# Towards a web-scale data management ecosystem demonstrated by SAP HANA (Hacia un ecosistema de gestión de datos a escala web demostrado por SAP HANA)

**Abstracto:**

Con los años, la gestión de datos se ha diversificado y se ha movido en múltiples direcciones, principalmente debido a un crecimiento significativo en el espacio de la aplicación con diferentes patrones de uso, un cambio masivo en las características subyacentes del hardware y, por último, pero no menos importante, el aumento de los volúmenes de datos. procesado. Una solución que coincida con estas restricciones tiene que hacer frente a un espacio de problemas multidimensional que incluye técnicas que tratan con una gran cantidad de tipos de datos específicos de dominio, modelos de datos y consistencia, escenarios de implementación e infraestructuras de procesamiento, almacenamiento y comunicación a nivel de hardware. Los motores de bases de datos especializados están disponibles y se posicionan en el mercado optimizando una dimensión particular, por un lado, mientras relajan otros aspectos (por ejemplo, implementación a escala web con consistencia relajada). Hoy es sentido común, que no hay un motor único que pueda manejar todas las diferentes dimensiones por igual y, por lo tanto, tenemos muy buenas razones para abordar este problema y optimizar las dimensiones con enfoques especializados en un primer paso. Sin embargo, abogamos por un segundo paso (reflexionando en nuestra opinión sobre el problema aún más difícil) de una integración profunda de motores individuales en un ecosistema de gestión de datos coherente y consistente que proporcione no solo componentes compartidos sino también una comprensión común de la semántica general del negocio . Más específicamente, un ecosistema de gestión de datos proporciona una "infraestructura" común para la gestión del ciclo de vida de software y datos, respaldo / recuperación, replicación y alta disponibilidad, contabilidad y monitoreo, y muchos otros temas operativos, donde los administradores y usuarios esperan una experiencia armonizada. Sin embargo, lo que es más importante desde la perspectiva de la aplicación, la experiencia del cliente nos enseña a proporcionar una visión empresarial coherente en todos los diferentes componentes y la capacidad de combinar a la perfección diferentes capacidades. Por ejemplo, dentro de los recientes escenarios de Internet de las cosas basados ​​en el cliente, existe un gran potencial en la combinación de gráficos ...

(Ver más)

**Publicado en:**[31a Conferencia Internacional IEEE 2015 sobre Ingeniería de Datos](https://ieeexplore.ieee.org/xpl/conhome/7109453/proceeding)

**Fecha de la conferencia:** 13-17 de abril de 2015

**Fecha de adición a IEEE *Xplore* :** 01 de junio de 2015

**ISBN electrónico:**978-1-4799-7964-6

**Información de ISSN:**

**Número de acceso de INSPEC:** 15180759

**DOI:**[10.1109 / ICDE.2015.7113374](https://doi.org/10.1109/ICDE.2015.7113374)

**Editorial:**IEEE

**Lugar de la conferencia:** Seúl, Corea del Sur

**SECCIÓN I.**

## **Introducción**

Los últimos años de investigación y desarrollo de bases de datos han estado dominados por algunas tendencias importantes, que cambiaron el enfoque de la comunidad académica, así como la atención de los actores económicos y las partes interesadas del mercado monetario hacia la gestión de datos. Temas como Main Memory Processing, Big Data analytics, No-SQL e Internet of Things (IoT), solo por nombrar a los representantes más destacados, provocaron una gran cantidad de actividades en un área dominada por los paradigmas de bases de datos tradicionales (datos relacionales modelo con consistencia ACID) durante décadas. Una gran variedad de aplicaciones desencadenó repensar todos los aspectos del problema de gestión de datos. Por ejemplo, las aplicaciones modernas requieren soporte especializado para:

* Diferentes tipos de datos: esto incluye todas las variaciones de conjuntos de datos estructurados a través de semiestructurados a desestructurados con soporte específico de la aplicación para el procesamiento de imagen y video, así como métodos integrales para el manejo eficiente de datos temporales y geoespaciales.
* Diferentes patrones de consumo: este requisito abarca desde OLTP tradicionales hasta consultas OLAP masivas sobre grandes conjuntos de datos, cada vez más como parte de iteraciones con tareas complejas de análisis de datos.
* Diferentes modelos de datos: las aplicaciones han establecido una multitud de modelos de datos como alternativas al modelo relacional clásico; Los ejemplos comprenden modelos de tabla amplia, centrados en gráficos o centrados en documentos, pero llegan hasta modelos tensoriales para aplicaciones científicas.
* Nociones de coherencia diferentes y controladas por la aplicación: una vez más, se requiere todo el espectro desde ACID hasta casi cero compatibilidad con la coherencia relacionada con DB de una cartera de aplicaciones moderna.
* Lenguaje de consulta y aplicación diferente: los lenguajes de consulta NoSQL permiten expresar tareas analíticas complejas de forma más natural; Además, la capa de administración de datos requiere cada vez más lenguajes específicos de dominio (DSL).
* Diferentes niveles de escalado: la elección de las opciones de implementación abarca desde escenarios de gran escalamiento a través de escalas modestas compatibles con centros de TI estándar hasta escenarios de escalamiento masivo en entornos de alojamiento.
* Diferentes capacidades de hardware: las tendencias de hardware tienen un tremendo impacto en las opciones de almacenamiento (memoria principal, memoria de memoria, disco o incluso en la memoria RAM no volátil), así como en la infraestructura de cómputo (CPU, GPU, FPGA) y de alta velocidad se interconecta con oportunidades como DMA remota.

