



InterSystems IRIS Demo: Deploy a Sharded Cluster

Version 2023.1
2024-07-11

InterSystems IRIS Demo: Deploy a Sharded Cluster

InterSystems IRIS Data Platform Version 2023.1 2024-07-11

Copyright © 2024 InterSystems Corporation

All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

Table of Contents

InterSystems IRIS Demo: Deploy a Sharded Cluster	1
1 How Can Sharding Help You?	1
2 How Does Sharding Work?	1
3 Try It! Deploy and Demo an InterSystems IRIS Sharded Cluster	2
3.1 Use ICM to Deploy the Cluster	3
3.2 Use Different Shard Keys to Distribute Rows Differently, Then Query the Sharded Tables	3
4 More Sharded Cluster Options	6
5 Learn More About Sharding	7
List of Figures	
Figure 1: A Basic Sharded Cluster	2

InterSystems IRIS Demo: Deploy a Sharded Cluster

This page introduces you to the InterSystems IRIS® data platform sharding feature and its use in a sharded cluster to horizontally scale InterSystems IRIS for data volume.

1 How Can Sharding Help You?

We are all managing more data than ever before and being asked to do more with it — and the response times demanded are growing ever shorter. Each business-specific workload presents different challenges to the data platform on which it operates — and as workloads grow, those challenges become even more acute.

InterSystems IRIS includes a comprehensive set of capabilities to scale your applications, which can be applied alone or in combination, depending on the nature of your workload and the specific performance challenges it faces. One of these, sharding, partitions both data and its associated cache across a number of servers, providing flexible, inexpensive performance scaling for queries and data ingestion while maximizing infrastructure value through highly efficient resource utilization. An InterSystems IRIS *sharded cluster* can provide significant performance benefits for a wide variety of applications, but especially for those with workloads that include one or more of the following:

- High-volume or high-speed data ingestion, or a combination.
- Relatively large data sets, queries that return large amounts of data, or both.
- Complex queries that do large amounts of data processing, such as those that scan a lot of data on disk or involve significant compute work.

Each of these factors on its own influences the potential gain from sharding, but the benefit may be enhanced where they combine. For example, a combination of all three factors — large amounts of data ingested quickly, large data sets, and complex queries that retrieve and process a lot of data — makes many of today's analytic workloads very good candidates for sharding.

Note that these characteristics all have to do with data; the primary function of InterSystems IRIS sharding is to *scale for data volume*. But a sharded cluster can also include features that *scale for user volume*, when workloads involving some or all of these data-related factors also experience a very high query volume from large numbers of users. And sharding can be combined with vertical scaling as well. With InterSystems IRIS, you can create just the right overall scaling solution for your workload's performance challenges.

2 How Does Sharding Work?

The heart of the sharded architecture is the partitioning of data and its associated cache across a number of systems. A sharded cluster partitions large database tables horizontally — that is, by row — across multiple InterSystems IRIS instances, called *data nodes*, while allowing applications to access these tables through any one of those instances. Each data node's share of the cluster's sharded data is called a *shard*. This architecture provides three advantages:

- Parallel processing

Queries are run in parallel on the data nodes, with the results combined, and returned to the application as full query results, significantly enhancing execution speed in many cases.

- Partitioned caching

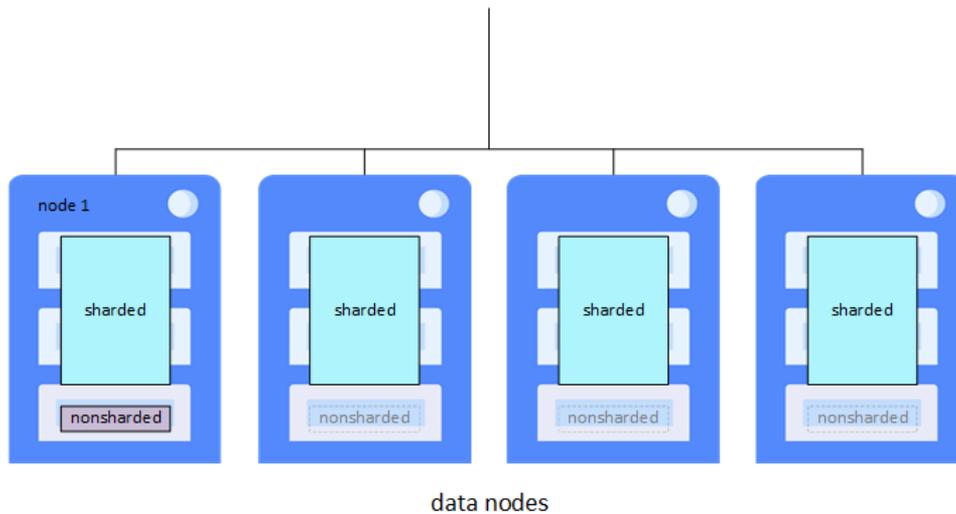
Each data node has its own dedicated cache, rather than a single instance's cache serving the entire data set, which greatly reduces the risk of overflowing the cache and forcing performance-degrading disk reads.

