INTEGRATING MICROSTRATEGY 2021 WITH AZURE SYNAPSE ANALYTICS

Whitepaper

MicroStrategy®
Analytics and Mobility

TABLE OF CONTENTS

INTRODUCTION **5**

About MicroStrategy 5

About Azure Synapse Analytics 5

About MicroStrategy Cloud Environment 6

OVERVIEW OF MICROSTRATEGY ARCHITECTURE 7

MicroStrategy Technology Philosophy: Leverage the Power of the RDBMS 7

Model-based Dynamic SQL Generation 7

Schema Abstraction 7

Aggregate Awareness 8

Multi-pass SQL 8

OVERVIEW OF AZURE SYNAPSE ANALYTICS DEDICATED SQL POOL ARCHITECTURE 9

Massively Parallel Processing 9

Data warehouse capacity settings 10

Dedicated SQL pool DDL Considerations 10

Declarative Referential Integrity 10

Distribution Style 10

Indexing 11

Loading data into Azure Synapse Analytics 12

OPTIMAL QUERY GENERATION 14

Intermediate Table Type 14

True Temporary Table 14

Derived Table 14

Table Creation Type 14

Full Outer join Support 15

Sub Query Type 15

SQL Global Optimization 15

Set Operator Optimization 16

Parallel Query Execution 16

Insert Post String and Select Post String 16

Pre/Post Statements 17

PERIODIC TASKS FOR OPTIMAL QUERY PERFORMANCE 18

Maintaining Table Statistics 18

Periodically Revisit Data Distribution 18

DATABASE WORKLOAD MANAGEMENT 20

Workload management in MicroStrategy 20

Workload management in Azure Synapse Analytics 21

MicroStrategy Integration with Azure resource classes 22

Monitoring of MicroStrategy Workload 22

SECURITY 23

Cluster and Connection Security 23

Encryption 23

MICROSTRATEGY AND AZURE SYNAPSE ANALYTICS CERTIFICATION 24

Azure Synapse Analytics Certification Status 24

APPENDIX 25

INTRODUCTION

This paper provides a brief overview of the MicroStrategy architecture and explains how this architecture takes advantage of the technology advances and business intelligence functionality of the Azure Synapse Analytics data warehouse.

The intent of this document is to provide clarification on how the two products work together and the necessary steps to ensure they are optimized. This document focuses mostly on configuration rather than installation steps. The goal is to highlight concepts that refer to detailed how-to steps in the relevant parts of the respective product documentation.

About MicroStrategy

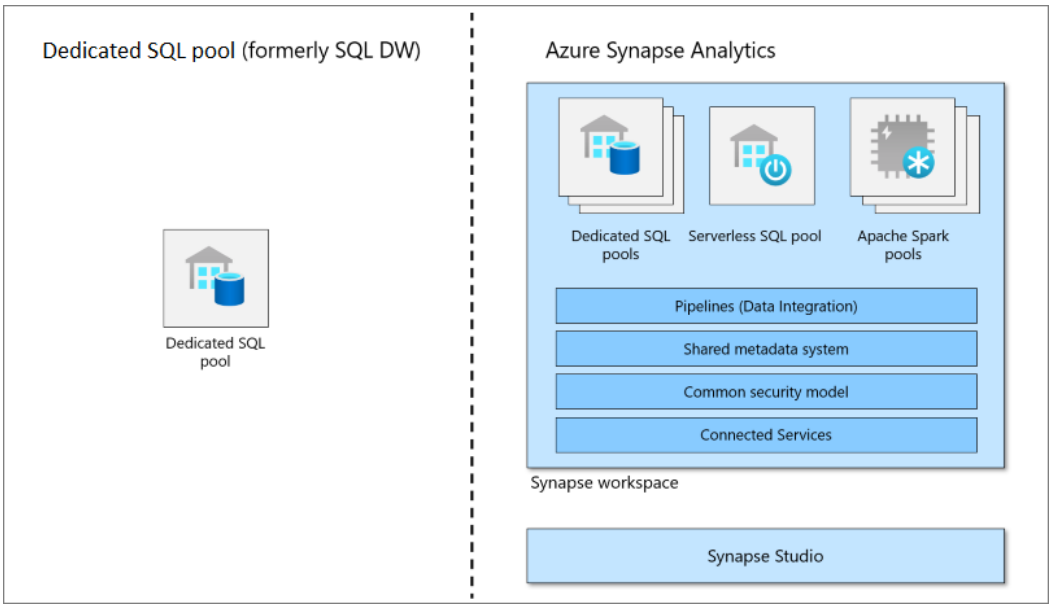
MicroStrategy is built to enable the Intelligent Enterprise — to quickly deploy sophisticated analytical and security applications at scale. Our platform architecture is uniquely suited to deliver high performance applications and meet the business intelligence demands of every user and every organization.

The MicroStrategy platform provides VLDB drivers for all supported data warehouse platforms to generate optimized SQL that takes advantage of database specific functionality. The full set of VLDB properties is documented in the MicroStrategy System Administration Guide. This guide discusses settings that are most relevant to implementing MicroStrategy with Azure Synapse Analytics Data Warehouse.

To learn more about MicroStrategy, visit http://www.microstrategy.com

About Azure Synapse Analytics

Azure Synapse Analytics is an analytics service that brings together enterprise data warehousing and Big Data analytics. Dedicated SQL pool (formerly SQL DW) refers to the enterprise data warehousing features that are available in Azure Synapse Analytics.



