DMM208 – New and Best Practices for Data Modeling with SAP HANA
Speakers

Las Vegas, Oct 19 - 23

- Werner Steyn

Barcelona, Nov 10 - 12

- Christoph Morgen
Agenda

- Data Modeling in SAP HANA
- New Approaches
- Best Practices
Disclaimer

This presentation outlines our general product direction and should not be relied on in making a purchase decision. This presentation is not subject to your license agreement or any other agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or to develop or release any functionality mentioned in this presentation. This presentation and SAP's strategy and possible future developments are subject to change and may be changed by SAP at any time for any reason without notice. This document is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP assumes no responsibility for errors or omissions in this document, except if such damages were caused by SAP intentionally or grossly negligent.
Data Modeling with SAP HANA

Overview SAP HANA Information Views
Virtual Data Modeling as a key SAP HANA concept

Virtual Data Flow Models

- No persistent Aggregations | single atomic copy of data
- In-Memory Engines | Performance
- Multidimensional Reporting Models
- Enterprise Applications Virtual Data Models
SAP HANA View Modeling – Overview

Flavors of SAP HANA View Modeling Approaches

- **Attribute Views**
  - Compose a dimensional view with a series of attributes derived from a collection of tables e.g. Master Data Views
  - Highly re-used and shared in Analytic- and Calculation Views
  - Used to build Hierarchies
  - Hierarchies are key elements in use with Analytic View for multi-dimensional reporting

- **Analytic Views**
  - Combines Fact-Tables with Attribute-Views to Star-Schema or OLAP Cube-like objects for multidimensional reporting.
  - Stores no aggregates and mass-aggregates on the fly
  - Hierarchies are key for multi-dimensional access (navigation, filtering, slicing and aggregation)

- **Calculation Views**
  - Great flexibility for advanced use
  - Approach to model custom scenarios like
    - Combined use of Multiple-Fact Tables/Analytics Views
    - Build Models on Normalized Data
    - Re-Use and stack views
    - Make use of custom scripted views

**Calculation Views of type “Dimension”**

**Calculation Views of type “CUBE/StarJoin”**

Calculation View
SAP HANA View Modeling – Overview

Virtual Data Models for Multidimensional Scenarios

Reporting Tools can usually directly consume Calculation Views.

Multidimensional Tools support Hierarchies for Navigation, Filtering and Aggregation and HANA Prompts (Variables & Input Parameters) for efficient Pre-Filtering of Data.

Calculation Views are usually build upon other Calculation Views and Column Tables.

Calculation Views

Column table

Table Name: HANA

<table>
<thead>
<tr>
<th>Name</th>
<th>SQL Data Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANDT</td>
<td>CHAR</td>
<td>3</td>
</tr>
<tr>
<td>MATNR</td>
<td>CHAR</td>
<td>8</td>
</tr>
<tr>
<td>ERBIA</td>
<td>CHAR</td>
<td>8</td>
</tr>
<tr>
<td>ERBXX</td>
<td>CHAR</td>
<td>8</td>
</tr>
<tr>
<td>LANDX</td>
<td>CHAR</td>
<td>8</td>
</tr>
<tr>
<td>ADRESA</td>
<td>CHAR</td>
<td>12</td>
</tr>
<tr>
<td>UPSTA</td>
<td>CHAR</td>
<td>15</td>
</tr>
<tr>
<td>PSTAT</td>
<td>CHAR</td>
<td>15</td>
</tr>
<tr>
<td>LVORR</td>
<td>CHAR</td>
<td>5</td>
</tr>
<tr>
<td>MTARR</td>
<td>CHAR</td>
<td>4</td>
</tr>
</tbody>
</table>
SAP HANA View Modeling – Overview

Virtual Data Models for Normalized Data Model Scenarios

SAP HANA Calculation Views provide the means to model sophisticated views based on normalized data structures.

Complex Calculation Views demand a more explicit intent and control of the modeled set-based data flow, i.e. slicing, aggregation and filtering of sets as input to joins, unions etc.

