JAMP会員　各位

Aras Innovator パフォーマンスチューニング事例紹介

富士フイルムにおける事例　2026年版

2026年7月24日

富士フイルムソフトウエア株式会社

第二開発本部
ネットワークソリューショングループ／平尾　義隆

JAMP

FUJIFILM
Value from Innovation

Agenda

- インフラ・データベースから見たAras Innovatorの特徴
- 富士フイルムにおけるAras活用の歴史とデータ量遷移
- 富士フイルムのサーバー及び実施済みのプラットフォームの設定について
- パフォーマンスチューニング事例
- インシデント対応事例
- 運用保守事例
- 今後の方針

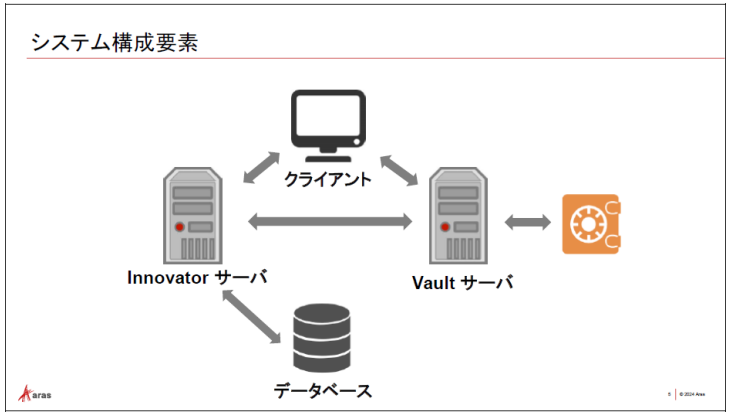
インフラ・データベースから見たAras Innovatorの特徴

Windows Server + IIS + SQL Serverで構成されるトラディショナルな三層構造。

Aras Innovatorのシステム構成要素

Administration1 Student Guideより

Microsoft .NET Core プラットフォームに基づくインターネットアプリケーションフレームワーク

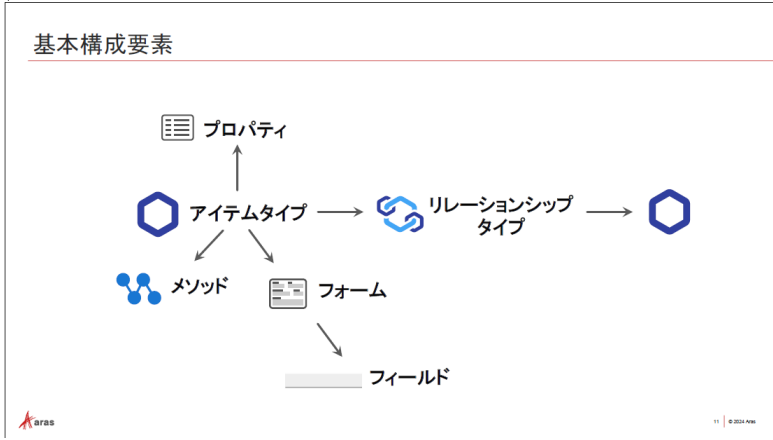


構成要素	内容
クライアント	Web ベースのブラウザインターフェースを使用しているため、クライアントリソースはほとんど必要ありません。 ブラウザは、Microsoft Edge、Mozilla Firefox またはGoogle Chrome が利用可能です。
Innovator サーバ	Microsoft IIS 上のMicrosoft .NET Core アプリケーションとして実行します。Microsoft Windows Server プラットフォームが必要です。
データベース	**全て**の構成ルール、コードならびにビジネスデータは、Microsoft SQL Server データベースに格納されます。
Vault サーバ	Vault サーバは、SQL Server データベース内のデータにリンクされたファイルの実体を管理します。
Vault	物理ファイルを格納するためにVault サーバに認識されるファイルディレクトリです。

Aras Innovator フレームワークの基本構成要素→テーブル名が分かり易い。

Administration1 Student Guideより

・（ほぼ）No Codeでビジネスアプリケーションを作るためのフレームワークが提供されている。（一部抜粋）



要素名	内容	データベースのテーブル名
アイテムタイプ	アイテムタイプは、Aras Innovator 上で管理するデータ（＝アイテム）の構造や振る舞い定義する、言わばテンプレートのようなものです。各アイテムタイプには、データベース内にそれぞれのデータを管理するためのテーブルが用意されます。 例） パーツ、ドキュメント、プロジェクト	＜Aras標準＞ CAD CAD_INSTANCE … ＜FF＞ HELI_IT_DOC0001
プロパティ	プロパティは、アイテムの属性情報です。各プロパティには、名称やラベル、データタイプ等、様々な設定を行うことができます。プロパティは、アイテムタイプで用意されるテーブルのカラムに対応します。 例） パーツ番号、パーツ名称、有効日	<Aras標準・共通> PROPERTY
リレーションシップタイプ	リレーションシップタイプは、アイテムタイプ間の関連（親子関係、参照関係、など）を定義したものです。 例） パーツの親子関係（部品表）、パーツ→ドキュメントの参照関係	＜FF＞ HELI_RT_DOC0001_DOC HELI_RT_DOC_FILE
フォーム	Aras Innovator 上の各データ（＝アイテム）は、アイテムタイプに設定したフォームを通して画面に表示されます。フォームは、ドラッグ＆ドロップ等を駆使して直感的に構成することが可能です。	<FF> HELI_FM_DOC0001
フィールド	フィールドは、フォーム上の GUI 部品です。アイテムのプロパティに対応します（プロパティと独立したフィールドを設けることも可能）。	<Aras標準・共通> FIELD
メソッド	メソッドは、固有のビジネスロジックを用意する場合などで使用される、プログラムコードを収めたものです。各メソッドを部品のようにアイテムタイプやフォームなどに設定することで、標準の挙動をカスタマイズすることが可能です。 プログラミング言語は、サーバメソッドには C#、VB.NET、クライアントサイドには JavaScript を用います。	<Aras標準・共通> METHOD

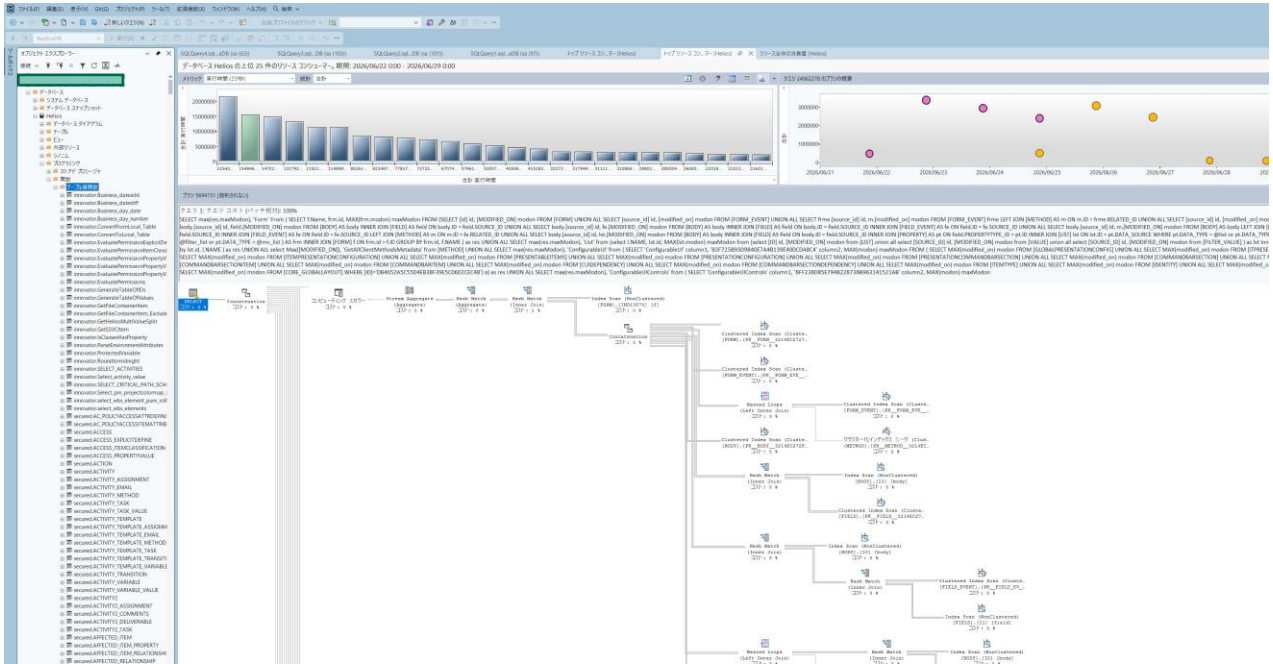
SQLクエリは**膨大な数**が実行されている・・・

・ユーザーがカスタマイズしたメソッドよりも、圧倒的にAras Innovatorが発行するSQLが多い。

・テーブル変数を多用され、大量に実行されるが短時間（0.0089ms等）で完了。
例：INSERT INTO @IDS (id) VALUES (@id);
IF NOT EXISTS(SELECT id FROM @IDS WHERE id=@id);

・平均651,890,948回／日のSQLが実行されている。

・テーブル値関数多数あり。



Aras Innovatorが発行するSQLの特徴

•画面構成はすべてシステム管理テーブルに入っており、動的に画面を作成してクライアントに返す方式を採用しているため、パスワード変更等をきっかけに再取得する等、ＳＱＬ実行件数が多い傾向となる。

•教科書通りのテーブル正規化が実施されている。
アイテムタイプ（画面）ごとにテーブルを複数生成するため、非常にテーブル数が多く、結合時の負荷が大きい。

•Arasシステムが動的生成しているSQLは基本的に**主キーを必ず選択している**。
（故にテーブルスキャンというパターンは少ない）
ただし、Index scanが多く全件取得操作が多いため処理時間を要する。
ArasシステムのWhere句指定が主キーのみで従属する列を
組み合わせた非クラスタ化インデックス定義を積極的に行っていない。

•アドホッククエリ（where ID = ‘12345’のような変数でないもの）**がほぼない**。
→ 実行プランは使い回している。
故に実行計画作成はスキップされ、キャッシュを使いまわしている。

Aras Innovatorアップグレードに伴いSQL実行数増加

富士フイルム事例では、V12 SP12からV14 Release31にアップグレードした際、
ユーザーがログイン完了してメニュー表示するまでのSQL実行数は以下の通り。

TOCアクセス権が199ある、ユーザーの例。

・1回目の比較だとV12と比較して**6.8倍**となっている。

・2回目（再度ログイン、キャッシュクリアなし）R31はほぼ同じSQL実行数。

同時アクセス時に応答待ちが発生したため、プリロードのスキップにより対処。

ログイン回数	V12 SP12	V14 Release31	プリロードスキップ
1回目	10,162	69,183	6,819
2回目	344	69,217	**723**

Aras Innovatorの**機能強化によりSQL実行数の増加傾向は続く（はず）。**

富士フイルムにおけるAras活用の歴史とデータ量遷移

2017年4月から順調に成長してます。

富士フイルムAras Innovator活用の歴史

2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026
【Aras】 V11 SP8 、Minervaテンプレート、全文検索(CBES)	4月運用開始！			【Aras】3月 V11SP8→**V11 SP15** へアップグレード	【Aras】7月 V11SP15→**V12 SP12** へアップグレード＋全文検索(Enterprise Search)切替え				【Aras】5月 V12SP12→**V14 Release31**へアップグレード	
【PF】 ・弊社DC仮想基盤 Windows Server 2012R2 / SQL Server 2012 standard edition							【PF】’23年12月下旬 ・弊社DC→Microsoft Azure移設 Windows Server 2019/SQL Server 2019 Standard Edition ・Client PC 8GB→16GB＆Windows11一斉切り替え			
【QMS機能】 QMS基本機能実装	【QMS機能】 基本機能追加対応	【QMS機能】 国内関係会社のQMSインフラ利用開始	【QMS機能】 出荷情報、国内関係会社対応 【性能対策】 DBメモリ増設	QMS機能追加、仕様変更対応等			【性能対策】 DBベストプラクティスに沿ったCPU強化＆サーバー統合等	【性能対策】 4～5月DTAを使って統計情報、インデックス追加	【性能対策】 MyInBasket/MyTask高速化等	【性能対策】 全文検索起動高速化/ファイル内検索解禁