- Parallel loading

Data can be loaded onto the data nodes in parallel, reducing cache and disk contention between the ingestion workload and the query workload and improving the performance of both.

A federated software component called the *sharding manager* keeps track of which data is on which data nodes and directs queries accordingly. Nonsharded data is stored on the first data node configured, called *data node 1* (which also stores code and metadata). From the perspective of the application SQL, the distinction between sharded and nonsharded tables is totally transparent.

Figure 1: A Basic Sharded Cluster



3 Try It! Deploy and Demo an InterSystems IRIS Sharded Cluster

In this exploration, you will

- Deploy a basic sharded cluster in a public cloud using InterSystems Cloud Manager (ICM).

ICM is an automated command-line tool that makes it easy to provision infrastructure in the cloud and deploy InterSystems IRIS and other services on the provisioned nodes. ICM takes care of all the needed InterSystems IRIS configuration, so if you specify a sharded cluster, the cluster is ready to use when deployment is complete.

- Create three sharded tables from the same data using different shard keys and observe the different distributions of the rows across the shards on the data nodes.

The shard key is the field or fields used to determine which rows of a sharded table are stored on which shards. By default, the RowID is used as the shard key, and this distribution is the most effective approach for most sharded tables because it offers the best guarantee of an even distribution of data and allows the most efficient parallel data loading.

There is one case, however, in which a user-defined key can be beneficial. When you shard two large tables that are frequently joined in queries on the field or fields used to join them, the rows to be joined are stored on the same shards, and a join can be executed locally on each shard rather than across the shards, providing a significant performance boost. This is called a *cosharded join*.

- Run the same query on all three sharded tables to demonstrate that the distribution of rows across shards is entirely transparent to the query.

To introduce you to sharding without bogging you down in details, we've kept this simple. A sharded cluster in production requires planning and a number of decisions, so be sure not to confuse this exploration of sharding with the real thing! For example, when designing a production cluster, you would review your schema and tables to decide which tables are to be sharded, then decide how many data nodes to deploy (typically on the order of 4 to 16) based on both the anticipated working set for the sharded data and the amount of memory available for the database caches on your servers. In this exploration, however, you'll deploy a basic cluster of two data nodes. The sources listed at the end of this document will give you a good idea of what's involved in using sharding in production. (For detailed information about sharding and sharded clusters, see [Horizontally Scaling for Data Volume with Sharding](#).)

These instructions assume you have InterSystems IRIS sharding licenses and access to InterSystems software downloads.

3.1 Use ICM to Deploy the Cluster

You can find the procedure for using ICM to deploy the sharded cluster on the Amazon Web Services public cloud platform in [Try It! Deploy InterSystems IRIS in the Cloud with ICM](#). You can use the entire procedure as it is presented there. When you get to [Customize definitions.json](#), verify that the value of each LicenseKey field is the filename of a InterSystems IRIS sharding license in the staging location specified by the LicenseDir field in the defaults.json file. When you have finished the deployment stage ([Deploy InterSystems IRIS](#)), remain in the ICM container.

Note: As described in [Deploying the Sharded Cluster](#), you can also install the InterSystems IRIS instances on existing physical, virtual or cloud machines and deploy the sharded cluster using the Sharding API or the Management Portal, or use one of several methods for automated deployment of sharded clusters.