SAP HANA Calculation Views typically feed data to Business Applications (i.e. SAP HANA XS build Applications, Enterprise Analytical Applications)
SAP HANA Modeling – Graphical Editors

Eclipse-based & Web-based Development Workbench

SAP HANA Studio

- **Modeler Eclipse-perspective**
  - **Systems view**: Supports basic Modeling artifacts. (no support for functions, roles, CDS objects, repository)

- **Development Eclipse-perspective**
  - **Project/Navigator/Repository view**: Support all design-time artifacts and repository functionality.

SAP HANA Web-based Development Workbench

- **Browser-based graphical View Editor**
  - Browser → http://hana-server:port/sap/hana/ide/editor/
  - **No Analytic- and Attribute-view Modeling support**
  - **Supports Calculation Models** (table functions, roles, procedures, script based calc-models, analytical privileges etc.)
  - **SQL Editor support**, Catalog, Security, Admin, etc.

Note: Web-based editors are future/mid-term innovation platform for SAP HANA development tools, however currently not yet feature complete (SAP HANA SPS10).
New Approaches

SAP HANA Information Models
New Approaches Overview

Key new approaches

- Calculation View Modeling
  - Calculation views-Star-join instead of Analytical/Attribute

- SQL-Script Integration
  - Table functions as data sources
    (instead of script-based Calculation Views)

- Analytic Privileges
  - Support for SQL based analytic privileges

- Hierarchies
  - SQL enablement & Time Dependent hierarchies

- Data Sources
  - CDS entities, tenant container database tables

- Web-based Editor

Specific Enhancements

- Column lineage analysis
- Comment & View Deprecation
- Copy & Replace nodes!
- Multilanguage-label support in Views
- Performance Analysis / Debugger enhancements
- Support Labels/Descriptions in value help dialogs

General Enhancements

- Implicit SQL Execution - Model unfolding
SAP HANA View Modeling – New Approaches

Modeling multidimensional Scenarios in Calculation Views

Multidimensional scenarios in Calculation Views

- Calculation Views of type Dimension
  - Composing Dimension-Attributes and -Hierarchies
- Calculation Views of type CUBE with StarJoin
  - Star-joins fact data with dimension-type calculation views like Analytic Views (with same performance guidance’s)
  - Implicit Calculation View optimization will evaluate to delegate processing to same execution behavior as Analytic Views
- Note: Table functions can be used as input sources for either Dimensions or Facts

Usage Scenario

- Implement new multidimensional scenarios / new projects
- Seek to refactor Attribute-/Analytic View scenarios

Note: As of SAP HANA SPS10 Calculation Views are not yet functional feature equivalent to Attribute- and Analytic Views (i.e. Temporal Join & Fuzzy search).
Enabling SQL Script with Calculation views

• Going forward use SQL Script **table functions** instead of script-based Calculation views

  – Side-effect free read-only functions using standard SQL
  – Consume data from tables, models, functions, predictive algorithms, etc.
  – Script based-calculation models can be re-factored into table functions

```sql
function QuarterToDateByYear(
    in ip_period varchar(1), in ip_date varchar(10)
) returns series_data_t
language sqlscript sql security invoker as
begin
    sd = select source_period_start, source_period_end from series_disaggregate_date ('interval 3 month', 'interval 1 day', :v_year, :v_next_year);
    if :ip_period = 'c' then
        return select bb.generated_period_start as generated_period_start, aa.generated_period_start as factdate, bb.generated_period_start as datesql, 'qtd' as timekpi, 'cy' as cypy_flag from ...
```
SQL Script approaches

- Standard SQL versus Calculation Engine (CE Functions) Plan Operators
  - Recommendation! When creating functions or procedures with SAP HANA SP09+ use standard SQL syntax instead of CE Functions. The SQL optimizer is able to leverage alternative execution engines and are not restricted only to the Calculation Engine

<table>
<thead>
<tr>
<th>SQL</th>
<th>CE function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT on column table</td>
<td>out = SELECT A, B, C from &quot;COLUMN_TABLE&quot;</td>
</tr>
<tr>
<td></td>
<td>out = CE_COLUMN_TABLE(&quot;COLUMN_TABLE&quot;, [A, B, C])</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>out = CE_COLUMN_TABLE(&quot;COLUMN_TABLE&quot;, [A, B, C])</td>
</tr>
<tr>
<td></td>
<td>col_tab: CE_COLUMN_TABLE(&quot;COLUMN_TABLE&quot;), out = CE_AGGREGATION([col_tab, SUM(0)], [A, B, C])</td>
</tr>
</tbody>
</table>

Recommended approach
New data sources for Calculation Views

- Multitenant database container tables
  - While modeling graphical calculation views, you can add data source from any of the isolated databases.

