Abstract

This white paper explains how MicroStrategy can be configured and used with Greenplum database in two-tier and basic three-tier architecture. This document provides a quick verification and validation of connectivity and interoperability of MicroStrategy with Greenplum. This abstract appears as the online abstract for EMC.com/Powerlink.

July 2011
Copyright © 2011 EMC Corporation. All Rights Reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided “as is.” EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com.

VMware, and are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. All other trademarks used herein are the property of their respective owners.

Part Number hXXXX
Table of Contents

Executive summary ................................................................................................................. 4
  Audience .......................................................................................................................... 4
Organization of this paper ..................................................................................................... 4
MicroStrategy ....................................................................................................................... 5
  Important MicroStrategy Features ..................................................................................... 6
  Installation and configuration of ODBC driver ................................................................. 7
  Compatibility Matrix ......................................................................................................... 8
Pitfalls and best practices ....................................................................................................... 9
Conclusion ........................................................................................................................... 12
References ........................................................................................................................... 12
Reference VLDB settings ...................................................................................................... 13
Executive summary

MicroStrategy is a popular Business Intelligence (BI) enterprise reporting tool that includes reporting, dashboard and advanced OLAP features. MicroStrategy provides an integrated platform for supporting business intelligence applications that leverage data warehouse databases such as Greenplum. It includes tools that support creation of metadata (MD), connectivity tools, office integration suite and report designing tools among others. MicroStrategy is connected to Greenplum through ODBC drivers supplied by Data Direct.

This paper primarily discusses two-tier architecture setup to enable field engineers to perform basic validation. MicroStrategy tool can be also configured to support 3-tier architecture which supports many advanced features such as OLAP cubes, data caching mechanism and enterprise level administration and scalability. Additional details on three-tier architecture are also included to guide field personnel.

Audience

This white paper is intended for EMC Greenplum field-facing employees such as sales, technical consultants, support, as well as customers who are using the MicroStrategy. This document is neither an installation guide nor an introductory material on MicroStrategy. That type of material is available elsewhere.

It documents the MicroStrategy connectivity and operation capabilities, and shows the readers how it can be used in conjunction with Greenplum to retrieve and present data to users. Though the reader is not expected to have any prior MicroStrategy knowledge, basic understanding of BI reporting tools such as MicroStrategy would help them understand better.

Organization of this paper

This paper walks through the following topics:

- MicroStrategy
- Important MicroStrategy Features
- Installation and configuration of ODBC driver
- Compatibility matrix
- Pitfalls and best practices
- Very Large Database (VLDB) settings in MicroStrategy three-tier architecture
- Specific scenarios for better operability with Greenplum
**MicroStrategy**

MicroStrategy is one of the most popular Business Intelligence (BI) reporting tools in the market that extends functionality to include ROLAP (Relational Online Analytical Processing). This means, a complex relational database structure can be presented as a multidimensional cube that is easy for end user and business user to understand easily. Like any BI reporting tool, MicroStrategy generates SQL for running against Greenplum and this SQL code looks identical to any SQL based tool. MicroStrategy performs certification tests internally in their company and has provided excellent support.

MicroStrategy maintains a metadata dictionary to store object information related to reports. Just like most BI tools store application objects and other relevant particulars in a meta data, MicroStrategy does the same using databases such as Microsoft Access and SQL server. MicroStrategy does not depend on either star schema or relational database normal forms for implementation of either metadata or its reports. It uses user designed model to capture hierarchies and relationships between various objects within MicroStrategy, which are typically modeled differently compared to database schema objects. Note that MicroStrategy uses rather confusing terminology for its internal objects; for instance, an attribute in MicroStrategy is same as an entity in entity relationship diagram.

- **MicroStrategy reporting components:** MicroStrategy organizes all objects under project source (indicates connection to a metadata database such as SQL server or Microsoft Access database and MicroStrategy Intelligence Server or I-Server) and projects (refers to data warehouse database connection and related objects or reference to database schema). Note that three-tier architecture uses Intelligence server which allows many advanced features including access to VLDB (Very Large Database) settings.

- **Different Very large database (VLDB) settings** are targeted for different databases. Over 100 VLDB settings are available for customizing behavior of MicroStrategy engine for SQL code generation.