Dedicated SQL pool represents a collection of analytic resources that are provisioned when using Synapse SQL. Rather than have users choose their hardware configurations, dedicated SQL pool offers performance through scale-units known as Data Warehouse Units (DWUs).

Dedicated SQL pool database is a shared-nothing, massively parallel processing (MPP), columnar database that decouples compute and storage to provide elastic performance at scale. Dedicated SQL pool is comprised of a control node and multiple compute nodes, each of which is built on Microsoft SQL Server technology.

Dedicated SQL pool functionality and SQL language support is targeted primarily for OLAP (Online Analytics Processing) rather than OLTP (Online Transaction Processing). Analytics means optimized for working with large amounts of data, and complex operations such as aggregation and joins against disparate data, whereas OLTP largely focuses on operations performing single row insertions and deletions, transactions, and locking, etc.

For more information on dedicated SQL pool architecture please refer to Azure Synapse Analytics online documentation.

About MicroStrategy Cloud Environment

The MicroStrategy Cloud Environment ("MCE" or "MCE Service") is a Platform-as-a-Service ("PaaS") delivery model designed to allow businesses to consume the MicroStrategy Analytics and Mobility platform in a single tenant architecture.

MCE offers a distributed compute architecture using various components provided by the cloud infrastructure vendors of either Microsoft Azure or Amazon Web Services. At the core of the solution are MicroStrategy Analytics and Mobility, components that provide users with a secure, scalable, and resilient business intelligence enterprise application platform.

It also includes the elements needed to operate, access, and manage the intelligence architecture. Users are provisioned with their own dedicated intelligence architecture based on a predefined configuration. Once provisioned, users can develop, tailor, and manage the application components to meet their respective needs.

Based on this operating model, customers control the Analytics and Mobility application stack, while MicroStrategy maintains the supporting cloud-based infrastructure. "MCE Service" means the MicroStrategy Cloud Environment service, a platform-as-a service offering that we manage on your behalf in an Amazon Web Services or Microsoft Azure environment that includes access to, collectively: (a) the "Cloud Platform" version of our Products (an optimized version of the MicroStrategy software platform built specifically for deployment in an Amazon Web Services or Microsoft Azure environment) licensed by you; and (b) the Cloud Environment, Cloud Support, and Cloud Infrastructure you have purchased for use with such Products.

OVERVIEW OF MICROSTRATEGY ARCHITECTURE

MicroStrategy Technology Philosophy: Leverage the Power of the RDBMS

The MicroStrategy architecture has its roots in the principles of Relational OLAP (ROLAP). A ROLAP architecture provides OLAP functionality to the end user (multidimensional framework, slice-and-dice interaction, drilling, etc.), but uses a relational database to resolve queries and perform calculations, rather than using a specialized proprietary multidimensional database.

While the virtues of ROLAP are fully extolled elsewhere, this is an important distinction for this article for two reasons. First, MicroStrategy applications have been able to achieve industry-leading data scale because of the reliance on the processing power and data management capabilities of the underlying RDBMS. The system design principle of partnering with the RDBMS, through features discussed in this document, is the cornerstone of achieving such scalability. Second, this focus on integration with the RDBMS allows a business intelligence system to tap into a powerful dynamic over time: technical advances in RDBMS technology are seamlessly accessible to a MicroStrategy-based BI system. The net benefit to the customer is a BI system that increases in value when either MicroStrategy or the RDBMS adds features to their products.

Model-based Dynamic SQL Generation

When a user runs a report or dashboard in MicroStrategy will push down the analytics to the RBDMS in form of optimized SQL queries and visualize the data of the result sets according to the report and dashboard specifications. Hence, to the RDBMS, a MicroStrategy application is an SQL-based application, in many ways like any other SQL application accessing the Azure Synapse Analytics Dedicated SQL Pool Database.

Schema Abstraction

The SQL Engine component of MicroStrategy will generate the queries dynamically at runtime. The SQL Engine performs its work based on a metadata model defined to the system. Note, that the MicroStrategy metadata is not used to store joins or schema-type information, such as star or snowflake. Instead, the metadata model stores content information for each table indicating that it contains a set of particular facts and a set of attributes. When a report request is submitted, the Engine breaks the report down into the individual components (i.e., attributes and facts), then begins searching the model to determine which combination of tables will be necessary and efficient in resolving the request.

Schema abstraction of the database columns (into MicroStrategy attributes and facts) provides the flexibility necessary to allow applications to be created quickly without having to change the structure of the data model. MicroStrategy can support virtually any type of star, snowflake, or hybrid physical design, including transactional schemas. The business model defined in MicroStrategy is easily able to span multiple stars/snowflakes in a single application and even a single query. MicroStrategy supports dimensional models well but does not require a dimensional model.

Aggregate Awareness

Query performance in many data warehouses is enhanced with aggregate tables. Aggregate tables, also called summary tables, store pre-computed results of data allowing users to query from a summarized set of data rather than the detail level data that would be stored in the fact table. In many cases, use of aggregate tables will improve query performance by orders of magnitude.

MicroStrategy's SQL Engine is aggregate-aware and determines the use of aggregate tables transparently at query time. MicroStrategy has allowed transparent navigation of aggregate tables, directing queries to summary tables when they exist without the user having to specify to use the table.