### A. Reacciones de la comunidad DB

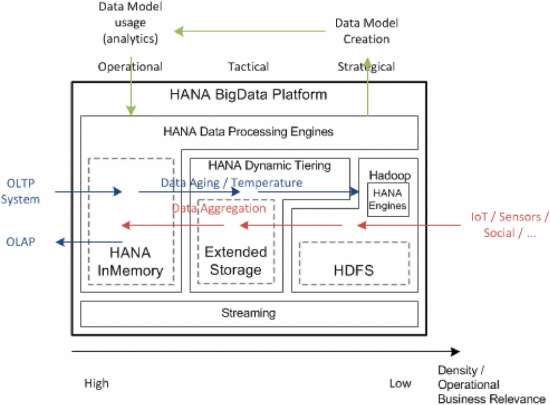
Con este nuevo interés y la posibilidad de proporcionar valor comercial para los clientes (menor costo total de propiedad y aumento de oportunidades de aplicación), una gran cantidad de nuevos motores de bases de datos ha evolucionado para enfrentar algunos de los desafíos que surgen. Especialmente los enfoques "NoSQL" intentan cuestionar las decisiones de diseño tradicionales en múltiples direcciones, por ejemplo, repensando los modelos de consistencia tradicionales, favoreciendo mecanismos de escalamiento extremo y proporcionando modelos de datos no relacionales como JSON o almacenes de valores clave puros. Representantes como MongoDB, CloudDB o Cassandra ya son bastante populares y atraen cada vez más la atención de los desarrolladores de aplicaciones que buscan flexibilidad más allá del modelo relacional tradicional. Procedente del otro extremo de un conjunto de datos sin esquema a priori, Los motores de gráficos se distinguen principalmente en el almacenamiento de elementos de datos individuales (entidades con tipos de entidad potencialmente individuales) como nodos en estructuras de gráficos con bordes para representar relaciones individuales entre las entidades. Con respecto a la consistencia y la escala, los motores de gráficos actuales como Neo4j, Allegrograph, etc. se comportan como sistemas relacionales estándar.

Recientemente, muchos proyectos en el ecosistema de Hadoop como Spark / Splash, Stinger, HBase, Hadapt, Hive y muchos otros recibieron mucha atención. Estos motores comparten el modelo de persistencia (HDFS), el entorno de tiempo de ejecución (MapReduce / TEZ), los servicios de gestión de recursos (Yarn) o combinaciones de los mismos. Como estos sistemas pueden cooperar e intercambiar parcialmente datos a nivel técnico, la aplicación debe proporcionar operaciones comunes de ejecución de consultas y semánticas, por ejemplo, proporcionando trabajos MapReduce específicos del escenario. Aunque existe una infraestructura técnica común, una infraestructura de procesamiento de consultas común (con componentes locales y globales) que permite una integración perfecta de lenguajes más allá de SQL, un repositorio común para conceptos empresariales de nivel superior (nuevamente con derivados locales),

### B. Ecosistema SAP HANA

Con SAP HANA damos los primeros pasos hacia nuestro objetivo de un ecosistema de gestión de datos integrado y desplegable dinámicamente. Nos esforzamos por combinar diferentes motores, modelos de datos y paradigmas de procesamiento de datos y ofrecer una solución para la aplicación que consiste lógicamente en un tiempo de ejecución de ejecución, una persistencia, una infraestructura y una experiencia de administración. Separamos cuidadosamente entre un nivel de modelo de almacenamiento físico y un nivel de modelo de datos lógico. Para el modelo de almacenamiento, confiamos en una interpretación flexible de los grupos de columnas como base para diferentes modelos de datos lógicos, como el modelo relacional estándar, modelos basados ​​en gráficos o tensoriales que pueden venir junto con lenguajes individuales que se asignan a un SQL común. lenguaje de consulta interno para la capa de almacenamiento.

La Figura 1 ilustra el impacto de nuevos dominios de aplicación como IoT además del procesamiento de datos tradicional de estilo OLTP / OLAP. Si bien SAP HANA está optimizado para ejecutar cargas de trabajo OLTP y OLAP a una velocidad extrema, los conjuntos de datos provenientes de sensores y otros dispositivos pueden cargarse inicialmente en entornos basados ​​en Hadoop y propagarse aún más dentro de un proceso de refinamiento de datos en las estructuras en memoria. Por el contrario, los datos transaccionales pueden envejecer y, según las reglas comerciales, trasladarse a un almacenamiento extendido y potencialmente a sistemas basados ​​en HDFS.



**Figura 1.** Posicionamiento del ecosistema de gestión de datos SAP HANA

### C. Esquema y objetivo

A continuación, primero daremos un resumen del estado de SAP HANA con respecto a las extensiones funcionales y la integración del contexto de la aplicación. Luego proporcionaremos una perspectiva detallada de la próxima extensión de escalamiento masivo, SAP HANA SOE (Extensión de escalamiento horizontal, nombre en clave "Velocity Engine") como otro motor especializado dentro de la familia SAP HANA. También describiremos la integración de Hadoop en el ecosistema de HANA. El documento se cerrará con una mirada en profundidad a diferentes escenarios del mundo real, donde una combinación eficiente de diferentes motores es crucial para el caso de uso del cliente.

