RESEARCH-ARTICLE

SAP HANA Cloud: Data Management for Modern Enterprise Applications

NORMAN MAY, SAP SE, Walldorf, Baden-Wurttemberg, Germany

ALEXANDER BÖHM, SAP SE, Walldorf, Baden-Wurttemberg, Germany

DANIEL RITTER, SAP SE, Walldorf, Baden-Wurttemberg, Germany

FRANK RENKES, SAP SE, Walldorf, Baden-Wurttemberg, Germany

MIHNEA ANDREI, SAP SE, Walldorf, Baden-Wurttemberg, Germany

WOLFGANG LEHNER, TUD Dresden University of Technology, Dresden, Sachsen, Germany

Open Access Support provided by:

SAP SE

TUD Dresden University of Technology

Published: 22 June 2025

Citation in BibTeX format

SIGMOD/PODS '25: International Conference on Management of Data
June 22 - 27, 2025
Berlin, Germany

Conference Sponsors:
SIGMOD

https://doi.org/10.1145/3722212.3724452
ISBN: 9798400715648

SAP HANA Cloud: Data Management for Modern Enterprise Applications

Norman May
Alexander Böhm
Daniel Ritter*
Frank Renkes
SAP SE
Walldorf, Germany
{firstname.lastname}@sap.com

Mihnea Andrei
SAP Labs France
Paris, France
mihnea.andrei@sap.com

Wolfgang Lehner
Dresden University of Technology
Dresden, Germany
wolfgang.lehner@tu-dresden.de

Abstract

Cloud Computing environments are a fantastic foundation for unprecedented opportunities in the context of data management. Scalability and availability can be achieved by consequently disaggregating compute and storage and leveraging different data centers around the globe. Although conceptually infinitely available resources promise an easy game for database systems, leveraging those opportunities on the one side and providing enterprise-scale data management solutions for a wide range of extremely challenging business applications on the other side is by far not trivial. Within this paper we share the fundamental principles of SAP HANA Cloud—the data management backbone for all SAP applications—by providing insights into design criteria and technological centerpieces of SAP HANA Cloud as a result of a stringent evolutionary shift from a purely on-premise, in-memory database engine to an on-Cloud, memory-storage hierarchy-aware data management platform. We will motivate the transition and the decision of certain developments by sharing key characteristics of some key applications run by SAP, which require enterprise qualities like availability, cloud-capabilities like resource elasticity, while managing overall costs and the flexibility to adapt the underlying data management system to their specific needs. The overall goal of the paper is to provide insights into the "story of SAP HANA Cloud" by sharing challenges and lessons learned within the evolution towards a composable cloud-based data platform, thus strongly arguing for a "one size fits all" solution, if done right.

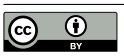
CCS Concepts

• **Information systems → DBMS engine architectures**; **Main memory engines**; *Online analytical processing*; *Enterprise applications.*

Keywords

SAP HANA Cloud, Cloud Architecture Patterns, Cloud Data Management, Multi-Tenancy, Data Lake, Multi-Model

*Corresponding author's email: daniel.ritter@sap.com



SIGMOD-Companion '25, Berlin, Germany

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3724452

ACM Reference Format:
Norman May, Alexander Böhm, Daniel Ritter, Frank Renkes, Mihnea Andrei, and Wolfgang Lehner. 2025. SAP HANA Cloud: Data Management for Modern Enterprise Applications. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3722212.3724452

1 Introduction

HANA Cloud is SAP's strategic data management system as well as data processing service. It is both the main database of SAP's business applications and the backbone of SAP's internal data platform. HANA Cloud is the evolution of SAP's on-premise HANA In-memory database engine, preserving its historical enterprise-level qualities and further adding Cloud qualities by systematically implementing Cloud-native architecture patterns to exploit the capabilities offered by the Cloud.

This paper describes the experience of developing the HANA on-premise database engine into a fully managed data platform operated by SAP. We outline the motivation and objectives of this evolution, the key decisions that have guided this evolution, and today's HANA Cloud as the outcome.

The fundamental decision at the origin of HANA Cloud was to use the on-premise HANA database system as the technological basis of the Cloud service. Consequently, HANA Cloud is not (yet) stricto sensu *"Cloud Native"* in each and every aspect. A main reason was the fact that many SAP applications already had adopted HANA as their database layer, and hence it was important to assure continuity as the HANA database evolved.

When SAP started to elaborate the plans for HANA Cloud, the on-premise HANA database system was the successful Hybrid Transactional Analytical Processing (HTAP) database used by SAP's flagship business application, S/4HANA [19]. S/4HANA itself can be considered the evolution of SAP's Enterprise Resource Planning (ERP) line of business applications: SAP R/2, SAP/R3, SAP ECC. It is the core component of SAP's Business Suite. Unlike the previous ERP versions, S/4HANA was designed specifically to run on the HANA database; and the HANA database was designed specifically to support S/4HANA.

The foundation of this HANA ↔ S/4HANA co-innovation was the observation that modern hardware made HTAP possible, and HTAP was substantially simplifying the customer landscape running business applications. The landscape simplification was obvious: if the application's main database supports, beyond its transactional workload, also advanced analytics, then there is no need

to extract the business data and implement analytics using a different specialized database. Data extraction and transformation, either ETL or ELT, are error-prone costly processes, that, in the ancient on-premise times, the customer IT department had to implement, maintain, and operate. Eliminating them brought both substantial cost savings, better robustness and real-time analytics on transactional data to the customer landscape [30].

Although obvious, the implementation of HTAP was not trivial: at that time, the database market was dominated by specialized databases, which differed in the physical representation of relational tables. Transactional databases were row oriented, to be friendly to point data access and whole row DML operations. Analytical databases were column oriented, to be friendly to range data access on a subset of the columns. The buffer cache behavior and the relative cost of disk IO vs. memory access resulted in a substantial performance gap between them. The advent of that time's modern hardware changed that, by bringing large memory capacity at a reasonable cost and substantial CPU processing power, including a large number of cores and single instruction multiple data (SIMD) instruction set support. That enabled in-memory computing—and in-memory computing was the foundation for delivering superb analytical performance in combination with sufficient throughput on the transactional side.

SAP HANA became the first commercial HTAP system. To be HTAP, HANA uses an in-memory columnar table format, which supports both the performance needed for S/4HANA's transactional workloads and best-in-class performance for its analytical workloads. Beyond being HTAP, the on-premise HANA database system was also multi-model: flat-relational, hierarchies for multi-dimensional schema, graph, document features [6, 7] as well as built-in support for text retrieval and geo-spatial data representations. For the history of the on-premise HANA database system, see [10, 25].

With the strategic transition of SAP to the Cloud, the on-premise S/4HANA business application evolved as the core component of SAP's Cloud suite of business services, accompanied by other business Cloud-native applications. Likewise, the on-premise HANA database system evolved to the HANA Cloud database service. The new S/4HANA still needed its "private" database offering HTAP qualities and multi-model functionality. The on-premise HANA database system has thus been the natural starting point for the HANA Cloud service. However, the on-premise inheritance was not sufficient. To be competitive in the Cloud, the SAP business applications needed their main database to gain Cloud qualities, as having a high availability SLA, being TCO effective, elastic, yet better having separate compute and store elasticity, but still and at the same time preserve and even enhance its enterprise qualities.

In order to outline the evolution of the SAP HANA database to the SAP HANA Cloud data platform, this paper starts in Sect. 3 with the specific requirements of enterprise-grade business applications. In Sect. 4, it then describes the evolution of HANA Cloud: the infrastructure running the service, the Cloud-native architecture patterns that were implemented within HANA Cloud, the Cloud qualities that it has acquired and incrementally enhanced, and the additional functionalities that it gained, like native support of multi-tenancy, a comprehensive underpinning of exploiting object stores and SQL-on-Files, as well as a seamless Spark integration. The paper motivates the choices that were made and highlights some insights gained along the way, such that others may benefit from our failures and success stories. It also illustrates why, in the specific context of SAP, the optimal choice for building a data management platform is *one size fits all* [39].

2 Preserving a Powerful Core: First Days and Expectations of SAP HANA Cloud

We start the description of SAP HANA Cloud's evolutionary steps towards a Cloud-enabled database system by briefly outlining the infrastructure as well as the expected Cloud properties of an enterprise-scale platform.

Cloud Infrastructure. A detailed description of the infrastructure running the service is outside the scope of this paper, which focuses on the evolution of the database technology, when offered as a service. It deserves nevertheless to be mentioned here at least, due to its impact on the service availability and total cost of ownership (TCO), specifically since the cost of operating the service is part of the TCO.

The first approach to a database service, before HANA Cloud itself became generally available, was using a semi-naive home-grown Cloud infrastructure. The cost of operations was surprisingly high, the availability was disappointing, and incident root cause analysis (RCA) showed that in most cases the downtime was caused by the control plane executing wrong actions based on wrong decisions, not by database engine failures. Substantial labor cost was added to the TCO by conducting the RCA itself, and by the human operator intervention to keep the service running in spite of a non-cooperating control plane. Yet worse, the disproportionate but mandatory focus on availability took all bandwidth and the teams could not invest in advanced cost reduction and high availability techniques.

