Oracle Database and MicroStrategy 10: A functional overview including recommendations for performance optimization
MicroStrategy and Oracle Have A Close Partnership

• MicroStrategy is an Oracle Gold certified partner
• Significant number of joint customers
• MicroStrategy participates in Oracle’s beta programs
• Oracle is one of the first tier platforms for MicroStrategy to deploy new features
• Weekly calls between Oracle and MicroStrategy
MicroStrategy Data Access Workflows

There are numerous ways for MicroStrategy to interact with Oracle

- **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 requests are 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
Push-down Analytics send analytical queries to Oracle

Key technical characteristics of Modeled Access
- Business questions cannot be answered by a single query
- Most queries access vast amounts of data

Challenges
- Interactive analysis demands fast query execution
- Data warehouse database platform are often IO bound
- Data transfer between data warehouse and BI server can be slow
- Multiple workload competing for resources

MicroStrategy and Oracle work together to tackle challenges
- MicroStrategy formulates “good queries”
- Oracle executes queries well
MicroStrategy and Oracle work together to tackle the challenges

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster data processing</td>
<td>Native Oracle SQL and Function pushdown, Multi-Pass SQL, Global Optimization, Parallel Execution</td>
</tr>
<tr>
<td>Remove IO bottleneck</td>
<td>• Base tables: In-Memory, Flash Cache, Partitioning, Indexing, Zone Maps</td>
</tr>
<tr>
<td></td>
<td>• Intermediate Tables: Derived tables, Global Temporary Tables, Regular Tables</td>
</tr>
<tr>
<td>Utilize network bandwidth</td>
<td>Concurrent database Connections, Parallel SQL Execution, Parallel Data Retrieval</td>
</tr>
<tr>
<td>Prioritize Workload</td>
<td>Workload Management</td>
</tr>
</tbody>
</table>
Leverage database analytics functions

MicroStrategy uses Oracle’s functions whenever possible

- As of now, MicroStrategy pushes 120+ functions to Oracle database
  - Or functions will be calculated on MSTR server
- We review new Oracle functions as soon as they are in Beta
- Users can use MicroStrategy user defined functions to utilize Oracle new functions
- Cubes defined in Oracle can be directly accesses by MicroStrategy
Multi-pass SQL for Analytical Sophistication

Multi-pass SQL optimized for Oracle database

- MicroStrategy provides analytical richness using Multi-Pass SQL
  - Result of a SQL construct used as input for the following
  - Temporary result is truly temporary in nature

- MicroStrategy can handle intermediate data results in multiple ways
  - Common Table Expression
  - Global Temporary Tables

- Temporary tables are required to implement some MicroStrategy functionality
  - Middle-tier processing of data (Analytical Engine)
  - MicroStrategy Partitioning feature
  - In case of sub-optimal execution plans for CTE
SQL Engine generates high quality SQL statements

MicroStrategy optimizes SQL generation with Global Optimization

- Eliminates unused and duplicate SQL passes
- Push Metric Filter Conditions into fact expression to reduce SQL passes
- Combines SQL passes if Metrics on the same fact table
  - Multiple SQL are combined into one if only select, where, or group by clause are different
- Combine SQL passes with set operations (minus, union, intersect)

<table>
<thead>
<tr>
<th>Report</th>
<th>Execution Time Reduction</th>
<th>SQL Pass Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Percent Reduction</td>
<td>Time Before / After</td>
</tr>
<tr>
<td>Report 1</td>
<td>67%</td>
<td>Before: 24 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>After: 8 sec</td>
</tr>
<tr>
<td>Report 2</td>
<td>60%</td>
<td>Before: 20 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td>After: 8 min</td>
</tr>
<tr>
<td>Report 3</td>
<td>56%</td>
<td>Before: 9 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>After: 4 sec</td>
</tr>
</tbody>
</table>

**SQL Global Optimization**

- Level 0: No optimization
- Level 1: Remove Unused and Duplicate Passes
- Level 2: Level 1 + Merge Passes with Different SELECT
- Level 3: Level 2 + Merge Passes, which only hit DB tables, with Different WHERE
- Level 4: Level 2 + Merge All Passes with Different WHERE
- Level 5: Level 2 + Merge All Passes, which hit the same warehouse fact tables
Utilize Oracle’s Parallel Query Execution

Apply Parallel Execution Best Practices

• Enable Parallel Query Execution in Oracle
  • Use PARALLEL_DEGREE_LIMIT to prevent excessive parallelism
  • Use Resource Plan to control DOP for different workloads
  • Set PARALLEL_DEGREE_POLICY=AUTO for Parallel Statement Queuing and in-memory parallel execution features
• Make sure the object statistics are current
• Collect system statistics so the optimizer knows the system’s capacity
  
    \texttt{dbms\_stats.gather\_system\_stats}

• Don’t hard code the Degree of Parallelism (DOP) on table level;
Utilize Oracle’s Parallel DML Execution