El objetivo general del documento es transmitir el mensaje de que un sistema moderno de gestión de datos debe abordar al menos tres aspectos para tener éxito con respecto a los desafíos modernos de gestión de datos:

* Proporcionando una variedad de estructuras de datos especializadas y algoritmos integrados: Dentro de SAP HANA, proporcionamos una gran cantidad de tipos de datos no tradicionales con algoritmos altamente ajustados. La cartera abarca desde análisis de texto semántico hasta cálculos científicos sobre matrices, como se describe en la Sección II.
* Escuchar la aplicación: la capa de gestión de datos puede beneficiarse enormemente del conocimiento de la aplicación para mejorar significativamente el rendimiento. Proporcionaremos algunos ejemplos en la Sección III.
* Orquestando motores de datos altamente especializados que se ejecutan en diferentes entornos que van desde dispositivos de nube de borde (por ejemplo, dispositivos móviles, estaciones base de red celular hasta grandes instalaciones basadas en Hadoop), consulte la Sección III.

**SECCION II.**

## **SAP HANA: más allá del procesamiento de datos relacionales**

Como motivado, un ecosistema moderno de gestión de datos a escala web se basa en dos pilares. Primero, el ecosistema tiene que proporcionar un amplio conjunto de funcionalidades y una estrecha integración del conocimiento de la aplicación: el modelo relacional puro con SQL simple, ya que el lenguaje de consulta ya no satisface la gran variedad de diferentes casos de uso y requisitos resultantes. En segundo lugar, el ecosistema requiere una implementación escalable y de alto rendimiento de esta funcionalidad para ser útil en escenarios de Big Data. Antes de sumergirnos en algunos detalles arquitectónicos de SAP HANA SOE, enumeramos extensiones funcionales específicas que ya existen en el sistema SAP HANA como la base para una pila de software eficiente para aplicaciones comerciales.

En 2010, SAP anunció el almacén de columnas en memoria como la base para la gestión de datos de SAP y el desarrollo futuro de aplicaciones. SAP HANA es un sistema de base de datos relacional totalmente compatible con ACID con todas las capacidades de vanguardia como copia de seguridad, recuperación y mecanismos de HA [1] , [2] . La Figura 2 muestra algunos componentes centrales del sistema HANA. Como se puede ver, el sistema comprende múltiples motores de procesamiento de datos para refinar los flujos de datos provenientes de diferentes motores de ejecución "en bruto". La sección resaltará algunos de los motores de procesamiento no estándar como texto, gráfico o capacidades para procesar datos de series temporales de manera eficiente. Figura 2 también da una impresión de los motores de ejecución actuales que van desde los motores basados ​​en memoria principal a través del motor de almacenamiento extendido (basado en tecnología IQ) hasta un marco de federación integral (SDA = acceso inteligente a datos) para llegar a una gran variedad de fuentes de datos externas.

**Figura 2.** Sistema SAP HANA

### A. Almacén de columna de memoria principal: unir OLAP y OLTP

Al cargar los datos completamente en la memoria principal, aplicar múltiples técnicas de compresión y optimizar las arquitecturas multinúcleo, el sistema HANA logra el acceso a los datos y las características de rendimiento, lo que permite recombinar las cargas de trabajo OLTP y OLAP en un solo sistema. Contrariamente a los métodos vigentes, el almacén de columnas de memoria principal también se utiliza para grandes cargas transaccionales, lo que ha demostrado ser lo suficientemente rápido en miles de instalaciones de clientes. Hasso Plattner y otros ya han descrito en [3]que las operaciones de lectura dominan masivamente las operaciones de escritura en los sistemas empresariales empresariales, lo que se vuelve aún más evidente si se eliminan las tablas redundantes del sistema. La combinación de ambas cargas de trabajo en un sistema permite evitar los costosos costos de replicación entre los sistemas OLTP y OLAP y proporciona acceso a todas las preguntas analíticas en tiempo real.

### B. Minería de datos: próximo nivel de análisis

Incorporamos algunas características críticas de minería de datos directamente en el motor de almacenamiento de columnas. Algunos ejemplos son el análisis de cestas distribuidas y una variedad de algoritmos de pronóstico. Además, el acceso al sistema R y otros proveedores de minería de datos como SAS son aplicables directamente fuera del sistema para aprovechar la gran cantidad de algoritmos y métodos que ya existen en la comunidad. El acceso a R se implementa como un operador especial en el gráfico de flujo de datos interno del motor de la base de datos, lo que permite que el optimizador acepte la llamada al sistema externo. Se integra más funcionalidad como llamadas de procedimiento almacenado en el modelo SQL nativo; Por lo tanto, podemos combinarlo a la perfección con cualquier otro tipo de acceso SQL.

### C. Motor de texto: unir datos estructurados y no estructurados

El análisis de datos textuales tiene muchas facetas en una solución de gestión de datos. Primero, pensamos en la búsqueda de texto simple que todos conocemos de los motores de búsqueda web. Para brindar soporte a los clientes de todo el mundo, muchos idiomas deben ser compatibles de forma nativa con funcionalidades como derivación, parte del etiquetado de voz y otros. Además de eso, se pueden admitir capacidades analíticas y de minería de texto sofisticadas como clasificación de texto, agrupamiento, análisis de sentimientos y otras operaciones de minería de texto. Además, podemos extraer entidades (como nombres, direcciones, empresas, ...) y sentimientos de documentos con un enfoque basado en reglas además de la funcionalidad del lenguaje natural. Las entidades y los sentimientos extraídos se pueden almacenar como datos estructurados. Dado que el procesamiento de texto está profundamente integrado en el motor HANA, Los resultados del análisis de texto ahora se pueden combinar con datos estructurados ya almacenados en la parte relacional de la base de datos. Esta integración cierra la brecha entre los diferentes métodos de análisis de información estructurada y no estructurada. El proceso de análisis de texto y extracción de características se activa automáticamente cuando se introducen documentos nuevos o modificados en el sistema de gestión de datos.