Based on that learning, the HANA Cloud control plane now employs Gardener [14], a Kubernetes distribution (K8s), and its operator architecture pattern: Custom Resource Descriptors and Controllers. The K8s' declarative nature and reliability of the reconciliation loop immediately increased availability and reduced its TCO. Gardner and Kubernetes provide a hyper-scaler agnostic infrastructure layer for HANA Cloud, which is a key enabler for SAP's multi-cloud strategy—not only for HANA Cloud. Furthermore, its modularity and extensibility were instrumental to enhance HANA with Cloud-native architecture patterns, which in turn enhanced the service's Cloud qualities.

Figure 1 shows the major building blocks of the HANA Cloud service infrastructure. The management plane is the customer-facing interface to manage and access HANA Cloud services including Web-based interfaces like HANA Cloud Central, an administration tool. The services in the management plane are built upon APIs that can also be used to programmatically interact with HANA Cloud. For each of the database products offered in the data plane, e. g., SAP HANA database, SAP-managed Spark, or HANA Cloud Datalake, there is a corresponding controller and custom resources in the control plane to define the desired target state. As discussed above, a change in the target state is implemented asynchronously via the associated controllers in a reconciliation loop. Various other such services realized as pair of custom resource and controller

exist, e. g., for managing database tenants. Furthermore, the control plane interacts with a set of BTP core services, which offer functions that are used by various services in SAP's Business Technology Platform (BTP), e. g., a key management system (KMS).

On Delivering Expected Cloud Properties. A key cloud quality of any service, and of an enterprise-grade database service even more so, is its **availability**. In HANA Cloud, service availability is an E2E quality, supported by the resilience and redundancy of all involved components. We distinguish planned downtime, which we eliminate through near-zero downtime operations (NZDO) and unplanned downtime, against which we protect through the resilience of the database engine itself, High Availability, and Disaster Recovery techniques. In case of failures, High Availability is achieved by replicating data synchronously across availability zones in the same region and Disaster Recovery via replication and backup across regions of a hyperscaler.

In addition to availability as one of the fundamental requirements, the team had to cope with the trio of the interrelated qualities: low **TCO**, high **performance**, and maximum **elasticity**. Achieving a low TCO is a challenge of an in-memory database upfront, which delivers superb performance for HTAP scenarios. Thus, elasticity is the key ingredient for a system allowing to identify the sweet spot for a specific application setting. From a technical perspective, the basis of elasticity is scalability: elasticity is the capacity to change quickly the scale of consumed resources. Enterprise applications have a broad range of scale requirements, from the tiny persistency of a multi-tenant service to the large data volumes of an application landscape data platform. HANA Cloud covers both by delivering scalability with the core engine as well as on a platform level.

Within the core engine, HANA Cloud is building upon a native disk-processing as an alternative to pure in-memory computing, as described in [38]. Consequently, yet another quality that HANA Cloud inherits from its on-premise predecessor is the broad cost-performance trade-off space, based on the dynamic choice to do either in-memory or buffer cache processing, at the column partition granularity, switching between them at run-time, and based on the same persistent data. Furthermore, HANA Cloud offers native support for multi-tenancy in order to cope with the extreme variability of tenant sizes—from very small (for test and demo users)

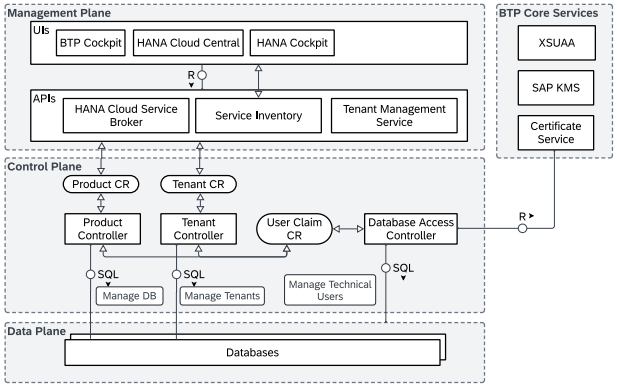


Figure 1: HANA Cloud service enablement

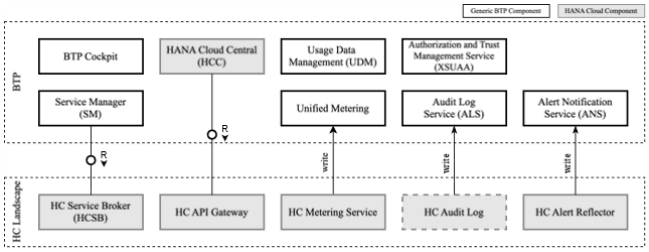


Figure 2: HANA Cloud Management Plane integrated into Business Technology Platform

to high-end enterprise scenarios in a resource-efficient manner. Applications using a complete database instance or native multi-tenancy facilitate the customer's decision to either choose using a customer-specific encryption key (CSEK) or a customer-controlled encryption key (CCEK). For CSEK, SAP handles the lifecycle of the key per customer, while for CCEK this is done by the customer. Moreover, multi-tenancy capabilities allow operational tasks like tenant copy, move, clone as well as the deployment of individual backup and recovery schemes.

Modern Cloud applications are composed of multiple services. They have a separate data processing need, beyond each service having a private main database: they need a data platform for cross-service data integration. HANA Cloud, through its tightly integrated *menage a trois* forms the backbone of SAP's internal data platform consisting of the native HANA database engine Spark and underlying object stores through its native support of delta tables and SQL-on-Files capabilities. Elasticity via scalability patterns is also achieved by the rapid deployment of Elastic Compute Nodes (ECN) [15] providing a single point of entry into the data plane with coarse-grained data replication for high-speed analytical processing. ECNs may be considered a result of applying Cloud-native architecture patterns implemented in HANA Cloud.

Last but not least, the state-of-the-art keeps evolving beyond Cloud technology, the latest example being GenAI. HANA Cloud is enhancing its inherited on-premise multi-model and ML/AI functionality with a native vector data type and vector engine, which allows to efficiently exploit pre-packed LLMs tightly interwoven with relational query processing, and with a Knowledge Graph based on a native triples-store and SPARQL interface.

2.1 On Delivering Common Enterprise Qualities

Before diving into more specific details of application deployment characteristics and challenges, we need to stress "the obvious" for enterprise-critical application support. Operating a fleet of SAP HANA Cloud clusters does not only require strong availability guarantees in an end-to-end manner, but also requires extensive supportability features and high code quality assured by a mature software development process. During the evolutionary journey from an on-premise to Cloud-style system, the team invested heavily into code quality and innovative testing methods [4].

The sheer number of parameters required (a) to substantially reduce the number of tuning knobs and (b) to provide self-tuning properties makes this challenging from operations perspective. Due to the complexity and internal characteristics of this topic, it will be out of scope for the rest of this paper. From a user perspective,

the HANA Cloud management plane consists of microservices that provide interfaces to govern, monitor, and manage their HANA Cloud instances. As a part of the BTP service, HANA Cloud utilizes standard BTP tools for common administrative operations, such as user and role management, billing and usage management, alert notifications, and audit log management (Figure 2). For HANA Cloud-specific administrative operations, such as upgrading HANA database binaries and browsing HANA-specific catalog and monitoring data, HANA Cloud Central (HCC), a dedicated BTP application, is provided.

Finally, the variety of supported business applications pushes the limits on the data modeling side. SAP HANA Cloud offers an end-to-end data modeling environment embracing the multi-model capabilities offered already by the on-premise variant of HANA. It goes without saying that the relational side requires a high coverage of modern SQL language features (e. g., window functions, lateral join, a rich set of built-in functions), procedural language support, and opportunity for externally provided packages of stored procedures. Everything expressed within a multi-model system is expected to be ACID compliant.

2.2 Expectations in a Nutshell

Before providing insights into characteristics and requirements of large business application suites offered and operated by SAP in the Cloud on-top of HANA Cloud, we have holistically provided some insights into the requirements that guided the development of SAP HANA Cloud. During the journey to a Cloud-enabled system, the team successfully preserved existing qualities like a high-performing DB core with best in class HTAP capabilities combining in-memory and disk-aware processing, rich multi-model and query features, as well as extensive capabilities for federation and replication) and added Cloud-native architecture patterns (elasticity, multi-tenancy, control plane for low TCO and native object store support for high-volume datasets) as well as novel features like AI-relevant technologies (e. g., HANA vector engine). Thus, without building from scratch, HANA Cloud has matured to a platform able to provide high HTAP performance on medium to large data sets, cloud store-native performance for analytical scenarios on big data sets, high expressiveness in terms of data model and queries, and full integration into SAP's BTP with competitive TCO.

3 Requirements for Modern Enterprise Applications

In this section, we outline specific requirements we see in enterprise applications running on HANA Cloud. Some of these requirements are quite common over a wide variety of Cloud applications, and we outline how they are supported by HANA Cloud. Beyond this baseline, some SAP applications have specific requirements that go significantly beyond what is typically supported by Cloud databases. We will present those challenges on a conceptual level and sketch how these challenges are addressed by SAP HANA Cloud. A more detailed description of the facilitating technological centerpieces are detailed in the subsequent sections.

3.1 Ariba Business Network: A Million of Skewed Tenants with Large OLTP Workloads

The Ariba business network (Ariba BN) is a Cloud-based environment where enterprise buyers and sellers can trade goods in a marketplace similar to eBay in the private context. As of now, Ariba has over five million buyers and vendors. Its three main applications for sourcing, procurement and the business network each have a data footprint of more than 100TB of compressed data. Typically, database instances of 6TB are used, and the customer data is sharded across the database instances.

