International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I10P127

Volume 72 Issue 10, 199-205, October 2024

Original Article

Navigating Integration Complexities in Hybrid BI and Data Lake Architectures

Savio Dmello

Business Intelligence and BPC SME, California, USA.

Corresponding Author : saviodmo@gmail.com

Received: 12 September 2024 Revised: 14 October 2024 Accepted: 25 October 2024 Published: 31 October 2024

Abstract - *In today's complex business environments, large organizations often operate within hybrid and heterogeneous IT system landscapes, integrating a range of on-premises and cloud-based business intelligence (BI) and data lake applications. These include platforms such as Databricks, Snowflake, SAP Analytics Cloud (SAC), SAP Datasphere, SAP BW/4HANA, and reporting tools like Microsoft Power BI and Tableau. Such environments cater to diverse business needs and require seamless integration to support complex data modeling, reporting, forecasting, and predictive analytics. However, integrating these systems, each with variations in SQL dialects, architecture, and data models, presents significant challenges, including issues related to user authentication, data security, data aggregation, and maintaining formula consistency and calculation behaviors across different processing layers. This study examines the inconsistencies and issues within these hybrid and heterogeneous environments, focusing on how integration across non-native systems can lead to discrepancies in data structure, query syntax, authentication protocols, and semantics. These differences can result in inaccurate calculations, negatively affecting algorithms, data accuracy, system security and query execution plans. The findings underscore the implications of these integration challenges, highlighting the need for a comprehensive redesign of data flows and calculation logic in hybrid and heterogeneous landscapes. This study also proposes recommendations to improve integration and ensure reliable data reporting outcomes in these environments.*

Keywords - *Aggregation, data security, Query execution plan, System Authentication, Enterprise Cloud and Premises systems, Business Intelligence, Formula collision, SAP, Databricks, Snowflake, System integration.*

1. Introduction

In today's digitally transforming business landscape, hybrid and heterogeneous systems have become essential in many organizations, integrating both on-premises and cloud-based technologies to support diverse and dynamic workloads. Hybrid systems unify different technologies—private on-premise infrastructure combined with public cloud services—within a cohesive architecture, allowing components to interact seamlessly and optimize performance, flexibility, and scalability. On the other hand, heterogeneous systems connect diverse technologies and platforms that are not inherently designed to operate together, requiring custom integrations to facilitate interoperability. These distinct architectural approaches cater to different organizational needs but also introduce complex challenges, especially in data integration, modeling, security and aggregation. As organizations increasingly adopt cloud-native tools like SAP Analytics Cloud (SAC) and SAP Datasphere alongside traditional systems such as SAP BW/4 and Databricks, system integration scenarios grow more complex. Typical use cases include hybrid environments where data originates from either cloud-based applications or on-premise transactional systems and is processed or reported using a variety of cloud and non-native tools, such as Power BI and Tableau. These diverse configurations highlight a critical integration issue: ensuring consistent data aggregation and accurate reporting across interconnected systems. For example, in scenarios where SAP Datasphere feeds live data into SAC, users often encounter aggregation discrepancies, leading to inaccurate calculations, such as inherited percentage calculations or formula collisions. Such issues occur because SAC and SAP Datasphere differ in how they handle calculations. SAC often pushes calculations to the SAP datasphere, resulting in unexpected outcomes when calculations are performed pre-aggregation instead of post-aggregation. In heterogeneous systems, query execution and data retrieval also face considerable challenges. As queries travel through different platforms, communication layers, and APIs, information loss in execution plans, network latency, and varied resource allocation can disrupt data integrity and slow down query performance. For instance, in complex reporting setups involving Power BI and SAP BW, queries need to be translated across multiple systems, risking performance degradation and inaccurate results. As organizations upgrade

from legacy systems to cloud-based solutions, these integration issues can complicate data migration, requiring a comprehensive redesign of data flows, calculation logic, and integration mechanisms. Furthermore, in heterogeneous systems, each component may have its own security protocols, access controls, and authentication methods, which are often not designed to work seamlessly with those of other systems. This diversity requires additional measures to ensure that data remains secure and accessible only to authorized users as it moves across platforms. This paper explores the aggregation, security and query performance challenges in hybrid and heterogeneous systems and explores solution approaches for developers and product engineers. By analyzing system architecture differences, real-time versus batch processing requirements, and the need for standardized execution plans, we propose strategies to streamline data integration and enhance reporting accuracy across interconnected environments.