’23年4月～
平尾担当

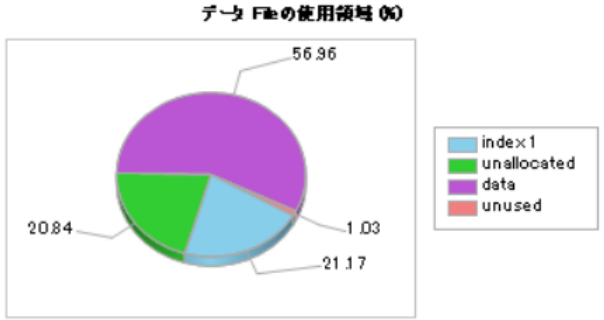
データベースのサイズ状況

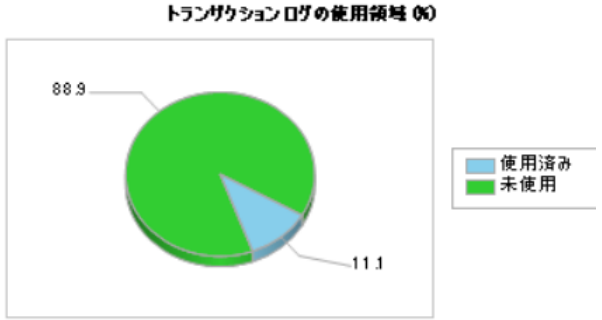
・DBサイズは10GBから開始、’23/12で98GB。Microsoft Azure移設時に設計見直しを行い、約340GBとした。（自動拡張による応答性能劣化を予防中）

・テーブル数は3,576。

予約済みの領域合計	342.48 GB
データ Fileの予約済み領域	248,288.00 MB
トランザクション ログの予約済み領域	102,407.31 MB

計測日	全テーブル レコード合計
2024/4/20	92,520,890
2026/4/8	153,140,879
2026/7/2	158,299,147





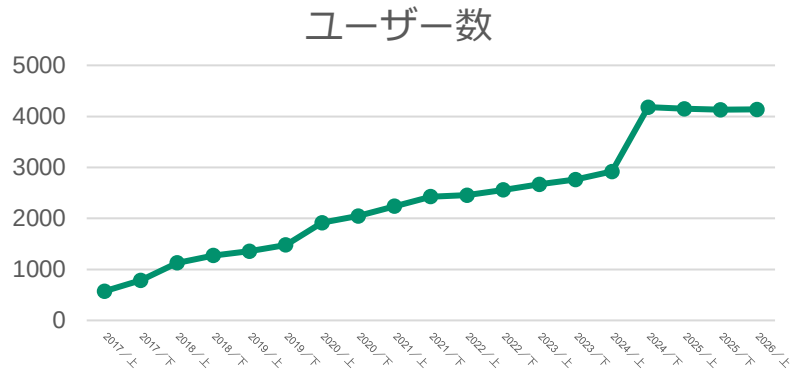
直近3年でレコード数は**1.7倍**！

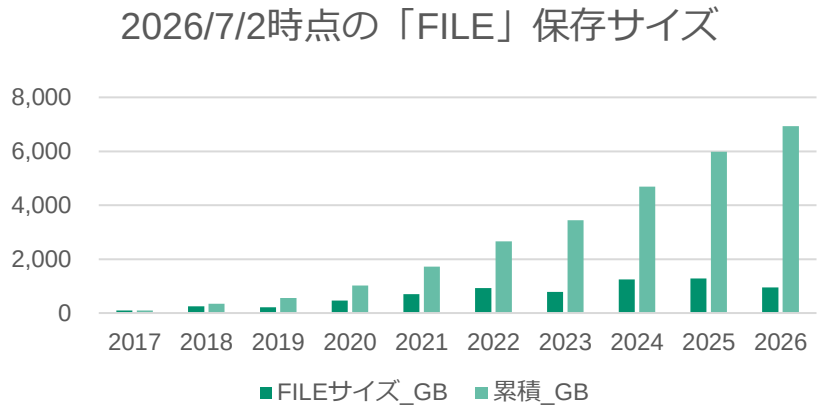
利用状況／規模

・利用者数は4,000人超、
国内外、富士フイルム関係会社にて利用

・ユーザーが添付するファイルサイズ合計　6,860GB(6.7TB)…

利用拡大に伴う大規模化
→性能対策必須





富士フイルムのサーバー
及び実施済みのプラットフォームの設定について

マイクロソフト、Aras社提供のベストプラクティスをぜひチェック！

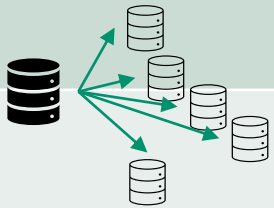
富士フイルムのAras向けサーバーについて

Application、Batch(ETL含む)、DB、File、Solr(全文検索)、Crawler(全文検索)の**6台** Microsoft Azureにて構築

	【Production server】	Application(Aras Innovator)	Application2(For batch processing/ETL)	Database(SQLServer)	Vault(File Server)	Solr(ES)	Crawlar(ES)
	VM Size	◆Instance Series：Dldsv5／Instance ：D16lds v5 (16Core、32GBのRAM、600GBのTemp Storage／Clock frequency 2.8GHz (3.5GHz Turbo))	◆Instance Series：Ddsv5／Instance ：D8ds v5 (8Core、32GBのRAM、300GBのTemp Storage／Clock frequency 2.8GHz (3.5GHz Turbo))	◆Instance Series：Ebdsv5／Instance ：E16bds v5 (NVMe) (16Core、128GBのRAM、600GBのTemp Storage／Clock frequency 2.45GHz (3.5GHz Turbo))	◆Instance Series：Ddsv5／Instance ：D8ds v5 (8Core、32GBのRAM、300GBのTemp Storage／Clock frequency 2.8GHz (3.5GHz Turbo))	◆Instance Series：Edsv5／Instance ：E16ds v5 (16Core、128GBのRAM、600GBのTemp Storage／Clock frequency 2.8GHz (3.5GHz Turbo))	◆Instance Series：Edsv5／Instance ：E8ds v5 (8Core、64GBのRAM、400GBのTemp Storage／Clock frequency 2.8GHz (3.5GHz Turbo))
Drive layout	Volume1 C/Windows	P10 − Premium SSD 128GB	P10 − Premium SSD 128GB	P10 − Premium SSD 128GB	P10 − Premium SSD 128GB	P10 − Premium SSD 128GB	P10 − Premium SSD 128GB
	Volume2 D/Temp or TempDB	Temporary Storage	Temporary Storage	Temporary Storage	Temporary Storage	Temporary Storage	Temporary Storage
	Volume3 E/Apps	P15 − Premium SSD 256GB	P20 − Premium SSD 512GB	P10 − Premium SSD 128GB	P70 − Premium SSD 16TB	P40 − Premium SSD 2TB	P20 − Premium SSD 256GB
	Volume4 F/Backup	E:Apps 80GB F:Backup 80GB G:Data 96GB(E/Fドライブ割り当て後の残り全容量)	E:Apps 100GB F:Backup 100GB G:Data 312GB(E/Fドライブ割り当て後の残り全容量)	P20 − Premium SSD 512GB	E:Apps 100GB F:Backup 100GB G:Data (E/Fドライブ割り当て後の残り全容量)	E:Apps 300GB F:Backup 100GB G:Data (E/Fドライブ割り当て後の残り全容量)	E:Apps ~~100GB~~→**220GB** F:Backup ~~50GB~~→**30GB** G:Data ~~106GB~~→**5.98GB** (E/Fドライブ割り当て後の残り全容量) 2023/12/25 File Crawler処理時にEドライブをかなり消費するため、サイズ分配を変更
	Volume5 G/data or Log			Premium SSD v2 512GB **			
	Volume6 H/data1	None	None	Premium SSD v2 256GB *	None	None	None
	Volume7 I/data2	None	None	Premium SSD v2 128GB *	None	None	None
	Volume8 J/data3	None	None	Premium SSD v2 128GB *	None	None	None
	Volume9 K/data4	None	None	Premium SSD v2 256GB *	None	None	None
				*15,000 IOPS/1,000MB/S **40,000 IOPS/1,000MB/S			

ポイント①　データベースの特性を踏まえた施策

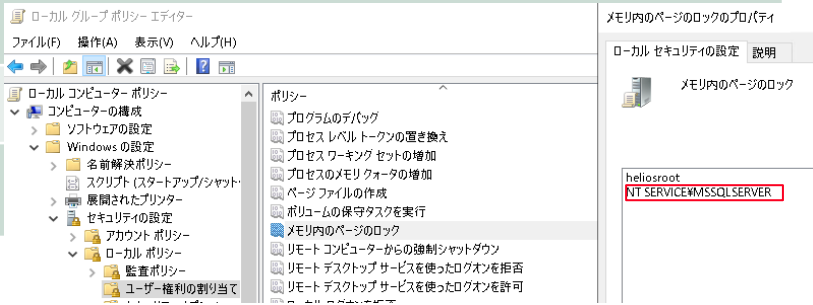
No.	施策	目的・効果
1	**データファイル分割**	読み書き時の複数ドライブ同時アクセスを実現可能となる。 →**1桁遅い**ストレージでも**最大４倍**の並列処理性能向上ができる。 Appendix①　参照
2	トランザクションログを **専用ドライブに割り当て**	トランザクションログは先行書き込みログであるため、 変更内容書き込み完了後、データ変更処理を行われる。 →データファイルと独立した専用ドライブにすることで 処理性能向上ができる。
3	tempDBを可能な限り**高速なドライブ**に割り当てかつ、８の倍数に合わせて分割	Aras innovatorはtempDBへの書き込み量が多い。 大規模なソート実施時や、Insert into @IDS(id) VALUES(@id)等実行時には、 tempdbへの書き込みを都度発生。 →**応答性能が非常に高いストレージへの配置**が有利。**★必須です！** ※Ebdsv5は、一時ストレージとして高速なローカル SSD ストレージ がVMに直結のため、tempDBを割り当てした。 Enterprise Edition使っている場合は、完全メモリ化も選択肢になる。Standard Editionでは利用出来無い。
4	**64kb**アロケーションユニットで ストレージを**フォーマット**	DBからテーブルへアクセスするとき、8kb X 8個の**64kb**、1エクステントという単位で処理している。 →**DBのアクセス単位とストレージアロケーションサイズを同一**にすることでOS標準フォーマットと比較すると**16倍のIO処理効率化**を実現。 ※OSの規定フォーマットサイズは4kb、故に１エクステントアクセスするのに**16回**ディスクへアクセスが必要！無駄です！！