- CDS (Entities and Types)
  - Graceful data structure lifecycle
SAP HANA View Modeling – New Approaches

Modeling with Virtual and Extended Tables

Consuming non-In Memory data in Calculation views

- **Dynamic Tiering**
  - Provides a disk-based columnar SAP HANA database extension using SAP HANA external tables (extended tables) *

- **Smart Data Access**
  - Provides a virtual access layer to outside SAP HANA data (e.g. other databases, Hadoop systems, etc.) using so-called virtual tables

- Model and approach these scenarios carefully and monitoring query performance
  - Remote data imposes a natural break in the execution
  - Ensure that filters and aggregation is pushed to remote sources
  - Use and observer join settings and optimizations
  - Implement input source pruning (explicit or implicit) techniques

*Supported external systems and restrictions are documented in SAP Note 1868209*
Hierarchy SQL Integration

- Hierarchy-based SQL processing capabilities enabled via SAP HANA View based hierarchies
  - SQL processing includes hierarchy-based filtering, aggregations as well as analytic privileges*.
- Usage scenario and enablement
  - Shared hierarchies from Dimension-Type and StarJoin-Type Calculation Views. Compound parent-child hierarchies are not allowed.
  - SQL use of hierarchies needs to be activated in the view properties (for all shared hierarchies). Once activated custom node-column name and hierarchy-expression parameter* can be given custom name

*Note: Use of hierarchy expression will be fully enabled, once hierarchy-expressions are documented in the SAP HANA SQL documentation and thus generally supported.
Hierarchy SQL Integration

- **Hierarchy-aggregation**
  - The node column can be used for **group by**:

  ```sql
  select "SalesRepHierarchyNode", sum("Revenue") as "Revenue", sum("Cost") as "Cost", sum("Margin") as "Margin"
  FROM "_SYS_BIC",mini/CvSalesCubeHier group by "SalesRepHierarchyNode"
  ```

<table>
<thead>
<tr>
<th>SalesRepHierarchyNode</th>
<th>Revenue</th>
<th>Cost</th>
<th>Margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASTERIX</td>
<td>12.360</td>
<td>3.057</td>
<td>75.26</td>
</tr>
<tr>
<td>JOHN</td>
<td>10.500</td>
<td>1.583</td>
<td>85.05</td>
</tr>
<tr>
<td>LISA</td>
<td>8.020</td>
<td>1.269</td>
<td>84.17</td>
</tr>
<tr>
<td>MAJESTIX</td>
<td>20.370</td>
<td>3.463</td>
<td>82.99</td>
</tr>
<tr>
<td>MIKE</td>
<td>2.570</td>
<td>0.314</td>
<td>87.78</td>
</tr>
<tr>
<td>OBELIX</td>
<td>7.310</td>
<td>0.304</td>
<td>95.84</td>
</tr>
<tr>
<td>THE_BIG_BOSS</td>
<td>30.960</td>
<td>5.046</td>
<td>83.7</td>
</tr>
</tbody>
</table>

Note, the Revenue and Cost column contain the hierarchical aggregated values, i.e. the Revenue for MAJESTIX is the sum of the revenue of all its child nodes. The calculated column Margin (percentage) is calculated correctly after the aggregation on each hierarchy level.

- **Filtering on a hierarchy node value**
  - The node column can be used for **filtering**

  ```sql
  select "SalesRep", sum("Revenue") as "Revenue", sum("Cost") as "Cost", sum("Margin") as "Margin"
  FROM "_SYS_BIC",mini/CvSalesCubeHier where "SalesRepHierarchyNode" = 'MAJESTIX'
  ```

<table>
<thead>
<tr>
<th>SalesRep</th>
<th>Revenue</th>
<th>Cost</th>
<th>Margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASTERIX</td>
<td>12.360</td>
<td>3.057</td>
<td>75.26</td>
</tr>
<tr>
<td>MAJESTIX</td>
<td>7.000</td>
<td>102</td>
<td>85.42</td>
</tr>
<tr>
<td>OBELIX</td>
<td>7.310</td>
<td>304</td>
<td>95.84</td>
</tr>
</tbody>
</table>

Note, that the filter on the node column constrains to the hierarchy subtree. All the descendants of the node are selected.
Create SQL-based Analytic Privileges

• Start with general Attribute based Analytic Privilege, then switch to SQL-based

• Use SQL Hierarchies within SQL Analytical Privileges

SalesHierarchyNode = MAJESTIX
Best Practices

SAP HANA Information Models
SAP HANA View Modeling – Best Practices

General Performance Guidelines

Client

Calculation Views

Avoid transfer data of large result sets between the HANA DB and client application

Do calculation after aggregation

Avoid Complex expressions, instead push down into model

Procedures / Functions

Aggregate data records (e.g. using GROUP BY, reducing Columns)

Join on Key Columns or Indexed Columns

Avoid calculations before aggregation on line item level

Replace sequential cursor looping with set processing

Column

Store

Filter data amount as early as possible in the lower layers (e.g. using Constraints, WHERE clause filters, Analytical Privileges, Partitioning)
Optimized execution for SQL-queries against Calculation Views

- The **initial Calculation Engine optimization** generates a single SQL statement across a stacked model, which is then passed to the SQL optimizer.
- The **SQL optimizer adds additional optimizations** and delegates operations to the best database execution operator, e.g. delegate star join-nodes to the OLAP processing engine where possible.

Additional SQL Optimization

- Since SAP HANA Rev 62 the SQL optimization can explicitly be determined setting the “Execute In:” property to “SQL Engine”.
- Since SAP HANA SPS9 the optimized execution behavior implicitly evaluates to leverage additional SQL optimization.
- SAP HANA SPS10 optimization enhancements include implicit optimized Star Join processing.

* Implicit SQL optimization details are documented in SAP Note 2223597.
SAP HANA View Modeling – Best Practices
Performance Analysis & Query Insight Tools

- Visualize Plan
- Operator list
- Timeline – root cause analysis
- Calculation View Debugger
- Explain Plan

<table>
<thead>
<tr>
<th>OPERATOR_NAME</th>
<th>EXECUTION_TIME</th>
<th>TABLE_TYPE</th>
<th>TABLE_SIZE</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW SEARCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WINDOW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AGGREGATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
• Ensure that filters are pushed down
• Aggregate and ensure Group By are pushed down (i.e. SDA remote DB)

```sql
select division, sum(prevamount), sum(currentamount)
from CvOrdersQuery
('placeholder' = ('$$year$$', '2013'))
where country in ('USA', 'GERMANY') and database = 'IQ'
group by division
```

```sql
where clause country
```
Out-source Value Help Information to dedicated views

- Benefit from faster value help dialogs
- Provide consistent LOVs across consuming views
- Dependencies between value help views
- Support Analytical Privileges

```sql
select
  teamname, playername, rank,
  sum(playergoals), sum(teamgoals)
from
CvSoccerPlayers(
  'placeholder'=('$$top_n_players$$','2'),
  'placeholder'=('$ip_team$','paderbom')
) group by teamname, playername, rank
order by teamname
```

The processing of the value help is directed to an external model instead of querying the central model.
Explicit pruning

```
select country, sum(previous), sum(current)
from CvSalesQuery
where year = 2015 and country in ('usa', 'germany') and SOURCE = 'Hot'
group by country order by country
```

Query execution direction relies on the Union Constant filter

Implicit pruning

```
select country, sum(previous), sum(current)
from CvSalesQuery (placeholder = ('$$ip_year$$', '2015'))
where country in ('usa', 'germany')
group by country order by country
```

Query execution direction relies on the Date Input parameter
Show Mouse sales in February!

<table>
<thead>
<tr>
<th>Order</th>
<th>Month</th>
<th>Product</th>
<th>Store</th>
<th>Customer</th>
<th>Quantity</th>
<th>Price</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>Ipad</td>
<td>TigerDirect</td>
<td>John</td>
<td>20</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Mouse</td>
<td>TigerDirect</td>
<td>Susan</td>
<td>2</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Mouse</td>
<td>TigerDirect</td>
<td>John</td>
<td>2</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Mouse</td>
<td>Amazon</td>
<td>John</td>
<td>2</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>Headset</td>
<td>Amazon</td>
<td>Susan</td>
<td>3</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Headset</td>
<td>Ebay</td>
<td>John</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>Ipad</td>
<td>Amazon</td>
<td>Susan</td>
<td>3</td>
<td>250</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Ipad</td>
<td>Ebay</td>
<td>Susan</td>
<td>3</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>Mouse</td>
<td>Amazon</td>
<td>Peter</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>Ipad</td>
<td>Amazon</td>
<td>John</td>
<td>3</td>
<td>250</td>
<td>40</td>
</tr>
</tbody>
</table>

Regular SQL

```sql
SELECT "Month", "Product", sum("Quantity") as SUM(Quantity), sum("Price") as SUM(Price)
FROM "DEMO"."demo.modeling.db::cds.StoreOrders"
WHERE ("Month" = '2' AND "Product" = 'Mouse')
group by "Month", "Product"
```

Solution!

Within Calculation Model Set
Keep Flag = True on Order ID

Generally the requested columns mandates the level of aggregation. Under certain circumstances this level of aggregation is to course and instead requires a finer level of aggregation.

- Setting the **Keep flag** will force additional columns into context altering the level of granularity.
Count the amount of stores that sold a Mouse to Johan and Susan!

```
select product, sum(quantity), sum(storecount) from model
where Customer in (john, susan) and product = mouse group by product
```

The Transparent Filter will omit columns from unnecessary being used part of Group By’s. This Flag is required when using …

- Stacked views where the lower views have count distinct measures
- Queries on the upper Calculation view contain filters on columns that are not projected

Solution! Set the Transparency Filter for the column Customer to true, on all models and nodes that references the Customer
Show the Sales & Total Sales by Region & Product!

Only the Join columns requested in the query will be brought into context and play a part in the Join execution.
Generally tables are pruned if no columns are requested and Join types such as Left-Outer or Referential are involved; under certain circumstances the Joined columns could be brought into context altering the level of aggregation.

This setting will ensure that the joined columns are omitted and will lead to smaller intermediate result sets which can improve performance significantly depending on the type of query.

*Note: This feature generally applies to earlier revisions*
Designing larger Virtual Data Models using HANA Views

Architecting complex SAP HANA View-based data models

- Learn from SAP HANA Live! Layering approach.
SAP TechEd Online
Continue your SAP TechEd education after the event!

- Access replays of keynotes, Demo Jam, SAP TechEd live interviews, select lecture sessions, and more!
- Hands-on replays

http://sapteched.com/online
Further Information

Related SAP TechEd sessions:
DMM360 – Advanced Data Modeling in SAP HANA, Hands-On Workshop
DMM163 – Introduction to Data Modeling in SAP HANA, Hands-On Workshop

SAP Public Web

What’s new in SAP HANA (incl. SAP HANA View Modeling)

SAP Education and Certification Opportunities
www.sap.com/education
https://www.youtube.com/user/saphanaacademy https://open.sap.com/courses

Watch SAP TechEd Online
www.sapteched.com/online
Feedback

Please complete your session evaluation for DMM208
Thank you!

Contact information:

Christoph Morgen
SAP HANA Product Management
SAP SE | Dietmar-Hopp-Allee 16 | 69190 Walldorf | Germany
christoph.morgen@sap.com | www.sap.com

Werner Steyn
SAP HANA Competence Center, SAP SE
werner.steyn@sap.com | www.sap.com
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. Please see http://global12.sap.com/corporate-en/legal/copyright/index.epx for additional trademark information and notices.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors.

National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP SE or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP SE or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE’s or its affiliated companies’ strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.