2. Complex System Integration Definition

2.1. Hybrid Systems

A hybrid system integrates different technologies or approaches within a unified, cohesive architecture. The components might belong to different categories (e.g., cloud vs. on-premise, transactional vs. analytical), but they are designed to work together seamlessly. Hybrid systems often aim to combine the strengths of different technologies to achieve better performance, flexibility, or scalability. In hybrid systems, the different components are designed to work closely together with shared data, interfaces, and processes. The integration is typically seamless and handled within the system itself. Hybrid systems often have a central control mechanism that manages the various components.

- Example: A hybrid cloud system combines private on-premise infrastructure with public cloud services, allowing organizations to flexibly scale their resources as needed.

2.2. Heterogeneous Systems

A heterogeneous system involves different technologies, platforms, or components that are not necessarily unified under a single architecture or framework. These components may interact or communicate, but they retain their individual identities and often require custom integrations to work together. Heterogeneous systems focus on enabling interoperability between distinct systems, where each component may serve a specific purpose or use different underlying technologies.

Lower level of integration: In heterogeneous systems, the components may not be designed to work together natively, so custom integrations (APIs, middleware) are often needed to bridge gaps between the systems. The components often function more independently, and there might be manual or external orchestration to enable communication.

- Example: An enterprise with different databases (e.g., Oracle, SQL Server, and SAP) used across various departments or applications

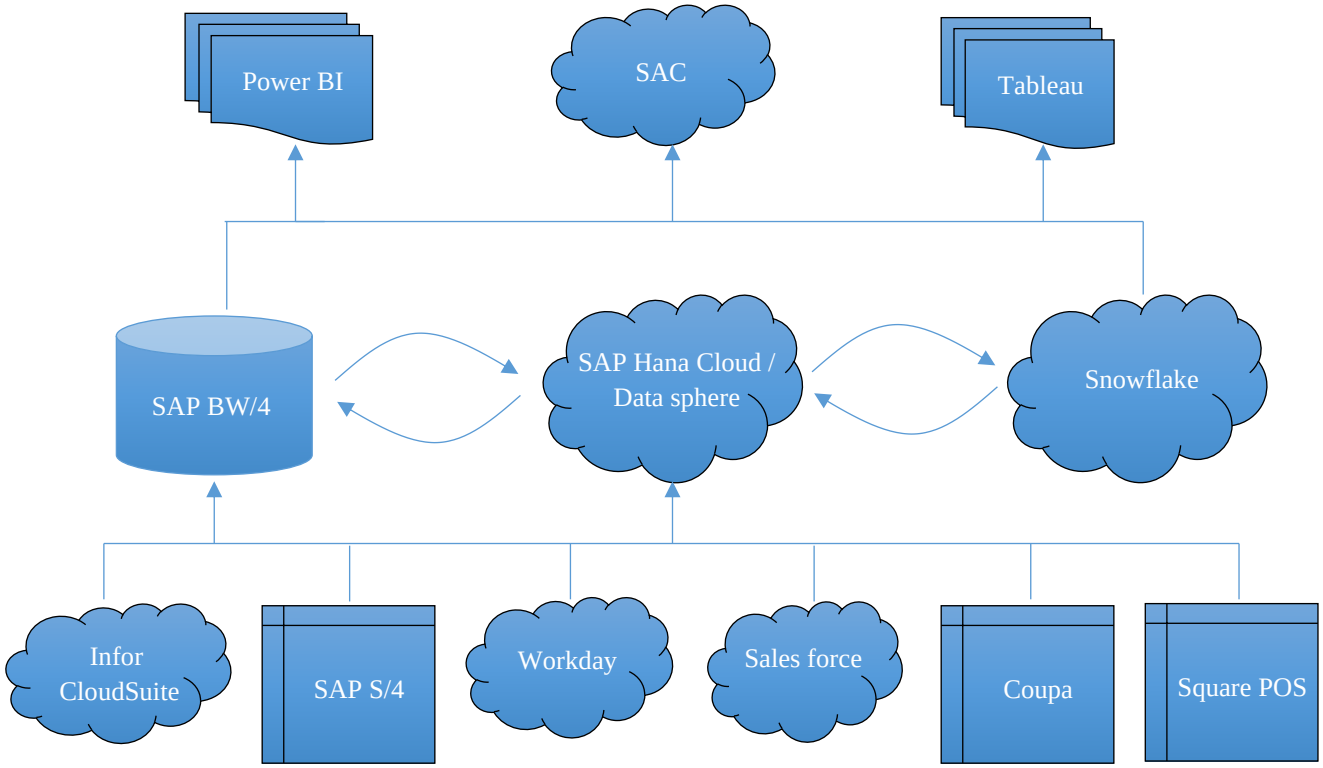

Fig. 1 Heterogenous system integration

3. System Integration Scenarios

To better understand the aggregation challenges, it is important to outline the typical system integration scenarios developers encounter. When the communication is between homogenous systems, the connection is the live connection, which facilitates real-time calls between systems. When the communication is between heterogeneous systems, the connection is either through middleware or OData services, and communication is done through a batch process.