- **A simplified architecture diagram of MicroStrategy** is below. There is no intelligence server in two-tier architecture. As mentioned above, meta data database can be either Microsoft SQL server or Access, not Greenplum database.
  - Project: refers to database and its objects or schema.
  - Project source: refers to both I-server (3-tier) and Meta data database.
  - Connectivity: through ODBC drivers from Data Direct.
Important MicroStrategy Features

MicroStrategy has a number of advanced features that benefit usability of the product with superior performance. Important ones are listed below:

- **Aggregate awareness**: MicroStrategy automatically directs pre-aggregated tables (if they exist) for compute intensive queries involving aggregation. For instance, if an item such as total sales is located in more than one table, MicroStrategy can use smallest table to fetch this information as this results in quicker database response.

- **Multi pass SQL** as opposed to a complex SQL involving many sub queries resulting in an efficient execution of queries or reports. In some cases, it is not possible to generate SQL without a multi pass to answer a business question. For instance, if a report has two metrics (data that can use SQL sum or count) coming from two different fact tables, MicroStrategy uses at least two SQL passes while running a report.

- **Application partitioning** allows partitioning of data logically at application side. This way, running a report results in more restrictive (where clause condition) as opposed to full parsing of data.

- **Various VLDB settings**, including pre and post conditions before and after SQL is run against database. Within these pre and post, a few levels exist; meaning, we can run different pre settings before running a SQL. These are typically used for passing to database with ‘alter session’ command.
  - VLDB settings can be accessed at project level or at report level, depending on desired way. Here is a way to access VLDB settings for a project:
    - Start MicroStrategy desktop and login to a project source.
    - Right click on the project and choose project configuration.
    - Expand data instances tree on the left. You will see this dialog box, click on VLDB settings for project level.
Custom SQL and pass through SQL generation.
OLAP cubes that can be entirely based on memory.
Meta data that is very scalable and can reach beyond 1TB.
Reports can consist of documents with customized HTML format.
MicroStrategy Intelligence Server (IS) allows threads with different priority and may be based on cost and other application factors. This can be related back to our Greenplum database scheduler.
Two popular forms of query generation is by using derived tables or true temp tables.

Installation and configuration of ODBC driver
Launch ODBC administrator and configure as shown below. ODBC driver for Greenplum is supplied by data direct. MicroStrategy also provides a driver and this may not be used for connecting to Greenplum. Please make note of 64-bit and 32-bit drivers based on your environment. For Linux, there no longer support for 32-bit MicroStrategy application and thus you are limited to 64-bit application.
Steps for ODBC configuration are self explanatory with screen shot images below:

Open driver set up window to check connectivity by clicking ‘Test Connect’ button.

**Compatibility Matrix**
Here is a short list of Greenplum version compatibility with MicroStrategy running on different operating system platforms.
### Greenplum Database Partner Interoperability

This table contains Certified solutions for Greenplum Database Partner Interoperability support.