●マイクロソフトではベストプラクティスがあります！あまりにも量が多いので厳選した内容をピックアップしました●

ポイント②　Microsoft Azure／Windows の特性を踏まえた施策

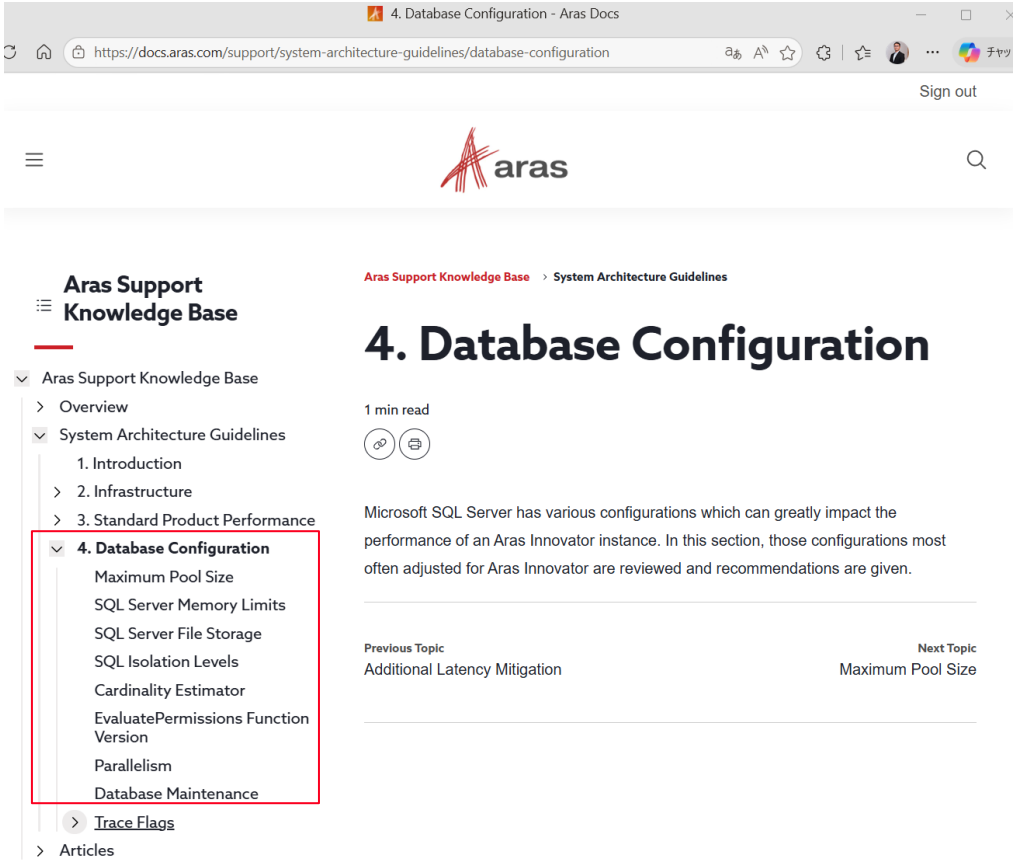
No.	施策	目的・効果
1	VM選定	高速なローカルストレージ付きを原則としてApp、Vault等の特性に応じて最新世代（第５世代）のVMを選定した。
2	ストレージ選定	MSが提供するストレージの測定ツールである「diskspd.exe」にて移行元と移行先のストレージ性能を計測 →この結果を元にAzureストレージのIOPSを決定。 ※VMには上限IOPSがある。このため、高速なストレージを多数接続してもVMの上限IOPSを超えて使うことができないので要注意。
3	ホストキャッシュ有効化	PremiumSSD、StandardSSDではVMが持つホストキャッシュを読み取り／書き込みを有効化を実施。 →繰り返す読み書きに効果あり。 ※UltraDisk、PremiumSSD v2は仕様上、設定不可。
4	一時ストレージへOSの仮想メモリを配置	通常のストレージよりもVM直結により高速処理が期待できる。特に大規模なI/Oとは分離したドライブへの配置が良い　→Azure temporary DriveのIOPSは実測値「76,486」でかなり高速。
5	Windows OS設定	**DBのみ**【Lock Pages in Memory】SQL Serverアカウントに対してポリシーを有効 →仮想メモリ（ディスク）へのデータのページングを防止。 ★必須！メモリ空いているのに応答が突然スローダウンした経験ありませんか？ 本設定にて**物理メモリ強制割り当て**できます。 OracleやPostgreSQLも同じく推奨！
6		【電源プラン】を「高パフォーマンス」に設定 →常にCPUの処理速度を高速にする。
7	IIS設定	アプリケーションプール →Aras Innovator AppPool ASP.NET Core→詳細設定 (全般)→キューの長さ　1,000→10,000



Aras社のSupport KBについて

[Aras Support Knowledge Base - Aras Docs]
こちらの[4. Database Configuration - Aras Docs]から引用しました。

他にも有用な情報があるので、一度見てみてください。



ポイント③　Aras 社のSupportKB情報について(1/2)

SupportKB 4 Database Configuration	内容	富士フイルム適用状況	備考
Maximum Pool Size	IIS→DBへの同時接続数を計測してArasから100を超えるユーザーがアクティブになるケースが想定されるシステムでは、InnovatorServerConfig.xmlにて「Maximum Pool size」で定義できる。	**未適用(保留)** →富士フイルム環境では平時40～60 今後負荷が増える場合は適用可否を判断する。	DBコネクト数が同時に100※を超える状況が続く場合、解放まで待ちが発生するので、その対策には有効。 ※大規模サイト
SQL Server Memory Limits	サーバーの最大メモリは、SQL Server で使用可能なメモリの合計から、オペレーティング システムおよびメモリを必要とするその他のプロセスに必要なメモリを差し引いた値に設定すること。	**既知** DBサーバーの物理メモリは128GB 最大メモリサイズは100GBとしている。	設定必須。 環境によるがもう少し最大メモリを増やしてもいいかなと思うがOS向けに７%は残したい。
SQL Server File Storage	最低でも以下の各ファイルセットは別々のソリッドステートドライブに保存する必要あり（合計 4 台の SSD）。 オペレーティングシステム(C:◆) Innovator データベースデータファイル (mdf) Innovator データベースのログファイル（ldf） SQL Server システムデータベースファイル tempDB →8つ分割は必要。	**既知** 「ポイント①　データベースの特性を踏まえた施策」 ・データファイルは4つのドライブに分散 ・ログファイルは高速なドライブに割り当て ・tempdbはVM直結のDドライブ	推奨の分割は必須。 かつ、物理的に分離できているドライブに配置が望ましい。 Innovatorデータベースファイル自体の分割はどこまでこだわるかは規模による。（富士フイルムはやや過剰かも・・・）
SQL Isolation Levels	同時使用率が高いデータベースの場合、 **Read Committed Snapshot** を有効にして分離レベルを変更することをお勧め。	**既知**	インストールデフォルトは READ COMMITTED (READ_COMMITTED_SNAPSHOT=OFF)なので変更が望ましい。
Cardinality Estimator	データベースに送信するクエリの最適な実行プランを決定するためにSQL Serverが使用するカーディナリティ エスティメーターは、システムパフォーマンスに最も大きな影響を与えるものの1つ	**既知** SQL Version : 2019 Compatibility Level : 150 Legacy Cardinality Estimator : OFF Evaluate Permissions V2 : 次項参照	組み合わせは以下の通り。

ポイント③　Aras 社のSupportKB情報について(2/2)

SupportKB 4 Databese Configuration	内容	富士フイルム適用状況	備考
Cardinality Estimator →Evaluate Permissions Version **2**	Aras Innovatorにおけるアクセス権限は、セキュアな関数（各ItemTypeごとに作成されるSQL Serverのテーブル値関数）によって管理されます。これらの関数は、InnovatorのACメカニズム（RBAC（ルールベースアクセス制御）、MAC（必須アクセス制御）、DAC（ドメインアクセス制御））によって宣言的に定義されたアクセス制御（AC）ロジックを実装しています。したがって、セキュアな関数によって実行されるAC計算のパフォーマンスは、Aras Innovatorインスタンスのパフォーマンスに大きな影響を与える要因となります。RBACはAras Innovatorにおける主要なACメカニズムです。そのロジック（指定された権限、識別子、チームなどによって定義される）は、EvaluatePermissions SQL Serverテーブル値関数にカプセル化されています。したがって、この関数の実装はパフォーマンス処理において重要な役割を果たします。Aras Innovator リリース 21 以降、EvaluatePermissions 関数の代替バージョンである EvaluatePermissions バージョン 2（v2）が利用可能になりました。ご注意ください、このバージョンは SQL Server バージョン 2019 以降（CE レベル 150）が必要です。Legacy Cardinality Estimator オプションとは互換性がありません。EvaluatePermissions v2 の目的は、特定の複雑なアクセス制御の実装における AC 計算のパフォーマンスを向上させることです。v2 関数は、大規模な Identity 構造、多数の Permissions/Access Items、または多数の Team Memberships がある場合に、最も一般的にパフォーマンスが向上します。ただし、よりシンプルな RBAC の実装では、EvaluatePermissions v2 関数のパフォーマンスが低下する可能性があります。Arasは、Aras InnovatorのインスタンスでEvaluatePermissions v2をテストし、最も重要なエンドユーザーシナリオでv2関数がより良いパフォーマンスを示すかどうかを確認することを推奨します。両方のバージョンの関数は有効であり、どちらが実装に最適かは顧客の判断に委ねられます。	**適用**	ArasのSQLを見るとselect count(*) が頻繁に実行されているがAras社のBill TurnerさんがArasは「計算」をするという表現をしてたので、AC計算が該当するかもしれない。 カプセル化するという表現があるが、テーブル値関数を使用してロジックをカプセル化することで、再利用性や可読性を向上させることを目的にしている。 CREATE FUNCTION dbo.GetActiveUsers() RETURNS TABLE AS RETURN (SELECT UserID, UserName, IsActive FROM Users WHERE IsActive = 1); ここまでがカプセル化 これを使って実行が以下の例 SELECT * FROM dbo.GetActiveUsers(); 注意！ View等にヒント句を入れるとNGです。 以下が表示され、実行できないケースがあります。 
Parallelism	並列化のコストしきい値については、デフォルトの値である5は非常に低く、多くのクエリがすでに高速に実行されているため、恩恵を受けないかもしれない並列化を利用することになります。また、必要以上のスレッドがサーバに占有され、並列スレッドが完了した後に再参加するまでの待ち時間が長くなります。これは、CXPACKETの待ち時間が非常に長くなることでよく見られます。並列処理が最も有用な複雑なクエリでのみ使用されるように、この値を100にすることを推奨	**既知**「並列処理の最大限度」8 **適用**「並列処理のコストしきい値」5 →100評価中	富士フイルム環境ではコストしきい値のみ変更して様子見。

ポイント①、②、③をぜひ御社の環境と比較してください！

- Aras InnovatorもSQL Serverもデフォルトである程度動くが故、見逃すポイント！
  特にＯＳ設定については、設定するだけなので、**リスク小、効果大**です！

- Arasカスタマイズ担当とサーバー構築担当が異なると**設計・設定が抜けてしまうケースが多いです。**レコード新規作成、更新に性能課題がある場合は、サーバー能力（特にストレージ応答速度）が支配的なので特に重要。

- データベースサーバーの**データファイル分散配置は効果大**です。
  超高速ストレージ採用されている場合、富士フイルムのように4つのドライブに分散配置は不要。
  ただ、論理分割はおすすめ。参照：Appendix①

- サーバー更新予定の方は、JAMPにて実施したサーバースペックアンケート(こちら)をぜひご確認ください。Aras社にサーバースペックの見積依頼すると非常に高いスペック（＝高額）が提案される。（安全側に倒している？）
  サーバースペックアンケートを参考に調整することをお勧め。
  →ぜひ、**皆様のサーバースペック、JAMP事務局または平尾までお知らせくださいm(__)m**

パフォーマンスチューニング事例

環境を整える（DB周り）ケースと外科的手法（アプリ改修）するケース

パフォーマンスチューニング事例紹介

No	分類	課題	対策	対策前	対策後
1	Index追加	My In Basketを開くのに時間がかかる。	・連結対象7つのテーブルのうち、3つ「フィルター条件付き付加列インデックス」を適用 ・無効ユーザー数の削減 →Aras Innovator 管理者向けSQL応答性能チューニングのあれこれ ※現在、joinのヒント句は削除している。	約100秒	約25秒
2	不要データ削除	PDF生成キューのチェックに要するCPU実行積算時間が多い。	・スプール管理のログテーブルを不要分削除（25万件） ・月1回の定期削除を運用に取り組み	21ms	測定限界以下
3	Index追加	SQL実行時間がやたら長く、インデックス化が完了しない。	・[FILE]テーブルに対するSQLを分析、「フィルター条件付き付加列インデックス」を適用 →数千回繰り返されるので効果は高い。	約2.6秒	約0.5秒
4	ロジック改修	ユーザーに回付された案件一覧を表示する機能（MyTask）を実装しているが、履歴が多数あるケースでは開くのに数分かかってしまう。	・クライアントとDBのキャッチボールをやめるために、DB内に「仮想表」を作成し、1つのSQLで実行するようにロジックを変更	258秒	2.84秒
5		全文検索「Enterprise Search」を初回起動すると2分以上応答が帰ってこないので使いにくい。	・全文検索起動時の実行されるSQLをすべて解析し、明らかに不要な繰り返し処理を特定、Aras社に修正要求実施。Patch-077749.01.zip（ArasES.dll）を適用	96秒	12.4秒
6		ディスカッション機能について応答が遅いので使いづらい。	・更新履歴も含めて表示しようとするが、History全件検索を都度実施するので遅すぎる。	60秒	数秒

No.3～6の事例をご紹介します。

パフォーマンスチューニング事例　No.3 (1/3)　【課題／原因】

【課題】SQL実行時間がやたら長く、インデックス化が完了しない。

【原因】全文検索の「File」クローニング（更新されたファイルを検出し、読込、インデックス化）にて大量検索するが、デフォルトではAras社はインデックスを用意していないので、非常に重い処理となっていた。

SQL実行内容を確認

SQL①　WITH org_query AS (
SELECT TOP (CAST(@u_offset_14 as BIGINT) + @u_fetch_15) [FILE].[ID] AS [ID],
[FILE].keyed_name AS [ID_$_KEYED_NAME],
@FILE_id_12 AS [ID_$_TYPE],
@$intCanGet AS id_$_permission,
ROW_NUMBER() over (ORDER BY [FILE].[FILENAME], [FILE].id) as [$sort_order],
ROW_NUMBER() over (ORDER BY [FILE].[FILENAME], [FILE].id) as [$row_number]
FROM [secured].[FILE](@$t_perm_tp_1, @$t_iden_list_2, @$t_identity_list_id_3, @$t_cur_user_id_4, @perms_having_can_get10, @$t_env_attrs_6)