3.1. Scenario 1

Data originates from cloud-based applications like Salesforce. Data is stored and processed in a cloud-based data warehouse or modeling system like SAP Datasphere or Hana Cloud. Reports and analytics are generated using cloud-based tools like SAP Analytics Cloud (SAC).

3.2. Scenario 2

Data originates from on-premise transaction applications such as SAP CRM or databases. Data is stored and processed in an on-premise or cloud-based data warehouse or modeling system like SAP BW/4. Reports and analytics are generated using non-native tools, which could be either on-premise or cloud-based.

3.3. Scenario 3

Data originates from on-premise transaction applications such as SAP CRM or databases. Data is stored and processed in a cloud-based data warehouse or modeling system like SAP HANA Cloud or SAP Datasphere. Reports and analytics are generated using non-native tools such as PowerBI or Tableau, which could be either on-premise or cloud-based.

3.4. Scenario 4

Data originates from on-premise transaction applications such as SAP CRM or databases. Data is stored and processed in a cloud-based data warehouse or modeling system like SAP HANA Cloud or SAP Datasphere. Reports and analytics are generated using native tools such as SAP Analytics Cloud SAC.

Fig. 2 (Scenario 1)

Fig. 3 (Scenario 2)

Fig. 4 (Scenario 3)

Fig. 5 (Scenario 4)

4. Examples of Common Issues in Hybrid and Heterogenous System Environments

4.1. Aggregation Behavior: E.g: GP% Calculation in SAP SAC and Datasphere

Consider the case where a business user attempts to calculate the year-over-year (YoY) difference in Gross Profit Percentage (GP%) using live data from SAP Datasphere through SAC. In this scenario, the GP% for two time periods (e.g., August 2024 and July 2024) is calculated at the Datasphere level and appears correct. However, the results are incorrect when the same calculation is performed in SAC.

4.1.1. Desired Calculation (Post-Aggregation)

The business user wants to subtract the GP% of one period from another after aggregation. For example, GP% for 2024/08 is 0.42, and GP% for 2024/07 is 0.37. The correct result should be $0.42 - 0.37 = 0.05$.

4.1.2. Incorrect Calculation (Pre-Aggregation)

In SAC, the system calculates the difference in Sales, Costs, and Gross Profit (GP) at the granular level before calculating the GP%. As a result, the GP% is recalculated based on these differences, leading to incorrect results. This issue arises because SAC, when connected live to Datasphere, pushes down much of the calculation logic to the source system. This behavior can disrupt the expected calculation order. When this issue appears in activities like data mining or training of AI models, it can affect the quality of Data.

4.2. Formula Collision

Formula collision point:

- Following the column formula (Profit Margin): $220{,}000 / 170{,}000 \approx 1.29$
- Following the row formula (total): $(1.25 + 1.33) = 2.58$

The results are different in this case, clearly demonstrating the formula collision. The query designer would need to specify which formula takes priority to resolve this collision. Furthermore, passing the formula calculation between heterogeneous systems is challenging.

4.3. Query Execution plan

In a heterogeneous system where a report is built in cloud reporting tools like Power BI and planning tools like SAP BPC, but the data is stored in on-premises SAP BW, query execution and data retrieval involves several communication layers between Power BI, SAP BW, and the underlying database. Once the report is executed or refreshed, Power BI will issue a request (query) for the data to SAP BW via the established connection.

Ower BI uses a connector like ODBO/MDX (for OLAP models) or REST API (for OData models) to communicate with SAP BW. Depending on the data model, this query can either be in the form of MDX or DAX. In DirectQuery mode, the request from Power BI is translated into a format SAP BW understands. In this process, the critical information in the execution plan is lost, causing performance issues.

Challenges in Query Execution on Heterogeneous Systems

4.3.1. Data Source Diversity

- Different databases may use different query languages (e.g., SQL for relational databases vs. SPARQL for RDF stores).
- Data might be stored in different formats (e.g., relational tables, JSON documents, key-value pairs).

4.3.2. Network Latency and Bandwidth

- Distributed systems often involve network communication between nodes or data centers.
- Query planners must consider the cost of transferring data across networks.

4.3.3. Resource Allocation

- Different systems may have varying computational resources (e.g., CPU, memory) and storage capabilities.
- The query planner must optimize resource usage across these systems.

