Scientific Research Publishing

Journal of Computer and Communications, 2017, 5, 1-10
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

High Performance Analytics with SAP HANA Virtual Models

Abani Pattanayak

SAP HANA Center of Excellence, SAP America, Dallas, TX, USA
Email: abani.pattanayak@sap.com

How to cite this paper: Pattanayak, A. (2017) High Performance Analytics with SAP HANA Virtual Models. *Journal of Computer and Communications*, **5**, 1-10.
https://doi.org/10.4236/jcc.2017.57001

Received: April 10, 2017
Accepted: May 6, 2017
Published: May 9, 2017

Abstract

Informed decision-making, better communication and faster response to business situation are the key differences between leaders and followers in this competitive global marketplace. A data-driven organization can analyze patterns & anomalies to make sense of the current situation and be ready for future opportunities. Organizations no longer have the problem of "lack of data", but the problem of "actionable data" at the right time to act, direct and influence their business decisions. The data exists in different transactional systems and/or data warehouse systems, which takes significant time to retrieve/process relevant information and negatively impacts the time window to out-maneuver the competition. To solve the problem of "actionable data", enterprises can take advantage of the SAP HANA [1] in-memory platform that enables rapid processing and analysis of huge volumes of data in real-time. This paper discusses how SAP HANA virtual data models can be used for on-the-fly analysis of live transactional data to derive insight, perform what-if analysis and execute business transactions in real-time without using persisted aggregates.

Keywords

SAP HANA, In-Memory Computing, Virtual Data Model, Advanced Analytics, Big Data

1. Introduction

SAP HANA is a high-performance analytics platform based on an in-memory columnar database developed and marketed by SAP. In a columnar database, all values in a column are stored together, so access of individual data elements (e.g. employee name) is significantly faster than traditional row-store database. Columnar database can use dictionary and other compression techniques to store data efficiently and this compression enables even faster columnar operations like *Count*, *Sum*, *Min*, *Max* and *Avg*. With SAP HANA, all data is stored in memory (RAM) which gives the CPUs quick access to data for processing and SAP HANA can process around 1b scans/second/core and 10 m rows/second join performance [2]. SAP HANA comes as an appliance with multi-core CPUs, multiple CPUs per board and multiple boards per server-all running in parallel provides serious computational power.

SAP HANA provides a sophisticated platform to model business requirements as virtual data models on top physical tables. These requirements may range from simple calculations to join/union/rank database operations. It can also perform complex calculations involving *geo-spatial*, *text analysis*, *predictive analysis* and *machine learning* involving billions of records in a distributed/ scale-out environment.

2. SAP HANA Architecture

2.1. Data Stores

SAP HANA as a database offers complete data lifecycle management using data temperature classification (*hot*, *warm* and *cold*), where the most active dataset (*hot data*) is stored as in-memory column tables, the less active dataset (*warm data*) is stored as disc-based column tables and inactive dataset (*cold data*) may be stored in near-line storages fully accessible from the HANA. The near-line storage could base on commercial database, like SAP Sybase IQ or open source Hadoop clusters.

The SAP HANA data stores and database engine are illustrated in Figure 1.

Column Store: Provides storage and management of in-memory column-tables. These tables are loaded into memory during initial access and continue to stay in-memory unless explicitly unloaded from memory by an administrator or by the HANA database itself using LRU (*least recently used*) algorithm. These tables can be loaded into memory during database restart operation. The most critical and most active dataset is stored as the in-memory column tables.

Row Store: Even though most of the tables created in SAP HANA are column tables, SAP HANA also supports creation of row-store tables in-memory. Row store tables are mostly used as system statistics table and they may be used by applications to meet specific technical requirements. Like the in-memory column store tables, these tables are loaded into memory during initial access and continue to stay in memory unless explicitly unloaded from memory by an administrator or by the HANA database using LRU algorithm.