3.2 Use Different Shard Keys to Distribute Rows Differently, Then Query the Sharded Tables

To see how a sharded cluster partitions a sharded table across the shards on the data nodes based on the shard key you use, you are going to connect to the cluster and take the following steps:

- Create and populate a small nonsharded table.
- Create three sharded tables with the same fields as the nonsharded table, using three different shard keys.
- Populate the sharded tables from the nonsharded table.
- Select the rows from the sharded tables to see how the rows are distributed differently across the shards (data nodes) in each case.

When a table is sharded, RowID values are assigned from different ranges for each shard, starting at widely-separated ranges; rows in the same range are on the same shard. When dealing with a small table, as in this case, the shard RowIDs are easy to distinguish, clearly showing you which rows are on one shard and which on the other (although you cannot tell which shard is which). Rows of tables sharded on RowID are distributed in a way that has no relationship with the data in the rows, and which would likely be different if you emptied and reloaded a table. A user-defined shard key, on the other hand, distributes rows based on the values of the field or fields in the key, and will therefore be the same when you empty and reload the table (assuming the values and the number of shards have not changed).

- Query all three tables for the maximum length of one of the fields to show that the distribution of rows across shards is transparent to the query.

Use the procedure that follows for this part of the exploration:

1. Open a Terminal window for either data node instance.

- On the ICM command line, issue the command **icm session** command, using the **-machine** option to specify the name of one of the data nodes and the **-namespace** option to specify the cluster namespace **IRISCLUSTER** (or the namespace you specified with the Namespace field in the defaults file, if different). If you are uncertain about the names of the nodes, you can use the **icm inventory** command to display the node names:

```
$ icm inventory
Machine          IP Address      DNS Name          Region
Zone
-----          -
Acme-DATA-TEST-0001 00.53.183.209  ec2-00-53-183-209.us-west-1.compute.amazonaws.com  us-west-1
c
Acme-DATA-TEST-0002 00.53.183.185  ec2-00-53-183-185.us-west-1.compute.amazonaws.com  us-west-1
c
```

Choose a node (it does not matter which) and issue the command:

```
icm session -machine Acme-DATA-TEST-0002 -namespace IRISCLUSTER
```

The **icm session** command opens a Terminal window for the InterSystems IRIS instance on the node you specify.

- If you deployed your cluster manually using the %SYSTEM.Cluster.Sharding API, open a Terminal window on either instance using the [procedure described for your instance in *InterSystems IRIS Basics: Connecting an IDE*](#) and switch to the **IRISCLUSTER** namespace (or the namespace you specified when initializing the cluster, if different).

```
Node: Acme-DATA-TEST-0002, Instance: IRIS
USER>set $namespace="IRISCLUSTER"
IRISCLUSTER>
```

2. Open the Terminal SQL shell:

```
IRISCLUSTER>do $SYSTEM.SQL.Shell()
[SQL]IRISCLUSTER>>
```

3. Create and populate the nonsharded table, test.nonsharded, with the following SQL statements.

```
CREATE TABLE test.nonsharded (field1 CHAR(5), field2 CHAR(5))
INSERT INTO test.nonsharded (field1,field2) VALUES ('one','one')
INSERT INTO test.nonsharded (field1,field2) VALUES ('one','two')
INSERT INTO test.nonsharded (field1,field2) VALUES ('one','three')
INSERT INTO test.nonsharded (field1,field2) VALUES ('two','one')
INSERT INTO test.nonsharded (field1,field2) VALUES ('two','two')
INSERT INTO test.nonsharded (field1,field2) VALUES ('two','three')
INSERT INTO test.nonsharded (field1,field2) VALUES ('three','one')
INSERT INTO test.nonsharded (field1,field2) VALUES ('three','two')
INSERT INTO test.nonsharded (field1,field2) VALUES ('three','three')
```

Use `SELECT` to see the contents of the table:

```
SELECT * FROM test.nonsharded

field1 field2
one     one
one     two
one     three
two     one
two     two
two     three
three   one
three   two
three   three
```

