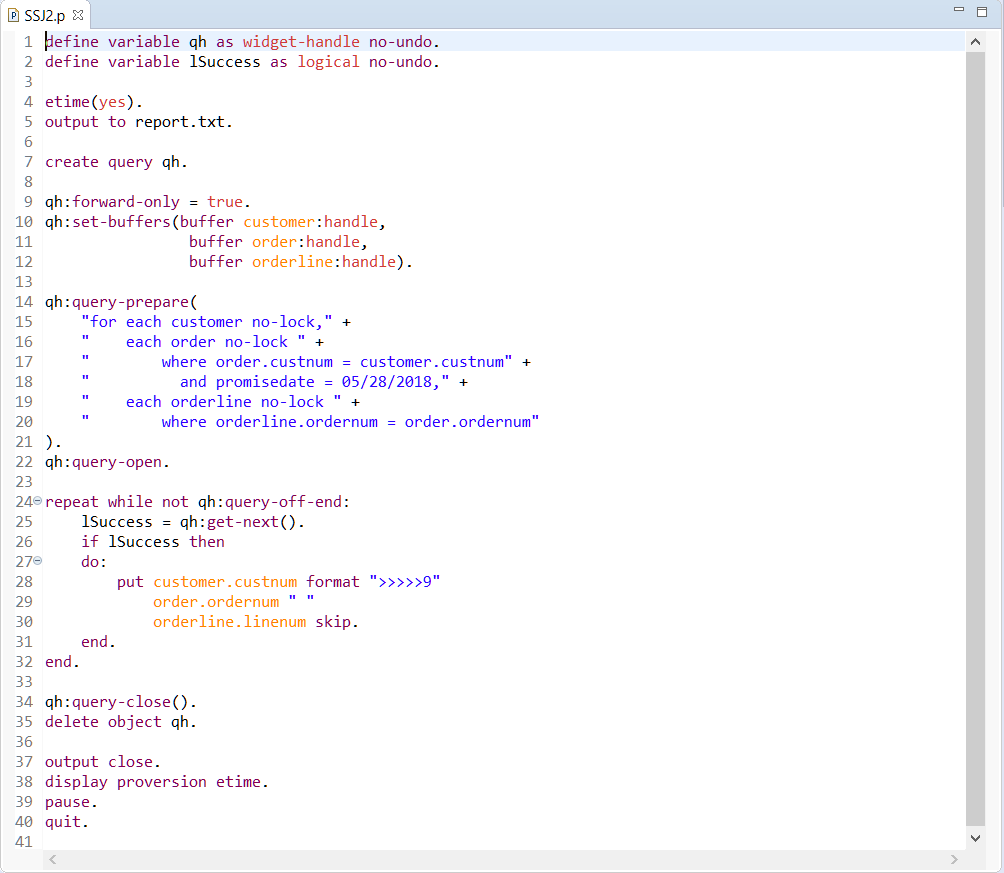
1. prodb sports2020 sports2020 **-ssj 0**
2. proserve sports2020 -S 20000
3. mpro sports2020 -S 20000 -p SSJ1.p

The difference with the steps in the previous section is that the “-ssj 0” option is used to disable SSJ.

When you test performance, you might want to shut down and restart the database broker so that improvements in the query execution are not associated with the caching of the blocks or records.

The performance when you disable SSJ (-ssj 0) would be much slower.

## **Sample Code Using FORWARD-ONLY Dynamic Queries**



To run the sample program using SSJ on a system with OpenEdge 12.1 installed, follow these steps:

The connection to the database is done using the network parameter “-S 20000”. A client/server connection is established. The program SSJ2.p runs and displays the elapsed time for the query.

The sample program uses a FORWARD-ONLY dynamic query with NO-LOCK.

## **QryInfo Logging**

You can analyze the execution of a query using the QryInfo logging option.