Multi-pass SQL

One of the key elements to providing analytical sophistication in analytical applications is MicroStrategy's ability to generate multi-pass SQL. Multi-pass SQL is required to answer analytical questions that cannot be answered with a single SQL query block. Examples of questions / scenarios that require multi-pass SQL include:

- Set qualification: "Show me sales by region over the last six months, but only for customers who purchased one of the 5 most popular products."
- Split metrics: query returns sales data from a sales star schema and inventory data from an inventory star schema
- Metrics calculated at different levels of aggregation
- Metrics calculated with different filtering criteria
- Simulating outer joins on RDBMS platforms that do not support them natively
- Querying multiple tables due to application-level partitioning

Support for these scenarios, especially when combined, provide a framework for significant analytic questions and value to the users of the system. One of the main optimizations the MicroStrategy SQL Engine makes is to generate SQL that performs these multi-pass queries as efficiently as possible.

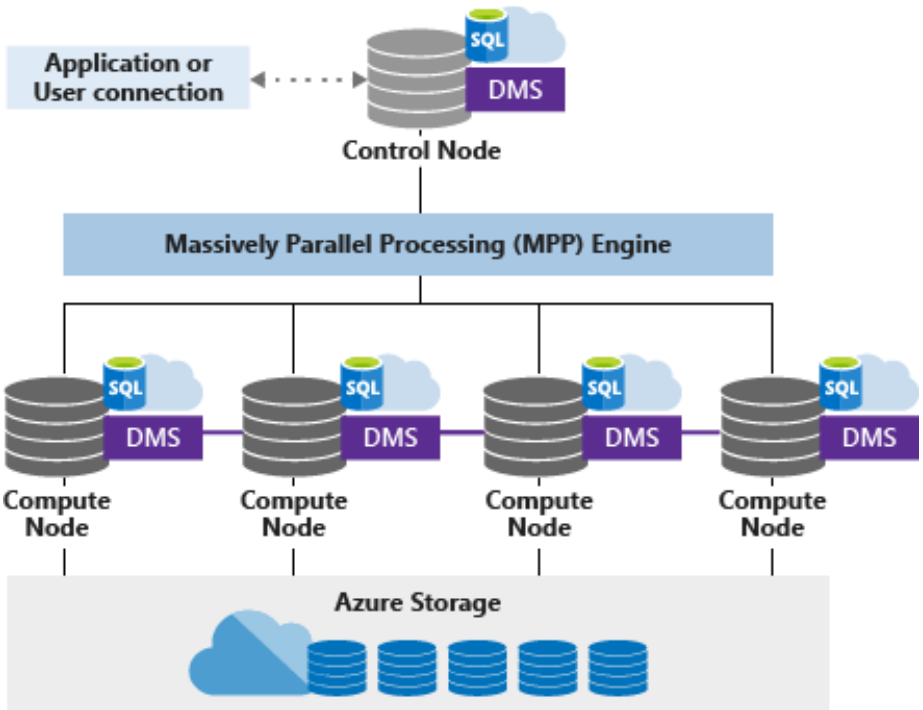
OVERVIEW OF AZURE SYNAPSE ANALYTICS DEDICATED SQL POOL ARCHITECTURE

Massively Parallel Processing

Dedicated SQL pool (formerly SQL DW) leverages a scale-out architecture to distribute computational processing of data across multiple nodes. The unit of scale is an abstraction of compute power that is known as a data warehouse unit. Compute is separate from storage, which enables you to scale compute independently of the data in your system.

Dedicated SQL pool is based on a massively parallel processing (MPP) architecture in which multiple homogeneous hardware nodes are clustered together to process user queries. The major components of a dedicated SQL pool MPP cluster are:

1. **Compute nodes** that perform the work of processing SQL statements in parallel:
    a. **A Data Movement Service** (DMS) that is a system-level internal service that moves data across the nodes as necessary to run queries in parallel and return accurate results.
    b. **Azure Storage** that is leveraged to store user data
2. **A control node** that manages connections to the database and coordinates the activities of the compute nodes. Applications connect and issue T-SQL commands to a Control node.



Data warehouse capacity settings

A dedicated SQL pool represents a collection of analytic resources that are being provisioned. Analytic resources are defined as a combination of CPU, memory, and IO.

These three resources are bundled into units of compute scale called Data Warehouse Units (DWUs). A DWU represents an abstract, normalized measure of compute resources and performance.

A change to your service level alters the number of DWUs that are available to the system, which in turn adjusts the performance, and the cost, of your system. The ideal number of data warehouse units depends very much on your workload and the amount of data you have loaded into the system.

For more information about memory and concurrency limits allocated to the various performance levels and resource classes in Azure Synapse Analytics refer to: https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/memory-concurrency-limits

Dedicated SQL pool DDL Considerations

Declarative Referential Integrity

The following referential integrity constraints cannot be applied in the dedicated SQL pool:

- primary keys
- foreign keys
- unique constraints

If uniqueness or existence must be enforced, then it must be done in code.

Distribution Style

A distribution is the basic unit of storage and processing for parallel queries that run on distributed data in dedicated SQL pool. When dedicated SQL pool runs a query, the work is divided into 60 smaller queries that run in parallel.

Each of the 60 smaller queries runs on one of the data distributions. Each Compute node manages one or more of the 60 distributions. A dedicated SQL pool with maximum compute resources has one distribution per Compute node. A dedicated SQL pool with minimum compute resources has all the distributions on one compute node.

