InterSystems IRIS デモ：シャード・クラスタの導入

Version 2024.1
2024-06-03

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

InterSystems IRIS デモ : シャード・クラスタの導入
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

図一覧

InterSystems IRIS デモ：シャード・クラスタの導入

ここでは、InterSystems IRIS® データ・プラットフォームのシャーディング機能の概要と、シャード・クラスタでこれを使用して、データ量に応じて InterSystems IRIS を水平方向に拡張する方法を説明します。

1 シャーディングがどう役立つのか

ビッグ・データの熱を感じていますか。

準備できているかどうかに関係なく、我々はかつてないほど多くのデータを管理し、そこからより多くの成果を上げることを求められています。そして、応答時間の短縮化がますます要求されています。1,000 万人の患者を治療するのか、1 日数十億件の金融オーダーを処理するのか、星雲を追跡するのか、工場の 1,000 台のエンジンを監視するのかにかかわらず、データ・プラットフォームは、現在のデータ作業負荷をサポートするだけでなく、需要の増大に合わせて拡張可能でなければなりません。それと同時に、求められるパフォーマンス標準を維持し、ビジネスの中断を回避できなければなりません。ビジネス固有の作業負荷ごとに、それぞれが運用されるデータ・プラットフォームに対して異なる課題が突きつけられ、作業負荷が拡大するにつれて、その課題はさらに深刻になります。

InterSystems IRIS には、アプリケーションを拡張するための包括的な機能セットが含まれており、直面する作業負荷や固有のパフォーマンス課題の性質に応じて、単独または組み合わせによって適用することができます。この中の 1 つであるシャーディングでは、データとそれに関連するキャッシュを複数のサーバ間で分割して、クエリおよびデータ取り込みに応じた柔軟で安価なパフォーマンス拡張を実現しながら、きわめて効率的にリソースを使用することにより、インフラストラクチャの価値を最大化します。InterSystems IRIS のシャード・クラスタは、広範囲のアプリケーションのパフォーマンスを大幅に向上させることができますが、特に、以下の 1 つ以上を含む作業負荷に対して顕著です。

- 大容量または高速のデータ取り込み（またはその組み合わせ）。
- 比較的大きなデータ・セット、大量のデータを返すクエリ（またはその両方）。
- 大量のデータ処理を実行する複雑なクエリ（ディスク上の大量のデータをスキャンするものや、大量の計算作業を必要とするものなど）。

これらの要因はそれぞれ個別でも、シャーディングから得られる潜在的恩恵に影響しますが、結合すれば恩恵も大きくなる場合があります。例えば、大量のデータの高速の取り込み、大規模データ・セット、大量のデータを取得して処理する複雑なクエリという 3 つの要因すべてを組み合わせると、現代の分析上の作業負荷の多くがシャーディングに非常に適した候補になります。

これらの特性にはすべてデータ操作の必要があります。したがって InterSystems IRIS のシャーディングの主要な機能は、データ量に応じた拡張です。ただし、シャード・クラスタはユーザ数に応じた拡張機能にも対応できます。これらのデータ関連要因の一部またはすべてを含む作業負荷に、多数のユーザからの大量クエリも発生した場合です。シャーディングは垂直方向の拡張と組み合わせることもできます。InterSystems IRIS では、ユーザの作業負荷のパフォーマンス課題に応じて、適切な全体拡張ソリューションを作成できます。

2 シャーディングの仕組み

シャードのアーキテクチャの中心部は、データとそれに関連するキャッシュを複数のシステム間で分割することです。シャード・クラスタは、データ・ノードと呼ばれる複数の InterSystems IRIS インスタンス間で大規模なデータベース・テーブルを水平方向に、つまり行ごとに分割すると同時に、これらのインスタンスのいずれかを介してアプリケーションがこれらのテーブルにアクセスできるようにします。クラスタの分割 (シャード化) されたデータの各データ・ノードの割り当て分をシャードと呼びます。このアーキテクチャは以下に示す 3 つの利点をもたらします。