Dynamic Tiering: SAP HANA supports creation of disc-based column tables, with an additional component called *dynamic tiering* [3] installed and configured in a separate node of the SAP HANA system. The dynamic tiering column tables are persisted into the disc and loaded into memory only during data access and subsequently unloaded from memory. These types of tables are used to store less active datasets

Smart Data Access: The *smart data access* (SDA) [4] [5] technology provides virtual/remote access to external database objects. Any operation on these virtual

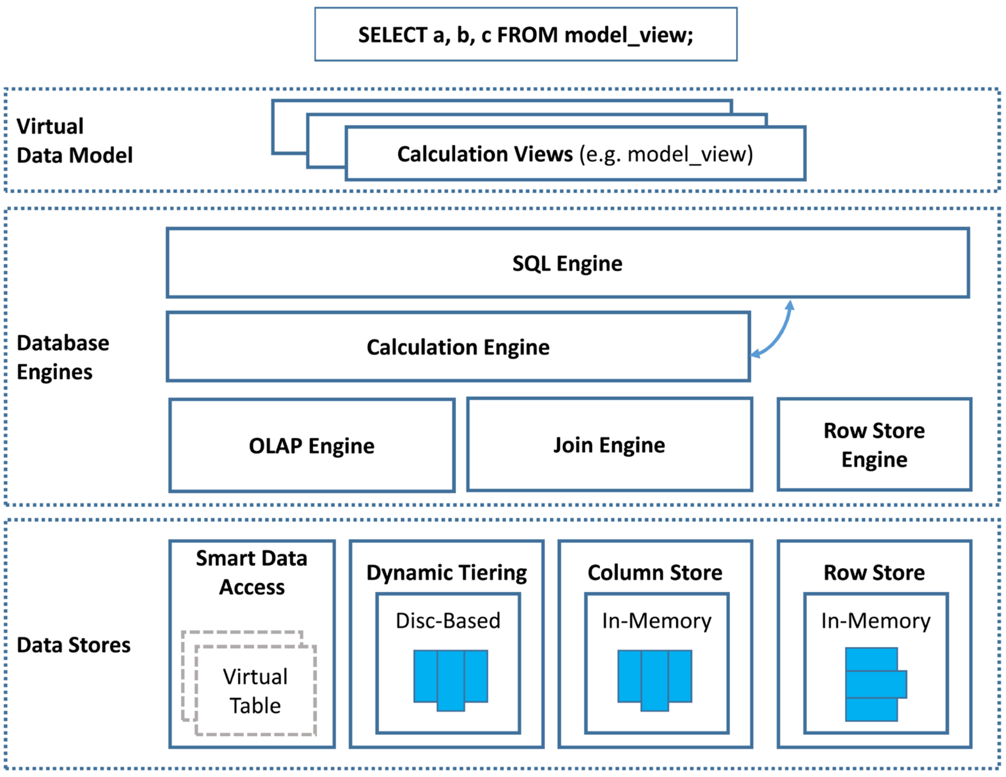


Figure 1. SAP HANA architecture.

tables will be delegated/federated to the remote database. These remote databases can be supported commercial databases (Oracle, SQL Server, DB2 etc.) or open source Hadoop systems supported by SAP.

2.2. Database Engines

SAP HANA has various database engines to perform various operations efficiently and appropriate engines will be engaged depending on the type of operations and calculations.

OLAP Engine: The OLAP engine is the fastest engine SAP HANA and as the name suggests is very efficient in OLAP operation-joins, aggregations and calculations to model star-schema.

Join Engine: As the name suggests, the join engine is used to perform join operations (inner, right outer, left outer and full outer) between tables. This is used to perform joins master data tables to model dimensions.

Calculation Engine: The calculation engine is used to perform field level calculations and union/rank operations on the dataset. The calculation engine will delegate as much computation as possible to the lower OLAP engine and join engine for efficient query processing.

Row Engine: The row engine is used to access and perform operations on ROW tables. Row engine is also used to perform certain SQL operations (like the Window functions) on column store tables.

SQL Engine: The SQL engine is responsible for processing SQL queries in SAP HANA. The SQL Parser (part of SQL Engine) first checks the syntactic and semantic correctness of the SQL query. The SQL query is then parsed through the SQL optimizer to create an acyclic tree-structure of operations and estimated cost is assigned to each operation in the tree structure. Depending on the complexity involved, the SQL Optimizer may generate multiple execution plan and the most cost-effective execution plan will be forwarded to the SQL Executor. The SQL executor will then forward the request to the appropriate engines *i.e.* Row store engines to deal with row tables and Column store engines (Calc. Engine, Join Engine and OLAP engine) for the column tables.