WHERE (([FILE].[INDEXED_ON] IS NOT NULL) /*WhereClauseHelper marker*/)
 ORDER BY ROW_NUMBER() over (ORDER BY [FILE].[FILENAME], [FILE].id)
)
SELECT *
FROM org_query
WHERE [$row_number] > @u_offset_14
ORDER BY [$sort_order]

OPTION(RECOMPILE)

SQL②　SELECT COUNT(*)
FROM (SELECT [FILE].ID
    FROM [secured].[FILE](@$t_perm_tp_1, @$t_iden_list_2, @$t_identity_list_id_3, @$t_cur_user_id_4, @perms_having_can_get16, @$t_env_attrs_6)
 WHERE (([FILE].[INDEXED_ON] IS NOT NULL) /*WhereClauseHelper marker*/)) [FILE]

SQL③　SELECT COUNT(*)
FROM (SELECT [FILE].ID
    FROM [secured].[FILE](@$t_pwarn_5, @$t_iden_list_2, @$t_identity_list_id_3, @$t_cur_user_id_4, @perms_having_can_get17, @$t_env_attrs_6)
 WHERE (([FILE].[INDEXED_ON] IS NOT NULL) /*WhereClauseHelper marker*/)) [FILE]

目的は異なるがいずれもinnovator.[FILE]へ「INDEXED_ON」（全文検索時のインデックス付与処理実行日時）が「入っているもの」をすべて取得したというSQL

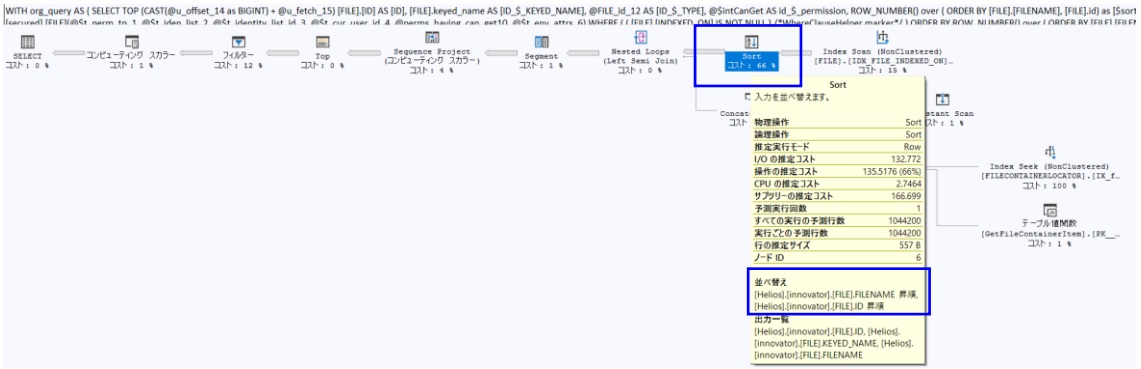
※SQL②と③は引数違いで実質同じ

INDEXED_ONがあるものを取得しろという命令だが、INDEXED_ONの日付自体は取得対象にしていない。
しかも「FILENAME」と「ID」でソート実行している。
※ソート処理はCPU負荷高い！

パフォーマンスチューニング事例　No.3 (2/3)　【解析】

SQL	無対策 推定cost	DBが考える不足インデックス候補	DB推奨インデックス追加
SQL①　WITH org_query AS (SELECT TOP (CAST(@u_offset_14 as BIGINT) + @u_fetch_15) [FILE].[ID] AS [ID], [FILE].keyed_name AS [ID_$_KEYED_NAME], @FILE_id_12 AS [ID_$_TYPE], @$intCanGet AS id_$_permission, ROW_NUMBER() over (ORDER BY [FILE].[FILENAME], [FILE].id) as [$sort_order], ROW_NUMBER() over (　ORDER BY [FILE].[FILENAME], [FILE].id) as [$row_number] FROM [secured].[FILE](@$t_perm_tp_1, @$t_iden_list_2, @$t_identity_list_id_3, @$t_cur_user_id_4, @perms_having_can_get10, @$t_env_attrs_6) WHERE (([FILE].[INDEXED_ON] IS NOT NULL　) /*WhereClauseHelper marker*/) ORDER BY ROW_NUMBER() over (　ORDER BY [FILE].[FILENAME], [FILE].id)) SELECT * FROM org_query WHERE [$row_number] > @u_offset_14 ORDER BY [$sort_order] OPTION(RECOMPILE)	108.721 120.689(全体)	CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [innovator].[FILE] ([INDEXED_ON]) INCLUDE ([KEYED_NAME],[FILENAME])	31.0128 205.777(全体)
SQL②　SELECT COUNT(*) FROM (SELECT　[FILE].ID 　　FROM [secured].[FILE](@$t_perm_tp_1, @$t_iden_list_2, @$t_identity_list_id_3, @$t_cur_user_id_4, @perms_having_can_get16, @$t_env_attrs_6) 　WHERE (([FILE].[INDEXED_ON] IS NOT NULL　) /*WhereClauseHelper marker*/)) [FILE]	108.686 111.536(全体)	CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [innovator].[FILE] ([INDEXED_ON]) INCLUDE ([TEAM_ID],[CREATED_BY_ID],[MANAGED_BY_ID],[OWNED_BY_ID],[PERMISSION_ID])	32.2909 39.8544(全体)

DB推奨インデックスだとSQL①が遅くなる結果となった。
原因は104万件に対してFILENAMEとIDでソートするため。
↓
・インデックス項目であらかじめソートしておけばいい。
・Indexed_onの存在確認しか、検索条件にない。
↓
SQL①、②をまとめてしまえば、一石二鳥では？



パフォーマンスチューニング事例　No.3 (3/3)　【結果】

SQL	default 推定cost	DB推奨インデックス追加　推定cost	2つのインデックスを統合 推定cost
SQL①　WITH org_query AS (SELECT TOP (CAST(@u_offset_14 as BIGINT) + @u_fetch_15) [FILE].[ID] AS [ID], [FILE].keyed_name AS [ID_$_KEYED_NAME], @FILE_id_12 AS [ID_$_TYPE], @$intCanGet AS id_$_permission, ROW_NUMBER() over (ORDER BY [FILE].[FILENAME], [FILE].id) as [$sort_order], ROW_NUMBER() over (　ORDER BY [FILE].[FILENAME], [FILE].id) as [$row_number] FROM [secured].[FILE](@$t_perm_tp_1, @$t_iden_list_2, @$t_identity_list_id_3, @$t_cur_user_id_4, @perms_having_can_get10, @$t_env_attrs_6) WHERE (([FILE].[INDEXED_ON] IS NOT NULL　) /*WhereClauseHelper marker*/) ORDER BY ROW_NUMBER() over (　ORDER BY [FILE].[FILENAME], [FILE].id)) SELECT * FROM org_query WHERE [$row_number] > @u_offset_14 ORDER BY [$sort_order] OPTION(RECOMPILE)	108.721 120.689(全体)	31.0128 205.777(全体)	10.3298 13.2577(全体)
SQL②　SELECT COUNT(*) FROM (SELECT　[FILE].ID 　FROM [secured].[FILE](@$t_perm_tp_1, @$t_iden_list_2, @$t_identity_list_id_3, @$t_cur_user_id_4, @perms_having_can_get16, @$t_env_attrs_6) WHERE (([FILE].[INDEXED_ON] IS NOT NULL　) /*WhereClauseHelper marker*/)) [FILE]	108.686 111.536(全体)	32.2909 39.8544(全体)	30.9439 39.8544(全体)

SQL①は 2,599ms → 537ms
SQL②は1,356 ms → 891ms

たいへんよくできました

技

★作成したインデックス

```
CREATE NONCLUSTERED INDEX [IDX_FILE_INDEXED_ON_3] ON [innovator].[FILE]
(
[FILENAME] ASC,[ID] ASC, [INDEXED_ON] ASC
)
INCLUDE([MODIFIED_ON],[KEYED_NAME],[TEAM_ID],[CREATED_BY_ID],[MANAGED_BY_ID],[OWNED_BY_ID],[PERMISSION_ID])
WHERE ([INDEXED_ON] IS NOT NULL)
 ON [HeliosIdx2]
```

FILENAMEとIDを項目にすると、昇順で作られるため、Sort処理は回避できる

付加列を指定することでインデックスのみで検索が完結

インデックスを一つに集約することで**更新時の負荷軽減しつつ、検索速度も担保！**
DTAや生成AIの提案は個別最適化となるので、**全体最適化を意識し管理者が決断すべし。**

インデックスを使うか否かの判断は「FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY」の優先順位がある。
インデックス作成対象をWhere句付き（フィルター）とすることで不要分が対象外となる。

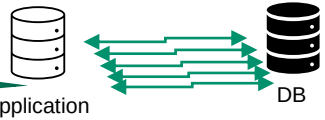
パフォーマンスチューニング事例　No.4

【課題】ユーザーに回付された案件一覧を表示する機能（MyTask）を実装しているが、履歴が多数あるケースでは開くのに数分かかってしまう。

【原因】

SQLで取得した一覧の一項目はIDなので、さらにループさせて個別にマスタからレコード取得していた。(都度DBへ接続)

手続き型処理に慣れているプログラマーはSQLでやれることを知らない典型例

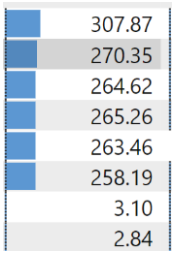


【対策】

SQL単体ではできないので、「一時テーブル」という機能を使って、DBのSQLで**仮想テーブル**を作成、そこにSQLで取得した一覧を書き出し、DBカーソルにてループさせてSQLでマスタからレコード取得する方式に切り替え。

【結果】

約100倍高速化！



DBに閉じて検索実行することで1回に集約

技

```
CREATE TABLE #TempResults
(
 z_item_label NVARCHAR(32) ,z_doc_name NVERCHAR(32),z_type_name
NVARCHAR(32),z_id NVERCHAR(32),z_is_current NVARCHAR(32)　– is_curren列を追加
);
-- 最初のクエリで一時テーブルにデータを取得
INSERT INTO #TempResults (z_act_label,z_act_id)
SELECT DISTINCT
  act.[type_name] as ‘z_type_name’,act.[id] AS ‘z_id’
FROM innovator.[ テーブルA ] AS act

-- 各テーブルから動的にz_is_currentを取得して更新
DECLARE @UpdateSql NVARCHAR(MAX) = N'';
SELECT @UpdateSql = @UpdateSql + N'
    UPDATE t
    SET
        t.z_item_label = temp_data.keyed_name,
        t.z_doc_name = temp_data.z_doc_name,
        t.z_is_current = temp_data.IS_CURRENT
    FROM #TempResults t
    INNER JOIN (
        SELECT id, keyed_name, z_doc_name, IS_CURRENT
        FROM innovator.' + QUOTENAME(z_type_name) + '
    ) AS temp_data
    ON t.z_item_id = temp_data.id
    WHERE t.z_type_name = ''' + z_type_name + ''';'
FROM (SELECT DISTINCT z_type_name FROM #TempResults) AS t;

-- 動的SQLを実行
EXEC sp_executesql @UpdateSql;-- 結果を`IS_CURRENT`が1のみにフィルタリングして表示
SELECT * FROM #TempResults WHERE z_is_current = '1';
-- 一時テーブルを削除
DROP TABLE #TempResults;
```