- 並列処理

  クエリはデータ・ノード上で並列で実行され、結果は結合されて、完全なクエリ結果としてアプリケーションに返されるので、多くの場合、実行速度が大幅に向上します。

- 分割されたキャッシュ

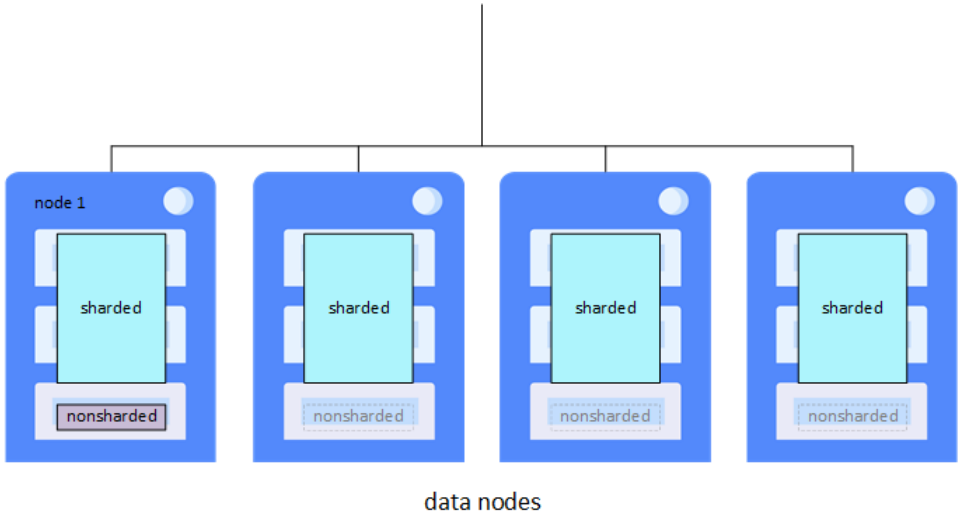
  1 つのインスタンスのキャッシュがデータ・セット全体を提供するのではなく、各データ・ノードに専用のキャッシュが存在します。これにより、キャッシュのオーバーフローやディスク読み取りのパフォーマンス低下のリスクが大幅に軽減されます。

- 並列ロード

  データをデータ・ノードに並列でロードできるので、取り込みの作業負荷とクエリの作業負荷との間でキャッシュとディスクの競合が減少し、双方のパフォーマンスが向上します。

シャーディング・マネージャと呼ばれるフェデレートされたソフトウェア・コンポーネントが、データ・ノードにあるデータを追跡し、それに応じてクエリを転送します。シャード化されていないデータは、データ・ノード 1 (コードとメタデータも格納します) と呼ばれる最初に構成されたデータ・ノードに格納されます。アプリケーション SQL から見ると、シャード・テーブルとシャード化されていないテーブルの区別は完全に透過的です。

図 1: 基本のシャード・クラスタ



3 体験 : InterSystems IRIS シャード・クラスタの導入とデモ

この体験では、以下の操作を行います。

- InterSystems Cloud Manager (ICM) を使用して、基本のシャード・クラスタをパブリック・クラウドに導入します。

ICM は自動化されたコマンド行ツールで、これにより、クラウドにインフラストラクチャを簡単にプロビジョニングし、プロビジョニングされたノードで InterSystems IRIS とその他のサービスの導入が簡単になります。必要な InterSystems IRIS の構成を ICM がすべて行うので、シャード・クラスタを指定した場合、導入完了時にクラスタが使用できる状態になります。

重要　InterSystems IRIS リリース 2023.2 以降、ICM は非推奨になっています。ICM は今後のリリースで削除されます。

- 異なるシャード・キーを使用して同じデータから 3 つのシャード・テーブルを作成し、データ・ノード上のシャード間にわたる行のさまざまな分散を確認します。

  シャード・キーは、シャード・テーブルのどの行がどのシャードに格納されるのかを決定するために使用するフィールドです。既定では、RowID がシャード・キーとして使用されます。この分散は、データの均等分散を最適に保証し、最も効率的な並列データ・ロードを可能にするため、ほとんどのシャード・テーブルに対して最も効果的なアプローチになります。ただし、ユーザ定義のキーが有利になる事例が 1 つあります。クエリに結合される頻度の高い 2 つの大規模なテーブルを、結合に使用するフィールドでシャードした場合、結合される行は同じシャードに格納され、結合をシャード全体ではなく、各シャードでローカルに実行できるので、大幅にパフォーマンスが向上します。これは、コシャード結合と呼ばれています。

- 3 つのシャード・テーブルすべてで同じクエリを実行して、シャード全体の行の分散がクエリに対して完全に透過的であることを示します。

詳細にとらわれずにシャーディングを紹介するために、この例はシンプルなものにしてあります。プロダクションのシャード・クラスタには計画と多くの意思決定が必要なので、ここに示すシャーディングの操作例と本番での操作を混同しないようにしてください。例えば、プロダクションのクラスタを設計する場合、シャード・データの想定作業セットとサーバのデータベース・キャッシュに使用できるメモリ容量の両方に基づいて、シャードするテーブルを決定して、導入するデータ・ノードの数 (通常は 4 ～ 16 台程度) を決定するために、スキーマとテーブルを見直す必要があります。ただし、この体験では、2 つのデータ・ノードによる基本のクラスタを 1 つ導入します。このドキュメントの最後に示すソースでは、シャーディングをプロダクション環境で使用するために必要な良策が提供されています。"スケーラビリティ・ガイド" の "シャーディングによるデータ量に応じた水平方向の拡張" の章では、シャーディングとシャード・クラスタについて詳しく説明しています。

