SAP HANA Smart Data Access – Modeling Recommendations

Dhirendra Gehlot & Abani Pattanayak --- SAP HANA Consulting, Center of Expertise
June 2, 2016
Contents

- SAP Smart Data Access (SDA)
- SDA Modeling Recommendations
- SDA Learning Resources
- SAP HANA Modeling Generic Best Practice
SAP HANA Smart Data Access
Data virtualization for on-premise and hybrid cloud environments

Benefits
• Enables access to remote data access just like “local” table
• Smart query processing including query decomposition with predicate push-down, functional compensation
• Supports data location agnostic development
• No special syntax to access heterogeneous data sources

Heterogeneous data sources
• HANA, ASE, IQ, MaxDB, ESP, SQLA
• Teradata, Microsoft SQLServer, Oracle, IBM DB2, IBM Netezza
• Cloudera, Hortonworks, MapR
SAP HANA Smart Data Access Components

- Performance and query optimization
- Leverage remote compute engines
- Query monitoring and statistics
- Powerful transformations and information validation
- “Smart” placement and caching recommendations

Data Consumers

Transactions + Analytics Applications

Data virtualization Components

Relational Views
Data-type Transformation
In-Flight Data Cleansing & Transformation
Query Optimization
Data Statistics
Recommendation Engine
Data Caching
Built-In Adapters
Adapter SDK

Other data sources
(SDK, databases, files, web services, etc.)

Remote Data Sources

Teradata
Hadoop
Oracle
ASE etc.

Current
Future

© 2016 SAP SE. All rights reserved.
SAP HANA Smart Data Access
High Level Component Diagram

**SAP HANA Studio**
- Schema Management
- Configuration
- Security Management
- Query Monitoring
- Modeler
- Administration

**Federation Perspectives**
- Query optimization and execution
- Identifies query fragment which can be sent to the target for remote processing
- Performs functional compensation
- Adapter framework enables data type conversions
- Driver manager loads database specific drivers
- Access methods enable integration with other HANA query processing components

**SAP HANA**

**Persistence Layer**
- (Repository & Catalog)
  - HANA Views
  - Base Tables
  - Virtual Tables
  - Analytical Model
  - Federation Model

**Query Processing**
- Query Optimization
  - Remote Query Execution
  - Federation Support

**Federation Adapter Framework**
- Access Methods
- Data Type Conversions
- Virtual Access Layer
- Driver Manager
  - ASE ODBC Driver
  - Teradata .ODBC Driver

**Added for HANA Federation**
# SAP HANA Smart Data Access Benefits

<table>
<thead>
<tr>
<th><strong>Rapid Adoption</strong></th>
<th><strong>Open</strong></th>
<th><strong>In-Memory</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Start projects and develop applications quickly, in a non-disruptive way using existing infrastructure to gain competitive advantage</td>
<td>In line with the SAP HANA open platform approach use data from a variety of heterogeneous data sources in an easy non-disruptive manner</td>
<td>Bring in-memory computing to data virtualization. All the data stores benefit from the central in-memory core of HANA. And it is highly optimized meaning data access is fast</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Low Cost</strong></th>
<th><strong>Enterprise Analytics</strong></th>
<th><strong>Easy Adoption</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>No need to do load data from other data sources to start projects – saves time, effort, cost, is non-disruptive</td>
<td>External data source may have application built on them. SDA enables enterprise analytics without moving data.</td>
<td>Easy to setup via virtual tables. Then just start writing the applications. Quick time to market = competitive advantage</td>
</tr>
</tbody>
</table>
SAP Smart Data Access
Modeling Recommendations

- Encapsulate virtual tables in HANA Calculation view
  Encapsulate one or more virtual table (no native HANA tables in this view) in a calculation view. This calculation view must be set to execute in calculation view.

- Use constant mappings in the UNION step to efficiently prune the model at run time
  The constant mappings in the calculation view prune the model at run time. It is specially useful when HANA model combines data from multiple sources using SDA. First each data source can be qualified with a constant mapping. Next define a mandatory variable with default value to use HANA source. This would set the default query behavior to return data resides in HANA and would not run a query to other sources. Depending on users input, query can be run on HANA, external sources, or at all sources.

- Build HANA Models using UNION to make future update transparent to front end analytics
  Build HANA models using UNION of multiple data sources. This approach makes future model updates, specially data source changes, transparent to existing analytics and users and requires minimal update effort. For example, build HANA model to combling HANA and IQ data using UNION option. Later removing IQ source or adding Hadoop or other source can be done easily before the UNION layer. The data source change would not impact logic or reporting element from UNION layer onward. Thus making these changes transparent to existing analytics.
SAP Smart Data Access
Modeling Recommendations

• **Build Separate model for each remote source. Than UNION all models**
  This will allow join to take place at the data source and only result set will be moved to HANA for 'federation' thus giving better performance.
  For example, if some master data and transaction data resides in HANA, and some master data and transaction data resides in IQ, the modeling approach should be
  - Create Analytic/Calc view using HANA tables
  - Create Calc view using virtual table on IQ table
  - UNION above two views

• **Push virtual tables JOIN to Remote Source**
  Join virtual tables from same remote source. Set it to execute in SQL engine. SDA supports join relocation which pushes join execution to the SDA remote source.