In the Ariba business network (BN), participants, i.e. vendors and buyers, are represented as tenants, which are grouped into communities (see Fig. 3). These communities are created by arrival of customers considering data growths over time, which occasionally requires a re-balancing of customers between the communities. Obviously, the data of each tenant needs to be isolated securely using customer-specific or customer-controlled encryption keys. However, as tenants, especially buyers, may be quite small, it is not practical from a TCO perspective to devote a dedicated database to every tenant. Instead, Ariba may share the resources of a single database instance between multiple tenants. Overall there are more than 100k tenants in the Ariba BN with their size highly skewed: A bit more than 100 tenants have a transaction data size larger than 30 GB of data (excluding binary LOBs, which are kept separately within the HANA Data Lake File infrastructure) with the largest tenant having more than 660GB of transactional data. As shown in Fig. 3, the data is not well-balanced between the communities as there are millions of very small tenants with very low data volume.

From this figure, we derive requirements for HANA Cloud regarding the organization and efficient support of tenants. Very large tenants or business-critical customers shall be isolated so that it is possible to backup and recover their data separately from other tenants. Such special customers may also want to use customer-controlled encryption keys to have a tighter control over their data. Small tenants could be grouped together and encrypted using customer-specific keys to limit security issues. Consequently, further lifecycle operations like backup, restore, or database movements would be done on the granularity of groups of tenants. It shall be possible to easily move tenants between communities, and thus between HANA Cloud database instances, to evenly distribute their compute and memory requirements, e. g., into communities of approx. 4TB in size.

Conclusion: Ariba illustrates the need for native support of multi-tenancy inside the database. By using different encryption keys per customer a strong security isolation is achieved, which needs to be complemented by resource isolation between tenants. Multi-tenancy enables not only a seamless scale up and down for tenants within a database instance, but it also allows to move, copy, backup, and restore tenants.

3.2 S/4 HANA: Low Latency HTAP on Large Hot Data

SAP S/4HANA is SAP's flagship ERP product comprising modules covering essential business processes, such as accounting, sourcing and procurement, manufacturing, supply chain, asset management,

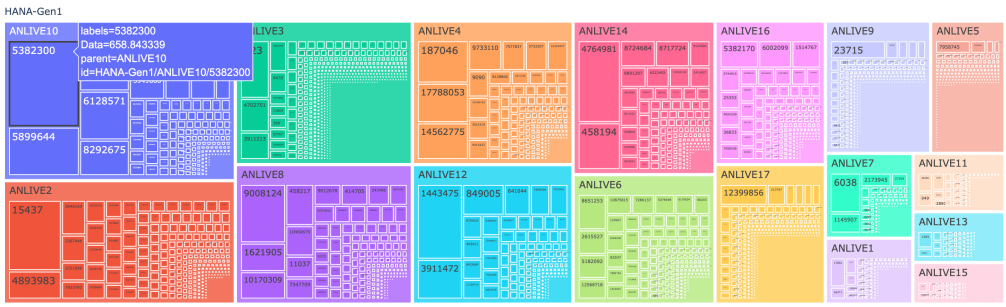


Figure 3: Size distribution of tenants in Ariba business network grouped by community

sales, project management, and engineering [5, 9, 34]. Additionally, 63 country and 17 industry-specific solutions, e. g., for retail or utilities exist indicating the great degree of customizability and extensibility of this product suite. Various deployment options are available, e. g., on-premise, private Cloud or public Cloud, which all run on SAP HANA as its database. The application logic of SAP S/4HANA is mainly implemented in the ABAP (Advanced Business Application Programming) language.

The ABAP application server interacts with SAP HANA via SQL leveraging various specific features of SAP HANA, e. g., passing table-valued parameters to queries in a HANA-native format. The S/4HANA workload is composed of highly concurrent transactional workload, e. g., to process orders, invoices, payments, or stock of materials, supporting millions of transactions per hour. Unlike benchmarks like TPC-C, even the transactional workload is dominated by queries and not updates [20]. Concurrently, S/4HANA users can perform complex analytical queries on the same tables without any replication or transformation of data, e. g., analyzing sales order processing, allowing to analyze the latest transactional snapshot of committed transactions. As shown in Fig. 4, such analytical queries can be quite complex with over 280 relational operators including 274 table or partition accesses, 130 joins, 8 union all operations, 61 group-by operations with complex expressions, or shared sub-plans [25].

As formulating such complex logic in SQL correctly is challenging, SAP defines the virtual data model (VDM), a deeply nested DAG of view definitions. At the lowest level, these views provide an abstraction layer on top of the physical database schema. Higher layers in the view stack represent reusable building blocks for a well-defined business context. In the ABAP application code developers would reference a single view for projecting away irrelevant columns to perform a query as shown in Fig. 4. Clearly, interactive analytic query processing on these normalized base tables poses significant challenges for the query optimizer and query processing engine in SAP HANA [25].

S/4HANA systems are run by the largest enterprise customer resulting in high expectations regarding availability, disaster tolerance, robustness, and performance. Large customers, for example within the retail industry wish to analyze their transactional data down to the level of single line items. In one such system the database schema of S/4HANA systems is mainly optimized for transactional processing with 194k tables and 111k views with an overall table size of 20TB of compressed data. The largest table for accounting documents (ACDOCA) has 153 billion records, 9.7 TB in compressed size distributed over 490 partitions. Three other tables in that system also have more than 100 billion records, and all tables in top 10 store more than 10 billion records. Four tables out of the 194k tables in this S/4HANA system are larger than 1 TB in size; note that these in-memory tables benefit from effective compression schemes. There are even bigger customers than this exemplary S/4HANA instance, which then meet the sizing restrictions of today's scale-up hardware. To handle these, we could successfully deploy S/4HANA on scale-out clusters of the HANA database with multiple nodes. In such a scale-out scenario, data placement is the key to avoid high-latency cross-node operations, e. g., joins in OLTP paths. To this end, SAP HANA offers an automated data placement advisor and benefits from the application classifying tables into table groups that should be kept co-located on a single node.

Conclusion: The ability to do real-time analytics on the latest transactional snapshot is perceived as a distinguishing feature of these applications running on top of SAP HANA in the Cloud. As data does not have to be replicated from a transactional system to a data warehouse costs are reduced not only for operating multiple database systems but also for the associated ETL processing. At the same time the demanding workloads of these applications challenge especially the query optimizer, the query execution engine, and transaction management to achieve high throughput and reliably low response times balancing the demands of the transactional and the analytical workload [28].

3.3 Data Lakehouse: Complex OLAP on Large Data Lakes with Delta Updates at Low Cost

With the data gravity leading towards data lakes, Data Lakehouses (LHs) for analytical query processing became crucial [18, 42], e. g., for deep learning [17], and in this work especially analytical enterprise applications [3, 21, 23, 35]. These use cases are enabled by their interoperability with various tools in their ecosystem based on open formats [42], cost efficiency, high throughput [16], and reliable data lake storage (e. g., AWS S3 [36]). LHs typically feature elastic compute (e. g., Apache Spark [11, 41], Presto, Trino [37]) for processing data physically stored in data formats like the column-oriented Parquet [29, 40] or ORC [13]. For efficient elastic query processing with relational table semantics and analytical capabilities on the data, open table formats (OTFs) such as Delta Lake tables [31] or Apache Iceberg [12] manage the underlying data through a separate

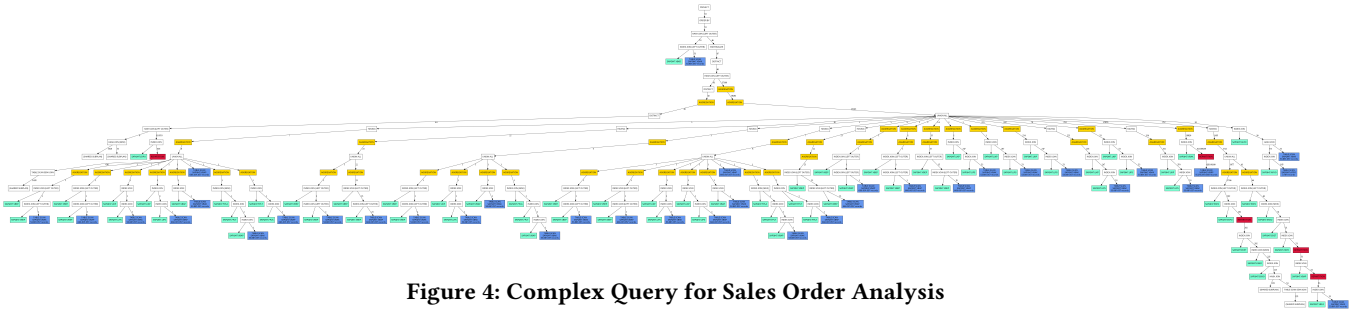


Figure 4: Complex Query for Sales Order Analysis

metadata layer on the object store [2, 42]. Despite their design specifying certain relational DBMS functionality like transactions for single tables and rows, indexes, and time travel [2, 42], analytical query processing for enterprise applications remain challenging.