<table>
<thead>
<tr>
<th>Information Management Software</th>
<th>Database</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>MicroStrategy 9.0.2</td>
<td>EMC Greenplum 4.0</td>
<td>HP-UX 11i: v2 (HP-UX 11.23), v3 (HP-UX 11.31)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM AIX: 5., 5.2, 5.3, 6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Windows 2003 and 2008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oracle Solaris 10: SPARC, x86; Oracle Solaris: 8 SPARC, 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Various LINUX distributions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[For complete list, please see references towards end of this document]</td>
</tr>
<tr>
<td>MicroStrategy: 8.1.1, 8.1.2, 9.0.0</td>
<td>EMC Greenplum 3.0</td>
<td>HP-UX 11i: v2 (HP-UX 11.23), v3 (HP-UX 11.31)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM AIX: 5., 5.2, 5.3, 6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Windows 2003 and 2008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oracle Solaris 10: SPARC, x86; Oracle Solaris: 8 SPARC, 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Various LINUX distributions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[For complete list, please see references towards end of this document]</td>
</tr>
<tr>
<td>MicroStrategy: 8.1.1, 9.0.0, 9.0.1</td>
<td>EMC Greenplum 3.2</td>
<td>Hewlett Packard HP-UX 11i: v2 (HP-UX 11.23), v3 (HP-UX 11.31); IBM AIX: 5.1, 5.2, 5.3, 6.1; Oracle Solaris 10: SPARC, x86; Oracle Solaris: 8 SPARC, 9 SPARC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[For complete list, please see references towards end of this document]</td>
</tr>
<tr>
<td>MicroStrategy: 8.1.1, 9.0.1</td>
<td>EMC Greenplum 3.3</td>
<td>HP-UX 11i: v2 (HP-UX 11.23), v3 (HP-UX 11.31)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM AIX: 5.1, 5.2, 5.3, 6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Windows 2003 and 2008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oracle Solaris 10: SPARC, x86 Oracle Solaris: 8 SPARC, 9 SPARC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[For complete list, please see references towards end of this document]</td>
</tr>
<tr>
<td>MicroStrategy: 8.1.2, 9.0.0, 9.0.1</td>
<td>EMC Greenplum 3.1</td>
<td>HP-UX 11i: v2 (HP-UX 11.23), v3 (HP-UX 11.31)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM AIX: 5.1, 5.2, 5.3, 6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Windows 2003 and 2008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oracle Solaris 10: SPARC, x86 Oracle Solaris: 8 SPARC, 9 SPARC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[For complete list, please see references towards end of this document]</td>
</tr>
<tr>
<td>MicroStrategy: 9.0.0, 9.0.1</td>
<td>EMC Greenplum 3.1</td>
<td>Oracle Solaris 10: SPARC, x86 Oracle Solaris: 8 SPARC, 9 SPARC</td>
</tr>
<tr>
<td>MicroStrategy: 9.0.0, 9.0.1</td>
<td>EMC Greenplum 3.2</td>
<td>Different versions of Windows 2003 and 2008; Different distributions of LINUX.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[For complete list, please see references towards end of this document]</td>
</tr>
</tbody>
</table>

### Pitfalls and best practices

Here are some challenges while using MicroStrategy.

1. **Custom groups**: MicroStrategy typically generates two or more SQL statements for custom groups. As an example, give me top 10 customers and bottom 5 regions from sales for this year. This generates two SQL, one for top 10 customers and another for bottom 5 regions. If this involves any grouping or where direct querying is not possible, generates additional
SQL using temporary tables. If we are using these queries against a large fact table, this creates problems.

2) **MicroStrategy partition**: MicroStrategy supports partitioning at application side. If this feature is implemented, a look up partition table is created on the database (this holds just partition key values and the corresponding physical table) and individual partition tables that are segregated based on the partition key value. If we have 3 year data, it is possible to create three partitioned tables with just three entries in the partition look up table.

Please see reference item for more details on partitioning scheme.

Partitioning can only use true temporary (TT) tables, not derived tables. This means, report based on MicroStrategy partitioned tables will generate SQL code for true temporary tables. This SQL code uses a number of SQL passes or individual SQL queries to just support a single report. We advise partition implementation at database level and derived tables at MicroStrategy level to efficiently run reports. There may be special cases where true temp tables may be beneficial, in such cases, that report may be run using this setting.

3) **Aggregate tables**: MicroStrategy recommends using aggregate tables. For instance, we have a fact table on Greenplum database that has a large number of transactions or entries for a day. Aggregate table can group them on a daily basis. MicroStrategy is “aggregate aware” and thus able to rewrite a query to make use of aggregate table instead of underlying big fact table. This enhances performance.

4) **Cross joins**: Check for cross joins between a fact able and a dimension table (or) between dimension tables. Most of them time this does not make sense from database side. Just to clarify, if a table has 10 rows, another table has 22 rows, cross join will result in 220 rows by joining all rows.

5) **VLDB settings**: Databases nowadays can generate better SQL code than that can be obtained from MicroStrategy by changing VLDB settings (see below for “Derived Vs True Temp Tables”).

6) **Derived Vs True Temp Tables**: This is just another technical jargon. What this means is that a big SQL statement is divided into smaller sub queries to achieve the same. The big SQL statement is referred to as derived table; the smaller equivalent code snippet ones are known as true temp tables. Derived tables are preferred in most databases, if result data set or rows returned (or rows generated during intermediate steps) is not too large.