### D. Extensiones de planificación: exitosas y sin embargo pasadas por alto

La funcionalidad de planificación y los operadores se pasan por alto en la comunidad de investigación. Las aplicaciones de planificación para la planificación de ventas, planificación financiera o planificación de operaciones son muy atractivas para los clientes como, por ejemplo, el éxito comercial de compañías como Anaplan 1 u otras. La planificación, en esta definición, es el proceso de definir y alinear los números de ventas, financieros u otros para el futuro previsible (normalmente por un período de 1 año). El proceso de planificación requiere una gran funcionalidad de la base de datos basada en la CPU, como procesos de desagregación o copia, que proporcionan instantáneas lógicas o versiones y otros operadores. Hemos integrado estos operadores directamente en el motor relacional y accedemos a ellos con extensiones SQL.

### E. Graph/hierarchy Engine: the Embedded Graph Story

Many business applications like transport and logistics, supply chain traceability, business networks, fraud detection, and others work inherently with graph structures and benefit directly from explicit graph representations inside the data management layer. As Paradies et al. have explained in [4], explicit graph structures help applications to express complex business logic more explicitly and execute the operations more effectively. As Finis et al. have discussed in [5] hierarchies as a special kind of a graph are used in almost all kinds of business applications. Special support for time dependent and versioned hierarchies is therefore a crucial functionality in the database layer for business applications. As the graphs and hierarchies are not isolated objects in the application context, graph data has to be combined during modeling but especially during query execution with other types of data, especially the structured relational data, but also geospatial and other data types. Therefore we have decided to embed a graph engine into the HANA data management leveraging the main memory column store. This allows to interpret data in columns (structured relational data) as graph or hierarchy structures by defining hierarchy or graph views on top of the relational data. Within these graph structures, state of the art graph processing functionality (like distance, siblings, shortest path, and others) is provided. We are currently defining extensions to SQL in order to integrate most of the operations into the standard processing model as well as providing a domain-specific language to fully exploit the graph data model without the constraints imposed by the relational representation.

### F. GIS, Time Series: Bringing More Semantics to the Data

Geospatial and time series are supported as native data types inside the relational engine. They are built with mechanisms of the main memory column store and implement powerful compression mechanisms, which is especially useful for sensor data. As operations on both data types provide results in form of tuples, implementing them as data types deep in the engine is a natural fit. We extended the SQL syntax in order to allow the definition of points or polygons, and to support query operators like WithinDistance, Contains or Area. Combination with relational data is obvious in areas like geo-location based analytics, with queries like, e.g., “get the areas in US with the most revenue” or “get all customers within a distance of 10 kilometer having payments due”. But also a combination with graph data for example in routing algorithms is a common use case. Time series data types are mainly interesting for the financial sector or Internet of Things for storing sensor data. Besides providing large compression factors, they provide functionality like resolution adoption, comparison functions, correlation, transformations, and others. Again, integration into the relational world is the key, because the queries always combine the selection of elected sensor data with time series functionality.

### G. Scientific Engine: Use the Full Table Scan Speed for Heavy Computation

In [6], Kernert et al. show the significant advantage of bringing linear algebra operations like eigenvalue calculation on large matrices into a main memory column store. Beside the scientific area, many applications in the financial area and business warehouse optimization benefit from having these kinds of functionality inside the data management layer. Since decision-support data is persisted and kept consistently within the data management ecosystem, no redundant copying from other data sources to external libraries is needed, reducing the pain to control consistency and define and monitor data movement processes. Furthermore, the corresponding metadata of data sets can be updated synchronously and consistently with the numerical data. Even more interestingly for database management research, large matrices are no static objects in common analytic workflows. As they are manipulated in an iterative process, the data manipulation capabilities of a database will meet the analytical demands better than the tedious maintaining of multiple data files and therefore significantly improve the efficiency of the domain expert by relieving it from organizing large file repositories.