DBのTempDBに**中間テーブル**を作成できるので、そこにいったん書き出す。

中間テーブル「TempResults」の「z_type_name」は各アイテムタイプ名をもとに各テーブルから情報を取得し、中間テーブルに反映する。

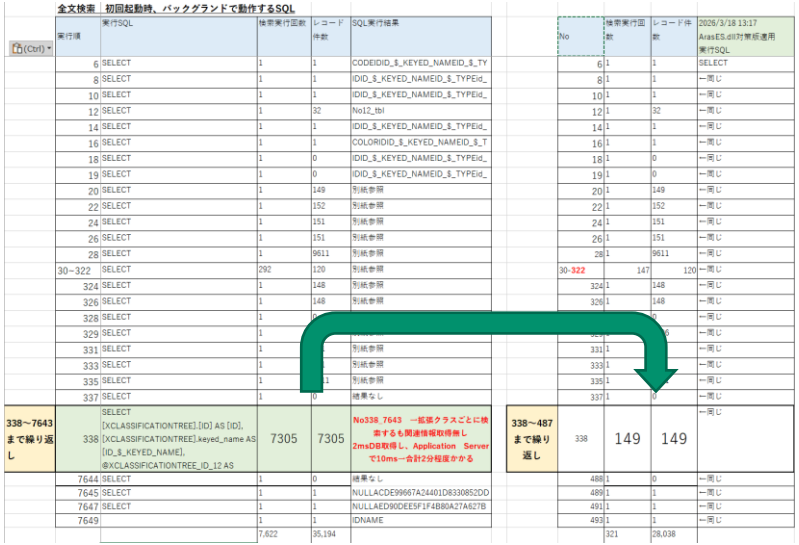
結果を返して中間テーブル削除

パフォーマンスチューニング事例 No.5

【課題】全文検索「Enterprise Search」を初回起動すると**2分以上応答が帰ってこない**ので使いにくい。

【原因】SQL実行回数が必要以上に多い。

【解析】初回起動相当の動作を実行、SQLプロファイラーでSQLを全て取得。
→実行されているSQLを仕分け、個別実行して結果を確認。
「XCLASSIFICATIONTREE_ITEMTYPE」へ7,305回アクセス。
DB応答は2ms程度だか、Innovator Serverにて10ms処理。
約12ms X 7,305回≒87秒→**ここが疑わしいとAras社へ連絡。**



全文検索　初回起動時、バックグランドで動作するSQL

	実行順	実行SQL	検索実行回数	レコード件数	SQL実行結果
	6	SELECT	1	1	CODEIDID_$_KEYED_NAMEID_$_TY
	8	SELECT	1	1	IDID_$_KEYED_NAMEID_$_TYPEid_
	10	SELECT	1	1	IDID_$_KEYED_NAMEID_$_TYPEid_
	12	SELECT	1	32	No12_tbl
	14	SELECT	1	1	IDID_$_KEYED_NAMEID_$_TYPEid_
	16	SELECT	1	1	COLORIDID_$_KEYED_NAMEID_$_T
	18	SELECT	1	0	IDID_$_KEYED_NAMEID_$_TYPEid_
	19	SELECT	1	0	IDID_$_KEYED_NAMEID_$_TYPEid_
	20	SELECT	1	149	引続参照
	22	SELECT	1	152	引続参照
	24	SELECT	1	151	引続参照
	26	SELECT	1	151	引続参照
	28	SELECT	1	9611	引続参照
	30～322	SELECT	292	120	引続参照
	324	SELECT	1	148	引続参照
	326	SELECT	1	148	引続参照
	328	SELECT	1	0	引続参照
	329	SELECT	1	0	引続参照
	331	SELECT	1	1	引続参照
	333	SELECT	1	1	引続参照
	335	SELECT	1	11	引続参照
	337	SELECT	1	0	結果なし
338～7643まで繰り返し	338	SELECT [XCLASSIFICATIONTREE].[ID] AS [ID], [XCLASSIFICATIONTREE].keyed_name AS [ID_$_KEYED_NAME], @XCLASSIFICATIONTREE_ID_12 AS	7305	7305	No338_7643　一拡張クラスごとに検索するも関連情報取得無し 2msDB取得し、Application Serverで10ms→合計2分程度かかる
	7644	SELECT	1	0	結果なし
	7645	SELECT	1	1	NULLACDE99667A24401D8330852DD
	7647	SELECT	1	1	NULLAED90DEE5F1F4B80A27A627B
	7649		1	1	IDNAME
			7,622	35,194	

	No	検索実行回数	レコード件数	2026/3/18 13:17 AravES.dll対策版適用 実行SQL
	6	1	1	SELECT
	8	1	1	←同じ
	10	1	1	←同じ
	12	1	32	←同じ
	14	1	1	←同じ
	16	1	1	←同じ
	18	1	0	←同じ
	19	1	0	←同じ
	20	1	149	←同じ
	22	1	152	←同じ
	24	1	151	←同じ
	26	1	151	←同じ
	28	1	9611	←同じ
	30-322	147	120	←同じ
	324	1	148	←同じ
	326	1	148	←同じ
	328	1	0	←同じ
	329	1	0	←同じ
	331	1	1	←同じ
	333	1	1	←同じ
	335	1	11	←同じ
	337	1	0	←同じ
338～487まで繰り返し	338	149	149	←同じ
	488	1	0	←同じ
	489	1	1	←同じ
	491	1	1	←同じ
	493	1	1	←同じ
		321	28,038	

【対策】Aras社からパッチリリース(AravES.dll)
XPath/AML のキャッシュ処理を追加（した模様）
12.4秒で起動。時間短縮成功！
クローニング時にも**AravES.dll**が実行されているので、
初回起動以外でも高速化に寄与している。

【Aras改善リストに登録済み】
I-076203 "Search at the top of the main menu takes a very long time"

Besides searches from the top search text box and crawling, the patch also affects:
1. Facet properties settings –
GetFacetProperties() calls InitSearchConfigurations() when opening facet configuration in the admin UI
2. IndexedType template validation –
ValidateUpdatingIndexedTypeTemplates() calls InitSearchConfigurations() when saving title/subtitle/additional_info templates on ES_IndexedType
3. ES_SetPropertyData –
SetPropertyData() was changed separately (XPath/loadAML caching). This affects the indexed property grid on the ES_IndexedType form only and has no impact on crawling or search.

パフォーマンスチューニング事例 No.6

★JAMPメンバからのノウハウ共有事例　感謝！

【課題】ディスカッション機能について応答が遅いので使いづらい。

【原因】VIEW [innovator].[SECUREMESSAGE]は
[innovator].[STOREDSECUREMESSAGE]（テーブル）と
[innovator].[HISTORYSECUREMESSAGE] （ View ）を2つを単純結合している。
→[innovator].**[HISTORYSECUREMESSAGE]** は
「HISTORY」「HISTORY_CONTAINER」「ITEMTYPE」3つのテーブル結合しており、
HISTORYは非常にレコード数(2,844万件)が多いため、レコード取得に時間を要する。

【対策】ディスカッションにて「過去履歴」を見る必要がない場合、重いSQLをスキップできる。innovator.[HISTORYSECUREMESSAGE] as hsmを削除して高速化実現。

【注意】Aras社提供のViewなので、Aras Innovatorのアップグレード時に
初期化されるので、忘れずに対処が必要。

ノウハウ/事例を共有しませんか？　**性能向上でストレス軽減！**

ユーザーの生産性向上に寄与します！

三人寄れば
文殊の知恵

データベースチューニングアドバイザーによる効果事例

技

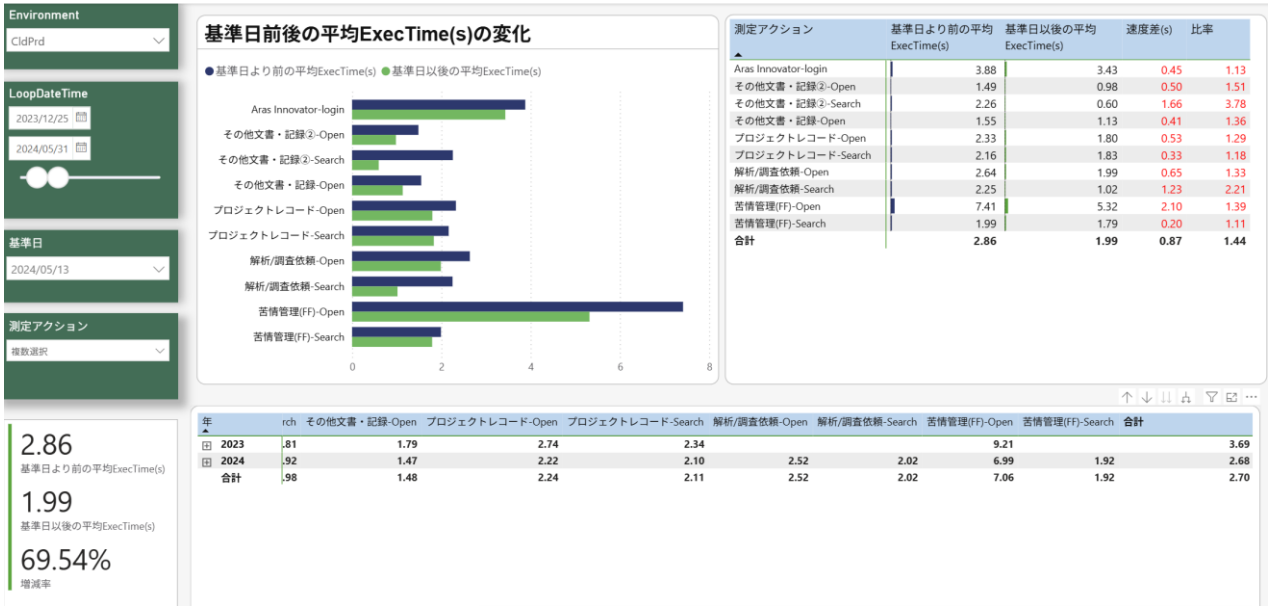
MSツールお任せで高速化事例については以下の通り。

2023/12/25～2024/5/28 【Aras Innovator V12 SP12】のインデックス追加事例

【対策】Aras Innovator 管理者向けSQL応答性能チューニングのあれこれ　P21～P22

DTAツールによるインデックス／統計情報追加（2024/5/13前後比較）→**31%改善**

MyInBasketやMyTaskといった重い処理を含めると239%改善



パフォーマンスチューニング事例のまとめ

- 応答が数分近くかかって戻るような性能課題については、インデックス追加だけでは解決しない可能性がある。
  ロジックの問題ではないか？という観点で点検しないと解決は難しい。Aras社実装起因かも？

ロジック改修＞インデックス追加＞不要データ削除

「ロジック改修」No.4,5,6
「インデックス追加」No.1,3
「不要データ削除」No.2

- **DBのメモリ不足は致命的。メモリから仮想メモリに書き込みが発生すると劇的に遅くなる！**
  例：検索時に大量データをソート実行時、tempDBに書き込みが発生すると間違いなく遅くなる。実行計画にて
  temoDBへの書き込みを実施しているかは確認が可能。応答時間がかかるSQLの実行計画を確認することをお勧め。

- **闇雲にインデックス追加という一本足打法はやめましょう。**
  今回はDB中心に事例紹介だが、もしかするとクライアントPCのメモリ不足かもしれないし、ネットワーク帯域の
  問題（VPNアクセスとか）かも。。。

インシデント対応事例

失敗から学ぶ。

インシデント事例　リソース不足起因

クラウド移設時の想定との差異で発生したリソース不足起因のインシデントと対策

No	インシデント	原因(気づき)	対策
1	朝始業時に「テーブルの更新ができない」というエラーが発生。	トランザクションログを100GBとし、2倍の256GBストレージを確保していたが、No.1に関連して大量のトランザクションがあると深夜のバックアップ時にシュリンクできず、空き領域を食いつぶしていた。	暫定対策：復旧モデル「完全」から「単純」に変更した後、頃合いを見て圧縮。 恒久対策：トランザクションログの格納ストレージを512GBに拡張
2	DBバックアップが失敗	利用者が急増した関係でDBファイルの拡大が想定より進行し、空き領域が不足。 移送時のバックアップ削除漏れも重なった。	恒久対策： ストレージを十分余力がある１TBに拡大。 かつ、不要なバックアップは削除運用を徹底

インシデント事例　操作起因

ユーザー操作起因のインシデントと対策

No	インシデント	原因(気づき)	対策
1	Arasにログインできない、 スピナーが回り続ける（30分でタイムアウト）	**管理者権限**を有するユーザーにて TOCメニュー 　アドミニストレータ> ファイルハンドリング> **ファイル** 右クリックメニュー 　ナビゲート> **逆展開** →innovator.FILEに関連するテーブル全検索が始まり、DBが高負荷になってしまう。	暫定対策：SSMS→利用状況モニタで高負荷SQLのセッションIDを特定、kill セッションIDを実行 恒久対策：操作禁止
2	朝始業時間は問題なかったが、一斉ログインによる高負荷状態が発生。ログイン待ちとなった。4時間程度で自然解消。	クラウド移設後に、初ログインしたユーザーが多数だったため、**大量の更新が実行され、トランザクション処理が間に合わなくなった。（消去法にて原因推定）** 個々の検索で致命的な検索処理はなく、当初解析は難航したが、トランザクションログが100GB→230GBまで肥大化したことをヒントにUser関連の更新処理が多数発生したことを確認。 ①2024年1月16日 10:00～10:10　同時ログイン：約260人 　→インシデント発生 ②2026年6月04日 10:10～10:20　同時ログイン：約150人 　→対策後、インシデント発生せず。	データファイル、トランザクションログを独立ドライブに分散配置していたが、IOPSは共通の**15,000**としていた。トランザクションログの書き込み待ちを打開する必要がある。 ※**tempDBのレスポンスも怪しい**がインシデント発生時平尾不在で詳細不明。 恒久対策： ①トランザクションログのストレージについてIOPSを15,000→**40,000**に高速化を行った。 ②運用回避として同時アクセスが発生しないように分散するように使用方法を変更。

「同時アクセス人数」が多いと想定外のインシデントが発生する可能性があるため、
同時接続（想定される最大数相当）でログイン等の操作テストを実施することを推奨する。

運用保守事例

インシデント発生を予防するために。

健康状態のチェック

運用コスト低減しつつ、早期対応を目指す。

No	監視対象	チェック内容／自動実行内容	アクション
1	サーバーのリソース	CPU使用率、メモリ使用率、ストレージ空き容量をあらかじめ設定した閾値を超えたときにメールにて管理者へ通知	【発生時のみ】 CPU/メモリなら、APM（Jennifer）等にて状況確認 ストレージなら、サーバーにログインし、不要ファイル削除、もしくはストレージ拡張実施。
2	サービス稼働	IIS／SQL Serverサービスをポーリングし、停止している場合、メールにて管理者へ通知	【発生時のみ】 速やかに再起動等のアクション実施
3	応答性能	専用PCに、PowerAutomateにてAras Innovatorを操作し、操作ログをShare Pointへアップロードする仕組みをつくり、PowerBIにて分析するツールを作成。	【定期監視】 週に1回程度、トレンドを確認。 極端に遅くなる状況であれば、個別対処。
4	ETLの実施ログ	外部システムとのデータ連携は原則処理結果をメールにて管理者に通知	【定期監視】 運用管理者により、目視で確認。失敗があれば、手順に従い対処
5	APMによるログ監視	Jenniferを導入、Aras InnovatorのWebサーバーを監視 IISリスタート、CPU使用率上昇時	【発生時のみ】 内容確認し、個別対処

SQL Serverにおけるメンテナンス

・メンテナンスプランにて自動実行

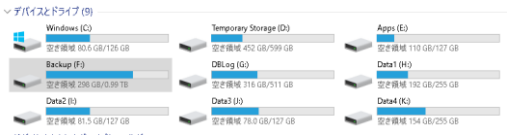
No	メンテナンスプラン	実行内容	備考
1	DBバックアップ	毎晩23時実行、5分程度で完了、 圧縮設定で約63GB。7世代保持	日々のバックアップ ＆　検証環境への横展開（不定期）
2	インデックスの再構成	土曜日を除く毎日、4時実行、20分以内に完了。 断片化率を10%超えたものが対象。	リーフの再構成のみ
3	インデックスの再構築	土曜日5時実行、再構築対象次第、30分以内に完了。 断片化率を20%超えたものが対象。	再構築なので実質作り直し
4	統計の更新	毎日21時50分に実行、55分以内に完了。 フルスキャン＆すべての既存の統計を対象	**更新を定期実行しないとSQL Serverのオプティマイザは正しく判断できなくなる！** テーブル、インデックス追加直後の統計情報に基づき実行計画を作成するため。

・SSMS（Microsoft SQL Server Management Studio)を使ったクエリストアで「リソース全体の消費量」を日々確認

Appendix⑨参照

・週1回リモートアクセスしてサーバーのドライブ空き状態を目視確認

<u>意外と大事です。アラートメールは閾値超えないと飛ばない！傾向を把握しましょう。</u>



不要データ・ファイルの棚卸

肥大化防止のため、以下実施。

No	件名	内容	メンテナンス内容
1	PDF生成キュー削除	パフォーマンス事例紹介No.2 「PDF生成キューのチェックに要するCPU実行積算時間が多い。」 PDF生成キューをテーブル管理しているが、証跡用PDFを必要とするアイテムタイプが対象となる。ＰＤＦ生成後、キュー管理は不要になるが、削除が想定されていなかった。	不要となった「PDF生成完了」のレコードを削除するSQLを月1回の定期メンテナンスにて実施。
2	**孤立ファイル削除**	添付ファイル保存を数回実施すると、確定版以外は**親アイテムと紐づいていない「FILE」アイテムが保存回数分、保存されたままとなる。** →添付ファイルを一度登録、添付したファイルを誤記修正等して、再度保存すると、最初に保存したものは親アイテムと紐づかない。 富士フイルムでは、今年4月までに合計、 **436.9GB（Vault の6.1%を占有）**となっていた。	親アイテムに紐づかないFILEを検索するSQLを作成及びAMLでFILEの削除を実行 四半期に一度程度実施する予定。
3	ユーザー棚卸	Aras InnovatorのTierは「User」の有効ユーザー数となっている。Tier超えを防ぐ＆セキュリティ観点で棚卸しが必要。	半期に一度、アクセス履歴を確認し、アクセスがない場合は無効化を行う。
4	インデックス棚卸	DTA等で一気にインデックス追加し、個別対応でインデックスを追加するとオプティマイザーの判断で使われないインデックスが発生→更新負荷及びリソース無駄となる。	不定期にチェックし、無効化もしくは削除を実施。

今後の方針

生成AIも活用しつつ。。。サーバー更新も見据えて種まき。

今後は・・・

＜富士フイルム＞

- LLMを活用した実行計画分析サポートし、人的負荷の軽減
- サーバーOSのEOLに合わせたHW刷新計画、OS/DB最新化（新しい方が賢い）
- 運用コスト最適化　※HW性能向上にあわせたサーバーのダウンサイジング

＜JAMP＞

- **事例を出し合ってナレッジ化→JAMPサイトやインフラ分科会にて情報共有。**
  大規模サイトの運用管理者の皆様、情報発信いかがですか？

- **Aras Innovator起因の共通課題は具体的な課題（問題個所）を示せばAras社は動きます！**
  ＃皆様、是非ご一緒に！

Appendix

是非ご利用ください。

Appendix①

I/O処理待ち対策→並列処理向上させるアプローチ

■テーブルの規模、利用頻度分析による割り当て設計

1. レコード件数＋サイズのリスト化
2. ドライブを４つに分類、高負荷SQLを分析、テーブルアクセス時にドライブへのI/Oを均等にすべく、ドライブ割り当てを設計

論理名	ストレージ				備考
	H	I	J	K	
Helios（Primary）				3	Aras Innovatorが自動生成するテーブル数
Helios01	7				
Helios02		8			
Helios03			5		
Helios04				2	
Helios05				2	
Helios06			1		
Helios07		478			
Helios08		10			カスタマイズに関連するテーブル数
Helios09				10	
Helios10				10	
Helios11			9		
Helios12	9				
Helios13		8			
Helios14				8	
Helios15		8			
Helios16			8		
Helios17			2286		
Helios18				704	
HeliosIdx1	26				チューニング時に追加したインデックス数
HeliosIdx2		58			
HeliosIdx3			35		
HeliosIdx4				135	

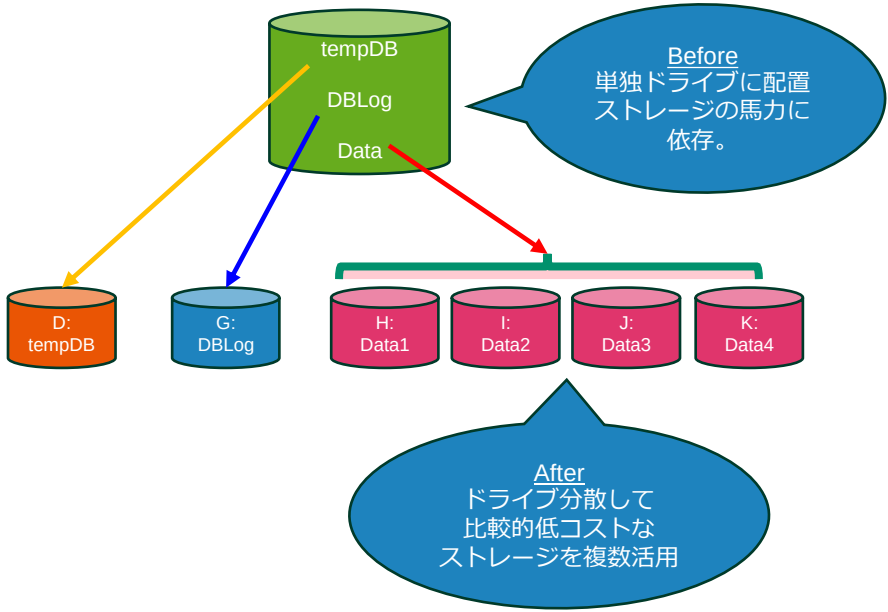
Aras自動生成
Arasシステム管理テーブル。
→HistoryやFile情報等

HELI_…
Arasシステム向けテーブル。
レコード件数・使用頻度を鑑みて配置決定

テーブルとは別のドライブにチューニング用インデックス配置

■SQLServerの構造に準じたドライブ割り当て見直し

1. DBのtempDB、トランザクションログ、データファイルを合計６つのドライブに配置
2. ６年稼働を見越して自動拡張が頻発しないようにサイズを計算し、テーブル作成SQLに反映。



Appendix②

ストレージ応答速度測定について

クラウドストレージの性能指標として**IOPS**と**スループット**の２種ある。

これは相反する関係を持っていて１秒当たりの応答回数を重視すると一度に扱うサイズが小さい方が有利になるし、巨大なアーカイブファイルを扱うならスループットを重視する。

DBでは一度に処理するサイズが小さいのでIOPSをベースに考えるのが良い。
（ファイルサーバーの場合は、IOPSは小さくして、スループット重視でOK）

MS提供の「DiskSpd.exe」を測定ツールとして利用。
diskspd.exe -r -t16 -c1G -b8k -d60 -S -w20 -P G:¥f1.dat G:¥f2.dat G:¥f3.dat G:¥f4.dat
（Gドライブに対して16スレッド、8kbブロックサイズ、60秒テスト実行、20%書き込み、80%読み込み）

出力結果の中でという合計行が出力される。赤文字部分が8kbのサイズで測定した
ストレージのIOPS実測値となる。

```
Total IO
thread |       bytes     |     I/Os     |    MiB/s    | I/O per s |  file
------------------------------------------------------------------------------------------
     0 |      23486464 |        2867 |       0.37 |      47.78 | G:¥f1.dat (1GiB)
 ・ ・ ・
    63 |      23502848 |        2869 |       0.37 |      47.82 | G:¥f4.dat (1GiB)
------------------------------------------------------------------------------------------
total:       1505026048 |   183719 |     23.92 |  3062.00
```

IOPS
新旧比較に活用しましょう。

スループット

Appendix③

ストレージ種別　IOPS実測値

データベースにおいて読込はDBが確保するキャッシュを活用するが、書き込み（insert , update ,delete）時にDBがディスクキャッシュを無効する。（電源断等、異常発生時にデータ損出を最小限にするための設計）
このため、**ストレージのIOPS性能が書き込み応答性能のボトルネックになる。**
以下は、測定実績を参考情報として掲載する。

機種	CPU	ストレージ	IOPS	備考
Dell R360 ＋H755（Raidカード）エントリーサーバー(2026年製)	Intel® Xeon® 6357P	HDD SAS　2.4TB ﾊｰﾄﾞ ﾄﾞﾗｲﾌﾞ SAS FIPS-140 10K 512e 2.5ｲﾝﾁ ﾎｯﾄﾌﾟﾗｸ 10krpm　RAID1（HDD2個）	1,237	昨今のSSDと比較するとHDD SAS 1万回転でもこの程度。
		HDD SAS　2.4TB ﾊｰﾄﾞ ﾄﾞﾗｲﾌﾞ SAS FIPS-140 10K 512e 2.5ｲﾝﾁ ﾎｯﾄﾌﾟﾗｸ 10krpm　RAID1＋0（HDD4個）	2,492	RAID1+0(1つのI/O分割してHDD2個に同時書き込みをさらに2セット) ざっくり2倍のIOPS
HP Z2 workstation (2024年製) ※ワークステーション	14thGen Intel Core™ i7-13700K	SSD 1TB M.2 SamsungMZVL21T0	380,853	昨今のPCに搭載されるSSD Bitlocker有効化→外すともう少し早い
		HDD 16TB　7,200kpm TOSHIBA MN08ACA16T	319	一般的なHDD Bitlocker有効化
PRIMERGY RX2540 M4 (2019年製？)	Xeon Gold 6128　3.40GHz*2	HP 3PAR	225,783	高速な3PAR（7年ぐらい前設置）
Microsoft Azure 弊社Aras InnovatorのDBサーバー	Xeon Platinum 8370C 2.8GHz 16core	P10 - Premium SSD	4,530	P10の最大IOPSは500。バースト機能（30分）にて9倍の速度達成
		Temporary Storage	76,484	VM近傍に配置されるSSD
		Premium SSD v2	15,304	IOPSを15,000に設定。実測値も同じ