Conclusion: Enterprise-scale data warehouses and lakehouse deployments feature complex analytical queries using features beyond standard SQL, e. g., data cubes or multi-dimensional queries. In addition, advanced privilege modeling and multi-query, multi-table consistency is required for certain regions of the overall data sets.

3.4 Integrated Business Planning: Fluctuating OLAP & ML Workloads at Low Costs

SAP Integrated Business Planning (SAP IBP) for Supply Chain [1] is a Cloud-based solution for supply chain planning. The application helps enterprises to forecast product demand, build optimized execution plans for their supply chain networks, and respond effectively to unexpected changes. As a consequence of the application characteristics, SAP IBP is one of the most demanding applications suites to be supported by SAP HANA.

First of all, IBP requires a data management platform that not only stores data but more importantly functions as an analytical engine supporting what-if simulations, a platform for traditional planning algorithms, and compute-intensive machine learning (ML) algorithms. To meet these demands, the DBMS must offer analytical in-memory performance and built-in ML capabilities, without moving a large amount of data to external systems for processing.

Secondly, the use of and resulting workload for the underlying data management platform is very sporadic, i. e., the customer workloads are known only shortly in advance, which requires a flexible provisioning technique: while spikes in the workload may quickly lead to temporary under-provisioning of resources, simple over-provisioning to cope with an increasing number of such fluctuating resource demanding workloads in large-scale is obviously also not feasible.

Conclusion: IBP as one of SAP's core application suites requires (a) support of extensive analytics and ML functionality executed close to the data and (b) a sophisticated workload prediction to facilitate high compute elasticity.

3.5 Beyond the Gang of Four

Although S/4HANA, Ariba, Data Lakehouses, and SAP IBP are spanning already a gigantic design space of requirements on an underlying "one-size-fits-all" data management platform, SAP HANA is the backbone of many more applications offered directly by SAP as well as by the worldwide ecosystem consisting of partners and customers. For example, SAP Cloud Application Lifecycle Management requires—similar to Ariba discussed in Sect. 3.1—support for thousands of very small tenants in a cost-effective and scalable way.

Furthermore, SuccessFactors for human capital management (HCM) in the Cloud features a demanding HTAP workload. SuccessFactors has deployed several hundred TB of data across hundreds of database instances processing billions of transactional and analytical statements per day. A distinguishing feature of SuccessFactors and Ariba is that more than 70% of the data footprint of these applications are based on LOB data, which must be accessible online to the application.

Conclusion: Although SAP is primarily known for the ERP-suite, it offers a large portfolio of different applications with many diverging requirements in performance as well as data structure and workload complexity. HANA Cloud as the underlying eco-system has to provide a robust, scalable, and low-TCO solution.

3.6 Requirements—Overview

In the following, we review the characteristics of modern enterprise applications. We remark that to the best of our knowledge no other database system covers all the requirements presented in Tab. 1. In the rows of the tables, we summarize the requirements of the business applications we used as examples to derive requirements for modern Cloud database systems. The heading of the columns groups these requirements along the Cloud qualities, i. e., costs, elasticity, availability, variety, and volume. The columns refer to solutions we considered during the implementation of SAP HANA Cloud to address these requirements. The cells indicate, how relevant the Cloud qualities and related solution options are for each of the presented business applications.

Modern enterprise applications require operational database systems with additional capabilities not found in current systems. Notably, the required capabilities are barely overlapping. The common approach of having one best-of-breed system for every requirement would require a zoo of well-integrated but different data systems. As an alternative to this approach, this paper presents how these requirements can be addressed within a single database system.

4 HANA Cloud Data Platform as Persistency Layer for Enterprise Applications

Starting from an on-premise in-memory database engine, the SAP HANA Cloud team has been confronted with a wide spectrum of requirements coming from business applications and found itself

Table 1: Requirements of SAP Business Applications

	Costs				Elasticity				Availability		Variety		Volume
	Cache	Expr.	SoF	MT	WL	Auto Scale	Repl.	Fed. / UDF	Restart	HA/DR	MM	ML	
S/4 HANA ("Complex HTAP")	👎	👍	👍	👍	👍	👍	👎	👎	👍	👍	👎	👎	👍
Ariba Network ("many tenants")	👎	👍	👍	👍	👍	👎	👎	👎	👍	👍	👎	👎	👍
Successfactors ("HTAP and many LOBs")	👍	👍	👍	👍	👍	👎	👎	👎	👍	👍	👎	👎	👍
SAP IBP ("fluctuating workloads")	👎	👍	👎	👎	👍	👍	👎	👍	👎	👎	👎	👍	👎
SAP "Data Lakehouse" ("data sizes")	👍	👍	👍	👍	👎	👍	👍	👍	👎	👎	👍	👍	👍

Required: 👍 Potentially required: 👍 Not required: 👎

Temperature-based data placement (Cache), Expressiveness / compiled query engine (Expr.), SQL-on-Files (SoF), Scalability (Auto Scale), Federation / user-defined functions / transformations (Fed. / UDF), Replication (Repl.), Multimodel (MM), Machine Learning (ML)

always in the area of conflict between delivering functional extensions and achieving Cloud qualities with respect to scalability, robustness, security, and essentially low TCO. In addition, SAP's core business in offering large-scale end-to-end enterprise solutions in many different flavors requires a strong core of customer-tailored solutions for core requirements of these applications complemented by open technologies where a strong ecosystem is the key.

As a result, HANA Cloud Data Platform consists of three major components, which are seamlessly integrated and are providing enterprise scale data offerings. The **HANA Cloud Database Core Engine**—as the direct successor of the SAP HANA on-premise in-memory engine—provides a full-scale HTAP system with multi-model and big data capabilities. Section 5 will dive into more details of the evolution and functional extensions of this component. The **HANA Cloud Data Lake Files** (HDLF) component builds the storage backbone for all SAP applications (see below). HDLF is obviously extensively used by the HANA core engine for direct SQL-on-Files processing. As the third major component, SAP HANA Cloud embraces **Apache Spark** as a fully managed service augmenting it with enterprise-critical services, thus resulting in a first class citizen of the SAP HANA Cloud offering. In the remainder of the section, we briefly sketch the relevant characteristics of and contributions to HDLF and Spark.

4.1 HANA Cloud Data Lake Files (HDLF)

HDLF serves a need for access to industry-standard Object Stores using a uniform access mechanism for all major Cloud providers namely AWS S3, Azure Files, Google Cloud Store, Alibaba OSS. HDLF provides REST-based access for data ingress/egress as well as metadata operations in combination with SAP's authentication and authorization mechanisms based on standards. Cross-tenant federation is supported including full data encryption using managed keys. HDLF provides a unified interface of the basic services offered by object stores of hyperscalers satisfying the need of elastically scalable storage capacity in an enterprise-scale environment. In such an environment data confidentiality, integrity, and availability is non-negotiable, and data management must follow applicable law and regulatory requirements. Allowing applications, SAP as well as non-SAP, to store and access data in HDL Files, allowing business analysts, data engineers, and data scientists to work with data in HDLF, using SAP as well as non-SAP technologies, is key to its success/adoption. HDLF thus allows for authentication, policy enforcement and definition, storage provisioning, and a virtual namespace under which all storage managed by HDLF is made accessible to applications and users.

One of the services provided by HDLF is a governed access to all connected storage and their data lifecycle via policies governed to enforce security and compliance. Publishers are able to author policies and store them in a governance policy store; if no policies exist a default policy applies. Policies govern access to and lifecycle of their data. They are defined in the context of a resource or a resource set that is defined by a URI within HDL Files. A resource URI can address data or metadata describing data. Policies are able to grant or deny authorizations for a given consumer for a given operation on a given URI. They can refer to metadata of data, locality of data, i.e. the region or availability zone of the hyperscaler, as well as the attributes of the consumer to the extent that these are available through the request context. A strong focus is given to a request's origin, path, and all involved identities.

SAP HANA Cloud uses the concept of zones to group tenants of different applications and services into a common context. Within a zone it is guaranteed that all applications share a common set of users, and services can therefore allow sharing of data within this zone. For governed access to data across regions of one hyperscaler or even between different hyperscalers HDL Files can offer a single virtual namespace for governed access to storage. This includes support for cross-zone federation without violating the strong access and lifecycle governance requirements for HDL Files.

Merits for HANA Cloud: HDLF is SAP's abstraction and extension of industry-standard object stores leveraging the scalability and availability of existing infrastructures on the one hand and enhancing the core service with business-critical capabilities in the context of security and compliance, authentication and authorization as well as a single virtual namespace across different hyperscalers and provider regions on the other hand.

4.2 Managed Apache Spark

Many SAP lines of business (LoBs), SAP partners, and customers utilize capabilities of Apache Spark for streaming and batch processing workloads, ingesting data from various sources, e. g., LoB-managed

Kafka clusters. As industry-leading solution for distributed processing of big data workloads and machine learning with a comprehensive eco-system of libraries and flow definitions, Apache Spark is an attractive solution for these stakeholders. Additionally, LoBs wish to offer curated data as data products to their customers using standardized file formats and sharing technologies. This requirement sparked a development to embrace Apache Spark as a technology platform and include it as a first-class citizen into the SAP HANA Cloud Data Platform. In addition to infrastructure services, specifically focusing on open table formats (OTF) such as Delta Lake [31] and Apache Iceberg [12] is of utmost relevance from a technological perspective to allow a secure and efficient integration in and cooperation with HDLF as the underlying uniform object store. Spark jobs are fully Kubernetes-integrated Cloud-native workloads. To aid fair scheduling among multiple tenants over a single resource pool and to mitigate challenges that arise when operating big data workloads as Kubernetes jobs, a dedicated batch job scheduler with more sophisticated workload-aware queue management is employed.