If MicroStrategy application side partitions are used, we can’t choose SQL processing with derived tables as partitioning at MicroStrategy has to use temporary tables for processing. Derived tables or true temporary tables can be set either at report level or at project level. It is also possible to use both combinations within a project using a fall back mechanism.
7) **Data distribution**: Data may be distributed in Greenplum using either explicit columns for distribution (or) not specifying any column (in this case, either primary key or first column is used by Greenplum for choosing distribution key) (or) random distribution of data. These settings are not only important for distribution of data as such, but also for distribution of intermediate data as queries are getting executed. How do we set these values is specific to a particular database implementation.

These are some of the best practices, required further assessment:
1) Limit number of parallel MicroStrategy threads or sessions between 10 and 40.
2) Though we may use resource manager at both MicroStrategy and Greenplum to allocated different resources such as CPU, it best left to Greenplum database as this is where most of the data activity happens.
3) Different settings may be done prior to execution of a report of a query through pre setting under MicroStrategy VLDB settings. For instance, we can change a threshold value for broadcasting of a table (treating like a dimension table) as opposed to a fact table by using “gp_segments_for_planner”.

8) **Pre / Post settings**: Most databases support ‘alter session’ command to change or pass certain parameters at session level. This can be achieved in MicroStrategy with this setting.

9) **Greenplum query support**: Certain queries such as correlated sub queries (CSQ) and multi-row sub queries are not fully supported with Greenplum. To overcome such short comings, we can use ‘sub query type’ setting at VLDB setting. As pointed out earlier, make changes at report level to see how this affects a report rather than making changes at project level.
To check effectiveness, uncheck the default check box. Use different radio buttons to see how a query gets executed.

10) **SQL view:** For any report, we can see SQL that is generated for that report. This can be done through report main window and choosing view menu. You are presented with design, grid and SQL views.

11) **Greenplum Database and SQL trace logs:** One can monitor database logs at the master pg_log to check how queries are getting executed. To enable better monitoring one way use SQL trace logs at windows.

![ODBC Data Source Administrator](image)

**Conclusion**

This document outlines useful topics related to MicroStrategy and its interoperability with Greenplum. Though MicroStrategy is matured enterprise software with many advanced and complex features, a few important items can be checked and verified for effective implementation. Your feedback is very important to make this document useful to all interested parties.

**References**

3. MicroStrategy support site: [https://resource.microstrategy.com/support/](https://resource.microstrategy.com/support/)
Reference VLDB settings

Rarely documented elsewhere or available from MicroStrategy support sites. These are also not listed in their documentation. These are typical settings, not recommended settings as they vary based on implementation.

Freeform SQL
Ignore Empty Result for Freeform SQL (FF Ignore Empty Result) = 0 (Do not turn off warnings for freeform SQL statements with empty results, such as updates)

Governing
Maximum SQL/MDX Size (Maximum SQL Size) = 65536
SQL Time Out (Per Pass) (Report Time Out) = 0

Indexing
Allow Index On Metric (Allow Index On Fact) = 0 (Don't allow the creation of indexes on metric columns)
Secondary Index Type (Create Individual Index) = 0 (Create Composite Index for Temporary Table Column Indexing)
Index Post String (Index PostString) =
Index Prefix (Index Prefix) =
Index Qualifier (Index Qualifier) =
Intermediate Table Index (Intermediate Table Index) = 1 (Create partitioning key (typically applicable to MPP systems))
Max Columns in Index (Max Columns in Index) = 0
Primary Index Type (Primary Index Control) = 1 (Create primary index/partitioning key (where applicable) if the intermediate table index setting is set to create a primary index)
Secondary Index Order (Secondary Index Order) = 0 (Create index after inserting into table)