There are three types of table distributions available:

1. **Hash-distributed**

In the HASH distribution, the data is organized in shards based on a hash function that is used to deterministically assign each row to one distribution. A hash distributed table can deliver the highest query performance for joins and aggregations on large tables.

2. **Round Robin**

A round-robin distributed table distributes data evenly across the table but without any further optimization. A distribution is first chosen at random and then buffers of rows are assigned to distributions sequentially.

3. **Replicated**

In a replicated table, a copy of the data is retained on every shard. This removes the need to transfer data among compute nodes before a join or aggregation. Replicated tables are best utilized with small tables.

Use the following strategies, depending on the table properties:

Type	Great fit for...	Watch out if...
Replicated	- Small dimension tables in a star schema with less than 2 GB of storage after compression (~5x compression)	- Many write transactions are on table (such as insert, upsert, delete, update) - You change Data Warehouse Units (DWU) provisioning frequently - You only use 2-3 columns, but your table has many columns - You index a replicated table
Round Robin (default)	- Temporary/staging table - No obvious joining key or good candidate column	- Performance is slow due to data movement
Hash	- Fact tables - Large dimension tables	- The distribution key cannot be updated

Indexing

Dedicated SQL pool offers several indexing options:

1. **clustered columnstore indexes**
2. **clustered indexes**
3. **nonclustered indexes**
4. **non-index option also known as heap**

Choosing an appropriate indexing strategy depends on table size, data cardinality and query patterns. In general, rowstore indexes perform best on queries that seek into the data, when searching for a particular value, or for queries on a small range of values, whereas columnstore

indexes give high performance gains for analytic queries that scan large amounts of data, especially on large tables.

Use the following strategies, depending on your needs:

Type	Great fit for...	Watch out if...
Heap	- Staging/temporary table - Small tables with small lookups	- Any lookup scans the full table
Clustered index	- Tables with up to 100 million rows - Large tables (more than 100 million rows) with only 1-2 columns heavily used	- Used on a replicated table - You have complex queries involving multiple join and Group By operations - You make updates on the indexed columns: it takes memory
Clustered columnstore index (CCI) (default)	- Large tables (more than 100 million rows)	- Used on a replicated table - You make massive update operations on your table - You over partition your table: row groups do not span across different distribution nodes and partitions

Loading data into Azure Synapse Analytics

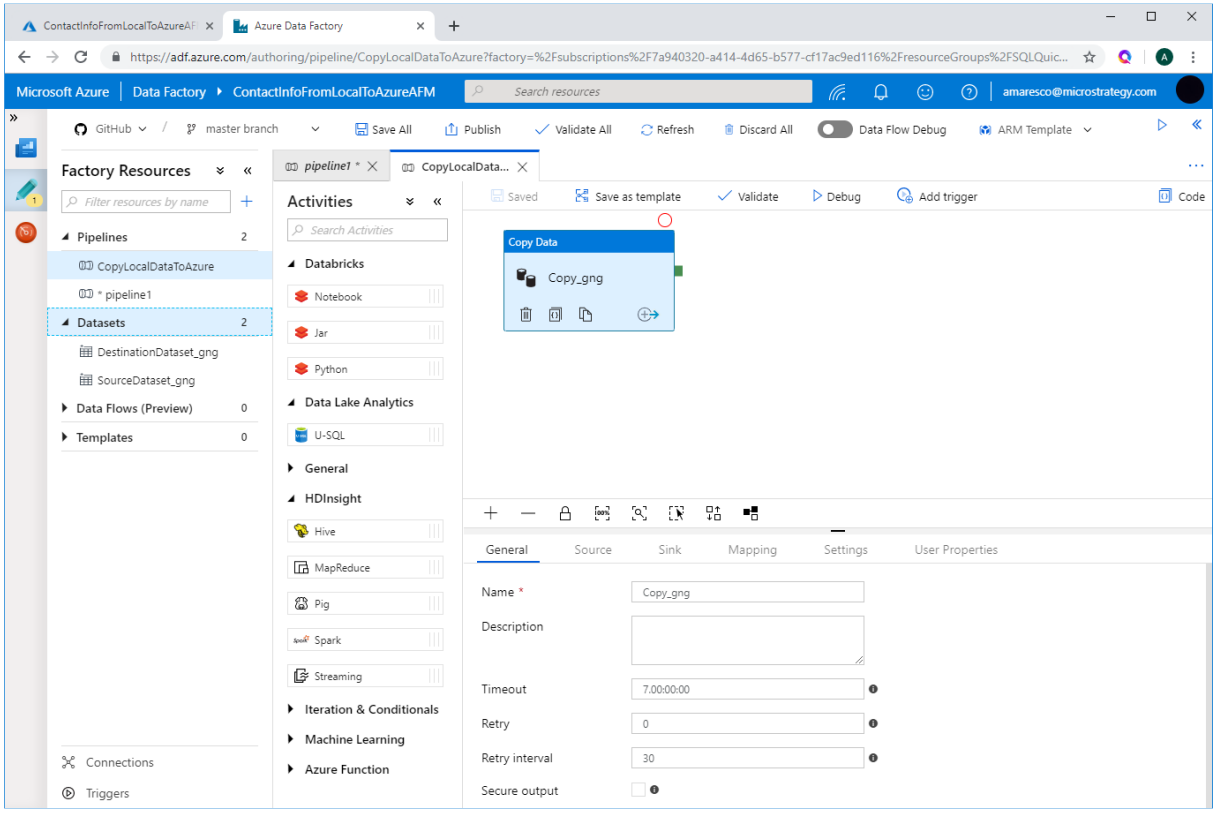
In the world of big data, raw, unorganized data is often stored in relational, non-relational, and other storage systems. However, on its own, raw data doesn't have the proper context or meaning to provide meaningful insights to analysts, data scientists, or business decision makers.