To run the sample program without SSJ with QryInfo logging on a machine with OpenEdge 12.1 installed, follow these steps:

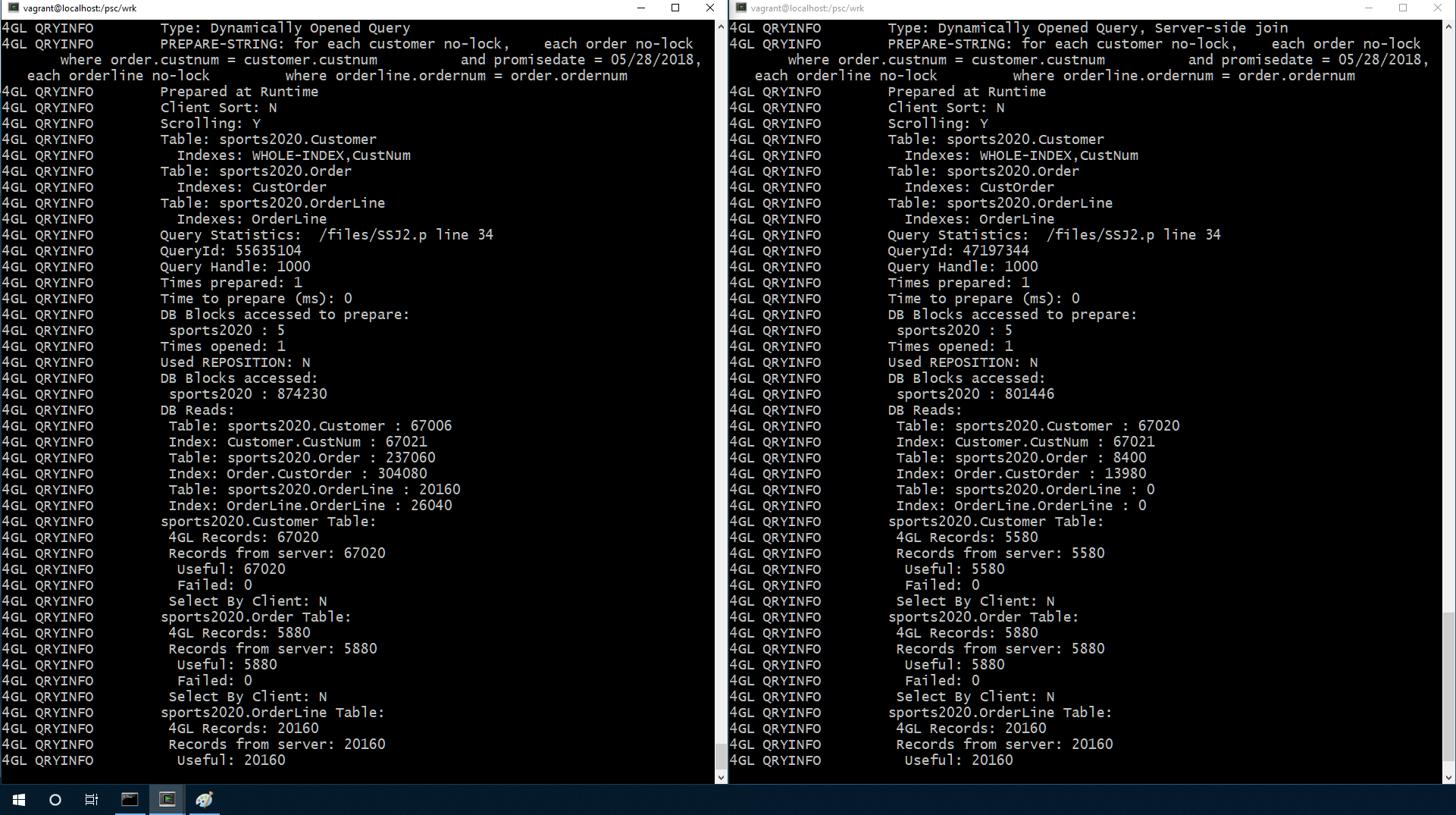
1. proserve sports2020 -S 20000 -ssj 0
2. mpro sports2020 -S 20000 -p SSJ2.p \