4. Create three sharded tables with the same fields that are in `test.nonsharded`, using the default shard key (`RowID`) for the first one, the **field1** field for the second one, and the **field2** field for the third, and populate each using an `INSERT INTO` statement selecting from `test.nonsharded`:

```
CREATE TABLE test.rowid (field1 CHAR(5), field2 CHAR(5), SHARD)
INSERT INTO test.rowid (field1,field2) SELECT field1,field2 FROM test.nonsharded

CREATE TABLE test.field1 (field1 CHAR(5), field2 CHAR(5), SHARD KEY (field1))
INSERT INTO test.field1 (field1,field2) SELECT field1,field2 FROM test.nonsharded

CREATE TABLE test.field2 (field1 CHAR(5), field2 CHAR(5), SHARD KEY (field2))
INSERT INTO test.field2 (field1,field2) SELECT field1,field2 FROM test.nonsharded
```

5. Use `SELECT *,%ID` to display the contents of each sharded table with its `RowID` on the shard.

```
SELECT *,%ID FROM test.rowid ORDER BY %ID

field1 field2 ID
one     one    1
one     two    2
two     one    3
two     two    4
three   three   5
one     three   256000
two     three   256001
three   one     256002
three   two     256003
```

This distribution does not reflect the values of `field1` or `field2` (rows with all three values of each are located on both shards). If you delete, recreate, and reload `test.rowid`, the distribution may be different.

```
SELECT *,%ID FROM test.field1 ORDER BY %ID

field1 field2 ID
one     one    1
one     two    2
one     three   3
two     one    256000
two     two    256001
two     three   256002
three   one    256003
three   two    256004
three   three   256005
```

Sharding on the **field1** field distributes the rows so that those with the same value of **field1** are placed on the same shard. In this case, rows with value **one** are on one shard and those with values **two** and **three** on the other, but which values end up on which shard depends on how many shards and how many values there are.

```
SELECT *,%ID FROM test.field2 ORDER BY %ID

field1  field2  ID
one     one     1
two     one     2
three   one     3
one     two     256000
one     three   256001
two     two     256002
two     three   256003
three   two     256004
three   three   256005
```

Here, distribution is determined by the value of the **field2** field.

- Finally, as an example of how sharding distributes work to the data nodes, use the following `SELECT` statement with all three sharded tables:

```
SELECT MAX(LENGTH(field2)) FROM <table>
```

In each case, the result is **5**, the length of the longest value, **three**, because the distribution of rows across the shards is entirely transparent to the query. The `MAX(LENGTH(field2))` expression is computed independently on each shard, and sharding manager chooses the `MAX()` of the results they return. For example, when the query is run on the `test.field2` table, one shard returns **3**, because it has only the value **one** in the **field2** field, while the other shard returns **5**; the sharding manager then chooses **5** as the larger of the two results.

If you like, use **EXPLAIN** to show the query plan indicating explicitly how work is sent to the shards:

```
EXPLAIN SELECT MAX(LENGTH(field2)) FROM <table>
```

4 More Sharded Cluster Options

Additional options for a sharded cluster include the following:

- You can add data nodes at any time and rebalance existing sharded data across the expanded set of data nodes. Rebalancing cannot coincide with queries and updates, and so can take place only when the sharded cluster is offline and no other sharded operations are possible. (See [Add Nodes and Rebalance Data](#).)
- To add high availability for the data on the cluster, you can deploy data nodes as mirrored failover pairs. (See [Mirror for High Availability](#).)
- For advanced use cases in which extremely low query latencies are required, potentially at odds with a constant influx of data, *compute nodes* can be added to provide a transparent caching layer for servicing queries. When a cluster includes compute nodes, read-only queries are automatically executed in parallel on the compute nodes, rather than on the data nodes; all write operations (insert, update, delete, and DDL operations) continue to be executed on the data nodes. This division of labor separates the query and data ingestion workloads while maintaining the advantages of parallel processing and distributed caching, improving the performance of both. (See [Deploy Compute Nodes](#).)

5 Learn More About Sharding

To learn more about sharding, see

- [Data Platform Scalability Technology Overview](#) (video)
- [Introduction to Sharding](#) (online course)
- [Sharding Basics](#) (online course)
- [Deploying InterSystems IRIS in Containers and the Cloud](#) (learning path)
- [Scalability Guide](#)
- [InterSystems IRIS Demo: InterSystems Cloud Manager](#)