For a secure communication between Spark applications and the HANA database a trust relation must be established. Consequently, LoBs are expected to configure trust between HANA Cloud, HANA Database and Spark, as well as to establish trust between HANA Cloud, HANA Database and HDL Files, allowing a HANA tenant (or HDI container [26]) to communicate with its corresponding file container within the same LoB customer subaccount with strong security. In order to facilitate a secure communication of Spark jobs with the systems in HANA Cloud, Apache Spark can securely retrieve credentials from the BTP Credential Store service and thus leverage the same for authentication with systems external to Apache Spark.

In HANA Cloud, the data lakes are implemented by HDL Files. HDL Files provide client libraries for Apache Spark, allowing Spark applications to read or write Delta tables. Thus, LoBs can produce data in files, described as a table, specific to a target audience, and provide access to the data using delta sharing as open protocol. OTF Files (Tables) stored in HDL Files, just like other files, are served to authenticated users. Together, these technologies enable building a distributed data mesh.

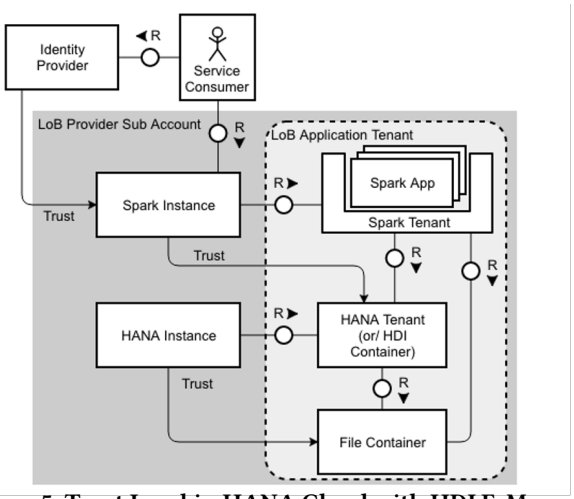


Figure 5: Trust Level in HANA Cloud with HDLF, Managed Apache Spark, and HANA Core Engine.

Merits for HANA Cloud: A fully managed and tightly integrated Apache Spark deployment facilitates to use the existing eco-system of libraries and pre-defined dataflows and is therefore heavily used for data wrangling activities.

5 HANA Cloud Database Core Engine

Due to the existence of the well-proven technology of the HANA core engine as well as the dependency of many applications and application suites on HANA's functional and non-functional properties, the team decided on a strategy between the extreme cases of "completely starting from scratch" and "just putting an on-premise engine into a container". Over the last years, HANA Cloud development pursued a systematic process of applying Cloud architectural patterns to the overall system by carefully balancing between incrementally re-writing parts of the engine to make HANA a good Cloud citizen and extending the functionality with respect to requirements from application side. As a result, SAP HANA evolved from a pure in-memory database engine to an ecosystem consisting of multiple components—internally called "Cloud Bricks", which can be "leveraged" for a specific application thus addressing the main requirements discussed above with respect to *economic viability*, *costs*, *elasticity*, *availability*, and *variety*. Subsequently, we will point out the main building blocks of the current HANA Cloud.

5.1 In-Memory Query Processing

The HANA Execution Engine (HEX) is the new query execution engine of the SAP HANA database and specifically tuned for query performance and resource efficiency. Next to a rich set of features, HEX provides extensibility for different non-relational functionalities appearing in the context of enhanced analytics and AI/ML, such as vector type. A major differentiating feature of HEX compared to other database execution engines (such as Meta's Velox or Databrick's Photon) is the ability to process HTAP workloads, for example, by providing a context-aware parallelization strategy. Thus, HEX is able to efficiently process queries that touch a few rows of a single table and run less than a millisecond to queries that scan, join, and aggregate tables of very high cardinality. Although query processing in HEX is optimized for data residing in memory, HEX works together with NSE to access block storage (see Sect. 5.2). For scale-out, HEX implements homogeneous distributed query processing including a set of optimizations to minimize the number of requests being sent within the cluster. For example, to avoid the access of remote tables, distributed query processing in HEX triggers the early materialization of values. HEX is built around the concept of pipelining. Conceptually, a pipeline consists of a sequence of operators, which operate on individual tuples, and a pipeline breaker at the end holding the result. HEX implements a version of push-based pipelining that passes a chunk of tuples—sized to fit into cache—of an intermediate result from one operator to the next. The HEX framework can fuse the code of multiple operators of a pipeline into a single LLVM program, which is then either compiled into machine code or interpreted. Just-in-time compilation based on the LLVM tool chain is particularly helpful to speed up the processing of complex expressions on large data sets.

Late materialization allows HEX to postpone reading of columns from the column store to the point in the query execution plan, where the columns are needed by an operator for the first time. HEX also implements several performance optimizations that leverage existing partitioning information. Compile-time partition pruning decides at query optimization time which partitions can be skipped for a given filter to still calculate the correct query result. Runtime pruning does the same for filters with runtime parameters. Next to partition pruning, dynamic pruning allows to skip reading of partitions based on intermediate results.

Merits for HANA Cloud: The well-proven in-memory database engine is still the main pillar for HTAP query processing. The HEX framework allows to process not only relational queries but facilitates to coordinate and execute complex and data-intensive business processing tasks.

5.2 Disk-centric Query Processing

In addition to its in-memory column store, SAP HANA supports a page-based columnar storage option, called **Native Storage Extension** (NSE) [38]. Upon access, pages of the dictionary-compressed columns or dictionaries are loaded from disk into an in-memory page buffer and evicted following an LRU-based eviction strategy. This disk-based data processing adds another option on the cost-performance spectrum allowing to provision smaller DRAM capacity and loading data into memory on demand. It also extends the usage scenarios for SAP HANA to cases where the working set of data does not necessarily fit into the memory of the host system. With a recent extension, SAP HANA now allows to choose between the traditional in-memory column representation or the page-based column organization when loading a column from disk based on the metadata defined for the column. We refer to [38] for further details on NSE.

The SAP HANA execution engine, HEX, supports spilling intermediate results stored in pipeline breakers to disk complementing the disk-based table storage discussed above. This further pushes the boundaries of query processing beyond the DRAM capacity of a single host.

Merits for HANA Cloud: NSE offers new options on the price-performance spectrum allowing to load pages from disk following the conventional page buffer technology. NSE is complemented by HEX query operators that can spill intermediate results to disk. This enables application scenarios for SAP HANA that were previously limited by the available DRAM capacity of the host.

5.3 Cloud Object Store Query Processing

Parquet files stored in HDLF may be exposed via an open table format (OTF) like delta tables [31]. Access to delta tables in an object store in the same cluster or even in a remote object store of another hyperscaler can be provided via delta sharing. SAP HANA allows for query processing on files stored in the object store, in particular delta tables, using the federation capabilities of HANA. Delta tables are represented in the SAP HANA metadata as virtual tables with a defined schema, allowing for the same way to access as for tables [27].

Figure 6 shows the high-level architecture for **SQL-on-Files processing** (SoF) in SAP HANA Cloud. The delta sharing service

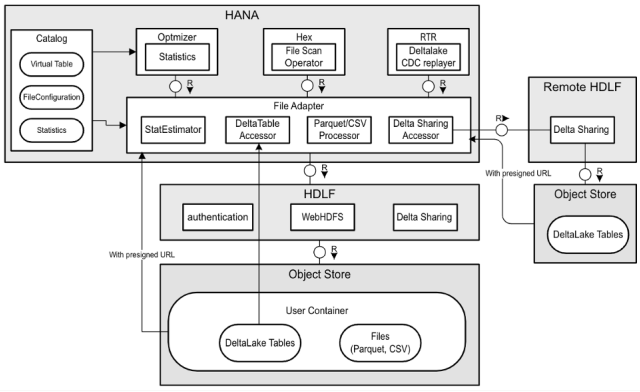


Figure 6: SQL-on-Files Architecture

offers access to Parquet files via pre-signed URLs, and via this technology also access to delta tables in remote object stores is available. As another alternative HANA can access files directly based on file name and path, i.e. without delta sharing involved. In either case, the File Adapter of SAP HANA parses the Parquet files and applies filters or projection that can be pushed down to the level of a file access to reduce the amount of data to be processed as early as possible. The HEX query execution engine then receives the column chunks produced by the File Adapter and continues query processing in the same way as for local tables.

The delta table metadata and file statistics either provided by the delta sharing server or stored in the footer of the Parquet files allows for skipping entire files when scanning a delta table at runtime and thereby improving performance and reducing data transfer costs. File statistics, such as row count, distinct count, or minimum and maximum are used when optimizing queries accessing delta tables.

Merits for HANA Cloud: SQL-on-Files processing operating on open file formats is the foundation for PB-scale data management and allows to seamlessly tap into large data lake environments while preserving the business-application perspective (complex business objects, privilege management, etc.) and expressivity of HANA SQL.

