SQL Server and MicroStrategy: Functional Overview Including Recommendations for Performance Optimization

MicroStrategy World 2016
Technical Integration with Microsoft SQL Server

Microsoft SQL Server is a data platform for analytical and transactional operations

- Microsoft offers a rich family of relational database products in the SQL Server family

- This session discusses the integration with
  - Microsoft SQL Server
  - Microsoft SQL Database on Cloud/Microsoft Azure

- MicroStrategy offers a different set of optimizations for the following products
  - Microsoft SQL Server Parallel Data Warehouse
  - Microsoft Analytics Platform System
  - Microsoft SQL Server Analysis Services
MicroStrategy Is Most Commonly Used To Send Analytical Queries to Microsoft SQL Server

Analytical queries have specific technical characteristics that differentiate them from operational queries

• Analytical queries involve processing of massive amounts of data
  o Accessing large data volume
  o Processing massive data volume

• **Challenge**: Achieve interactive response times

• Microsoft SQL Server offers some key features to help with that:
  o Compression
  o Partitioning
  o In-memory columnstore
  o In-memory OLTP, etc.
MicroStrategy Data Access Workflows

There are numerous ways for MicroStrategy to interact with Microsoft SQL Server

- **Adhoc Schema**
  - For Analysts familiar with data in database
  - Schema is created automatically on the fly
  - Optimal time-to-value

- **Modeled Schema**
  - BI Architect creates logical model of data in MicroStrategy
  - Analyst or Consumers use model objects (attributes and metrics) to express their analytical needs
  - MicroStrategy generates translated to multi-pass SQL to database

- **Live Connect**
  - User actions result in interactive queries against data source
  - Good for frequently changing data

- **In-Memory Dataset**
  - Dataset is imported from database into Multi-dimensional In-Memory
  - Can improve performance and user scale accessing less frequently updated data sets
Many Integration Points Tackle Common Challenges

• SQL Server specific SQL Syntax
  o Ordered Analytic (OLAP) functions
  o CASE expressions
  o Full outer joins
  o Set operators
  o Sub queries

• Multi-pass SQL for analytical sophistication
  o Use of temporary tables or derived tables
  o Indexing
  o MicroStrategy Parallel SQL Execution

• Support for key SQL Server features
  o Parallel Query Execution
  o Indexed Views
  o Clustered Columnstore Indexes
  o Compression
  o Partitioning
  o Data level security (Pass-through auth)
  o SQL Server Linked Servers
  o Integration with Table-Valued User-Defined functions
  o Querying XML data

• Extensions to SQL Server functionality
  o Aggregate awareness with physical summary tables
  o Middle-tier computation of calculations not available in SQL Server
  o Middle-tier caching via Intelligent Cubes
  o Report caching
MicroStrategy Generates SQL Server-Specific Syntax

*Take advantage of SQL Server’s stack of rich analytics*

Push down 120 + **Functions**
- Mathematical,
- String,
- Statistical,
- Date-Time functions
- 20+ OLAP functions

20+ SQL-Server specific tunable settings
- Full outer joins,
- Set Operators,
- Implicit/Explicit Table Creation Type,
- Query banding,
- Indexing,
- Sub-Query Type, etc.
Multi-Pass SQL Helps Answer Complex Analytical Questions

Choice of Intermediate tables have a significant impact on Query Performance

MicroStrategy offers multiple approaches for Intermediate tables

Default: Global or True temporary table

A simple configuration setting allows switching:

VLDB: Intermediate Table Type

Intermediate result sets are created, populated and dropped by MicroStrategy.

Global temporary tables, indicated with the “##” prefix, are created in the tempdb and last for only one session
Indexes On Large Temp Tables Improve Performance

Report level setting creates secondary index on Intermediate Tables

```
select a11.CUSTOMER_ID  CUSTOMER_ID
into
##ZZMQ00
from  CUSTOMER_SLS a11
where (a11.TOT_UNIT SALES > 10.0)

create index ##ZZMQ00_i on ##ZZMQ00 (CUSTOMER_ID)
```

In-memory columnstore Indexes

CREATE CLUSTERED COLUMNSTORE INDEX cci_?? ON ?

‘??’ is a MicroStrategy wildcard that will be replaced with the name of the MicroStrategy data mart table
Executing multiple passes in parallel in MicroStrategy

*Get the most out of the hardware you already have!*

- Standard Edition only uses single core to process any given query – analytical queries are often CPU bound.
- MicroStrategy’s “Parallel Query Execution” identifies the passes in a multi-pass SQL job that don’t depend on each other and can be safely submitted in parallel.
- By doing so the hardware resources on a Standard Edition SQL Server can be better utilized.

**Required VLDB Settings to enable this:**
- Query Optimizations -> Parallel Query Execution -> Enable
- Parallel Query Execution Improvement Estimate in SQL View -> Enable
- Maximum Queries per Report -> 2 (default)
Utilize Multiple Cores for SQL Query Execution

Parallel SQL Execution can improve performance for SQL Server Enterprise Edition

SQL Server Enterprise offers Parallelism feature via a server setting:

Max Degree of Parallelism: Should be no more than the number of cores available to the SQL Server instance.
Improve query speed dramatically using Indexed Views

*SQL Server can take advantage of pre-aggregated data structures*

- Indexed views are a SQL Server feature that provides for the creation, maintenance, and automatic navigation of aggregated data.

- Pre-summarization allows the database to perform row access and aggregation ahead of time.

- Any SQL generated by an end user request will be submitted to SQL Server where the optimizer may choose to substitute the Indexed View for the base table.