### H. NoSQL Extension: Bringing Flexibility into the Relational World

El "mundo NoSQL" a menudo se define por tres criterios principales: estructuras de datos flexibles como el modelo de documento o almacenes de valores clave, escalamiento masivo y fácil y la libertad y la opción de relajar la consistencia. Los tres temas también deben integrarse en un sistema estándar de gestión de datos. En HANA, las llamadas "tablas flexibles" se pueden definir donde la definición de columna no es un DDL, sino que se activa implícitamente a través de una operación DML (similar al paradigma de almacenamiento de documentos o valores clave). Se puede consultar una tabla flexible a través de SQL, como sería el caso si fueran columnas estándar. Técnicamente, los metadatos sobre columnas desconocidas se crean automáticamente tan pronto como se insertan los registros con valores para las nuevas columnas. El número de columnas no está limitado (técnicamente hay un límite de 32 bits para expresar el número de columnas). Los métodos de compresión interna también pueden manejar columnas muy dispersas para lograr tasas de compresión. Además, estamos integrando un modelo de datos basado en documentos (similar al formato JSON) en HANA mediante la introducción de un "documento" del tipo de datos para una columna de una tabla relacional. El contenido (el documento) está estructurado en un formato JSON arbitrario. Los propios documentos son consultados por un lenguaje similar a XQuery que está incrustado en la instrucción SQL. El resultado de una consulta de "documento" es un conjunto de filas de la tabla que contiene el documento como una celda. Obviamente hay múltiples opciones, cómo usar este modelo. Por ejemplo, los usuarios pueden simplemente almacenar sus documentos JSON con la clave del documento como columnas estándar en la "tabla principal". Las claves de combinación adicionales también se pueden almacenar en la tabla principal.1 : Ncardinalidad entre encabezado-elemento y elemento-subpunto. Si además podemos garantizar desde la perspectiva de la aplicación, las entradas correspondientes de la tabla siempre se insertan / actualizan / eliminan juntas y las solicitudes de recuperación están leyendo principalmente el "objeto" completo (que refleja el procedimiento estándar para muchos objetos en software empresarial como SAP ERP), como un "objeto" podría almacenarse en un documento JSON como una especie de índice materializado sobre los datos relacionales. Se modelará como un tipo especial de índice de unión en la base de datos y será explotado de forma transparente por el proceso de recuperación. Con tales mecanismos no estándar, incluso las aplicaciones existentes y los modelos de datos pueden beneficiarse del almacenamiento de documentos.

### I. Escalar: reconciliar el Hadoop y el mundo relacional

En los últimos años hubo una clara tendencia a escalar utilizando clústeres de hardware básico especialmente para las infraestructuras de Hadoop. Pero también en otros entornos de carga de trabajo analítico, la tendencia hacia la ampliación frente a la ampliación mediante la adición de más recursos a un solo servidor es obvia. Investigación muy reciente [7]afirma que la mayoría de los volúmenes de datos que vemos en los clientes podrían manejarse con uno o solo unos pocos servidores más grandes, lo que proporciona beneficios en el rendimiento del costo total de propiedad y la densidad del servidor. Hasta ahora, nos centramos con SAP HANA principalmente en casos de uso con hasta unos pocos cientos de Terabytes de datos que pueden ser manejados por un número modesto de servidores de alta gama. Sugerimos incluso a nuestros principales clientes un modelo donde los datos activos se almacenarán en un único servidor de base de datos y los datos más fríos se distribuirán en un conjunto de nodos más baratos. Con la tendencia de Big Data, especialmente en relación con la gran cantidad de datos generados por los sensores, vemos una gran demanda de almacenamiento y análisis de datos en el rango alto de Petabyte. Además de almacenar grandes conjuntos de datos recopilados de algunos dispositivos, surge la necesidad de integrar dichos conjuntos de datos con bases de datos existentes como datos ERP, datos maestros, datos logísticos y para ejecutar algoritmos sobre dichos conjuntos de datos integrados. Por ejemplo, un productor de jabón para baños quiere comenzar una planificación de ruta para que su equipo de servicio llene los dispensadores, una vez que los sensores han detectado, que algunos están casi vacíos. Incluso quieren llenarlos antes, si han notado que se llevará a cabo un evento importante en los lugares donde han instalado dispensadores. Además, la compañía quiere dirigir al equipo de servicio en la dirección correcta. Para esos requisitos, una combinación de logística, datos maestros y datos de planificación es la clave. Por lo tanto, actualmente estamos ampliando el sistema SAP HANA con una opción de escalamiento masivo y una profunda integración con el sistema Hadoop. No discutimos llevar nuestras aplicaciones estándar a un sistema de gestión de datos de escala masiva. Todavía creemos en las ventajas de una configuración pequeña. Pero con los nuevos requisitos en diferentes áreas y la necesidad de una integración profunda, sostenemos que necesitamos ambos profundamente integrados. EnEn la Sección IV describimos las unidades de procesamiento único de baja huella, la infraestructura de escalamiento horizontal y la integración de Hadoop de nuestra solución.

**SECCION III.**

## **Cerrar la brecha entre las aplicaciones y la gestión de datos**

Con más de 100.000 clientes para sus diferentes soluciones comerciales (ERP, Analytics, HCM, redes y otros), SAP ha adquirido mucha experiencia en el desarrollo de aplicaciones. Casi todas las aplicaciones se crean sobre un motor de base de datos relacional. Se ha creado una gran variedad de herramientas, bibliotecas y servicios en la capa de aplicación para apoyar la eficiencia de los desarrolladores de aplicaciones. Muchas de estas herramientas reutilizan componentes que pertenecen lógicamente a la capa de datos, pero se implementan en la capa de aplicación para lograr la independencia de la base de datos subyacente. El problema de este enfoque se manifiesta en la realización de muchos cálculos centrados en datos dentro de la capa de aplicación y no dentro de la capa de base de datos lo más cerca posible de los conjuntos de datos originales. Esto implica que se debe transferir una cantidad significativa de datos de la base de datos a la esfera de la aplicación, lo que generalmente tiene un impacto severo en el rendimiento. Los dos ejemplos siguientes demuestran este efecto: si la conversión de moneda2se implementa a nivel de aplicación, las consultas analíticas han incluido el campo de moneda en la lista "agrupar por" para recuperar la información de la moneda. Dependiendo de la distribución de divisas, esto puede multiplicar los datos que se transferirán entre las capas. Un problema aún más difícil consiste en tratar con estructuras de datos jerárquicas. Dado que muchos sistemas de bases de datos no brindan soporte básico para las jerarquías, se implementan algoritmos complejos de resolución de jerarquía en las capas de aplicación. Por ejemplo, para contar los nodos secundarios transitivos de un nodo dado, todo el subárbol de la jerarquía respectiva debe moverse de la base de datos a la aplicación. Con la funcionalidad de jerarquía adecuada en el sistema de base de datos, solo se necesita comunicar el número de nodos a la aplicación que realiza la llamada. Otros ejemplos, entre otros, son la conversión de unidades, Soporte de calendario de fabricación, planificación financiera y optimizaciones en combinación con simulaciones completas. Con HANA, comenzamos a introducir sistemáticamente la funcionalidad en la base de datos y a construir bibliotecas / extensiones específicas de aplicaciones comerciales en la capa de base de datos con un impacto positivo significativo para el tiempo de ejecución. Argumentamos que este tipo de extensiones también son necesarias para otros tipos de aplicaciones.