以下の手順では、ユーザが InterSystems IRIS シャード・ライセンスを持ち、インターシステムズ・ソフトウェアのダウンロードにアクセスできることを前提としています。

3.1 ICM を使用してクラスタを導入

ICM を使用してシャード・クラスタを Amazon Web Services パブリック・クラウド・プラットフォームに導入する手順は、"InterSystems IRIS デモ :InterSystems Cloud Manager" に示されています。特に、"体験 : ICM によるクラウドへの InterSystems IRIS の導入" というセクションを参照してください。全体的な手順は、そこに示されたとおりに使用できます。definitions.json をカスタマイズする場合、各 LicenseKey フィールドの値が、defaults.json ファイルの LicenseDir フィールドで指定されているステージング場所にある InterSystems IRIS シャード・ライセンスのファイル名になっていることを確認します。導入ステージ (InterSystems IRIS の導入) の完了時には、ICM コンテナ内にそのまま残ります。

注釈　"スケーラビリティ・ガイド" の "シャード・クラスタの導入" で説明しているように、InterSystems IRIS インスタンスを既存の物理マシン、仮想マシン、またはクラウド・マシンにインストールし、シャーディング API または管理ポータルを使用してシャード・クラスタを導入することも、シャード・クラスタの自動導入方法のいずれかを使用することもできます。

3.2 異なるシャード・キーを使用して行をさまざまに分散して、シャード・テーブルにクエリを実行

使用するシャード・キーに基づいてデータ・ノード上のシャード間でシャード・テーブルをシャード・クラスタにより分割する方法を確認するには、クラスタに接続し、以下の手順を実行します。

- 小さなシャード化されていないテーブルを作成して生成します。
- このシャード化されていないテーブルと同じフィールドを持つ 3 つのシャード・テーブルを、3 つの異なるシャード・キーを使用して作成します。
- シャード化されていないテーブルからシャード・テーブルを生成します。
- シャード・テーブルの行を選択して、それらの行が各事例ごとに異なる方法でシャード (データ・ノード) 間で分散される仕組みを確認します。

  テーブルをシャードすると、シャードごとに RowID 値がさまざまな範囲から割り当てられます (最初は広く離れた範囲から)。同じ範囲の行が同じシャードになります。この例のように小さいテーブルを処理する場合、シャード RowID の識別は容易なので、それぞれのシャードにどの行が割り当てられているのかは明確に示されます (ただし、シャード同士を識別することはできません)。RowID でシャードされるテーブルの行は、行に格納されたデータとは何の関係もない方法で分散されるので、テーブルを空にして再ロードすると、別の方法で分散される可能性があります。一方で、ユーザ定義のシャード・キーは、キーのフィールドの値に基づいて行を分散するので、テーブルをいったん空にしてから再ロードしても同じ方法で分散されます (シャードの値と数が変更されていないという前提です)。
- 3 つのテーブルすべてに、いずれかのフィールドの最大長に対するクエリを実行して、シャード全体の行の分散がクエリに対して透過的であることを示します。

この部分の体験では、以下の手順を使用します。

1. どちらかのデータ・ノード・インスタンスでターミナル・ウィンドウを開きます。

   - ICM コマンド行で、いずれかのデータ・ノードの名前を指定するための -machine オプション、およびクラスタ・ネームスペース **IRISCLUSTER** (異なる場合は既定値ファイルの Namespace フィールドで指定したネームスペース) を指定するための -namespace オプションを使用して、icm session コマンドを発行します。ノードの名前が不明な場合は、icm inventory コマンドを使用してノード名を表示できます。

     ```
     $ 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
     ```

     ノード (どちらのノードでもかまいません) を選択し、以下のコマンドを発行します。

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

     icm session コマンドにより、指定したノードの InterSystems IRIS インスタンスでターミナル・ウィンドウが開きます。

   - %SYSTEM.Cluster.Sharding API を使用してクラスタを手動で導入した場合は、"InterSystems IRIS の基礎 : IDE の接続" で説明している該当のインスタンスについての手順を使用して、どちらかのインスタンスでターミナ