Joins
Base Table Join for Template (Base Table Join for Template) = 0 (Temp table join)
Cartesian Join Warning (Cartesian Join Warning) = 0 (Execute)
Cartesian Join Evaluation (Cross Join Order) = 0 (Do not reevaluate cartesian joins.)
Downward Outer Join Option (Downward Outer Join) = 0 (Do not preserve all the rows for metrics higher than template level)
DSS Star Join (DSS Star Join) = 0 (No star join)
From Clause Order (Fact Table Switch) = 0 (Normal FROM clause order as generated by the engine)
Full Outer Join Support (Full Outer Join Support) = 1 (Support)
Preserve all final pass result elements (Incomplete Lookup Table) = 0 (Preserve common elements of final pass result table and lookup/relationship table.)
Attribute to join when key from neither side can be supported by the other side (Join Attribute Option) = 0 (Join common key on both sides)
Join Type (Join Type) = 1 (Join 92)
Lookup Table Join Order (Lookup Table Join Order) = 0 (Partially based on attribute level (behavior prior to 8.0 SP1 release))
Max Tables In Join (Max Tables In Join) = 0
Max Tables in Join Warning (Max Tables in Join Warning) = 1 (Cancel execution)
Nest Aggregation Outer Join (Nested Aggregation Outer Join) = 0 (Do not perform outer join on nested aggregation)
Preserve all lookup table elements (Outer Join To Lookup Table) = 0 (Preserve common elements of lookup and final pass result table.)

MDX
MDX Add Fake Measure (MDX Add Fake Measure) = 1 (Add a fake measure to an Attribute-only MDX report)
MDX Level Number Calculation Method (MDX Add Generation Property) = 0 (Use actual level number)
MDX Add Non Empty (MDX Add Non Empty) = 1 (Add the non empty keyword in the MDX select clause only if there are metrics on the report)
MDX Cell Formatting (MDX Cell Formatting) = 0 (MDX metric values are formatted per column)
MDX Non Empty Optimization (MDX Non Empty Optimization) = 0 (No nonempty optimization)
Format for date/time values coming from data source (MDX Property Key Date Format) = DD.MM.YYYY
MDX TopCount Support (MDX TopCount Support) = 1 (Use TopCount instead of Rank and Order)
MDX Verify Limit Filter Literal Level (MDX Verify Limit Filter Literal Level) = 0 (Do not verify the level of literals in limit or filter expressions)

Metrics
Absolute Non-Agg Metric Query Type (Absolute Non-Agg Metric Query Type) = 0 (Use subquery)
Compute Non-Agg before/after OLAP Functions (e.g. Rank) Calculated in Analytical Engine (Compute Non-Agg After Analytics) = 0 (Calculate non-aggregation before OLAP Functions/Rank)
COUNT Compound Attribute (COUNT Compound Attribute) = 0 (COUNT(Expression) enabled)
COUNT(column) Support (COUNT(column) Support) = 0 (Use COUNT(Column))
Default to Metric Name (Default to Metric Name) = 0 (Do not use the metric name as the default metric column alias)
Integer Constant In Metric (Integer Constant In Metric) = 0 (Add '.0' to integer constants in metric expressions.)
Max Metric Alias Size (Maximum Metric Alias Length) = 63
Metric Join Type (Metric Join Type) = 0 (Inner Join)
Non-Agg Metric Optimization (Non-Agg Metric Optimization) = 0 (Optimized for less fact table access.)
Null Check (Null Check) = 2 (Check for NULL in temp table join only)
Separate COUNT DISTINCT (Separate COUNT DISTINCTs) = 0 (One pass)
Zero Check (Zero Check) = 1 (Check for zero in all queries)