4.3.4. Data Location

- The physical location of data can affect query performance.
- Queries that join tables from different databases must account for data transfer costs between locations.

4.3.5. Security and Access Control

- Different systems may have different access control mechanisms.
- The query execution plan must ensure that it respects security policies across all systems.

4.4. Security and Authentication Mechanisms

Each database may use a unique authentication method (e.g., LDAP, Kerberos, token-based), complicating secure communication between systems. Access control models vary widely, and a user role with access to certain data in one database may not align with user roles in another, making access permissions hard to manage across systems.

5. Analysis

5.1. System Architecture

The most obvious reason is the system architecture, specifically microservices versus monolithic architectures. In a monolithic architecture, all components are combined into a single process, allowing for faster communication between components since they exist within the same process. In contrast, microservices divide the application into smaller, independent services that communicate via APIs. This

approach allows individual services to be scaled independently based on demand, leading to more efficient resource usage.

5.2. Real-Time vs. Batch Processing

Another important consideration is the difference between real-time and batch processing. If data needs to be processed in real time, upstream systems may perform calculations directly. Conversely, processing can be deferred to downstream systems for less time-sensitive data. In non-native hybrid system architectures, batch processing is often the only option.

Example: A streaming analytics system processes incoming data instantly and passes only alerts to the upstream system, which handles user notification.

Table 1. Data before aggregation

Comp	Fiscal per	Dept	Sales	Cost	Gross Profit	GP%
8010	202301	HR	500	400	100	0.20
1060	202301	Finance	420	180	240	0.57
2000	202301	Operations	600	400	200	0.33
1000	202301	Inventory	400	300	100	0.25
5000	202301	Engineering	500	250	250	0.50
1000	202301	Marketing	300	100	200	0.67

Table 2. Data after aggregation

	Fiscal Per	Sales	Cost	Gross Profit	GP%
Aggregation	202408	1800	1050	750	0.42
Aggregation	202308	920	580	340	0.37

Table 3. Result displayed in the reporting system

	Sales	Cost	Gross Profit	GP%
Difference	880	470	410	0.47

Table 4. Formula Collision

Quarter	Sales	Costs	Profit Margin (Sales / Costs)
Q1	100,000	80,000	1.25
Q2	120,000	90,000	1.33
Total Q1 + Q2	220,000	170,000	Formula Collision Result ?

6. Impact

The challenges identified in integrating hybrid and heterogeneous systems are critical for developers and enterprise IT teams aiming to optimize data accuracy, operational efficiency, and reporting consistency. When calculations and aggregations differ across platforms like SAP BW/4HANA, SAP Datasphere, Power BI, and Tableau, developers often face substantial obstacles in preserving data integrity during system upgrades or cross-system data flows. A key challenge is the lack of visibility into how instruction sets are processed by downstream systems, which complicates efforts to ensure calculation consistency and precise aggregation behaviors. This ambiguity in calculation logic not only increases the risk of incorrect reporting outcomes but also impedes seamless system upgrades, such as migrating from legacy SAP BW to cloud-native solutions like Datasphere. The impact of these integration challenges is magnified during platform transitions, where developers may attempt to transfer existing data models and calculation logic, only to find that differing aggregation and execution methods lead to inconsistent results. For instance, in SAP BW/4HANA, aggregations are typically handled at the data warehouse level, whereas cloud platforms like SAC and Datasphere may distribute calculations across multiple layers, creating potential discrepancies. These issues highlight the necessity for a more unified, transparent approach to calculation handling and data flow management across platforms. Organizations risk data misinterpretation, decision-making delays, and additional development resources to troubleshoot integration issues without such standardisation.

7. Solution Approaches for Product Engineers and Developers

7.1. Post-Aggregation Calculations in Upstream Systems such as SAC

Ensure that calculations like GP% are performed after aggregation in SAC. This can be achieved using calculated measures in SAC stories that explicitly aggregate values before applying the calculation logic.

7.2. Pre-Aggregating Data in downstream systems such as Datasphere

Developers can pre-calculate measures like GP% in the Datasphere model, ensuring that SAC receives already

aggregated data for reporting. For example, a calculated column in Datasphere might aggregate GP and Sales before computing GP%.

7.3. Custom Formulas locally in downstream reporting systems such as power BI or SAC

If pre-aggregation and post-aggregation adjustments do not resolve the issue, custom formulas can be defined locally within the downstream systems such as SAC stories or Analysis for Office or BPC EPM system to handle specific calculation logic. This ensures the system does not attempt to recalculate values based on raw data points.