Otras optimizaciones de la interfaz de la aplicación / base de datos no están agregando funcionalidad novedosa, pero permiten un mejor uso del conocimiento de la aplicación en la capa de la base de datos. Un ejemplo bastante simple es el mantenimiento de diccionarios de columnas de tabla. La codificación de dominio o diccionario es una técnica común para comprimir las columnas: para una columna de cadena, todos los valores distintos se insertan en un diccionario ordenado y la columna misma solo almacena las referencias al diccionario. Uno de los principales desafíos en este contexto es mantener eficientemente el diccionario ordenado en el caso de inserciones y eliminaciones. En HANA, esto se garantiza al tener una estructura de búfer llamada delta store que registra todos los cambios. Una fase de fusión incorpora estos conjuntos de datos en la columna principal. Para mantener la clasificación del diccionario dentro de este proceso de fusión,[8] . En el nivel de la aplicación, se crean tablas grandes con columnas de clave obviamente de alta cardinalidad a partir de una gran cantidad de datos transaccionales, que se crean principalmente fuera del sistema. Muy a menudo, las claves se generan mediante la concatenación de información del contexto de la aplicación más un contador incremental para lograr la unicidad. Al conocer el mecanismo de cómo se generan las claves, el mantenimiento y la fusión del diccionario se pueden hacer de manera mucho más simple y eficiente. Al incorporar el conocimiento de la aplicación, se puede lograr un orden de clasificación estable sin recurrir en algunas situaciones, mejorando el proceso de fusión.

Un tema de integración aún más importante es el envejecimiento de datos: existen múltiples formas de implementar el envejecimiento de datos, que se basan exclusivamente en estadísticas de bases de datos. Al permitir que la aplicación defina las reglas de antigüedad y las almacene en los metadatos de la base de datos, el mecanismo de antigüedad adquiere un significado semántico que permite una poda de partición mucho mejor que cualquier enfoque basado únicamente en estadísticas de acceso. Se pueden utilizar métodos estadísticos para proponer nuevas reglas de aplicación al desarrollador de la aplicación o al administrador del sistema. Considere un ejemplo con pedidos y facturas, dos objetos independientes en una aplicación comercial. La regla de vencimiento del pedido de ventas podría definirse como "antigüedad de un pedido de ventas, si está cerrado y la fecha de cierre es anterior a 3 meses y el pedido de venta no es de este año". Para las facturas, la regla de antigüedad podría definirse como "si la factura se paga por al menos 3 meses y la factura no es de este año". Cuando se solicitan todas las facturas abiertas de los últimos 3 meses, está claro que las particiones más antiguas (ya antiguas) del objeto de la factura se pueden podar. Un caso más interesante es, si alguien quiere ver todas las órdenes abiertas y las facturas correspondientes, una consulta estándar en este contexto. Dadas las reglas de antigüedad independientes de los objetos, la unión debe calcularse entre los pedidos "no envejecidos" y la tabla de factura completa. Al analizar las reglas de antigüedad, la consulta podría dividirse en dos partes, una especifica que todos los pedidos de venta abiertos del año actual deben unirse a la partición de factura "no envejecida", la segunda solicita que se abran los pedidos de años anteriores debe unirse con la tabla de objetos de factura completa. Suponiendo que, con suerte, no haya órdenes abiertas del año anterior, esa optimización podría ser beneficiosa. Podríamos extender las reglas de envejecimiento dadas por ". y si el pedido de cliente correspondiente está en el conjunto de antigüedad ". Esta regla exigiría que una factura solo pueda ser vencida, si la orden de venta correspondiente también está vencida. La aplicación de esta regla adicional significaría que la unión solo se puede ejecutar en las particiones "no antiguas". Lógicamente, se crea un gráfico de dependencia para el envejecimiento. Debe haber reglas y comprobaciones que garanticen que, a pesar de las dependencias, muchos objetos puedan envejecer y que no haya un ciclo en el gráfico de dependencia. Esta regla exigiría que una factura solo pueda ser vencida, si la orden de venta correspondiente también está vencida. La aplicación de esta regla adicional significaría que la unión solo se puede ejecutar en las particiones "no antiguas". Lógicamente, se crea un gráfico de dependencia para el envejecimiento. Debe haber reglas y comprobaciones que garanticen que, a pesar de las dependencias, muchos objetos puedan envejecer y que no haya un ciclo en el gráfico de dependencia. Esta regla exigiría que una factura solo pueda ser vencida, si la orden de venta correspondiente también está vencida. La aplicación de esta regla adicional significaría que la unión solo se puede ejecutar en las particiones "no antiguas". Lógicamente, se crea un gráfico de dependencia para el envejecimiento. Debe haber reglas y comprobaciones que garanticen que, a pesar de las dependencias, muchos objetos puedan envejecer y que no haya un ciclo en el gráfico de dependencia.