2.3. Virtual Data Model

Virtual Data Model is a key concept of SAP HANA platform which enables rapid development of analytic applications. These virtual data models fully support set based operations enabling real-time analytics and on the fly calculations without the use of persistent aggregates. SAP HANA virtual data model can be built using *Attribute Views*, *Analytics Views*, *Graphical Calculation Views* and *Scripted-Calculation views* [6].

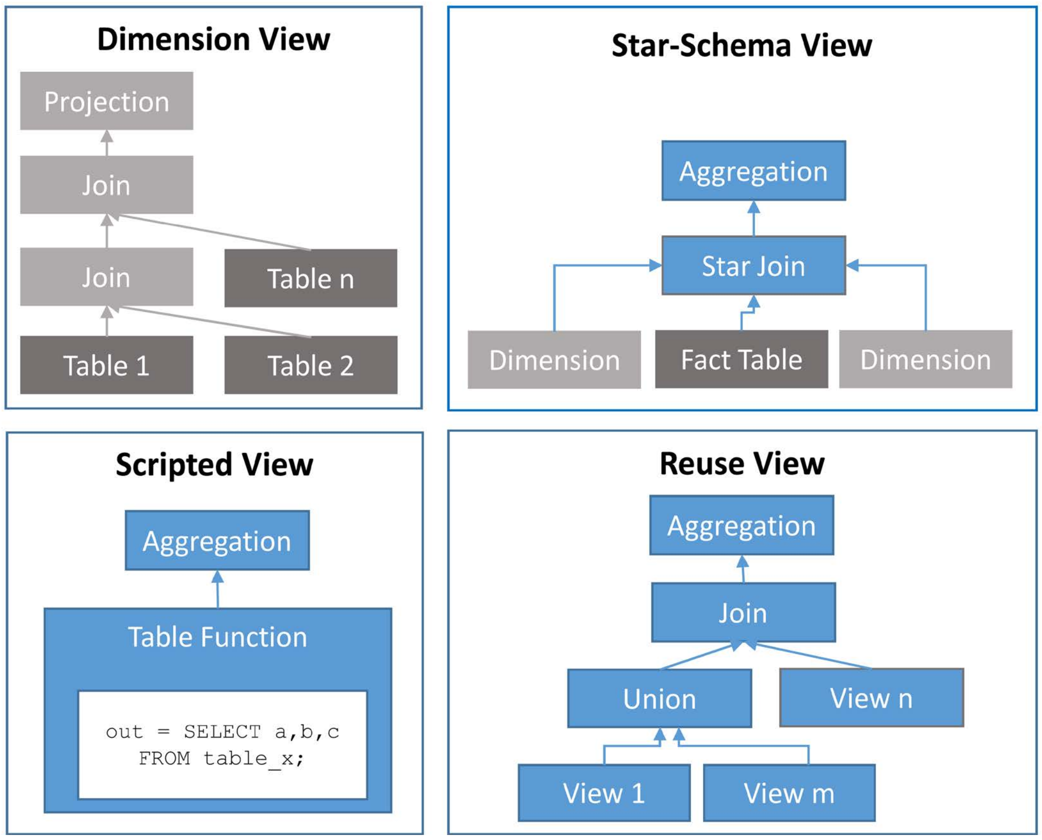
However, starting with SAP HANA SP09, SAP recommends building virtual data model using only SAP HANA graphical calculation views. The graphical calculation views support persisted objects (e.g. column store tables & warm-store tables), virtual objects (e.g. database views, script-based table functions & smart data access virtual tables) and other calculation views. Database operations *like projection with filter*, *aggregation*, *joins* (*inner*, *left outer*, *right outer*, *referential*, *text joins*), *star-join*, *union* and *rank* can be graphically modelled using calculation views. It supports an extensive list of field level calculation (e.g. boolean, mathematical, date, string and data conversion) functions to model business requirements. Complex calculations with procedural logic can be modeled as script-based table function, which can then be used in graphical calculation views.