Pre/Post Statements
Drop Database Connection (Drop Database Connection) = 0 (Drop database connection after running user defined SQL)
Element Browsing Post Statement (Element Browsing PostStatement) =
Element Browsing Pre Statement (Element Browsing PreStatement) =
Insert Mid Statement 1 (Insert MidStatement 1) =
Insert Mid Statement 2 (Insert MidStatement 2) =
Insert Mid Statement 3 (Insert MidStatement 3) =
Insert Mid Statement 4 (Insert MidStatement 4) =
Insert Mid Statement 5 (Insert MidStatement 5) =
Insert Post Statement 1 (Insert PostStatement 1) =
Insert Post Statement 2 (Insert PostStatement 2) =
Insert Post Statement 3 (Insert PostStatement 3) =
Insert Post Statement 4 (Insert PostStatement 4) =
Insert Post Statement 5 (Insert PostStatement 5) =
Insert Pre Statement 1 (Insert PreStatement 1) =
Insert Pre Statement 2 (Insert PreStatement 2) =
Insert Pre Statement 3 (Insert PreStatement 3) =
Insert Pre Statement 4 (Insert PreStatement 4) =
Insert Pre Statement 5 (Insert PreStatement 5) =
Data mart SQL to be executed before inserting data (Insert SQLs) =
Cleanup Post Statement 1 (Post Cleanup Statement 1) =
Cleanup Post Statement 2 (Post Cleanup Statement 2) =
Cleanup Post Statement 3 (Post Cleanup Statement 3) =
Cleanup Post Statement 4 (Post Cleanup Statement 4) =
Cleanup Post Statement 5 (Post Cleanup Statement 5) =
Data mart SQL to be executed prior to data mart creation (Post SQLs) =
Data mart SQL to be executed after data mart creation (Pre SQLs) =
Report Post Statement 1 (Report PostStatement 1) =
Report Post Statement 2 (Report PostStatement 2) =
Report Post Statement 3 (Report PostStatement 3) =
Report Post Statement 4 (Report PostStatement 4) =
Report Post Statement 5 (Report PostStatement 5) =
Report Pre Statement 1 (Report PreStatement 1) =
Report Pre Statement 2 (Report PreStatement 2) =
Report Pre Statement 3 (Report PreStatement 3) =
Report Pre Statement 4 (Report PreStatement 4) =
Report Pre Statement 5 (Report PreStatement 5) =
Table Post Statement 1 (Table PostStatement 1) =
Table Post Statement 2 (Table PostStatement 2) =
Table Post Statement 3 (Table PostStatement 3) =
Table Post Statement 4 (Table PostStatement 4) =
Table Post Statement 5 (Table PostStatement 5) =
Table Pre Statement 1 (Table PreStatement 1) =
Table Pre Statement 2 (Table PreStatement 2) =
Table Pre Statement 3 (Table PreStatement 3) =
Table Pre Statement 4 (Table PreStatement 4) =
Table Pre Statement 5 (Table PreStatement 5) =

Query Optimizations
Additional Final Pass Option (Additional Final Pass) = 0 ((default) Final pass CAN do aggregation and join lookup tables in one pass)
Apply Filter Options (Apply Filter to All Passes) = 0 (Apply filter only to passes touching warehouse tables)
Count Distinct with Partitions (Count Distinct with Partitions) = 0 (Do not select distinct elements from each partition)
Custom Group Banding Count Method (Custom Group Banding Count Method) = 0 (Treat banding as normal calculation)
Custom Group Banding Points Method (Custom Group Banding Points Method) = 1 (Use standard case statement syntax)
Custom Group Banding Size Method (Custom Group Banding Size Method) = 0 (Treat banding as normal calculation)
Dimensionality Model (Dimensionality Model) = 0 (Use relational model)
Engine Attribute Role Options (Disable Engine Attribute Role) = 0 (Disable Engine Attribute Role Feature)
Attribute Element Number Count Method (Element Number Count) = 0 (Use Count(Attribute@ID) to calculate total
element number (will use count distinct if necessary))
  Unrelated Filter Options (Keep Unrelated Filter) = 0 (Remove unrelated filter.)
  MD Partition Prequery Option (MD Partition Prequery Option) = 0 (Use count(*) in prequery)
  Multiple data source support (Multiple data sources: Heterogeneous Database or Gateway) = 0 (Use Multisource Option to
  access multiple data sources)
  Remove Group By Option (Non-Groupby) = 0 (Remove aggregation and Group By when Select level is identical to From
  level)
  Data population for Intelligent Cubes (Normalization For Cube Report) = 1 (Normalize Intelligent Cube data in Intelligence
  Server)
  Data population for reports (Normalization For Normal Report) = 0 (Do not normalize report data)
  OLAP function support (OLAP New Behavior) = 0 (Preserve backwards compatibility with 8.1.x and earlier)
  Rank Method if DB Ranking Not Used (Rank Method) = 0 (Use ODBC ranking (MSTR 6 method).)
  Remove Aggregation method (Remove Aggregation method) = 0 (Remove aggregation according to key of FROM clause)
  Remove Repeated Tables For Outer Joins (Remove Repeated Tables For Outer Joins) = 1 (Enable optimization to remove
  repeated tables in full outer join and left outer join passes)
  SQL Global Optimization (Remove Unreferenced and Duplicates) = 4 (Level 4: Level 2 + Merge All Passes with Different
  WHERE)
  Set Operator Optimization (Set Operation) = 1 (Enable Set Operator Optimization (if supported by database and [Sub
  Query Type]))
  Sub Query Type (Sub Query Type) = 3 (WHERE (COL1, COL2...) IN (SELECT s1.COL1, s1.COL2...))
  Transformation Formula Optimization (Transformation Optimization) = 1 (Use transformation formula instead of join with
  transformation table when possible)
  Unrelated Filter Options For Nested Metrics (Unrelated Filter Options For Nested Metrics) = 0 (Use the 8.1.x behavior)
  WHERE Clause Driving Table (WHERE Clause Driving Table) = 1 (Use fact table)