5.4 Data-Mesh Query Processing

SAP HANA Cloud leverages powerful federation capabilities to seamlessly make queries to various external data sources - that include both relational and non-relational databases and even unstructured or semi-structured sources such as spreadsheets and file systems. However, in order to implement an efficient Data Mesh, HANA provides the **Fabric Virtual Table** (FVT) as option for local replicas of remote data—either with synchronous or asynchronous replication—for better query performance and availability. Thus, users may conveniently toggle the options between full federation and replication modes for a remote data source. For remote data federation and replication, SAP HANA provides two major artifacts—remote data sources and virtual tables.

A remote data source refers to an external entity such as a database service, to which HANA's federation can make a connection. HANA can refer to various data source types like databases using

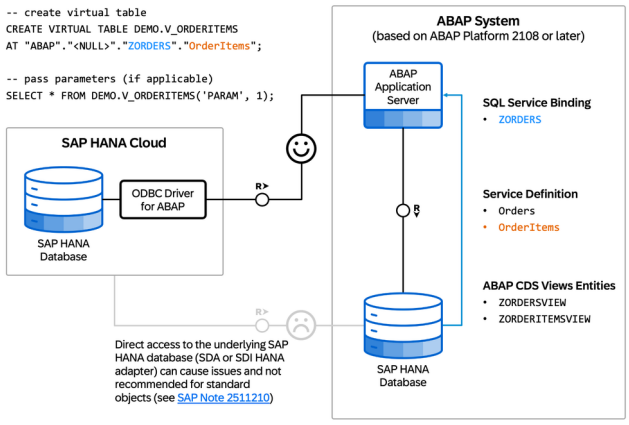


Figure 7: Federation into SAP's Application Layer

ODBC or services exposing a REST API. A virtual table is a synonym for a catalog object to abstract a remote data source that physically exists outside HANA. It is therefore a proxy object for reading and manipulating data in the remote as if it were a local HANA table. Like any other local tables, virtual tables can be read or manipulated with HANA's SQL statements, which are translated into compatible and optimal equivalent statements or operations to be executed on the remote system.

The following applies to the Fabric Virtual Table feature:

- A fabric virtual table cannot have multiple replica tables.
- Multiple fabric virtual tables can share a single replica table when the source table in the remote system is the same.
- A replica table can be configured to use in-memory tables or NSE.
- Replicas can be created with a single snapshot or be incrementally updated as DML operations are executed on the remote source.

Fabric Virtual Tables rely on Real Time Replication (RTR) to replicate a remote object in real time. For snapshot replication, Fabric Virtual Table utilizes an `INSERT INTO ... SELECT` statement.

While the core technology is used in many "traditional" federation scenarios, SAP HANA also pushes the envelope in data integration by remotely accessing ABAP Entities. ABAP is SAP's application server hosting the application logic for a large number of SAP applications, especially for SAP S/4 (see section 3.2). Since accessing data at the database level does miss many of the application semantics in "refining" the raw data stored in the database, ABAP-federation allows to "see" business entities like any front-end application. Examples of application semantics "augmented" within the application server are the consideration of access rights defined via potentially complex business rules or the composition of business artifacts due to some additional information not available at the database layer (for example, a business partner may be resolved into a supplier or customer depending on the previous business transaction).

Figure 7 illustrates the access to business entities at the level of the ABAP application server. Within the federation framework both aggregations and joins are pushed down to the remote ABAP system and executed there. If the ABAP server provides statistics over its runtime data, the query optimizer will consider it and create the optimal execution plan for this particular data access pattern.

Merits for HANA Cloud: Providing a data management solution in large customer scenarios with many operational systems potentially around the globe is based on extensively using data mesh technology. Fabric Virtual Tables are instrumental in providing functionality as well as performance (via transparent and object-based replication).

5.5 Query Processing with Elastic Compute Nodes

SAP HANA in its on-premise version already provided comprehensive support for scale-up and scale-out. On the scale-up side, HANA implemented cost-aware NUMA strategies for efficient access to data in memory [32, 33]. On the scale-out side, SAP HANA provides a cluster-based setting with client-side routing for efficient transactional processing considering data locality [22]. Analytical query processing exploits scale-out capabilities for increased throughput. In addition, **SAP HANA Elastic Compute Node** (ECN) [15] is a feature that provides elasticity for CPU and memory to the overall HANA Cloud database offering and is a prime example of the evolution towards a fully elastic architecture that separates storage and compute.

A new HANA node type called Elastic Compute Node was introduced as a part of the HANA landscape configuration. When a node has a compute role, it runs with a special semantics for persistence as it does not allow the creation of persistent tables and is thus excluded from HANA System Replication.

Although ECN is therefore perceived to be persistence-less, it has a persistence layer and uses persistence volumes serving all types of temporary tables, handling LOB, and serving table replicas. However, an ECN volume is not a part of global persistence operations like disaster recovery (DR) and backup and restore (BNR). In this regard, ECN does not offer durability in the sense of ACID. As a consequence, an ECN is excluded from log replay and from distributed transaction (DTX) commit logs but is notified during DTX handling to clean up necessary transaction related objects.

ECN has its own SQL endpoint, users can open a connection directly to an ECN. For transparent elasticity with a single entry point, client-side statement routing and extended workload management features are used. Workload management is extended to specify the routing location to cover the ECN routing use case. Certain types of workloads like all statements of a specific user or queries with a specific session variable can be routed to ECN. For ECN-aware query processing, the SQL optimizer generates a query plan to make use of as much computing power of the ECN as possible. The optimizer generates query plans accordingly, where non-leaf nodes will always be executed in ECN while leaf nodes will be executed in the data location. By default, for persistent tables data locations will be the primary HANA node where the table partitions are located. However, to reduce data transfer during query processing, HANA Advanced (Synchronous) Table Replication can be leveraged by ECN to install a local replica. The existence of replicas in ECNs is controlled by the application and comes obviously with some overhead. The initial synchronization may take several minutes to

Figure 8: Customer Scenario of ECN usage

hours depending on the size of table, and replication needs to be re-initialized after takeover or recovery.

Merits for HANA Cloud: Elastic Compute Nodes are one core mechanism in HANA to implement elasticity in the level of database processes. Business logic, e. g., quarter end calculations, very often allows to "plan ahead" and deploy additional HANA resources using DB/application-co-design strategies. As a concrete example, Fig. 8 shows the timeline of an elasticity scenario for an IBP application (Sect. 3.4) instance scheduling big, overlapping planning jobs to start every 5 minutes. The timeline shows an instantiation of up to 43 ECNs with a size between 200GB and 4TB and an overall peak of 23TB main memory for the planning process. As can be seen, using ECNs HANA Cloud can breath autonomously.

5.6 Multi-Model and Vector Processing

In order to accommodate requirements to support use cases relying on vector embeddings (like Retrieval Augmented Generation (RAG) [24]), HANA has been extended by **Vector Database capabilities** to find the most relevant entries in a given Vector database. The extension consists of:

- a new SQL data type `REAL_VECTOR` that represents a vector of single-precision floating-point number elements,
- a specialized implementation of column fragments in the column store,
- functions to create vectors from textual and binary formats and to serialize them again to textual and binary formats,
- functions to compute the cosine similarity and the L_2 distance of two vectors,
- an index used to accelerate top-k most similar vectors searches based on the idea of hierarchical navigable small worlds [8],
- integration of vector processing in the query optimizer and query processing engines of SAP HANA.

While vector support is reflected by a new relational data type, SAP HANA Cloud offers an alternative to a strict tabular modeling and storing of data. On the one hand, the **SAP HANA JSON Document Store** is a dedicated store in HANA to store and process JSON data. It is completely independent from the Column and Row Stores. Document Store runs in a dedicated HANA service. SQL statements containing references to the document store are centrally optimized and the logical plans for accessing the content of the document store are sent to the Document Store for execution. Document Store also runs in the same transactional domain and uses the HANA persistence, hence providing full ACID capabilities.

Merits for HANA Cloud: Multi-Model capabilities have already proven extremely beneficial for complex structured data sets (esp. considering graphs) and document-style data sets. The vector data type supports upcoming application scenarios natively embracing LLMs and related technologies, thus strengthening HANA Cloud as the central eco-system for data-related services.

5.7 Application-specific Query Processing

In addition to multi-model support, HANA Cloud offers HANA Cloud Plugins as a delivery mechanism for **Application Function Libraries** (AFL). An AFL consists of procedures that are released by stakeholders from within SAP. Business applications may either push-down specific application logic into the data management platform or leverage specific extensions developed outside of the HANA core. A prime example is the **Predictive Analytics Library** (PAL), which is extensively used for analytical scenarios and is providing comprehensive support of a phletora of highly specialized statistical and mathematical functions. Requests of an AFL-PAL-procedure are routed internally to a specific service partially running in its own Kubernetes Pod but shared (since stateless) within a HANA cluster.

It is worth mentioning that also ECNs are capable of running Application Function Library (AFL), since ECNs are a clone of a regular HANA node with all components and functionalities. As the ECN is separated from the main HANA node, an ECN crash does not affect regular execution but at the same time it provides all the advantages of executing the AFL code close to the data logically within the core engine (and within the same pod).