SAP HANA Studio (and SAP HANA Web Workbench and Web IDE) provides a unified information modelling and design environment for rapid end-to-end application development (data acquisition, data modeling and user interface)

Figure 2 illustrates various types of calculation views used in building the SAP HANA virtual data model.

Dimension View: Calculation views (*type*: *Dimension*) are used to model dimension view typically using "projection" and "join" nodes on one or more master data tables. One or more fields are used to define the key for the dimension view.

Star-Schema View (Cubes): Calculation view (*type*: *Cube with star-join*) is used to model cubes for multi-dimensional analysis of transaction data (fact table). The *star-join* node is used to join one-fact table (or view) with one or more dimension views. It also supports *temporal-join* (time-dependent joins) to analyze transaction data with slowly-changing dimensions. These views must have at least one measure, which is aggregated using an *aggregate* node.



```
out = SELECT a,b,c
  FROM table_x;
```

Figure 2. Virtual data models using calculation views.

Reuse View: Calculation view (*type*: *Cube*) is used to implement business logics on top of the dimension and star-schema views. The KPI views can be modeled for specific business functions using a combination of *projection*, *aggregation*, *join*, *union or rank* nodes. The complete virtual data model can be built using view-on-view layout with multiple layers of KPI Views forming Reuse Views.

Scripted View: Complex calculations with procedural logic, predictive algorithms, R-libraries etc. which can't be modeled as graphical calculation views are modeled using scripted-table functions. These table functions can then be used in graphical calculation view to compute complex KPIs, part of the virtual data model.

2.4. Execution of SQL Queries on Calculation Views

The SQL engine in SAP HANA is responsible for processing SQL queries on SAP HANA database tables/views and SAP HANA virtual data models based on calculation views

With the release of *SAP HANA SP*06-*revision* 62, optional SQL optimizations [7] were introduced for graphical calculations views. However, these optional SQL optimizations were implicitly enabled for all graphical calculation views starting with *SAP HANA SP*09-*revision* 90 [8]. These optimization and execution behavior of a SQL query against a graphical calculation view can be described as below.

1) Based on the selected fields in the SQL query, SAP HANA composes a single *execution model* from all included graphical calculation views. All included graphical calculation views are un-folded to the table/database objects and any table/database objects not contributing to the SQL execution are eliminated to build a single acyclic data-flow graph for the SQL query as shown in Figure 3.

2) The rule based optimizations (e.g. applying filters at the lowest level to narrow the intermediate result sets, combining multiple aggregations into one operation and similarly combining multiple join operation into one operation) are applied to optimize the acyclic data-flow graph.

3) All column store operations are then converted to equivalent SQL operators to create a single SQL statement for the complete data-flow graph. The SQL statement is then passed to the SQL-optimizer.

4) The *SQL Optimizer* can then apply further optimizations like join reordering across the complete data flow graph. The SQL optimizer applies a cost-based estimate for each operation and may generate multiple execution plan (based on the best order of operation and best choice operator variants for optimal performance). The most cost-effective execution plan will be then forwarded to the SQL-executor.