Parallel DML is not enabled by default

Enable parallelism Insert and Update (DML)

- Parallel DML is not enabled by default
- Use “alter session enable parallel dml” as MicroStrategy Pre-Statement
- Parallel DML has to commit before subsequent query to start
- Be aware that insert and select phase may use different level of parallelism in “Insert as select” statement
- Use ON COMMIT PRESERVE ROWS if use Global Temporary Table with parallel DML
MicroStrategy and Oracle work together to tackle the challenges

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster data processing</td>
<td>Multi-Pass SQL, Global Optimization, Native Oracle SQL and Function pushdown, Parallel Execution</td>
</tr>
<tr>
<td>Remove IO bottleneck</td>
<td>• <strong>Base tables:</strong> In-Memory, Flash Cache, Partitioning, Indexing, Zone Maps</td>
</tr>
<tr>
<td></td>
<td>• <strong>Intermediate Tables:</strong> Derived tables, Global Temporary Tables, Regular Tables</td>
</tr>
<tr>
<td>Utilize network bandwidth</td>
<td>Concurrent database Connections, Parallel SQL Execution, Parallel Data Retrieval</td>
</tr>
<tr>
<td>Prioritize Workload</td>
<td>Workload Management</td>
</tr>
</tbody>
</table>
Remove IO Bottleneck for Reading (1)

Intelligent Table Indexing and Partitioning to eliminate unnecessary IOs

• MicroStrategy transparently takes advantage of indexes, partitions, and Zone Maps defined on fact tables
• Additionally, MicroStrategy generates primary indexes on intermediate tables
  • System administrator can weigh columns and control the size of an index for a particular report
  • Intermediate Tables can be indexed to match fact tables which minimizes database processing that would be required to repartition the temp table to match the fact table primary index
• MicroStrategy can automatically partition the result table for faster data process
Remove IO Bottleneck for Reading (2)

Utilize Oracle new features to improve read IO performance

- **Tables can be put in RAM for higher bandwidth**
  
  Alter table table_name inmemory (12c in-memory option)
  Alter table table_name storage (buffer_pool keep)

- **Tables can be cached in Flash Cache for faster IOs** (Exadata feature)
  
  Alter table table_name storage(flash_cache keep)

- **Tables can be compressed for less IOs**
  
  Alter table table_name Compress for query high (Exadata)
  Alter table table_name memcompress for query high (in-memory option)
Pick right compression scheme for your application

Balance the cost and benefit

Size in memory (MB)

Loading data into memory (MB/Sec)
Remove IO Bottleneck for Writing

MicroStrategy offers different ways to handle intermediate data

**Derived Table:** (Only visible in Statement level)

With pa1 as (select
  a12.SUBCAT_ID SUBCAT_ID,
  a13.YEAR_ID YEAR_ID,
  sum(a11.TOT_UNIT_SALES) WJXBFS1
from ITEM_MNTH_SLS a11
join LU_ITEM a12
on (a11.ITEM_ID = a12.ITEM_ID)
join LU_MONTH a13
on (a11.MONTH_ID = a13.MONTH_ID)
group by a12.SUBCAT_ID,
  a13.YEAR_ID)

select pa1.SUBCAT_ID SUBCAT_ID,
  a11.SUBCAT_DESC SUBCAT_DESC,
  pa1.YEAR_ID YEAR_ID,
  pa1.WJXBFS1 WJXBFS1,
  pa2.WJXBFS1 WJXBFS2
from pa1,
...
join (select...
  ) pa2
on (pa1.SUBCAT_ID = pa2.SUBCAT_ID and
  pa1.YEAR_ID = pa2.YEAR_ID)
join LU_SUBCATEG a11
on (pa1.SUBCAT_ID = a11.SUBCAT_ID)

**Implicit Table:**

create table ZZSP00 as ( select
  a13.YEAR_ID YEAR_ID,
  a12.SUBCAT_ID SUBCAT_ID,
  sum(a11.TOT_UNIT_SALES) WJXBFS1
from ITEM_MNTH_SLS a11
join LU_ITEM a12
on (a11.ITEM_ID = a12.ITEM_ID)
join LU_MONTH a13
on (a11.MONTH_ID = a13.MONTH_ID)
group by a13.YEAR_ID,
  a12.SUBCAT_ID

**Explicit Table (recommended for flexible controls):**

create table ZZSP00 (Year_ID number, Subcat_id number, Tot_unit_sales number) nologging parallel;

Insert into ZZSP00
select a13.YEAR_ID YEAR_ID,
  a12.SUBCAT_ID SUBCAT_ID,
  sum(a11.TOT_UNIT_SALES)
from ITEM_MNTH_SLS a11
join LU_ITEM a12
on (a11.ITEM_ID = a12.ITEM_ID)
join LU_MONTH a13
on (a11.MONTH_ID = a13.MONTH_ID)
group by a13.YEAR_ID,
  a12.SUBCAT_ID
Remove IO Bottleneck for Writing (2)

Choose between regular table and Global Temporary Tables

- Write IOs are expensive
- Oracle write data to data files (mirrored), undo (mirrored), redo (multiplexed)

<table>
<thead>
<tr>
<th></th>
<th>Regular Table</th>
<th>Global Temporary Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize Write IO</td>
<td>No (Undo will be written to Redo even table is defined as no-logging)</td>
<td>Yes (12c) Temp Undo will not be written to Redo</td>
</tr>
<tr>
<td>Parallel Execution</td>
<td>Yes, Parallel Query, Insert, Update, and Delete</td>
<td>Partially, Parallel Query and Insert after 11.2.0.3</td>
</tr>
<tr>
<td>Compression</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>In-Memory</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Partitioning</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
MicroStrategy and Oracle work together to tackle the challenges

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster data processing</td>
<td>Multi-Pass SQL, Global Optimization, Native Oracle SQL and Function pushdown, Parallel Execution</td>
</tr>
<tr>
<td>Remove IO bottleneck</td>
<td>• Base tables: In-Memory, Flash Cache, Partitioning, Indexing, Zone Maps</td>
</tr>
<tr>
<td></td>
<td>• Intermediate Tables: Derived tables, Global Temporary Tables, Regular Tables</td>
</tr>
<tr>
<td>Utilize network bandwidth</td>
<td>Concurrent database Connections, Parallel SQL Execution, Parallel Data Retrieval</td>
</tr>
<tr>
<td>Prioritize Workload</td>
<td>Workload Management</td>
</tr>
</tbody>
</table>
MicroStrategy and Oracle work together to tackle the challenges

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster data processing</td>
<td>Multi-Pass SQL, Global Optimization, Native Oracle SQL and Function pushdown, Parallel Execution</td>
</tr>
</tbody>
</table>
| Remove IO bottleneck        | • Base tables: In-Memory, Flash Cache, Partitioning, Indexing, Zone Maps  
                               | • Intermediate Tables: Derived tables, Global Temporary Tables, Regular Tables |
| Utilize network bandwidth   | Concurrent database Connections, Parallel SQL Execution, Parallel Data Retrieval |
| Prioritize Workload         | Workload Management                                                          |
**Integration with Oracle Workload Management**

Prioritize Workload

Workload Management (WLM) is necessary to optimize access to shared 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

Both MicroStrategy and Oracle provide WLM
MicroStrategy Allows Prioritizing Workload In Many Ways

Make sure the high priority jobs are sent to database first

1. User Groups
   - Developer
   - Analyst
   - User
   - CEO
   - VP
   - Director

2. Application Type
   - Windows
   - Email
   - CRM
   - Web
   - Mobile

3. BI Project
   - Sales Project
   - Financial Project

4. Request Type
   - Sales Report
   - Expense Report
   - CRM Inquiry
   - Product Inquiry

5. Cost
   - 260
   - 840
   - 10
   - 20
   - 440
   - 930

Database

Database
Integration with Oracle Workload Management

Allocate database resource to MicroStrategy

- Oracle WLM can recognize MicroStrategy workload based on connection information
- Oracle WLM can allocate resource to MicroStrategy among all database workloads
- Oracle needs MicroStrategy to feed further information to Prioritize different types of jobs within MicroStrategy

Use Pre-Statement to provide Application Info:

Exec DBMS_APPLICATION_INFO.SET_MODULE ("Financial Reports")
Exec DBMS_APPLICATION_INFO.SET_ACTION ("Element Browsing")
Optimizing your connection to Oracle

Use MicroStrategy’s Connectivity Wizard to enable the optimized and supported parameters for your Oracle database connection.

The Wizard enables support for important features:

- Bulk insert support for database write-back operations
- N data type support
MicroStrategy works with Oracle Cloud

MicroStrategy can connect with Oracle Cloud with built-in Security

![Oracle Database Cloud Service](image-url)
MicroStrategy Unique Optimizations for Oracle

Technical Summary

• Multi-pass SQL for analytical sophistication
  – Use of temporary or permeant tables to store intermediate results
  – Use of derived tables
  – Global optimization
  – Control of primary indexes and statistics collection

• Oracle-specific SQL syntax
  – Analytical functions (OLAP functions)
  – Subqueries
  – OLAP cubes
  – Global Temporary Tables
  – Bulk inserts

• Seamless support for Oracle Performance features
  – Parallel execution
  – In memory option
  – Compressions
  – Workload Management

• Extensions to Oracle functionality
  – Aggregate awareness with physical summary tables
  – Middle-tier computation of calculations not available in Oracle
  – Middle-tier caching via Intelligent Cubes
  – Parallel data transfer
  – Connectivity optimization
Summary

- MicroStrategy and Oracle database continue to have a strong partnership
- Multi-faceted technical integration of products
- Continued optimization provides a seamless Data Analytic experience
Questions