Merits for HANA Cloud: SAP's application has a large number of specialized (and proprietary) application domain-specific process indicators to compute. While some can be computed within the application server, many indicators are compute- and data-intensive. HANA Cloud Plugins allow to deploy (SAP internal, yet HANA external) application code in managed environments. Although often underestimated, it has a high relevance for supporting application code "close to the data".

5.8 Native Multi-Tenancy

One of the lessons learned by running a phletora of different applications of varying deployment sizes is the necessity of an efficient (and ideally natively supported) multi-tenancy concept. HANA has devised and implemented support for schema-based tenant isolation providing the following core features in HANA Cloud for tenant-aware data management.

Even though a tenant's data is grouped at different granularities, a fundamental set of features need to be provided for all multi-tenancy database architectures. We outline the main features needed in HANA Cloud for tenant-aware data management.

(1) **Tenant Lifecycle Management:** Tenants are first class objects in HANA Cloud,their functionality is extended to cover tenant lifecycle management, i.e. to create, drop, move, copy, and recover from backup. **Tenant-level Copy** is an internal operation and thus the basis of implementing the tenant-level offline move, online move, recovery from backup, and

cloning. Since multiple tenants usually reside on one HANA instance, some of them need to be moved to a different HANA instance as they grow. On the one side a **Tenant-level Online Move** needs to be online because no downtime is accepted for such an operation. On the other side, a **Tenant-level Offline Move** is a disruptive operation, during which the application may not use the tenant. Finally **Tenant-level Recovery** is necessary, because usually only the data of a specific corrupted tenant needs to be recovered to a specific point in time. This feature complements the instance-level backup and restore feature, which addresses the need for durability in case of failures.

(2) **Tenant Content Management:** Database artifacts like users, tables, or views can be assigned to tenants. Schema-level artifacts created by a tenant-user are implicitly assigned to that tenant. The HANA Cloud DDLs are extended to support the specification of the tenant's group of objects.

(3) **Tenant-level Encryption**: HANA's encryption capabilities are extended with the functionality to encrypt each tenant's data-at-rest with a separate key even though data are not physically segregated anymore but do now reside in the same database. Tenant-specific data encryption is only ensured for data in the tables that are assigned to a tenant. Shared data and meta-data (catalog, users, shared containers) are encrypted only with the DB instance key. By default HANA native tenancy uses CSEK to encrypt the data of a tenant. In this setup, SAP creates and manages keys for different database tenants to assure isolation in the storage layer. As an alternative, customers can use CCEKs where the customer creates and manages the lifecycle of keys of a tenant in a KMS, e. g., by revoking the key, a customer can make sure that not even SAP is able to access the customer data in the main database persistency, the database log or the backup.

(4) **Isolation, Security and Authorizations**: Isolation and cost-efficiency are conflicting Cloud qualities. The main reason of introducing multi-tenancy to HANA Cloud is cost efficiency for small data volumes, and thus this Cloud quality takes precedence over the others. Nevertheless, isolation and security may not be ignored and must still be offered, resulting in a trade-off with cost-efficiency. Due to the schema-based tenant isolation with a user being the owner of the tenant, the multi-tenancy feature inherits various security features. The privileges assigned to the tenant user are transported during the tenant move. Also, care is taken to avoid cross-tenant dependencies, e. g., via foreign key constraints, which would interfere with tenant lifecycle operations.

(5) **Tenant-level Workload Management**: Another essential functionality is tenant-level quotas. While such quotas target memory and thread limits in the current phase, they will eventually be extended to limit network IO, disk IO, and disk size. This feature is based on the instance-level workload management features of SAP HANA [28].

Merits for HANA Cloud: Native support of multi-tenancy is a core driver for cost-efficiently operating already more than eight thousand productive tenants for SAP-internal applications. As a matter of fact, 25% of the services in SAP's BTP have adopted native HANA Cloud multi-tenancy already.

6 Summary and Conclusion

SAP HANA started out as an on-prem engine paving the way for in-memory database systems to satisfy demanding SAP applications. While moving strategically to the Cloud, the SAP HANA Cloud offering is now positioned as a data management platform to cater for all SAP application suites as well as applications provided by SAP partners and customers. Thus, SAP HANA morphed into an ecosystem with Cloud architectural patterns applied and able to satisfy a wide variety of requirements—most importantly and usually not explicitly addressed by the database research community—regarding coping with the complexity of the application scenarios—schema, data sets, users, workloads, deployments, security and auditing, etc.—by providing low TCO and not compromising overall query performance. Thus, SAP HANA Cloud …

- …provides reliable HTAP and multi-model query processing in "4+1" hyper-scaler independent multi-Cloud environments (currently operating on AWS, Azure, Google, Alibaba, and SAP Converged Cloud).
- …provides a data-mesh style access to different data sets leveraging locality-aware query processing.
- …architecturally de-couples petabyte-scale from gigabyte-scale query processing by offering query capability in-memory, on-disk, and on-Cloud.
- …tightly integrates into SAP's Business Technology Platform and application programming model for a one-stop shopping experience for application developer and modeler.
- …preserves the unprecedented performance for in-memory analytical query processing using the proven HANA database engine.
- …builds the strategic foundation for all SAP business applications.

Within this contribution, we conveyed not only the evolution towards SAP HANA Cloud, but also shared some demanding characteristics of typical SAP applications that push the boundaries of what is seen in traditional industry benchmark scenarios and which might spark research activities within the database community. Like any other software system, SAP HANA Cloud is and will be an ongoing endeavor. We share a snapshot of the current state with our community.

Acknowledgments

We are grateful to all colleagues in the HANA Cloud team for their hard work and their many contributions. We also thank our stakeholders and customers for their partnership in shaping and evolving HANA Cloud.

This work was partly funded by (1) the German Research Foundation (DFG) via a Reinhart Koselleck-Project (LE-1416/28-1), (2) the German Research Foundation (DFG) priority program SPP 2377 under grant no. LE 1416/30-1, and (3) the Horizon Europe research and innovation program under grant agreement no. 101189551 (CHORYS).

References

[1] [n. d.]. SAP Integrated Business Planning for Supply Chain. https://www.sap.com/products/scm/integrated-business-planning.html Accessed: 2024-12-05.

[2] Michael Armbrust, Tathagata Das, Sameer Paranjpye, Reynold Xin, Shixiong Zhu, Ali Ghodsi, Burak Yavuz, Mukul Murthy, Joseph Torres, Liwen Sun, Peter A. Boncz, Mostafa Mokhtar, Herman Van Hovell, Adrian Ionescu, Alicja Luszczak, Michal Switakowski, Takuya Ueshin, Xiao Li, Michal Szafranski, Pieter Senster, and Matei Zaharia. 2020. Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores. *Proc. VLDB Endow.* 13, 12 (2020), 3411–3424. doi:10.14778/3415478.3415560

[3] Nikos Armenatzoglou, Sanuj Basu, Naga Bhanoori, Mengchu Cai, Naresh Chainani, Kiran Chinta, Venkatraman Govindaraju, Todd J. Green, Monish Gupta, Sebastian Hillig, Eric Hotinger, Yan Leshinksy, Jintian Liang, Michael McCreedy, Fabian Nagel, Ippokratis Pandis, Panos Parchas, Rahul Pathak, Orestis Polychroniou, Foyzur Rahman, Gaurav Saxena, Gokul Soundararajan, Sriram Subramanian, and Doug Terry. 2022. Amazon Redshift Re-invented. In *SIGMOD 2022*. ACM, 2205–2217. doi:10.1145/3514221.3526045

[4] Thomas Bach, Artur Andrzejak, Changyun Seo, Christian Bierstedt, Christian Lemke, Daniel Ritter, Dong Won Hwang, Erda Sheshi, Felix Schabernack, Frank Renkes, Gordon Gaumnitz, Jakob Martens, Lars Hoemke, Michael Felderer, Michael Rudolf, Neetha Jambigi, Norman May, Robin Joy, Ruben Scheja, Sascha Schwedes, Sebastian Seibel, Sebastian Seifert, Stefan Haas, Stephan Kraft, Thomas Kroll, Tobias Scheuer, and Wolfgang Lehner. 2022. Testing Very Large Database Management Systems: The Case of SAP HANA. *Datenbank Spektrum* 22, 3 (2022), 195–215. doi:10.1007/s13222-022-00426-x

[5] Devraj Bardhan, Axel Baumgartl, Nga-Sze Choi, Mark Dudgeon, Piotr Górecki, Asidhara Lahiri, Bert Meijerink, and Andrew Worsley-Tonks. 2016. *SAP S/4HANA: An Introduction.* SAP Press.

[6] Stefano Belloni and Daniel Ritter. 2022. Benchmarking JSON Document Stores in Practice. *Datenbank-Spektrum* 22, 3 (2022), 217–226. doi:10.1007/S13222-022-00425-Y

[7] Stefano Belloni, Daniel Ritter, Marco Schröder, and Nils Rörup. 2022. DeepBench: Benchmarking JSON Document Stores. In *DBTest@SIGMOD '22: Proceedings of the 9th International Workshop of Testing Database Systems, Philadelphia, PA, USA, 17 June 2022,* Manuel Rigger and Pinar Tözün (Eds.). ACM, 1–9. doi:10.1145/3531348.3532176

[8] Leonid Boytsov, David Novak, Yury A. Malkov, and Eric Nyberg. 2016. Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search. In *CIKM 2016*. ACM, 1099–1108.