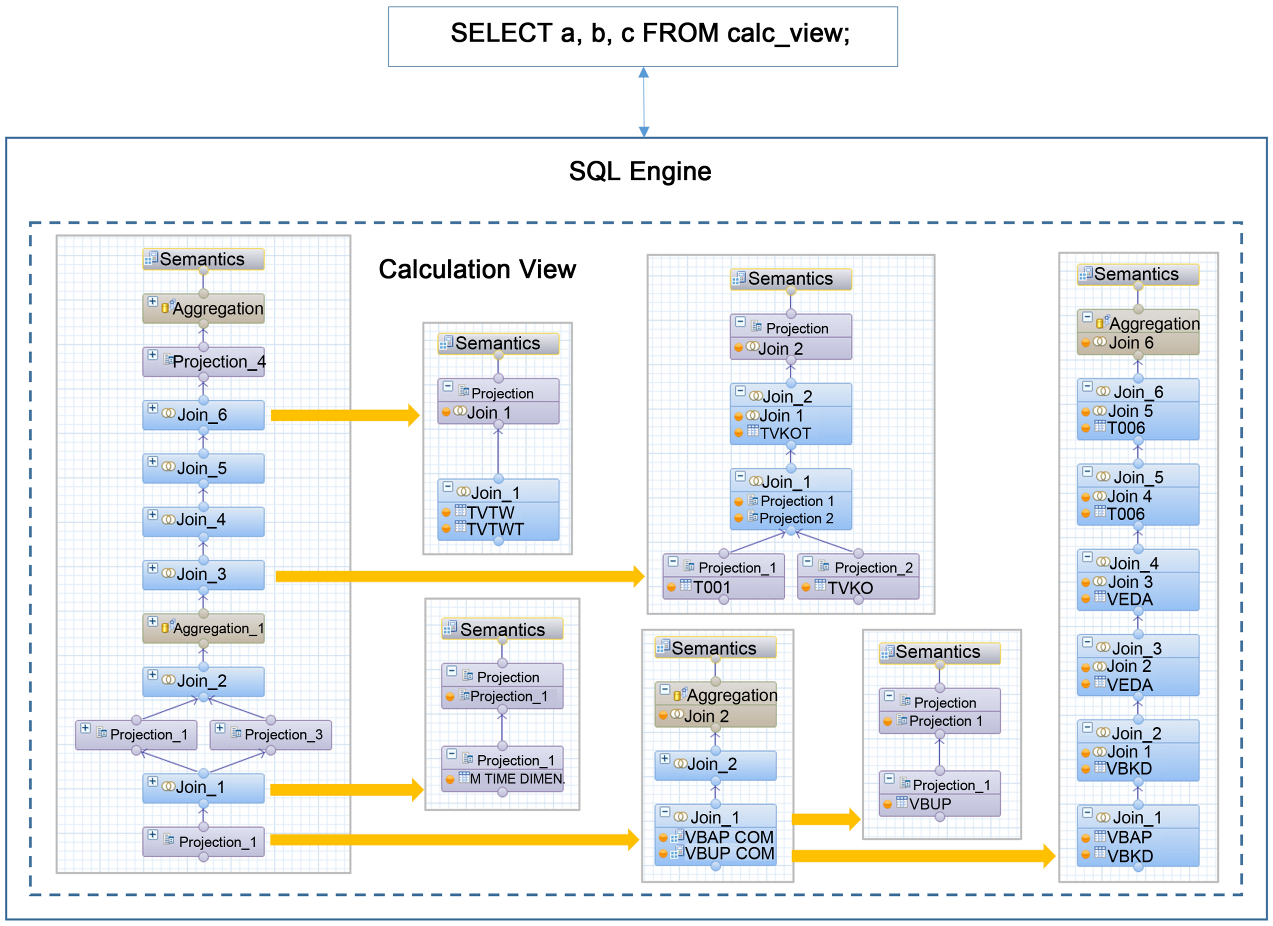


Figure 3. Calculation view unfolding [9].

5) The *SQL Executor* will the forward then database operations to the best SAP HANA column store engines (e.g. execution of a star-join node will be forwarded to the OLAP engine).

Following query insight tools available part of the SAP HANA Studio (integrated development environment) can be used to analyze the execution plan for performance optimization.

Explain Plan (SQL Editor: *Context Menu* → *Explain Plan*): Evaluates the execution plan that the database follows when executing an SQL statement. The most important information of the explain plan is the *Execution Engine* and *Subtree Cost*. If there are two SQL statements on the same list of tables, then the SQL with overall low *Subtree Cost* in the explain plan is more efficient and should be used.

Plan Visualization (SQL Editor: *Context Menu* → *Visualize Plan*): Generates a graphical view of the plan to help analyze the execution plan of an SQL statement. The Plan Visualizer displayed the time taken for compilation & execution, dominant operators in the plan, no of tables used, maximum no of rows processed, the total memory allocated and data transfer between nodes (in a scale-out environment). The name of the operators indicates the type of HANA engines (e.g. *BW**: OLAP Engine, *JE**: Join Engine, *ce**: Calculation Engine and *Sql**: SQL Engine).