ル・ウィンドウを開き、IRISCLUSTER ネームスペース (異なる場合はクラスタの初期化時に指定したネームスペース) に切り替えます。

```
Node: Acme-DATA-TEST-0002, Instance: IRIS

USER>set $namespace="IRISCLUSTER"

IRISCLUSTER>
```

2. ターミナル SQL シェルを開きます。

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

[SQL]IRISCLUSTER>>
```

3. 以下の SQL 文を使用して、シャード化されていないテーブル test.nonsharded を作成して生成します。

```
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')
```

SELECT を使用してテーブルの内容を表示します。

```
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. test.nonsharded と同じフィールドを使用して 3 つのシャード・テーブルを作成します。このとき、最初のフィールドには既定のシャード・キー (RowID)、2 つ目には field1 フィールド、3 つ目には field2 フィールドを使用します。さらに INSERT INTO 文を使用して test.nonsharded から各フィールドを選択して生成します。

```
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. SELECT *,%ID を使用して、各シャード・テーブルの内容とシャード上の RowID を表示します。

```
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
```

この分散には field1 や field2 の値は反映されていません (それぞれの 3 つの値すべてを含む行が両方のシャードに配置されています)。test.rowid を削除し、再作成して、再ロードすると、分散は異なる状態になる場合があります。

```
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
```

field1 フィールドのシャーディングが行を分散し、field1 と同じ値を持つ行が同じシャードに配置されるようになります。この例では、値が one の行が 1 つのシャード上にあり、値が two および three の行が別のシャード上にありますが、最終的にどの値がどのシャードに配置されるのかは、存在するシャードの数と値の数によって異なります。

```
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
```

ここでは、field2 フィールドの値によって分散が決定されます。

6. 最後に、シャーディングによってデータ・ノードに作業を分散する方法の例として、3 つのシャード・テーブルすべてに対して以下の SELECT 文を使用します。

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

各例の結果は 5 (最も長い値 three の長さ) になります。シャード全体の行の分散はクエリに対して完全に透過的だからです。MAX(LENGTH(field2)) 式はシャードごとに別々に計算され、シャーディング・マネージャはこれらが返す結果の MAX() を選択します。例えば、クエリを test.field2 テーブルで実行する場合、1 つのシャードは 3 を返します。field2 フィールドに one という値しか存在しないからです。一方、別のシャードは 5 を返します。このときシャーディング・マネージャは、この 2 つの結果の大きい方の 5 を選択します。

必要に応じて、EXPLAIN を使用して、作業がシャードに送信される方法を明示的に示すクエリ・プランを示します。

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

4 シャード・クラスタのその他のオプション

シャード・クラスタには、以下のような追加のオプションがあります。

- いつでもデータ・ノードを追加して、既存のシャード・データをデータ・ノードの拡張セット全体で再分散できます。再分散をクエリや更新と同時に実行することはできません。したがって、再分散を実行できるのは、シャード・クラスタがオフラインで、他のシャード操作が使用できない場合のみです (″スケーラビリティ・ガイド″ の ″データ・ノードの追加とデータの再分散″ を参照してください)。

- クラスタ上のデータに対して高可用性を追加するには、ミラーリングされるフェイルオーバー・ペアとしてデータ・ノードを導入できます。(″スケーラビリティ・ガイド″ の ″ミラーによる高可用性″)
- 常に大量のデータが取り込まれるような状況でも、クエリの遅延がきわめて小さいことが求められる高度な使用事例では、計算ノードを追加して、クエリを処理するための透過的なキャッシュ層を提供できます。クラスタに計算ノードが含まれている場合は、データ・ノード上ではなく、その計算ノード上で読み取り専用クエリが自動的に並行して実行されます。データ・ノードではすべての書き込み操作 (挿入、更新、削除、および DDL 操作) が引き続き実行されます。この作業分担により、クエリとデータ取り込みの作業負荷が分離される一方で、並列処理や分散キャッシュの利点を維持し、これら両方のパフォーマンスを向上させます (″スケーラビリティ・ガイド″ の ″作業負荷の分離とクエリ・スループットの向上のための計算ノードの導入″)。

5 シャーディングの詳細情報

シャーディングの詳細は、以下を参照してください。

- Data Platform Scalability Technology Overview (ビデオ)
- Introduction to Sharding (オンライン・コース)
- Sharding Basics (オンライン・コース)
- Deploying InterSystems IRIS in Containers and the Cloud (学習パス)
- スケーラビリティ・ガイド
- InterSystems IRIS デモ : InterSystems Cloud Manager