Big data requires a service that can orchestrate and operationalize processes to refine these enormous stores of raw data into actionable business insights. Azure Data Factory is a managed cloud service that's built for these complex hybrid extract-transform-load (ETL), extract-load-transform (ELT), and data integration projects.

You can use ADF to populate an Azure Synapse Analytics with data from your existing system and save time when building your analytics solutions. Azure Data Factory is the cloud-based ETL and data integration service that allows you to create data-driven workflows for orchestrating data movement and transforming data at scale. Using Azure Data Factory, you can create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores.

Azure Data Factory supports GitHub integration so that you can place your pipelines under version control. Also notice the range of Activities which are offered, and you can also add custom activities as well. You also configure your datasets in databases, storage, and external sources such as AWS S3.

This is a screenshot of the Azure Data Factory authoring tool to create an ADF pipeline:



For more information about Azure Data Factory please refer to: https://docs.microsoft.com/en-us/azure/data-factory/introduction

For detailed steps how to use the Data Factory Copy Data tool to load data into Azure Synapse Analytics refer to: https://docs.microsoft.com/en-us/azure/data-factory/load-azure-sql-data-warehouse

OPTIMAL QUERY GENERATION

Intermediate Table Type

The ability to generate multi-pass SQL is a key feature of the MicroStrategy SQL Engine as noted in Overview of MicroStrategy Architecture section. Azure Synapse Analytics supports a number of different ways to implement multi-pass SQL. Among the most performant ones are True Temporary Table and Derived Tables.

Derived Table

Derived Table syntax allows the SQL Engine to issue additional passes as query blocks in the FROM clause. Instead of issuing multiple SQL statements that create intermediate tables, the SQL engine generates a single large SQL statement. This can allow queries to run faster since there is no CREATE TABLE or DROP TABLE statements to catalog, no corresponding locks on the system tables, and no logging of records inserted into a physical table.

Note that not all reports are able to use derived tables. Two primary scenarios in which temporary tables must be used instead of derived tables are when a report uses a function supported in the MicroStrategy analytical engine that is not supported in Synapse or when a report uses the MicroStrategy partitioning feature. These situations do not cover 100% of the cases in which temporary tables must be used. The rest of the cases are relatively obscure combinations of VLDB settings, such as certain combinations of Sub Query Type plus outer join settings on metrics plus non-aggregable metrics.

Moreover, some reports may be running with suboptimal plans because they are using derived tables. It happens when reports have many derived tables, so the query becomes "too complex" and do not perform well. In such a scenario, True Temporary Table setting should be used.

True Temporary Table

Users can select True Temporary Table as Intermediate Table Type to force a multi-pass SQL execution, i.e. execute each pass (each query block) in a separate table. When the VLDB property Intermediate Table Type is set to "True Temporary Table", each pass results in a temporary table. A temporary table typically incurs less I/O and better performance than permanent tables and is recommended.

Table Creation Type

When creating temporary or physical tables in Synapse, MicroStrategy SQL Engine will use implicit table creation by default (rather than explicit table creation). When using implicit table creation, the tables created by MicroStrategy (true temporary tables) would be automatically created using CTAS statement. The CREATE TABLE AS SELECT (CTAS) statement is one of the most important T-SQL features available. CTAS is a parallel operation that creates a new table based on the output of a SELECT statement. CTAS is the simplest and fastest way to create and insert data into a table with

a single command. With CTAS you can specify both the distribution of the table data as well as the table structure type. By default, MicroStrategy Engine will use below design for temporary tables:

WITH (DISTRIBUTION = ROUND_ROBIN, HEAP)

This setting can be modified by editing Table Space VLDB Property.

Full Outer join Support

Full outer join support is enabled in the Azure Synapse by default. Levels at which you can set this are Database instance, report, and template.

Sub Query Type

There are many cases in which the SQL Engine will generate sub-queries (i.e., query blocks in the WHERE clause):

- Reports that use Relationship Filters
- Reports that use "NOT IN" set qualification, e.g., AND NOT <metric_qualification> or AND NOT <relationship_filter>
- Reports that use Attribute qualification with M-M relationships, e.g., show Revenue by Category, filter on Catalog
- Reports that "raise the level" of a filter, e.g., dimensional metric at Region level, but qualify on Store
- Reports that use non-aggregable metrics, e.g., inventory metrics
- Reports that use Dimensional extensions Reports that use Attribute to attribute comparison in the filter

The default setting for Sub Query Type for Azure Synapse is Option 2 – "Where EXISTS (select (col1, col2…))."

Some reports may perform better with Option 5 – "Use temporary table, falling back to IN for correlated subquery". Reports that include a filter with an "AND NOT set qualification" (e.g., AND NOT relationship filter) will likely benefit from using temp tables to resolve the subquery. However, such reports will probably benefit more from using the Set Operator Optimization discussed below.