[9] Jochen Doppelhammer, Thomas Höppler, Alfons Kemper, and Donald Kossmann. 1997. Database Performance in the Real World - TPC-D and SAP R/3 (Experience Paper). In *SIGMOD 1997*. ACM Press, 123–134. doi:10.1145/253260.253280

[10] Franz Färber, Norman May, Wolfgang Lehner, Philipp Große, Ingo Müller, Hannes Rauhe, and Jonathan Dees. 2012. The SAP HANA Database–An Architecture Overview. *IEEE Data Eng. Bull.* 35, 1 (2012), 28–33.

[11] The Apache Software Foundation. 2018. Apache Spark - Unified Engine for large-scale data analytics. https://spark.apache.org/. Accessed: 2024-12-05.

[12] The Apache Software Foundation. 2024. Apache Iceberg. https://iceberg.apache.org/. Accessed: 2024-12-05.

[13] The Apache Software Foundation. 2024. Apache ORC - High-Performance Columnar Storage for Hadoop. https://orc.apache.org/. Accessed: 2024-12-05.

[14] Gardner. 2024. A Managed Kubernetes Service Done Right. https://gardener.cloud/. Accessed: 2024-12-05.

[15] Boris Gruschko, Kihong Kim, Hyunjun Kim, Taehyung Lee, Michael Mueller, and Daniel Ritter. 2025. Elastic Compute in SAP HANA Cloud by Example of SAP Integrated Business Planning. *Datenbank-Spektrum* (2025), 1–12.

[16] Gabriel Haas and Viktor Leis. 2023. What Modern NVMe Storage Can Do, And How To Exploit It: High-Performance I/O for High-Performance Storage Engines. *Proc. VLDB Endow.* 16, 9 (2023), 2090–2102. doi:10.14778/3598581.3598584

[17] Sasun Hambardzumyan. 2023. Deep Lake: A Lakehouse for Deep Learning. In *CIDR 2023*. https://www.cidrdb.org/cidr2023/papers/p69-buniatyan.pdf

[18] Gartner Inc. 2023. Gartner Says Cloud Will Become a Business Necessity by 2028. https://www.gartner.com/en/newsroom/press-releases/2023-11-29-gartner-says-cloud-will-become-a-business-necessity-by-2028. Accessed: 2024-12-05.

[19] Kihong Kim, Hyunwook Kim, Jin Su Lee, Taehyung Lee, Mihnea Andrei, Alexander Böhm, Ralf Dentzer, Heiko Gerwens, Irena Kofman, Norman May, Daniel Ritter, and Guido Moerkotte. 2025. Enterprise Application-Database Co-Innovation for Hybrid Transactional/Analytical Processing: A Virtual Data Model and ts Query Optimization Needs. In *Companion of the 2025 International Conference on Management of Data, SIGMOD/PODS 2025, Berlin, Germany, June 22-27, 2025*. ACM, to appear.

[20] Jens Krueger, Changkyu Kim, Martin Grund, Nadathur Satish, David Schwalb, Jatin Chhugani, Hasso Plattner, Pradeep Dubey, and Alexander Zeier. 2011. Fast updates on read-optimized databases using multi-core CPUs. *Proc. VLDB Endow.* 5, 1 (Sept. 2011), 61–72. doi:10.14778/2047485.2047491

[21] Maximilian Kuschewski, David Sauerwein, Adnan Alhomssi, and Viktor Leis. 2023. BtrBlocks: Efficient Columnar Compression for Data Lakes. *Proc. ACM Manag. Data* 1, 2 (2023), 118:1–118:26. doi:10.1145/3589263

[22] Juchang Lee, Yong Sik Kwon, Franz Färber, Michael Muehle, Chulwon Lee, Christian Bensberg, Joo Yeon Lee, Arthur H. Lee, and Wolfgang Lehner. 2013. SAP HANA distributed in-memory database system: Transaction, session, and metadata management. In *ICDE 2013*. 1165–1173. doi:10.1109/ICDE.2013.6544906

[23] Justin J. Levandoski, Garrett Casto, Mingge Deng, Rushabh Desai, Pavan Edara, Thibaud Hottelier, Amir Hormati, Anoop Johnson, Jeff Johnson, Dawid Kurzyniec, Sam McVeety, Prem Ramanathan, Gaurav Saxena, Vidya Shanmugan, and Yuri Volobuev. 2024. BigLake: BigQuery's Evolution toward a Multi-Cloud Lakehouse. In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024*, Pablo Barceló, Nayat Sánchez-Pi, Alexandra Meliou, and S. Sudarshan (Eds.). ACM, 334–346. doi:10.1145/3626246.3653388

[24] Guoliang Li, Xuanhe Zhou, and Xinyang Zhao. 2024. LLM for Data Management. *Proc. VLDB Endow.* 17, 12 (2024), 4213–4216. https://www.vldb.org/pvldb/vol17/p4213-li.pdf

[25] Norman May, Alexander Böhm, and Wolfgang Lehner. 2017. SAP HANA–The Evolution of an In-Memory DBMS from Pure OLAP Processing Towards Mixed Workloads. *Datenbanksysteme für Business, Technologie und Web (BTW 2017)* (2017).

[26] Norman May, Alexander Böhm, Meinolf Block, and Wolfgang Lehner. 2015. Managed Query Processing within the SAP HANA Database Platform. *Datenbank-Spektrum* 15, 2 (2015), 141–152.

[27] Norman May, Wolfgang Lehner, P Shahul Hameed, Nitesh Maheshwari, Carsten Müller, Sudipto Chowdhuri, and Anil K Goel. 2015. SAP HANA-From Relational OLAP Database to Big Data Infrastructure. In *EDBT*. 581–592.

[28] Stefan Noll, Norman May, Alexander Böhm, Jan Mühlig, and Jens Teubner. 2019. From the Application to the CPU: Holistic Resource Management for Modern Database Management Systems. *IEEE Data Eng. Bull.* 42, 1 (2019), 10–21. http://sites.computer.org/debull/A19mar/p10.pdf

[29] Apache Parquet. 2024. Parquet. https://parquet.apache.org/. Accessed: 2024-04-30.

[30] Hasso Plattner. 2014. The Impact of Columnar In-Memory Databases on Enterprise Systems. *Proc. VLDB Endow.* 7, 13 (2014), 1722–1729. doi:10.14778/2733004.2733074

[31] The Linux Foundation Projects. 2024. Delta Lake. https://delta.io/. Accessed: 2024-12-05.

[32] Iraklis Psaroudakis, Tobias Scheuer, Norman May, and Anastasia Ailamaki. 2013. Task Scheduling for Highly Concurrent Analytical and Transactional Main-Memory Workloads. In *ADMS 2013*. 36–45. http://www.adms-conf.org/2013/psaroudakis_adms13.pdf

[33] Iraklis Psaroudakis, Tobias Scheuer, Norman May, Abdelkader Sellami, and Anastasia Ailamaki. 2016. Adaptive NUMA-aware data placement and task scheduling for analytical workloads in main-memory column-stores. *Proc. VLDB Endow.* 10, 2 (2016), 37–48. doi:10.14778/3015274.3015275

[34] Thomas Saueressig, Jan Gilg, Uwe Grigoleit, Arpan Shah, Almer Podbicanin, and Marcus Homann. 2022. *SAP S/4HANA Cloud: An Introduction* (2 ed.). SAP Press.

[35] Amazon Web Services. 2020. Amazon Redshift Spectrum adds support for querying open source Apache Hudi and Delta Lake. https://aws.amazon.com/about-aws/whats-new/2020/09/amazon-redshift-spectrum-adds-support-for-querying-open-source-apache-hudi-and-delta-lake/. Accessed: 2024-12-05.

[36] Amazon Web Services. 2024. Cloud Object Storage - Amazon S3 - AWS. https://aws.amazon.com/s3/. Accessed: 2024-12-05.

[37] Raghav Sethi, Martin Traverso, Dain Sundstrom, David Phillips, Wenlei Xie, Yutian Sun, Nezih Yegitbasi, Haozhun Jin, Eric Hwang, Nileema Shingte, et al. 2019. Presto: Sql on everything. In *ICDE 2019*. IEEE, 1802–1813.

[38] Reza Sherkat, Colin Florendo, Mihnea Andrei, Rolando Blanco, Adrian Dragusanu, Amit Pathak, Pushkar Khadilkar, Neeraj Kulkarni, Christian Lemke, Sebastian Seifert, et al. 2019. Native store extension for SAP HANA. *Proceedings of the VLDB Endowment* 12, 12 (2019), 2047–2058.

[39] Michael Stonebraker and Ugur Cetintemel. 2005. "One size fits all": an idea whose time has come and gone. In *ICDE 2005*. 2–11. doi:10.1109/ICDE.2005.1

[40] Deepak Vohra. 2016. *Apache Parquet.* Apress, Berkeley, CA, 325–335. doi:10.1007/978-1-4842-2199-0_8

[41] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster Computing with Working Sets. In *HotCloud 2010*. USENIX Association. https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets

[42] Matei Zaharia, Ali Ghodsi, Reynold Xin, and Michael Armbrust. 2021. Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. In *CIDR 2021*. http://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf