A Simple Overview on SAP HANA

Applies to:

SAP ERP. For more information, visit the EDW homepage

Summary

A simple overview on SAP HANA. A basic knowledge on SAP BW and DBMS is preferred for the reader.

Author: Abyson Joseph Chavara

Company: Applexus Technologies (P) LTD.

Created on: 16 September, 2011

Author Bio

Abyson Joseph Chavara is working as an SAP ABAP consultant at Applexus Technologies (P) LTD. He has an experience of 3 years in ABAP programming and 1 year in SAP PI.

Table of Contents

Introduction

SAP is evolving its in-memory technology with the introduction of SAP High- Performance Analytic Appliance (SAP HANA) software, a flexible, multi-purpose, data source agnostic in-memory appliance that combines SAP software components optimized on hardware provided and delivered by SAP leading hardware partners.

SAP HANA can enable organizations to analyze business operations based on large volumes of detailed information as it develops. Organizations can instantly explore and analyze all of their transactional and analytical data from virtually any data source in real time. Operational data is captured in memory as business happens, and flexible views expose analytic information at the speed of thought. External data can be added to analytic models to expand analysis across the entire organization.

Overview and Architecture of HANA

- In memory computing studio as a frontend for modeling and administration.
- HANA is connected ERP systems, Frontend modeling studio can be used for load control and replication server management.
- Two types of Relational Data stores in HANA : Row Store, Column Store.
- SAP BOBJ tools can directly report HANA.
- Data from HANA can also be used in MS Excel.
- Row Store – Traditional Relational Database, the difference is that all the rows are in memory in HANA where as they are stored in a hard drive in traditional databases.
- Column Store – The data is stored in columns like in SAP BWA.
- Persistency Layer: In memory is great by it is volatile and data can be lost with power outage or hardware failures. To avoid this HANA has a Persistency Layer component which makes sure that all the data in memory is also store in a hard drive which is not volatile.
- Session Management: This component takes care of logon services.
- Two processing engines – Well, data is in memory which is good but how do I extract/report on the data? HANA has two processing engines one is based on SQL which accepts SQL queries and the other one is based on MDX.
- HANA Supports Sybase Replication Server – Sybase Replication Server can be used for real time synchronization of data between ERP and HANA.

Modeling Studio

Using Modeling Studio you can,

- Specify which tables are stored in HANA, first part is to get the meta data and then schedule data replication jobs
- Manage Data Services to load the data from SAP BW and other 3rd party systems.
- Manage connections to ERP instances, current release does not support connecting to several ERP instances
- Use Data services to for the modeling
- Do modeling in HANA itself (This is independent of Data services).
- You can also do modeling can also be done in Business Objects Universes which is nothing but joining fact and dimensional tables.

Reporting

- Client tools can access HANA directly; Like MS EXCEL, SAP BI 4.0 Reporting tools, Dashboard Design Tool (Xcelsius) etc can also access HANA directly.
- Third party reporting tools can leverage ODBC, JDBC and ODBO (for MDX requests) drivers in HANA for reprint.
- HANA supports BICS interface.

Request Processing and Execution Control

- SQL Script, MDX statements are passed to calculation modules. Optimizer which is included in calculation engine optimizes for better performance.
- **Calc Engine :**
  - Modeler can define data sources as inputs and different operations (join, aggregation, projection) on top of them for data manipulation.
  - The calc engine will break up a model into sub processes for optimized performance on cost based.
  - System will use maximum resources to achieve max through put.
- **Planning Engine:** Will be included in next release. Will include planning functions like distribute and copy functions.

ROW Store

- One of the relational engines to store data in row format.
  - Pure in-memory store (Future versions will also have an option of disk based store).
  - In memory object store (in future) for live cache functionality.
  - Transactions Version Memory is the heart of row store.
- Row store architecture
  - Write operation mainly go into "Transactional Version Memory".
  - INSERT also writes to persisted segment.
  - Moves visible version from memory to persisted segment.
  - Clears outdated record versions from Transactional Version memory.
  - Row Store tables have a primary index.
  - Row ID maps to primary key.
  - Secondary indexes can be created.
  - Row ID contains the segment and the page for the record.
  - Indexes in row store only exist in memory.
  - Index definition stored with table meta.

Column Store

- Improves read functionality significantly, also improves write functionality.
- Highly compressed data.
- No real files, virtual files.
- Optimizer and Executer – Handles queries and execution plan.
- Delta data for fast write.
- Asynchronous delta merge.
- Consistent view Manager.
- Main store compressed and read optimized – Data is read from Main Store.
- Delta Store – Write optimized – for write operations.
- Asynchronous merge move the data from delta store to main store.
- Compression by create dictionary and applying further compression methods.
- Even during the merge operation, the columnar table will still be available for read and write operations. To fulfill this, a second delta and main storage are used internally.
- Merge operation can also be triggered manually with an SQL command.

Persistence Layer

- Persistence Layer is needed as Main memory is volatile.
- Provides Backup and Restore functionality.
- One Persistency Layer takes care of both row and column stores.
- Regular Save Points.
- Logs capturing DB transactions since last save point.
- Actions during system restart
    - ➢ Last save point must be restored plus undo logs must be read and uncommitted transactions saved with last save point and apply redo logs.
    - ➢ Complete content of row store is loaded into memory during start process.
    - ➢ Flags can be set for column store to specify which tables are loaded during system restart.

Modeling

- Modeling only possible for Column tables.
- Information Modeler only works for column tables.
- Replication servers create tables in column store per default.
- Data Services creates tables in column store per default.
- SQL to create column table: Create COLUMN TABLE.
- Stored can change with ALTER TABLE.
- System tables are created where they fit best.
- Schema SYS -> caches, administrative table of engine.
- Tables from static server.

In-Memory Computing Studio

- Build with java based eclipse.
- Navigator to access different HANA systems on left, Quick Launch View at the middle and Properties view at the bottom.
- Information Modeler Features:
    - ➢ Database views.
    - ➢ Choice to publish and consume at 4 levels of modeling.
        - ✓ Attribute view, analytic view ...
    - ➢ Physical tables and Information Models.
    - ➢ Import/export models, data source schemas, mass and selective load.
    - ➢ Landscapes.
- The models are just virtual definitions they don't store actual data.
- Analytic Views are like cube model where Transaction Data is connected to attribute view.
- Calc View – With custom functions and calculations.
- **Modeling Process Flow**
    - ➢ Import Source System Metadata.
    - ➢ Create Information Models.
    - ➢ Consume using BICS, SQL or MDX.

- **Information Modeler Terminology**
    - Attributes – Characteristics.
    - Measure – Key Figures.
    - Attribute Views – Dimensions.
    - Analytic Views – Cubes.
    - Calculation Views – Similar to Virtual provider concept in BW.
    - Hierarchies
        - ✓ Leveled – based on multiple attributes.
        - ✓ Parent-child hierarchy.
    - Analytic Privilege – Security Object.
- **Navigation View**
    - HANA instance -> Hana server name and instance number -> user database schema -> views functions and tables.
    - Information Models – Attribute, Analytic, Calculation Views and Analytic Privilege.
- **Attribute View :**
    - Attributes add context to data.
    - Attributes are modeled using attributes views.
    - Can be regarded as Master Data Tables.
    - Can be linked to fact tables in Analytical Views.
    - A measure e.g. weight can be defined as an attributes.
    - Table Joins and properties
        - ✓ Leftouter, Rightouter, full outer or text table.
        - ✓ Cardinality 1:1, N:1, 1:N.
        - ✓ Language Column.
    - Content Views and Functions will be shipped with HANA.
- **Analytics View:**
    - Similar to Cube.
    - Analytic Views does not store any data. The data is stored in column store table or view based on Analytical View structure.
    - Attributes and Measures – Like key figures.
- Data Preview – Similar to list cube functionality.
- **Calculation View:**
    - Define table output Structure.
    - Write SQL statement.
        - ✓ Ensure the selected field corresponds to previously defined output structures.
    - SQL Scripts unlike SQL procedure can't change any data they are read only.

Other Notes:

- External tools can connect to HANA using JDBC and ODBC drivers.
- HANA currently doesn't support complete MDX set; it supports EXCEL 2010 standard MDX.
- BWA Hardware can be upgraded to HANA given that hardware is relatively new.
- BWA Licenses can be transferred to HANA.
- ERP and BW can be connected to HANA using Data services or Sybase Replication
- No namespace concept in HANA at this moment, so two ERP instances can't be connected to HANA. However this issue can be avoided using datasources.
- CRM also can use HANA.
- Data from BW into HANA can be loaded into using Data Services and InfoSpokes.

Related Content

For more information, visit the EDW homepage

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.