Other reports may perform better with Option 3 – "WHERE (col1, col2) IN (Select s1. col1, s1. col2) …". This setting instructs the SQL Engine to generate a subquery in the WHERE clause using the IN operator.

SQL Global Optimization

This setting can reduce the number of SQL passes generated by MicroStrategy. In MicroStrategy, SQL Global Optimization reduces the total number of SQL passes with the following optimizations:

- Eliminates unused SQL passes, e.g., a temporary table is created but not referenced in a later pass

- Reuses redundant SQL passes (e.g., the same temporary table is created multiple times a single temp table is created)
- Combines SQL passes where the SELECT list is different (e.g., two temporary tables have same FROM clause, same JOINs, same WHERE clause, same GROUP BY SELECT lists are combined into single SELECT statement)
- Combines SQL passes where the WHERE clause is different (e.g., two temporary tables have the same SELECT list, same FROM clause, same JOINs, same GROUP BY predicates from the WHERE clause are moved into CASE statements in the SELECT list)

See the System Administration Guide for a complete description of the cases covered by this setting. The default setting for Synapse SQL is to enable SQL Global Optimization at its highest level.

Set Operator Optimization

This setting is used to combine multiple subqueries into a single subquery using set operators (e.g., UNION, INTERSECT, EXCEPT). However, there is no significant improvement against Synapse Analytics. The default value for Synapse Analytics is to disable Set Operator Optimization.

Parallel Query Execution

The Parallel Query Execution is an advanced property which determines whether MicroStrategy attempts to execute multiple queries in parallel to return report results faster and publish Intelligent cubes. When enabled it allows independent passes of SQL to be executed simultaneously against a database instead of sequentially.

Parallel Query Execution is disabled in Azure Synapse by default, but when enabled one must remember about possible implications towards database concurrency slots. The number of concurrency slots depends on how many resources you allocate to your Azure Synapse Analytics SQL Pool.

Parallel Query Execution produces viable improvements for big, multi-pass Intelligent Cubes when Direct loading is used in Data population for Intelligent Cubes VLDB setting.

Insert Post String and Select Post String

Insert Post String property allows you to define a custom string to be inserted at the end of the INSERT statements. Similarly, Select Post String property allows you to define a custom string to be inserted at the end of the SELECT statements.

Both can be helpful if statements such as OPTION (MERGE JOIN, LOOP JOIN) need to be included after every pass of SQL to optimize the execution, avoid the error message, or complete the task.

A helpful statement to add in Insert/Select Post String is:

OPTION (LABEL='custom string or MicroStrategy wild card')

It allows for query labeling that tags MicroStrategy issued queries that are executed on the dedicated SQL pool. These labels can be then used to quicky find query details in dynamic management views (DMVs).

A sample statement that labels given insert or select pass with a report name:

OPTION (LABEL='!o')

Pre/Post Statements

Using VLDB properties for Pre/Post Statements, MicroStrategy allows users to input customized SQL in between multi-pass SQL. There are eight statements to insert into a SQL statement:

- Cleanup Post Statement
- Insert Mid Statement
- Insert Post Statement
- Insert Pre Statement
- Report Post Statement
- Report Pre Statement
- Table Post Statement
- Table Pre Statement

For more information about customizing SQL Statements via Pre/Post Statements visit: https://www2.microstrategy.com/producthelp/Current/SystemAdmin/WebHelp/Lang_1033/Content/Customizing_SQL_statements__Pre_Post_Statements.htm

Table available there summarizes the Pre/Post Statements VLDB properties and additional details about each property, including examples and a list of wild cards, are available.

A common scenario when Pre/Post Statements are used is statistics maintenance for temporary tables created by MicroStrategy. By using Table Post Statement one can add explicit statistics creation for temporary tables, like (??? is a wildcard for inserting the table name):

CREATE STATISTICS <statistics_name> on ???(<column_name>);

PERIODIC TASKS FOR OPTIMAL QUERY PERFORMANCE

Maintaining optimal query performance is dependent upon Synapse tables having updated statistic and optimal distribution and indexing choices. As the data is added or altered in the database tables, the statistics can go stale, or choice of data distribution may become obsolete. Therefore, it becomes necessary to run maintenance tasks periodically to retain the optimal execution of Microstrategy queries with Azure Synapse Analytics dedicated SQL pool. This section provides an overview of the maintenance tasks to be run periodically on Synapse.

Maintaining Table Statistics

Table statistics are vitally important to the query planner to generate optimal query execution plans. Statistics are used by the query planner to generate optimal query execution plans on each of the compute nodes, as well as the distributed query plan. In Synapse, table statistics are created using CREATE STATISTICS statement and updated through UPDATE STATISTICS statement. Statistics can be managed automatically or manually. Depending on the table size and frequency of updates, users should choose which option is better suited for their workload.

In the context of a MicroStrategy deployment it is important that all queries submitted by MicroStrategy are supported with current statistics. There are two key aspects to this. Firstly, there needs to be a strategy to maintain current statistics on all tables that implement the data model used by MicroStrategy. The second aspect involves statistics collections for intermediate tables created by MicroStrategy as part of multi-pass SQL functionality.

Synapse automatically creates statistics for permanent tables when AUTO_CREATE_STATISTICS option is set to ON and one of the below statements is used:

- SELECT
- INSERT-SELECT
- CTAS
- UPDATE
- DELETE
- EXPLAIN when containing a join or the presence of a predicate is detected

When automatic statistics are created, they'll take the form: *WA_Sys<8 digit column id in Hex>_<8 digit table id in Hex>*.