Select/Insert
  Attribute Form Selection Option for Intermediate Pass (Attribute Form Select for Intermediate Pass) = 0 (Select ID form
  only)
  Attribute Selection Option for Intermediate Pass (Attribute Selection Option for Intermediate Pass) = 0 (Default) Select
  only the attributes needed
  Bulk Insert String (Bulk Insert String) =
  Custom Group Interaction With Report Filter (CG Custom Group Interaction With the Report Filter) = 0 (No interaction -
  static custom group)
  Constant Column Mode (Constant Column Mode) = 0 (Pure select, no group by)
  Datumart Column Order (Datumart Column Order) = 0 (Columns created in order based on attribute weight)
  SQL Date Format (Date Format) = mm/dd/yyyy
  Date Pattern (Date Pattern) = to_date('#0::text, YYYY/MM/DD')
  SQL Decimal Separator (Decimal Separator) = 0 (Use '.' as decimal separator (ANSI standard))
  Default Attribute Weight (Default Attribute Weight) = 0 (Highest weight)
  Disable Prefix in WH Partition Table (Disable Prefix in WH Partition Table) = 0 (Default) Use prefix in both warehouse
  partition pre-query and partition query
  Merge Same Metric Expression Option (Do Not Merge Same Metric Expression) = 0 (Merge same metric expression)
  GROUP BY ID Attribute (GROUP BY ID Attribute) = 0 (Group by expression)
  GROUP BY Non-ID Attribute (GROUP BY Non-ID Attribute) = 0 (Use Max)
  Insert Table Option (Insert Table Option) =
  Max Digits In Constant (Max Const Digits) = 0
  Insert Post String (Post Insert String) =
  Distinct/Group by option (when no aggregation and not table key) (SELECT DISTINCT at Same Level) = 2 (Use GROUP
  BY)
  Select Post String (SELECT PostString) =
  Select Statement Post String (SELECT PostString For Main SQL) =
  SQL Hint (SQL Hint) =
  SQL Timestamp Format (TimeStamp Format) = yyyy-mm-dd hh:mm:ss
  UNION Multiple INSERT (UNION Multiple INSERTs) = 0 (Do not use UNION)
  Long integer support (Use BigInt For Long Integer) = 0 (Do not use BIGINT)

Tables
  Attribute ID Constraint (Attribute ID Constraint) =
  Character Column Option (Character Column Option) =
  Column Pattern (Column Pattern) = #0.#1
Commit After Final Drop (Commit After Final Drop) = 0 (No commit after the final drop statement)
Commit Level (Commit Level) = 0 (No COMMIT)
Create Post String (Create PostString) =
Drop Temp Table Method (Drop Temp Table Method) = 0 (Drop after final pass)
Hexadecimal Character Transformation (Enable Unicode Support) = 0 (Do not apply hexadecimal character transformation to quoted strings)
FallBack Table Type (Fallback Table Type) = 1 (True temporary table)
Intermediate Table Type (Intermediate Table Type) = 3 (True temporary table)
Alias Pattern (Mid Name Alias Pattern) = AS
National Character Column Option (National Character Column Option) =
CREATE and INSERT support (Read-only DBRole) = 0 (CREATE and INSERT statements are supported)
Table Creation Type (Table Creation Type) = 1 (Implicit Table)
Table Descriptor (Table Descriptor) =
Table Option (Table Option) =
Table Prefix (Table Prefix) =
Table Qualifier (Table Qualifier) =
Table Space (Table Space) =
Maximum SQL Passes Before FallBack (Table Type Fallback Max Pass) = 0
Maximum Tables in FROM Clause Before FallBack (Table Type Fallback Max Table) = 0