3. Data Loading and Replication

SAP HANA is an ACID compliant database and can be used for OLTP (online transaction processing) systems, OLAP (online analytical processing) applications and hybrid (both OLTP and OLAP in one database) solutions. SAP business applications like SAP S/4HANA [10] and Business Suite powered by SAP HANA use SAP HANA as the underlying database. Also, SAP data warehouse solutions like SAP BW/4HANA [11] and SAP Business Warehouse powered by SAP HANA use SAP HANA as the underlying database. So, virtual data models (calculation views) can be built on top of these application tables.

For application using other databases, the data can be replicated to a SAP HANA database to take advantage of the SAP HANA platform. SAP Landscape Transformation Replication Server (SLT) can be used to replicate data in real-time from both SAP and non-SAP business applications running on supported databases (e.g. Oracle, SQL Server etc.). SAP HANA Enterprise Information Management (EIM), SAP Business Objects Data Service or other ETL tools can also be used to batch load data to SAP HANA. SAP HANA virtual data model can be built once these tables are loaded/replicated to the SAP HANA database.

Data in near-line storage or any other supported external database can also be accessed virtually from SAP HANA using SAP HANA Smart Data Access [4] [5] technology. External database tables/views are referenced and accessed as virtual tables in SAP HANA. SAP HANA virtual data model can also be built using these remote virtual tables. The data access performance of the virtual table depends on the capability of the remote system, network connection to the remote system and the amount of data transferred across the network.

4. Reporting & Visualizations

SAP HANA virtual data models are accessible by SAP Business Objects BI tools and Business Objects Cloud to create sophisticated visualizations. SAP HANA virtual data models are also supported by BI tools from other commercial vendors (Tableau, QlikView, IBM Cognos etc.). SAP HANA also support development of HTML5 based applications & visualizations using the in-built XS Server and XS Advanced framework.

5. Modeling Guidelines for Optimal Performance

We understand SAP HANA can process around 1b scans/second/core and 10m join rows/sec and with multi-core CPUs, multiple CPUs per board and multiple boards per server, SAP HANA can perform highly complex queries in seconds. However, with the following guidelines [12] may be used to design virtual data models for optimum performance using minimum memory and CPU resources.

SQL Queries: Implement the business logic part of the virtual data model, so the SQL queries should be simple *SELECT* statements on top of the virtual data model. Since SAP HANA is a columnar database, avoid using *SELECT* * and consider explicitly selecting appropriate columns. To minimize data-transfer across networks, consider using the *Top N*, *Limit*, *Where*, *Group By & Having* clauses part of your *SELECT* queries. Also, watch out for SQL functions (like *String_Args*) which execute in the context of row-engine.

Filters: Use appropriate design time filters (constant value and input parameters/placeholders) to limit the no of records read from the database tables. Avoid design time or runtime (*WHERE* clause & *Analytic Privilege*) filters on calculated columns. If filters must be applied on a calculated column, consider creating the calculated column in the database table.

Type of Calculation: Understand certain type of calculations like *Count* (*DISTINCT column*) is resource intensive compared to simple *Count* (*column*) aggregate function. These resource intensive calculations (if required) should be computed at the lowest level of your data model (e.g. part of the model with star-join). Watch out for KPIs with "Calculate before Aggregation" or with currency/unit conversion.

Join Pruning: Joins are one of the most expensive operations (vs. union or aggregation), so it is essential only requested tables are engaged during query execution. To enable join pruning during execution, consider using *Left Outer Join* (vs. *Inner Join*) where possible. Also specify *Cardinality* for all types of joins (1:1, n:1 etc.). Also, specify *Optimize Join* = *True* for left outer and text Joins to aid join pruning (SP09 onwards).

Model Pruning: Consider using *Constant* column mapping (SP05 onwards) or *Pruning Configuration Table* (SP11 onwards) in *Union* nodes to prune part of the model not explicitly requested during query execution.

Degree of Parallelization: Logically partitioning (e.g. *CY* & *PY* or 2015, 2016 &

2017 dataset) data model with *Unions*, will enable more parallel processing and will reduce query execution time. But this may lead to higher CPU usage, so keep this in perspective while designing the virtual data model.