• **Avoid JOIN between Remote Source and HANA**
  If you join HANA tables and remote tables, the data will be queried (aggregated at the remote source) based on the columns participating in the join.
SAP Smart Data Access
Modeling Recommendations

• Avoid JOINs between tables from multiple remote sources
  Joining multiple remote sources would create performance issue. It would require moving data from one remote source to another remote source via HANA to execute the join. For example, lets say the analytic use requires building HANA model to combine data from HANA, Teradata, and IQ.
  In this case joining Teradata tables with IQ tables in a model would create performance overhead because data from either Teradata or IQ will need to be brought over to HANA, then push it to IQ or Teradata where JOIN will execute.
  Instead you should create encapsulated calc view for only IQ tables, and create encapsulated calc view for only Teradata tables, and use these encapsulated models to build final analytic model.

• Avoid large data transfer between SDA remote sources
  In case of joins on high-cardinal its columns, there will be significant data transfer from SDA remote source to HANA. Consider moving these joins to the remote source, by replicating the HANA table to the remote source
SAP Smart Data Access
Modeling Recommendations

- Avoid JOINs between tables from multiple remote sources
  Joining multiple remote sources would create performance issue. It would require moving data from both remote sources to HANA to execute the join.
  For example, let's say the analytic use requires building HANA model to combine data from HANA, Teradata, and IQ.
  In this case joining Teradata tables with IQ tables in a model would create performance overhead because data from both Teradata or IQ will need to be brought over to HANA for the join to execute. If one of the remote table participating in the join is significantly small (say Teradata), then the join may be relocated to the remote server with bigger size table (say IQ). However in this case the smaller table (Teradata) have to be brought over to HANA and then copied to the remote server (IQ) for the join to execute. Instead you should create encapsulated calc view for only IQ tables, and create encapsulated calc view for only Teradata tables, and union these encapsulated models to build final analytic model.

- Avoid large data transfer between SDA remote sources
  In case of joins on high-cardinal its columns, there will be significant data transfer from SDA remote source to HANA. Consider moving these joins to the remote source, by replicating the HANA table to the remote source
Generate statistics on virtual tables.

In order for the SDA to optimize the data transfer between local data and remote data the SQL-Optimizer needs statistic information about the remote table.

There are two different types of statistics: Simple and Histogram. Histogram gives you full-blown statistics to optimize the query, and as the name implies, simple gives you a reduced amount of information (only high level: min/max, etc.). The only advantage of using simple over histogram is the fact that histogram can take longer to generate.

EX CREATE STATISTICS ON “SCHEMA”.“VIRTUAL_TAB”(“COL1”, “COL2”, “COL3”) TYPE HISTOGRAM;

For more information see SAP note: 1872652
**SAP Smart Data Access**

**Modeling Recommendations**

- **Set SDA parameters**
  
  There are SDA parameters that need to be configured to optimize remote query execution. Please review the parameters setting and change it as it applies to your scenario.

- **Use ‘Optimize JOIN’ property**

- **Specify JOIN Cardinalities**
  
  The cardinality of STAR-JOIN allows HANA to generate efficient Query path (i.e. generated SQL to SDA)
SAP Smart Data Access
Learning Resources

- The power of Smart Data Access (SDA) with SAP HANA SP08
  http://scn.sap.com/docs/DOC-58084
- Advanced Modelling: Heterogeneous Reporting using Smart Data Access (SDA) http://scn.sap.com/docs/DOC-64042
- SDA central note 1868209
  http://service.sap.com/sap/support/notes/1868209
- SDA Tutorials
  https://www.youtube.com/watch?v=CbUEhLMUKE4&list=PLkzo92owKnVx_X9Qp-jonm3FCmo41Fkzm
SAP HANA Modeling
Generic Best Practices

1. Avoid transfer of large result sets between the SAP HANA DB and client (e.g. by HAVING, TOP, LIMIT, ...)
2. Do calculation after aggregation
3. Avoid complex expressions (IF, CASE, ...)
4. Reduce data transfer between the engines
5. Do aggregate data records (e.g. using GROUP BY, reducing columns)
6. Join on Key Columns or Indexed Columns
7. Avoid calculation before aggregation on line item level
8. Filter data amounts as early as possible in the lower layers (CONSTRAINTS, WHERE clause, ..)
1. Move Business Logic into HANA
2. Keep Master Data on All Nodes in a scale-out deployment
3. Use LEFT OUTER join instead of INNER join
4. Use UNIONs instead of JOINs in Calculation View
5. Use CONSTANT MAPPING in UNIONs for efficient pruning
6. Use PROJECTION filtering in Calculation View
7. Use Restricted Measures wherever possible
8. Use Mandatory and Optional Variables to restrict the dataset
9. Avoid JOINING multiple large tables
10. Avoid JOINING Analytic Views
11. Avoid calculated columns in Attribute View or Analytic View
12. Keep single Fact table in Analytic View, wherever possible
13. Design Models to Process Logic in Parallel
14. Keep an eye of the query performance as you model
15. Use Execution plan and Visualization plan to analyze query performance