7.4. Common Rule-based Framework

These can be used to automate decision rules based on predefined criteria for resolving collisions.

7.5. Tool Agnosticism

In a diverse environment where multiple tools are used for data reporting and analysis, a tool-agnostic query analyzer allows seamless interaction across different systems. By ensuring that the query analyzer can operate independently of the specific tools being used, enterprises can standardize their approach to query execution and analysis. This flexibility helps organizations utilize the best tools for specific tasks without being constrained by compatibility issues.

7.6. API Gateway

Implement an API gateway to act as an intermediary between applications and databases, enabling secure API calls, monitoring, and rate-limiting. This helps control access, manage user sessions, and handle authentication centrally.

7.7. Unified Query Execution Plans

A centralized query analyzer would facilitate a unified execution plan, ensuring that all systems interpret queries similarly, leading to accurate and reliable data insights.

7.8. Federated Identity Management

Use a centralized IAM system (like Okta, Azure AD, or AWS IAM) to manage user identities across on-premise and cloud environments. Federated identity protocols, such as OAuth 2.0 and OpenID Connect (OIDC), enable seamless and secure authentication across systems.

7.9. Common Artificial Intelligence Machine Learning Models

These could analyze historical data to predict which formula prioritization yields optimal results based on past outcomes.

7.10. Future-Proofing Enterprise Solutions

As organizations evolve, the need for integration with new tools and technologies will continue to grow. A common codebase for the query analyzer ensures that enterprises can easily adapt to future changes in their technology stack without requiring extensive rewrites or adjustments to individual reporting tools. This adaptability is crucial for maintaining competitive advantage in a fast-paced business environment.

8. Conclusion

Integrating hybrid and heterogeneous systems requires a foundational shift toward standardized, tool-agnostic query processing and consistent aggregation behaviors. By developing a centralized query execution framework that operates independently of specific tools, organizations can improve the accuracy, reliability, and efficiency of reporting across diverse platforms.

Furthermore, enterprises must prioritize adaptable and future-proofed integration strategies that incorporate unified query analyzers, federated identity management, and consistent calculation handling across systems. Such a holistic approach will enable businesses to adopt new tools and technologies seamlessly, ensuring that their data landscapes remain agile and responsive to the demands of modern business applications. Ultimately, these advancements in system integration will empower enterprises to make more informed decisions, enhancing overall business performance and resilience in a rapidly evolving technological landscape.

References

[1] Qing Li et al., “Applications Integration in a Hybrid Cloud Computing Environment: Modelling and Platform,” *Enterprise Information Systems*, vol. 7, no. 3, pp. 237-271, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[2] Surabhi Saxena et al., “Hybrid Cloud Computing for Data Security System,” *2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, Coimbatore, India, pp. 1-8, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] Zhenxiao Luo et al., “From Batch Processing to Real Time Analytics: Running Presto® at Scale,” *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, Kuala Lumpur, Malaysia, pp. 1598-1609, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[4] Nada Chendeb Taher et al., "An IoT-Cloud Based Solution for Real-Time and Batch Processing of Big Data: Application in Healthcare," *2019 3rd International Conference on Bio-Engineering for Smart Technologies (BioSMART)*, Paris, France, pp. 1-8, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[5] Ramez Elmasri, and Shamkant B. Navathe, *Fundamentals of Database Systems*, Global Edition, Pearson Education, pp. 1-1272, 2016. [Google Scholar] [Publisher Link]

[6] Fotis Savva, Christos Anagnostopoulos, and Peter Triantafillou, "Explaining Aggregates for Exploratory Analytics," *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, pp. 478-487, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[7] Sanket Tavarageri et al., "A Data Analytics Framework for Aggregate Data Analysis," *Arxiv*, pp. 1-10, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[8] Navin Kabra, and David Johns DeWitt, "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans," *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle Washington USA, pp. 106-117, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[9] Saeed Shahrivari, "Beyond Batch Processing: Towards Real-Time and Streaming Big Data," *Computers*, vol. 3, no. 4, pp. 117-129, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[10] E. Bertino, and R. Sandhu, "Database Security - Concepts, Approaches, and Challenges," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 1, pp. 2-19, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[11] Yuxiang Gao, and Peng Zhang, "A Survey of Homogeneous and Heterogeneous System Architectures in High-Performance Computing," *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, New York, NY, USA, pp. 170-175, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[12] Kisung Park et al., "ProcAnalyzer: Effective Code Analyzer for Tuning Imperative Programs in SAP HANA," *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, Portland OR USA, pp. 2709-2712, 2020. [CrossRef] [Google Scholar] [Publisher Link]