**SECTION IV.**

## **SAP HANA Scale-Out Option**

In addition to the core HANA system, we outline the SAP HANA Scale-Out Extension. The core idea of SAP HANA SOE is to extend the “big” Scale-up oriented SAP HANA system with massive scale-out capabilities. In this section, we briefly sketch the general approach, the scale-out architecture, and integration into Hadoop-based data management infrastructures.

### A. Goals and Architectural Principles

En los próximos años, veremos mejoras de hardware que cambiarán enormemente la forma en que se construyen y usan los sistemas de bases de datos. La memoria transaccional de hardware, que obtuvo y obtendrá disponibilidad global con la integración de Intel en sus sistemas Haswell, ayuda a desarrollar algoritmos escalables y estructuras de datos. En particular, Neumann et al. [9] han demostrado que los sistemas transaccionales pueden beneficiarse significativamente en la ejecución de transacciones de bases de datos globales al dividirlas en múltiples transacciones de hardware y eliminar bloqueos explícitos. Con la próxima memoria no volátil, hay muchas opciones nuevas para optimizar el diseño de los sistemas de gestión de datos. Oukid y col. mostrado en [10]cómo se puede acelerar la recuperación de una base de datos mediante un diseño cuidadoso de las estructuras de datos subyacentes y un diseño optimizado de rehacer / deshacer registro. Con los nuevos tejidos de red, esperamos una mejora significativa de la latencia de la red y el ancho de banda, lo que permite un nuevo diseño de software de sistemas de escalamiento horizontal. Otra tendencia es el cambio a dispositivos móviles con nuevos procesadores energéticamente eficientes, que crean una inmensa red de potencia de cálculo. Esta red se hará aún más grande con los miles de millones de sensores y colectores que se esperan en el futuro cercano. Todos estos cambios crean nuevas oportunidades y desafíos para los sistemas de gestión de datos, que son de escala masiva, tienen servidores únicos pequeños y flexibles, proporcionan una ejecución eficiente en energía,

En este contexto, desarrollamos una extensión de HANA con poco espacio físico, rendimiento extremo y diseñada para una alta escalabilidad (en el rango de miles de nodos). SAP HANA SOE se adhiere al paradigma del modelo de almacenamiento HANA que se basa en un almacén de columnas de memoria principal. Algunos de los requisitos de compresión se relajan para permitir un cálculo más eficiente de la energía. Esto es, por ejemplo, cierto para comprimir las referencias y para recurrir a las tablas durante la fusión. Además, durante el tiempo de ejecución, el motor compila la instrucción SQL en código C y la traduce a un formato binario ejecutable. Como Dees y Sanders describieron en [11], existen ventajas de rendimiento significativas con este enfoque. El marco de compilación LLVM con Clang realiza la compilación del código C en código nativo. Se sigue un enfoque similar en[12] , sin embargo, generamos código C en lugar de código de bytes LLVM para admitir funcionalidades de depuración y mantenimiento más sofisticadas. En el nivel del operador, utilizamos muchos algoritmos y funciones que ya están optimizados en el motor de back-end de HANA. En la versión actual, el motor está muy optimizado para solicitudes de lectura con un uso intensivo de índices de unión. Además, generamos planes para un paisaje distribuido. Estos planes pueden conducir a fuertes resultados de aceleración en comparación con la ejecución de una sola máquina como se muestra en [13]si los planes están diseñados específicamente para una ejecución agrupada en combinación con algoritmos de comunicación eficientes. A finales de 2014 se entregó una primera versión de SAP HANA SOE a los clientes como parte de la solución analítica de SAP (Lumira Desktop y Lumira Teamserver). Como siguiente paso, el motor obtendrá las capacidades descritas anteriormente, como GIS, búsqueda de texto, soporte de gráficos, etc., lo que permite utilizar estas funcionalidades en un paisaje de escalamiento extremo.

### B. Componentes del sistema y relaciones

Nuestro mecanismo de transacción distribuida favorece la disponibilidad a expensas de la coherencia (por ejemplo, el teorema CAP [14] ). Como ya se describió, SAP HANA SOE está compuesto por un conjunto de servicios, empaquetados en ejecutables y desplegados en un grupo de nodos. La Figura 3 muestra los diferentes componentes del sistema, así como su relación con cierto detalle. En el núcleo se encuentra el ejecutable de procesamiento de consultas locales SAP HANA SOE (v2lqp) que contiene una consulta y un servicio de datos. El servicio de consulta es un contenedor alrededor del motor central y funciona en particiones de tabla horizontales que se crean durante la importación de datos. Estas particiones preempaquetadas permiten una distribución rápida de los datos al escalar o recuperar datos. Un servicio de datos se encarga de recuperar y almacenar las particiones de tabla horizontales.

The execution of distributed queries is controlled by a distributed query coordinator service (v2dqp) which translates each query to a directed acyclic graph of tasks. The tasks are being sent to the query service instances where they are compiled and executed. A transaction broker service executes, serializes, and persists transactions to a distributed shared log (v2transact). Similar to the Corfu approach described in [15], the log stores all changes in a transactional consistent way. We introduced several changes and optimizations to the original proposal including information on how we distribute the log across multiple locations and how we update the different database nodes.