- A MicroStrategy BI architect can also use the Indexed View as summary table resulting in simpler SQL queries.
**Querying XML Data in SQL Server**

*SQL Server “methods” can be used by MicroStrategy to display results*

- MicroStrategy SQL can generate the EXIST syntax via a Filter using `ApplyComparison`: `ApplyComparison("#0.exist(''/book/author[ln='"Doe"]') = 1", Book@XML)`

```sql
select a11.BOOK_ID  BOOK_ID,
       sum(a11.DOLLARS)  WJXBFS1
from  BOOK SALES  a11
join  LU_BOOK a12
      on  (a11.BOOK_ID = a12.BOOK_ID)
where  a12.BOOK_XML.exist(''/book/author[ln="Doe"]') = 1
group by  a11.BOOK_ID
```

- MicroStrategy can display elements from XML data using QUERY and VALUE methods: `ApplySimple("cast(#0.query(''/book/title') as nvarchar(250))", Book@XML)`

```sql
select a11.BOOK_ID  BOOK_ID,
       cast(a11.BOOK_XML.query(''/book/title') as nvarchar(250))  CustCol_1
from  LU_BOOK a11
```

- `Max(ApplySimple("#0.value(''/book[1]/title[1]', 'varchar(100)')", Book@XML)) {~}`

```sql
select a11.BOOK_ID  BOOK_ID,
       a11.BOOK_XML.value(''/book[1]/title[1]', 'varchar(100)')  WJXBFS1
from  LU_BOOK a11
```
Integration with Table-Valued User-Defined functions

**Users can provide input parameters to answer analytical questions of the data warehouse**

- SQL Server supports table-valued user-defined functions in which users can provide input parameters to answer analytical questions of the data warehouse.

- Table-valued functions are similar to stored procedures but unlike stored procedures can be referenced in a FROM clause much like a table.

Use a table-valued user-defined function in MicroStrategy with a Free Form SQL report:

```sql
CREATE FUNCTION EmployeesWithYearsExperience(@years int)
RETURNS @YearsExperience TABLE (  
    EmpId    int     primary key NOT NULL,
    EmpName  nvarchar(100) NOT NULL,
    HireDate date     NOT NULL,
    YearsExperience int NULL
  )
AS BEGIN
  INSERT INTO @YearsExperience (EmpId, EmpName, HireDate, YearsExperience)  
  SELECT [EMP_ID],
    [EMP_FIRST_NAME] + ' ' + [EMP_LAST_NAME],
    [HIRE_DATE],
    DATEDIFF (year, [HIRE_DATE], GETDATE())
  FROM   LU_EMPLOYEE
  WHERE  DATEDIFF (year, [HIRE_DATE], GETDATE()) >= @years;
  RETURN;
END;
```
Access Data To/From Linked Servers

**SQL Server Feature**

- Access data just not stored on the local SQL Server. The remote linked servers can be other SQL Server databases or non-SQL Server databases.
- The SQL Server syntax for referencing a table in a linked server is:
  \[
  \text{LinkedServerName..TableOwnerName.TableName}
  \]
- **Example:** If your linked server name is SQL2 and you have a table named LU_PROD owned by SCOTT, a view in the SQL Server database can be created as this:

```sql
CREATE VIEW LU_PROD AS
SELECT *
FROM SQL2..SCOTT.LU_PROD
```
Use Pass-through Authentication to Log On Securely

Single-Sign-On using Kerberos Integrated Authentication

- Microsoft (in its OS as well as SQL Server) configures Kerberos Integrated Authentication as default authentication mechanism.
  - **Benefit** – no unnecessary and potentially unsecure UID/PWD
- MicroStrategy supports integrated authentication to SQL Server – in the typical setup MicroStrategy will connect using the security context (user account) Intelligence Server is running at and execute all queries under that account.
  - This allows efficient report caching on the middle tier.

Some customers implement data security on the database layer, which requires MicroStrategy to run queries under the end user account. MicroStrategy supported this for a long time using UID/PWD.

---

**Diagram Description**

1. **MicroStrategy Web Client**: UID/PWD entered
2. **Directory**: Authenticated against user store
3. **Validate Token**: User Info passed
4. **Profile Passed**: Web Server (MicroStrategy Web)
5. **Profile Passed**: MicroStrategy Intelligence Server

**Flow**: MicroStrategy Web Client → Directory → Validate Token → Profile Passed → MicroStrategy Intelligence Server
**SSO Available with Intelligence Server on Linux/Windows**

*MicroStrategy 10.2 enables SSO with Intelligence Server on Linux*

- Starting with MicroStrategy Secure Enterprise 10.2, we brought along a **huge enhancement**.
- Now users will be able to use Kerberos integrated authentication to access databases when the MicroStrategy Intelligence Server is running on both Linux/Unix operating systems and Windows operating system.

- Refer to the MicroStrategy Community website [http://community.microstrategy.com](http://community.microstrategy.com) : TN272948 for more details.
Summary

• MicroStrategy and Microsoft continue to have a strong partnership. We work together to further optimize our integration to provide a seamless reporting experience.

Call-to-Action:

• Refer to existing best practices for developing MicroStrategy applications apply. Please see our detailed integration paper in the MicroStrategy Community Knowledge Base: TN48017

• Make sure to take advantage of DB features designed for analytical workloads

• Look for best practices to take advantage of data source strengths in MicroStrategy Community

• MicroStrategy customer requests / requirements should be submitted to the http://community.microstrategy.com website under the “Ideas” section.

• Contact Information: Farah Omer – fomer@microstrategy.com
Questions