Note, the automatic creation of statistics is not generated on temporary or external tables. Therefore, it is recommended to use appropriate statements to update statistic in Table Post Statement VLDB setting.

Periodically Revisit Data Distribution

With business changes and the new data is inserted into tables, the selected distribution style may no longer provide even distribution across Synapse nodes, thus hindering parallelism. It is

recommended to periodically review the data distribution technique for your tables as more data is added and the business changes.

For more details on defining table distribution, refer to the Distribution Style section in Dedicated SQL pool DDL Considerations chapter or Azure Synapse documentation:

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-distribute?context=/azure/synapse-analytics/context/context

DATABASE WORKLOAD MANAGEMENT

In general, a typical MicroStrategy workload consists of a mix of queries with significantly varying complexity. These queries originate from different MicroStrategy jobs, e.g., element requests, simple grids (select * from), complex analytical multi-pass reports, Dashboards that rely on multiple (simple or complex) queries, database write-back from Transaction Services, and Cube reports designed to fill MicroStrategy In-Memory Cubes.

Additionally, it needs to be assumed that other workloads are being simultaneously submitted to Synapse SQL pool from other sources.

Workload Management (WLM) is necessary to optimize access to database resources for concurrently executing queries. The goals of a functional workload management are to

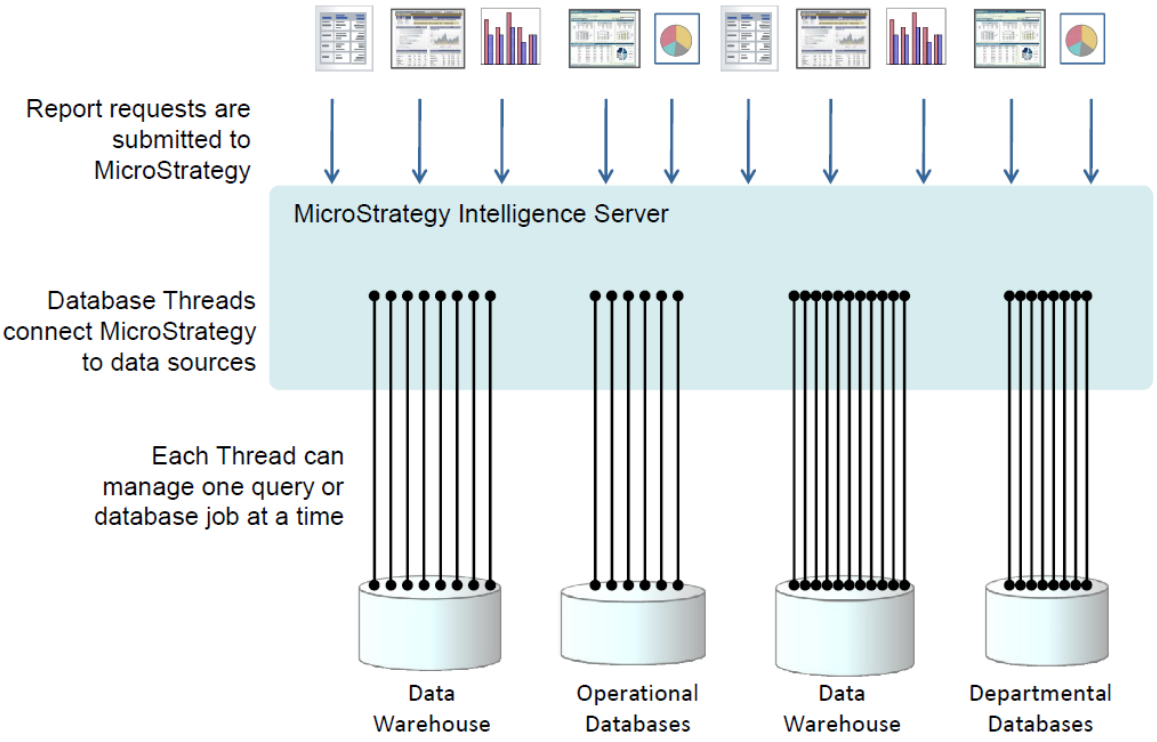
- Optimally leverage available (hardware) resources for performance and throughput
- Prioritize access for high priority jobs
- Assure resource availability by avoiding system lock-up by any small set of jobs

Effective workload management starts with comprehensive monitoring, allowing identifying bottleneck conditions, and then leveraging the available platform tools to implement a workload management

Workload management in MicroStrategy

In order to manage the varying complexity of a typical MicroStrategy deployment MicroStrategy possesses its own workload management. MicroStrategy internally breaks down every user request into one or more jobs that are processed independently. Each job advances through a series of processing steps, one of which might be the submission of multi-pass queries to a database. MicroStrategy is capable of processing and submitting multiple jobs in parallel to a database.

By default, MicroStrategy opens multiple connections called Database Threads to any data source. This is illustrated in the diagram below.



A typical BI environment encompasses queries ranging from very quick and urgent to long running and low priority. To avoid the scenario where a small number of expensive queries can block all access to database resources it all database threads are assigned priority classes (High, Medium, and Low). When report jobs are submitted, jobs are assigned a priority based on a list of different application parameters – User groups, Application type, Project, Request type and Cost. The MicroStrategy workload management routes each job according to their priority to their corresponding database threads. When no database threads are available jobs will be queued until a database thread of the appropriate class becomes available.