チューニング関連SQL

＜合計実行速度ワースト抽出＞

```
SELECT TOP 100
   RANK() OVER(ORDER BY total_elapsed_time DESC) AS 順位
 , total_elapsed_time / 1000 AS [合計実行時間（ミリ秒）] , total_worker_time / 1000 AS [合計CPU時間（ミリ秒）]
 , 100 * total_worker_time / total_elapsed_time AS [CPU時間割合(%)] , execution_count AS [実行回数]
 , total_elapsed_time / 1000 / execution_count AS [平均実行時間（ミリ秒）] , max_elapsed_time / 1000 AS [最大実行時間（ミリ秒）]
 , min_elapsed_time / 1000 AS [最小実行時間（ミリ秒）] , max_worker_time / 1000 AS [最大CPU 時間（ミリ秒）]
 , min_worker_time / 1000 AS [最小CPU 時間（ミリ秒）] , total_logical_reads AS [論理読み込み]
 , total_physical_reads AS [物理読み込み] , creation_time AS [初回実行時刻]
 , last_execution_time AS [最終実行時刻] , sql_handle , plan_handle
 , REPLACE( REPLACE(REPLACE(REPLACE( SUBSTRING(text, (statement_start_offset / 2) + 1,  ((CASE statement_end_offset WHEN -1 THEN DATALENGTH(text)
          ELSE statement_end_offset END - statement_start_offset) / 2) + 1 ), CHAR(13) + CHAR(10),' ' ), CHAR(13), ' ' ), CHAR(10), ' ' ), CHAR(9), ' '   ) AS sqltext, query_plan
FROM
 sys.dm_exec_query_stats qs
  CROSS APPLY sys.dm_exec_sql_text(sql_handle)
  CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle)
ORDER BY total_elapsed_time DESC OPTION(RECOMPILE);
```

結果　メッセージ

	順位	合計実行時間(ミリ秒)	合計CPU時間(ミリ秒)	CPU時間割合(%)	実行回数	平均実行時間(ミリ秒)	最大実行時間(ミリ秒)	最小実行時間(ミリ秒)	最大CPU 時間(ミリ秒)	最小CPU 時間(ミリ秒)	論理読み込み	物理読み込み	初回実行時刻	最終実行時刻	sql_handle	plan_handle	sqltext	query_plan
1	1	8337466	8245611	98	952114441	0	236	0	16	0	1926389551	102	2026-06-24 16:23:16.333	2026-07-03 10:24:47.790	0x03000...	0x05000700B...	INSERT INTO @IDS (id) VALUES (@id)	<ShowPlanXML xmlns="http://schemas.microsoft.com...
2	2	4742457	4658145	98	974509386	0	64	0	9	0	1949018730	72	2026-06-24 16:23:16.333	2026-07-03 10:24:47.790	0x03000...	0x05000700B...	IF NOT EXISTS(SELECT id FROM @IDS WHERE id=@id)	<ShowPlanXML xmlns="http://schemas.microsoft.com...
3	3	4314768	4299249	99	93152	46	131	36	83	36	270180540	15775	2026-06-25 04:00:01.873	2026-07-03 04:11:58.070	0x02000...	0x060007000...	SELECT i.name AS [Name], CAST(i.index_id AS int) AS [ID], ...	<ShowPlanXML xmlns="http://schemas.microsoft.com...
4	4	2114932	2018355	95	24768	85	1770	67	1483	66	4309664	1	2026-06-24 16:15:32.583	2026-07-03 10:24:23.727	0x02000...	0x060004004...	SELECT volume_mount_point, CONVERT(BIGINT...	<ShowPlanXML xmlns="http://schemas.microsoft.com...

♪　絞り込みやSQLtextの検索がやりづらい場合は「テキスト（タブ区切り）」で保存して、Excelでテキストファイルを開く方が見やすいのでお勧め。

Appendix⑤
チューニング関連SQL

＜インデックス使用状況収集＞

```
select
    SCHEMA_NAME(st.schema_id) as schema_name    , OBJECT_NAME(dp.object_id) as object_name    , si.name    , used_page_count    , reserved_page_count    , row_count    , user_seeks    , user_scans
    , user_lookups    , user_updates    , last_user_seek    , last_user_scan    , last_user_lookup
from
    sys.dm_db_partition_stats dp  left join  sys.indexes si  on  si.object_id = dp.object_id
    and
    si.index_id = dp.index_id  inner join sys.tables st on  st.object_id = dp.object_id
    and
    SCHEMA_NAME(st.schema_id) <> 'sys'  left join  sys.dm_db_index_usage_stats iu  on  iu.object_id = dp.object_id
    and
    iu.index_id = dp.index_id
order by row_count desc OPTION(RECOMPILE)
```

schema_name	object_name	name	used_page_count	reserved_page_count	row_count	user_seeks	user_scans	user_lookups	user_updates	last_user_seek	last_user_scan	last_user_lookup
innovator	HISTORY	I1	323,967	323,992	28,462,585	719	0	0	134,116	2026/7/7 15:13	NULL	NULL
innovator	HISTORY	PK__HISTORY__3214EC279113828A	2,860,678	2,861,522	28,462,585	268,590	4	719	134,116	2026/7/7 15:17	2026/6/29 13:47	2026/7/7 15:13
innovator	HISTORY	I0	297,806	297,814	28,462,585	0	2	0	134,116	NULL	2026/7/6 11:43	NULL

user_seeks：索引探索（Index Seek）の回数

user_scans：全体スキャン（Index Scan / Table Scan）の回数

user_lookups：キー参照（主に非クラスタードインデックスからクラスタードへの「bookmark lookup」）の回数

user_updates：INSERT/UPDATE/DELETE によるインデックス更新回数

User_seeks,User_scan,User_lookupsいずれも、極端に少ない、もしくはゼロがそろっている場合は削除候補となる。明らかに自分で設定したインデックス以外は、Aras Innovatorで追加した可能性があるので、 アイテムタイプの「インデックス」チェックボックスを確認すること！

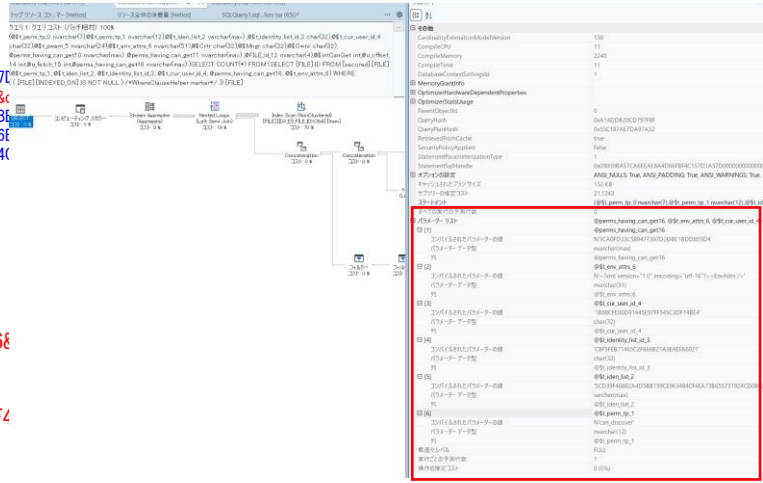
Appendix⑥

SSMSにて実行計画から単独実行する方法

※ごめんなさい、メンドクサイです・・・

1. Appendix④のSQL実行後、解析したいSQLの「query_plan」をクリック
2. 実行計画が展開されるので、左上のSelectクリックして「プロパティ」選択→「パラメータリスト」を開く
3. 右クリックでXMLを表示
   →No.2のパラメータ名で検索。

```
<ColumnReference Column="@perms_having_can_get16" ParameterDataType="nvarchar(max)" ParameterCompiledValue="N'3CA0FD33C589477397D
<ColumnReference Column="@$t_env_attrs_6" ParameterDataType="nvarchar(51)" ParameterCompiledValue="N'&lt;?xml version=&quot;1.0&
<ColumnReference Column="@$t_cur_user_id_4" ParameterDataType="char(32)" ParameterCompiledValue="'1B88CFE00D91445E97FF345C3DF14B
<ColumnReference Column="@$t_identity_list_id_3" ParameterDataType="char(32)" ParameterCompiledValue="'C8F5FEB71465C2F666B21A3E6
<ColumnReference Column="@$t_iden_list_2" ParameterDataType="varchar(max)" ParameterCompiledValue="'5CD39F46882A4D5BB159CE963484C
<ColumnReference Column="@$t_perm_tp_1" ParameterDataType="nvarchar(12)" ParameterCompiledValue="N'can_discover'" />
```



4. パラメータをdeclareにて変数定義　＋　SQL文を張り付け　SSMSで実行。

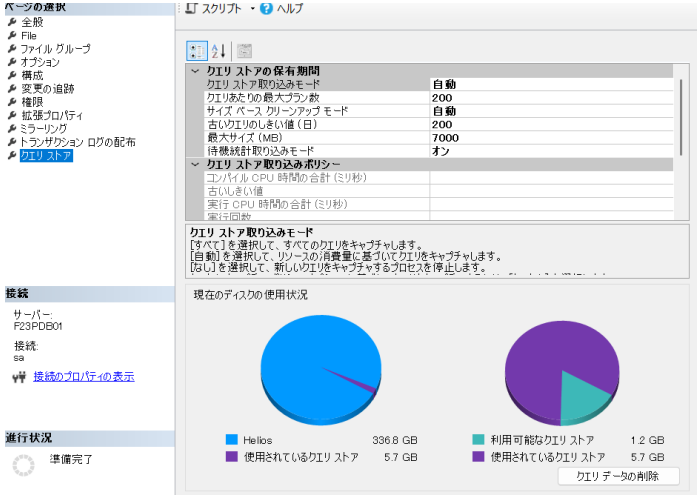
```
declare @perms_having_can_get16 nvarchar(max) =N'3CA0FD33C589477397D2DBE1BDD855D4'
declare @$t_env_attrs_6 nvarchar(51) = N'&lt;?xml version=&quot;1.0&quot; encoding=&quot;utf-16&
declare @$t_cur_user_id_4 char(32) ='1B88CFE00D91445E97FF345C3DF14BE4'
declare @$t_identity_list_id_3 char(32) ='C8F5FEB71465C2F666B21A3E6EFA6021'
declare @$t_iden_list_2 varchar(max) ='5CD39F46882A4D5BB159CE963484CF4F,A73B655731924CD0B027E4F4
declare @$t_perm_tp_1 nvarchar(12) =N'can_discover'

SELECT COUNT(*) FROM (SELECT [FILE].ID FROM [secured].[FILE]
(@$t_perm_tp_1, @$t_iden_list_2, @$t_identity_list_id_3, @$t_cur_user_id_4, @perms_having_can_get16, @$t_env_attrs_6)
 WHERE ( ( [FILE].[INDEXED_ON] IS NOT NULL ) /*WhereClauseHelper marker*/ )) [FILE]
```