-clientlog client.log -logentrytypes QryInfo -logginglevel 3

To run the sample program using SSJ with QryInfo logging on a machine with OpenEdge 12.1 installed, follow these steps:

1. proserve sports2020 -S 20000
2. mpro sports2020 -S 20000 -p SSJ2.p \

-clientlog client.log -logentrytypes QryInfo -logginglevel 3



In the screenshot, you can see the output of using QryInfo logging, with the sample program using a dynamic query. The left side shows the output when server-side join is disabled, and the right side shows the output with server-side join enabled (default).

The phrase “Type: Dynamically Opened Query, Server-side join” is shown to indicate the usage of SSJ.

The line “Records from server:” for “sports2020.Customer Table” demonstrates the performance gain by using SSJ. Only the Customer records in the result set, 5,580 records, are returned from the server. In contrast, the number of records returned when the join is resolved on the client is 67,020.

## **PROSERVE Parameters**

The server-side join functionality is controlled by the -ssj database parameter.

By default, the database uses -ssj 1. The use of server-side join can be turned off by specifying -ssj 0 as a startup parameter when starting the database broker with PROSERVE.

The multi-threaded server (MTS) functionality should also be enabled for the server-side join to operate. MTS is also enabled by default.

You can use –ssj 0 if you want to turn off the functionality, for example, if you are analyzing the gains from using SSJ in your application.

|  |  |
| --- | --- |
| **Parameters** | **Server-Side Join** |
| -threadedServer 1 -ssj 1 | Enabled (default). Multi-threaded server (MTS) with server-side join (SSJ). |
| -threadedServer 1 -ssj 0 | Disabled |
| -threadedServer 0 -ssj 1 | -ssj 1 is ignored because the multi-threaded server is not enabled. |
| -threadedServer 0 -ssj 0 | Disabled |

## **Videos**

The following videos provide an introduction and demo to the server-side join functionality:

* SSJ with OpenEdge 12

<https://docs.progress.com/bundle/videos-database-12/page/Server-side-Join-Processing-with-OpenEdge-12.html>

* Update to SSJ in OpenEdge 12.1

<https://progresssoftware.sharepoint.com/:v:/s/OpenEdgeA-Team/EXPqvP0OQDRDmG1s9WcmByYBoNBHbyIHHeL8zxh-TNufKQ?e=MfhoeG>

## **Resources**

* OpenEdge 12.0 Database Performance and Server Side Joins presentation by Rich Banville at the PUG Challenge Americas:
  + [PDF OpenEdge 12.0 Database Performance and Server Side Joinstp://pugchallenge.org/downloads2018/Banville\_Performance.pdf](http://pugchallenge.org/downloads2018/Banville_Performance.pdf)
* Progress Information Hub:
  + <https://docs.progress.com/bundle/database-performance/page/Server-side-join-processing-for-dynamic-queries.html>
  + <https://docs.progress.com/bundle/openedge-whats-new/page/New-server-side-join-processing.html>
  + [https://docs.progress.com/bundle/openedge-whats-new/page/New-server-side-join-ocessing.html](https://docs.progress.com/bundle/openedge-whats-new/page/New-server-side-join-processing.html)

<https://docs.progress.com/bundle/database-performance/page/Server-side-join-processing-for-dynamic-queries.html>

## **Conclusion**

The server-side join processing functionality in OpenEdge 12.*x* delivers great performance out of the box.

Here are some of the things that we learned:

* SSJ is enabled by default.
* SSJ is available with the FOR statement and FORWARD-ONLY dynamic queries.
* Actual performance gains depend on the data being queried and the reduction of the records sent over the network.
* The QryInfo logging option can be used to analyze the query execution.

Have you seen the performance improvement of SSJ in your application?

How much performance have you gained? 2 x, 3 x? Please let us know in the comments.

You can find the source code for this demo here.

Stay tuned. More enhancements to SSJ are coming in a future release.

Thank you for reading.