For more information on how to set these priorities refer to technical note https://community.microstrategy.com/s/article/KB5401-How-to-set-Group-Prioritization-in-MicroStrategy.

Administrators can, for each priority class and depending on the available database resources, specify the number of warehouse connections that are required for efficient job processing.

Workload management in Azure Synapse Analytics

Azure Synapse Analytics Dedicated SQL pool offers workload management that is governed by resource classes. Resource classes dictate the amount of memory and CPU cycles a user query is granted. A user can customize user resource class roles through *sp_addrolemember* and *sp_droprolemember* statements.

More information on configuring and customizing resource classes: https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/resource-classes-for-workload-management

In general, there can be multiple applications submitting queries to dedicated SQL pool in addition to MicroStrategy. Dedicated SQL pool query labeling allows identification of MicroStrategy queries

which in turn can inform users on the performance and queue times of their queries. A way to label queries was described in Pre/Post Statements section in Optimal query generation chapter.

A MicroStrategy report job can submit one or more queries to Azure Synapse. In case of multi-pass reports all passes for a MicroStrategy job are typically submitted in its own (or "a single") database session that can be traced in dynamic management views (DMV), like *sys.dm_pdw_exec_requests*.

MicroStrategy Integration with Azure resource classes

Apart from assigning MicroStrategy database user an appropriate resource class there is a way to force a different resource allocation for a given report despite of global workload management settings. To achieve that, one can specify a SQL session context using *sys.sp_set_session_context* system procedure and a desired workload management setting defined by a workload classifier. To do that in MicroStrategy use Report Pre and Post Statements in VLDB settings.

A sample Report Pre Statement syntax:

EXEC sys.sp_set_session_context @key = 'wlm_context', @value = 'classifier_name>

To clear session settings use a Report Post Statement:

EXEC sys.sp_set_session_context @key = 'wlm_context', @value = null

Monitoring of MicroStrategy Workload

The primary goals when monitoring workload is to ensure that it's utilizing the available hardware resources efficiently, that sufficient hardware resources are available to handle the given workload, and, in case the first conditions are not met, that it will provide the insights to identify issues and areas for improvement.

A good starting point for monitoring the workload of a MicroStrategy implementation is MicroStrategy Platform Analytics. Platform Analytics easily provides an overview of the BI workload, the consumed resources, such as time spent and CPU cycles, and allows identifying the time spent querying the database both from an aggregated view down to individual user requests.

SECURITY

There are three primary levels of security when using Azure Synapse Dedicated SQL pool:

- **Resource security**: Controlling access to SQL Data Warehouse through Azure
- **Connection security**: Controlling which user connections are permitted to connect to SQL Data Warehouse
- **Database object security**: Controlling which users have access to which database objects

Cluster and Connection Security

Azure resources such as Dedicated SQL pools are governed by role based access control (RBAC) through Azure Active Directory (AAD) and Azure Resource Manager (ARM). These roles define what privileges granted to Azure users against resources such as creation, modification, and deletion of resources within resource groups. It is also within Azure Resource Manager that security features such as Virtual Network ACL definitions and firewall rules are set up for Logical Services on which SQL Data Warehouse instances are hosted.

Client connections in Synapse Dedicated SQL pools are configured separately from Azure Resource Manager security. Users and Logins for Dedicated SQL pool are configured within the Logical Server and database instance through T-SQL commands. Currently both SQL Server Authentication (userid/password), as well as Azure Active Directory authentication is supported. Note, that only client connections and Azure services which have been whitelisted in firewall rules and access control lists can connect to the logical server and Dedicated SQL pool instance.

MicroStrategy supports authentication to Synapse dedicated SQL pools using SQL Server Authentication (userid/password).

Encryption

Transparent Data Encryption (TDE) helps protect against the threat of malicious activity by encrypting and decrypting your data at rest. When you encrypt your database, associated backups and transaction log files are encrypted without requiring any changes to your applications. TDE encrypts the storage of an entire database by using a symmetric key called the database encryption key.

You can encrypt your database using the Azure portal or T-SQL.

MICROSTRATEGY AND AZURE SYNAPSE ANALYTICS CERTIFICATION

Azure Synapse Analytics Certification Status

Azure Synapse Analytics is certified as warehouse platform with the MicroStrategy 2021 platform release on various flavors of Windows and Linux. Certification level is set to "Diamond", i.e., MicroStrategy strongly advocates for the use of these-industry-leading sources and proactively runs end-to-end regression and performance tests as soon as the source becomes generally available. For the full listing of all the certified/supported configurations, please refer to: https://www2.microstrategy.com/producthelp/Current/Readme/en-us/Content/certified_configurations.htm

APPENDIX

Some of the default VLDB Settings for Azure Synapse Analytics:

VLDB Category	VLDB Property Setting	Value
Tables	Intermediate Table Type	Derived table
Tables	Table Space	WITH (DISTRIBUTION = ROUND_ROBIN,HEAP)
Tables	Table Creation Type	Implicit
Query Optimizations	Sub Query Type	Where EXISTS(select (col1, col2…))
Joins	Full Outer Join Support	Supported
Query Optimizations	SQL Global Optimization	Level 4: Level 2 + Merge All Passes with Different Where
Query Optimizations	Set Operator Optimization	Enable set operator optimization
Select/Insert	UNION multiple INSERT	Do not use UNION