

## 記事

[Toshihiko Minamoto](#) · 2021年5月31日 3m read

## DeepSee: データベース、ネームスペース、マッピング (4/5)

以下の記事では、DeepSee のより柔軟なアーキテクチャ設計の概要を説明します。 [前の例](#) で説明したとおり、この実装には、DeepSee キャッシュや DeepSee の実装と設定、および同期グローバル用の個別のデータベースが含まれています。この例では、[DeepSee インデックス](#) を保存するための新しいデータベースを紹介します。DeepSee インデックスがファクトテーブルや次元テーブルとともにマッピングされないように、グローバルマッピングを再定義します。



### 例 3: 完全に柔軟なアーキテクチャ

#### データベース

APP-FACT データベースはファクトテーブルと次元テーブルしか保存しないのに対し、アナリティクスサーバーは、中間的な例で前に定義したデータベースに加え、インデックスを保存する APP-INDEX データベースを導入します。インデックスをファクトテーブルから分離するのは、インデックスのサイズが大き

くなる可能性があるため、パフォーマンスを向上させるために APP-FACT のブロックサイズを変更することができるからです。

[前の例](#)のように、ファクトテーブルとインデックスのジャーナリングはオプションで有効にできます。詳細については、[前の記事の注意事項](#)をお読みください。

## グローバルマッピング

次のスクリーンショットは、上記の実装例のマッピングを示しています。 ^DeepSee.Index グローバルのマッピングは、新たに作成された APP-INDEX データベースに保存されるように変更されています。中間の例と同様に、^DeepSee.Fact\* と ^DeepSee.Dimension\* グローバルのマッピングは引き続き、ファクトテーブルと次元テーブルを APP-FACT データベースに保存するために使用されています。クエリログと最後の MDX クエリは、オプションで DeepSee キャッシュとともに保存されます。

## コメント

このアーキテクチャの例は最も高い柔軟性を備えていますが、ネームスペースごとに 5 つのデータベースを作成する必要があります。[2 つ目の例](#)のように、DeepSee キャッシュはジャーナリングが無効になっている専用のデータベースにマッピングされており、同期グローバルは APP-DSTIME にマッピングされています。

ファクトテーブルとインデックスをマッピングすると、DeepSee の実装と設定をジャーナリングされる専用のデータベース ( APP-DEEPSEE ) に保存できるため、DeepSee 実装の復元を簡単に行えるようになります。多くの場合、対応するグローバルをファクトテーブルと共に APP-FACT に保存するだけで充分であるため、インデックス用のデータベースを個別に作成するのはオプションです。

この連載の最後の記事には、3 つの例で使用したデータベースの要約とリストを記載します。

[#マッピング](#) [#チュートリアル](#) [#デプロイ](#) [#分析](#) [#初心者](#) [#InterSystems IRIS BI \(DeepSee\)](#)

---

### ソースURL:

<https://jp.community.intersystems.com/post/deepsee-%E3%83%87%E3%83%BC%E3%82%BF%E3%83%99%E3%83%BC%E3%82%B9%E3%80%81%E3%83%8D%E3%83%BC%E3%83%A0%E3%82%B9%E3%83%9A%E3%83%BC%E3%82%B9%E3%80%81%E3%83%9E%E3%83%83%E3%83%94%E3%83%B3%E3%82%B0%E3%83%8845%E3%83%89>