Table Partitioning: SAP HANA provides multi-level partitioning of tables. Physical partitioning of table aids parallel access of data from the table. However, the partitioning scheme should be based on the dominant filters in the frequently used SQL queries.

Inter-node Network Data-transfer: This is applicable for SAP HANA *scale-out* environment with a master node and several worker nodes. Joins between tables physically existing in different nodes will result in cross-node data transfer and this can significantly impact query performance depending on the volume of the dataset. To reduce the inter-node data transfer, collocate the master and fact data tables in the same node. If the fact tables are partitioned across multiple/all worker nodes, consider replicating master data to all nodes.

SAP HANA Caching: If the underlying dataset is static for a certain period and if there is a significantly high concurrency requirement for the same dataset (e.g. sports score), enabling SAP HANA cache on calculation views will provide consistent query performance while consuming minimal system resources.

6. Conclusion

As discussed SAP HANA provides a sophisticated platform to manage the full life-cycle (hot, warm & cold) of business data and supports analysis of both operational and historical data. With SAP HANA, both business transaction system (OLTP) and analytics (OLAP) can co-exist in the same database using the same single source of facts-no need of data duplication/movement and hence no need of reconciliation between systems. SAP HANA information models (calculation views) can be used to build sophisticated virtual data models to provide real-time analytics on huge volumes of operational and historical data across multiple physical systems. These virtual data models are truly agile, and provide source-agnostic data access & integration of external data and on the fly analysis of entire dataset across organization. Moreover, advanced analytics capabilities of the SAP HANA platform (text analysis, geo-spatial, predictive, R-integration and machine learning) can be integrated with virtual data models to drive innovation in business process, optimize business process, reduce cost of operation and create new opportunities. Based on an IDC study, SAP HANA customers will realize an average of $19.27 million per year compared with an investment cost of 2.41 million over five years [13].

Acknowledgements

I want to express my gratitude for my supervisor Greg Smith, VP-Digital Business Services, SAP America & Dhirendra Gehlot, Director-Digital Business Services, SAP America for their encouragement and support in publishing this paper.

References

[1] What's SAP HANA. https://www.sap.com/product/technology-platform/hana.html

[2] SAP HANA Troubleshooting and Performance Analysis Guide. https://help.sap.com/viewer/bed8c14f9f024763b0777aa72b5436f6/2.0.00/en-US

[3] SAP HANA Dynamic Tiering. https://help.sap.com/viewer/p/SAP_HANA_DYNAMIC_TIERING

[4] SAP HANA Smart Data Access. https://help.sap.com/saphelp_nw74/helpdata/en/bb/df686dc0d94f779b5c11d06753da95/frameset.htm

[5] Smart Data Access: Data Virtualization in SAP HANA, by Abani Pattanayak, Williams Ruter, SAP PRESS.

[6] SAP HANA Modeling Guide, SAP Help Portal. https://help.sap.com/viewer/fc5ace7a367c434190a8047881f92ed8/2.0.00/en-US

[7] SAP Note 1857202. https://launchpad.support.sap.com/#/notes/1857202

[8] SAP Note 2291812. https://launchpad.support.sap.com/#/notes/2291812

[9] DMM208: New and Best Practices for Data Modeling with SAP HANA, by Werner Steyn, Christoph Morgen. http://events.sap.com/teched/en/session/26543

[10] SAP S/4HANA: The Next-Generation ERP Business Suite. https://www.sap.com/product/enterprise-management/s4hana-erp.html

[11] SAP BW/4HANA. https://www.sap.com/product/data-mgmt/bw4hana-data-warehousing.html

[12] HANA Modeling Good Practices, by Shivaji Patnaik, Abani Pattanayak, Imran Rashid & Gordon Dailey. https://blogs.sap.com/2014/03/26/hana-modeling-good-practices/

[13] Innovating with Real-time Data and Insights with SAP HANA for Better Business Outcomes and More Efficient Operations, Carl W. Olofson, Matthew Marden, November 2016, IDC White Paper.