クエリストア

SSMS→DB→スキーマ　右クリック「プロパティ」→「クエリストア」を有効

※時間変化を追いかけるのであれば、可能な限り長い日数が望ましい。

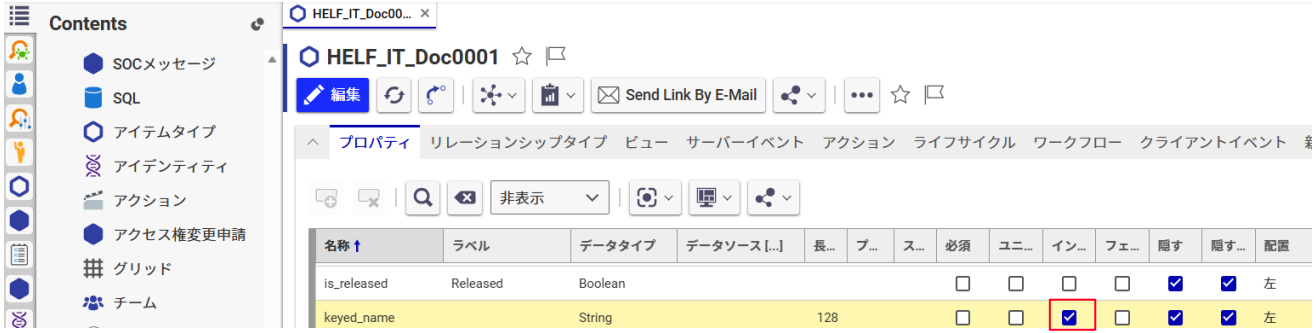


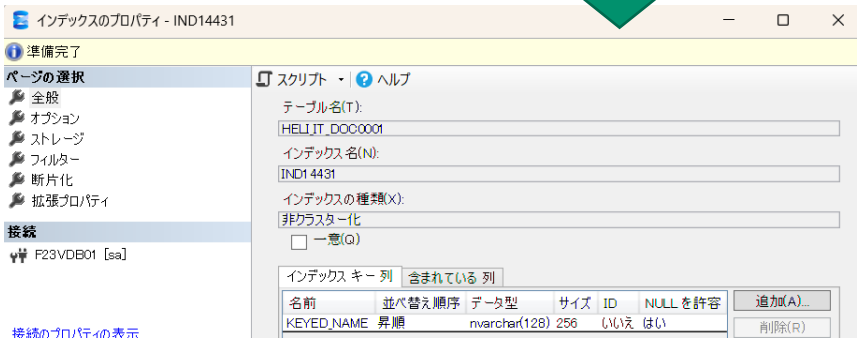
Appendix⑧

Aras Innovatorが提供するインデックス設定

管理者権限でログイン
TOCメニュー
→アドミニストレータ
→アイテムタイプ

「インデックス」に
チェックを入れると
インデックスが作られる。



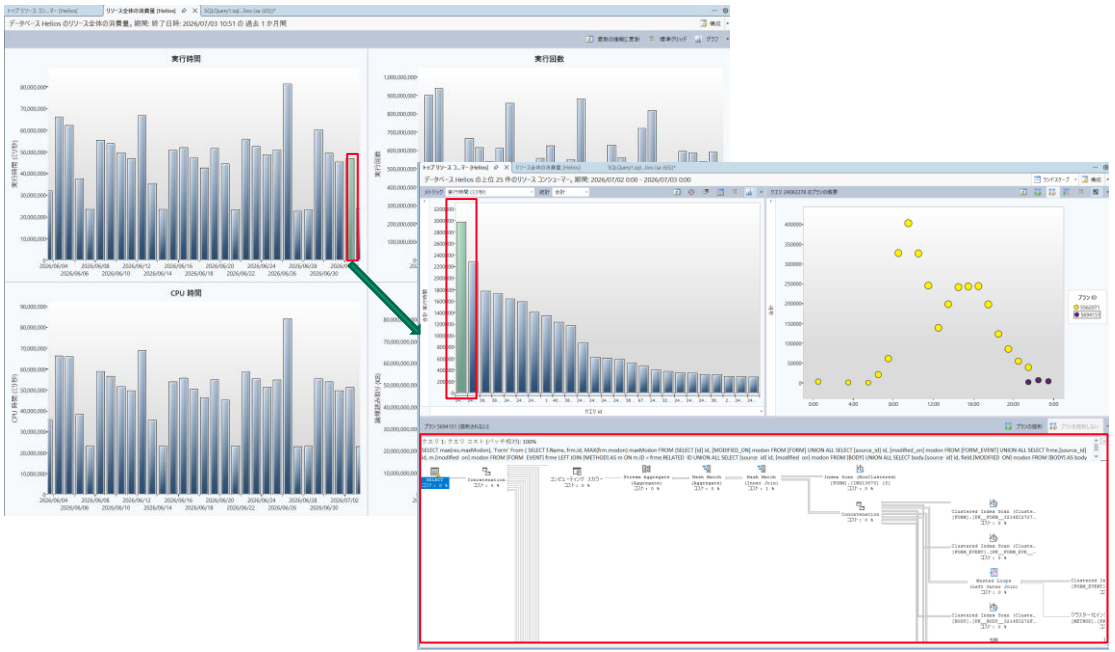


・<u>1項目しか選択できない。</u>
→ちょっと凝ったインデックスは設定不可。
<u>Aras社サポートに確認し、ユーザーによるインデックス追加OKとの見解を得ている。</u>

Appendix⑨
お勧めのインデックス作成の段取り・指針

1. 【一括適用】SSMS付属のデータベースチューニングアドバイザー（DTA）にて拾える推奨インデックスは重複チェックの上、統計情報、インデックスともに適用する。→手順等は、Aras Innovator 管理者向けSQL応答性能チューニングのあれこれ　の「①全体のパフォーマンス平均値を向上させたい」を参照。富士フイルムではDTA提案インデックス追加は15件実施。
2. 【個別対応】クエリストア→「リソース全体の消費量」→実行時間をクリック　→初期値はTop25が出るので、内容確認し、インデックス不足が提示されたら、追加を検討
   毎日10分見れば、傾向把握可能。

   実行時間積算が極端に増える場合は
   「何かある（はず）」
   詳細分析実施する場合、Appendix④のSQLを
   実行して、平均実行時間を中心に絞り込み。
   →稀にインデックス追加で解決する場合もあり。



3. 【指針】
   インデックス追加は必要最小限にする。
   効果が少ない（20%程度）であれば
   追加しない判断も必要。

インデックス分類(‘26/7/1現在)	件数
DTA提案インデックス	15
DTA提案統計情報	618
個別対応　インデックス	56
全文検索向けインデックス	187

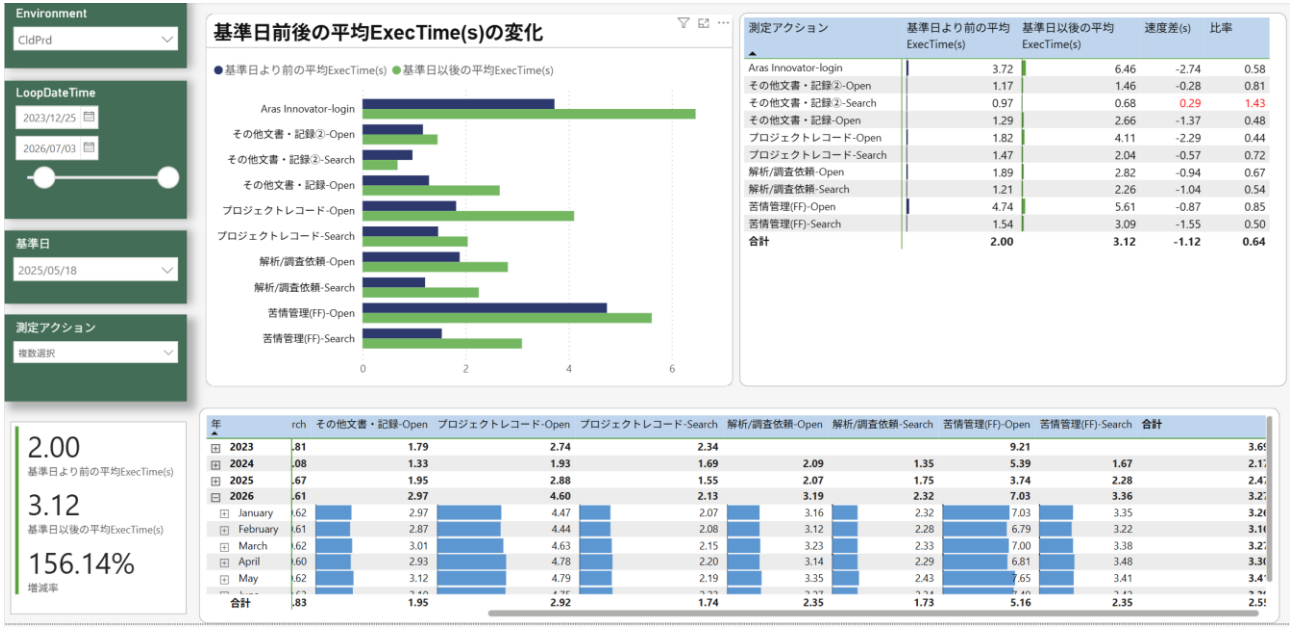
Appendix⑩

Aras V12 SP12→V14 Release31比較

2023/12/25～2026/7/3 【Aras Innovator V12 SP12→V14 Release31】

Aras Innovator Upgrade（2025/5/18 upgrade前後比較）→ 56%劣化(ユーザーは気が付かないレベルではあるが)

※Aras Upgradeによる**処理数増 AND レコード数1.7倍増**でもこの程度の劣化でコントロールできていると考える。



測定アクション	基準日より前の平均ExecTime(s)	基準日以後の平均ExecTime(s)	速度差(s)	比率
Aras Innovator-login	3.72	6.46	-2.74	0.58
その他文書・記録②-Open	1.17	1.46	-0.28	0.81
その他文書・記録②-Search	0.97	0.68	0.29	1.43
その他文書・記録-Open	1.29	2.66	-1.37	0.48
プロジェクトレコード-Open	1.82	4.11	-2.29	0.44
プロジェクトレコード-Search	1.47	2.04	-0.57	0.72
解析/調査依頼-Open	1.89	2.82	-0.94	0.67
解析/調査依頼-Search	1.21	2.26	-1.04	0.54
苦情管理(FF)-Open	4.74	5.61	-0.87	0.85
苦情管理(FF)-Search	1.54	3.09	-1.55	0.50
合計	**2.00**	**3.12**	**-1.12**	**0.64**

年	rch	その他文書・記録-Open	プロジェクトレコード-Open	プロジェクトレコード-Search	解析/調査依頼-Open	解析/調査依頼-Search	苦情管理(FF)-Open	苦情管理(FF)-Search	合計
⊞ 2023	.81	1.79	2.74	2.34			9.21		3.6
⊞ 2024	.08	1.33	1.93	1.69	2.09	1.35	5.39	1.67	2.1
⊞ 2025	.67	1.95	2.88	1.55	2.07	1.75	3.74	2.28	2.4
⊟ 2026	.61	2.97	4.60	2.13	3.19	2.32	7.03	3.36	3.2
⊞ January	.62	2.97	4.47	2.07	3.16	2.32	7.03	3.35	3.2
⊞ February	.61	2.87	4.44	2.08	3.12	2.28	6.79	3.22	3.1
⊞ March	.62	3.01	4.63	2.15	3.23	2.33	7.00	3.38	3.2
⊞ April	.60	2.93	4.78	2.20	3.14	2.29	6.81	3.48	3.3
⊞ May	.62	3.12	4.79	2.19	3.35	2.43	7.65	3.41	3.4
合計	.83	1.95	2.92	1.74	2.35	1.73	5.16	2.35	2.5

Appendix⑪
お勧め情報

【書籍】

絵で見てわかるSQL Serverの仕組み（平山 理）｜翔泳社の本

→オンラインマニュアルは細かすぎて頭に入らないので、書籍にて構造理解したほうが良い。

【マイクロソフトのオンラインマニュアル】

チェックリスト: ベスト プラクティスとガイドライン - SQL Server on Azure VMs | Microsoft Learn

→Azure VM向けとはなっているが、オンプレミスでもIaaSで構築していても基本は変わらない。

論理プラン表示演算子と物理プラン表示演算子リファレンス - SQL Server | Microsoft Learn

→SSMS（SQL Server Management Studio)にてクエリプランを見る際に、アイコン等の解説あり。大規模なソート実行時に遅くなっている場合は、tempDBへの書き込み頻度が高いことが考えられるので、テーブルスプールが発生しているか？等で判断可能。

Slow I/O - SQL Server and disk I/O performance | Microsoft Community Hub

→古い情報だが、探せばいろいろヒントはある。

【Aras Innovator関連】

Aras Support Knowledge Base - Aras Docs

→Aras Subscriber portalサイト内のナレッジ。

【本資料関連】

オンプレミスからAzureへのクラウドリフト～インフラ設計のポイントについて～

Aras Innovator 管理者向けSQL応答性能チューニングのあれこれ

FUJIFILM

Value from Innovation