The distributed log is initially designed to work on top of Non-Volatile Memory, but multiple implementation variants will be provided (also on top of HDFS). With the distributed log approach we decouple the transaction mechanism from the query processing which allows easier scale and provides a clear system design. Updates in the log are incrementally communicated to the different data services. If the resulting isolation level is not sufficient for a given query, the distributed query coordinator service can ask the transaction broker service directly for additional updates to be considered. Therefore we are able to achieve different transactional behaviors by distinguishing two types of database nodes. On the one hand, an OLAP node updates itself in a transactionally consistent way but not necessarily synchronously to the update request coming from the application. The updates can either be incorporated by regularly polling the log or by retrieving the latest snapshot of the data hosted by a particular node. On the other hand, OLTP nodes allow real time transactional update of the data by incorporating the log during the update transaction.

Utilizamos diferentes implementaciones de MVCC para optimizar múltiples cargas de trabajo. Un servicio coordinador acepta consultas de un usuario o aplicación y coordina el procesamiento de consultas distribuidas. El administrador de clúster proporciona servicios de clúster como estadísticas, supervisión y orquestación. El servicio de descubrimiento realiza un seguimiento de todos los componentes disponibles en el paisaje. Todos estos servicios (incluidos los bloqueadores de transacciones globales y los servicios de bases de datos) pueden aislarse mediante una infraestructura de contenedor como Docker 3. Como parte del sistema HANA, la extensión escalable admite particiones horizontales multinivel (rango y hash), con la capacidad de manejar una gran cantidad de particiones. La alta disponibilidad se logra al admitir múltiples réplicas con el mecanismo de replicación de registros descrito anteriormente. Al definir nodos de base de datos OLAP u OLTP, se admiten diferentes niveles de coherencia.

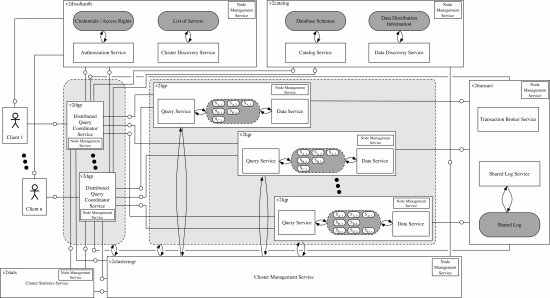
La supervisión y configuración general del clúster se realiza mediante un servicio de administración de clúster (v2clustermgr). Este servicio puede iniciar y detener dinámicamente otros servicios de procesamiento de consultas, así como orquestar el movimiento de datos. Puede acceder a información estadística sobre el uso actual del clúster para identificar puntos de acceso o monitorear los objetivos de rendimiento. Una autorización y un servicio de descubrimiento de clúster se agrupan (v2disc y auth) para almacenar los derechos de acceso al clúster y realizar un seguimiento de la disponibilidad de los servicios en todo el clúster. Un servicio de catálogo almacena y proporciona información de esquema y metadatos, un servicio de descubrimiento de datos realiza un seguimiento de la ubicación de las particiones de tabla horizontal correspondientes (v2catalog).

### C. Integración de Hadoop

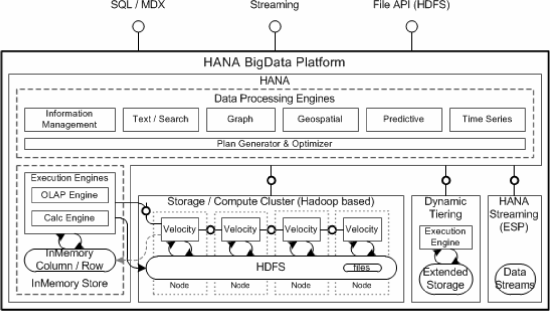
Con la creciente importancia del ecosistema Hadoop, las bases de datos y otros motores deben proporcionar versiones que se ejecuten de forma nativa dentro del ecosistema Hadoop [16]. Con la combinación de backend / escalamiento horizontal de HANA, actualmente estamos construyendo una integración profunda de Hadoop, que permite leer datos de Hadoop con mecanismos estándar de HANA SQL y Hadoop y combinarlos con datos en HANA como datos SAP ERP. La forma más simple de integración es un enfoque federado que empuja las sentencias SQL de HANA a Hive o marcos similares. Las consultas sobre los datos HDFS se ejecutan en Hadoop y los resultados se combinan en la capa HANA. Si bien esta combinación ya se entrega con SAP HANA, la opción de escalamiento horizontal proporciona una integración significativamente más profunda en la pila general del sistema de base de datos. Primero, permitimos instalar SAP HANA SOE de baja huella en cada nodo de Hadoop. Como estándar, proporcionamos un conector basado en archivos para SAP HANA SOE, que permite combinar el procesamiento de datos de SAP HANA SOE con trabajos estándar de MapReduce.La Figura 4 muestra la imagen más grande del ecosistema integrado de gestión de datos de SAP HANA que comprende el sistema en memoria HANA, el SOE de SAP HANA (que se muestra como nombre de código "Velocity") que se ejecuta dentro de la pila YARN, así como la transmisión (ESP) y la nivelación dinámica para frío datos. Con el fin de desempeñar también el papel de un centro de integración central para otras fuentes de datos, SDA ("Smart Data Access") permite la federación a una gran variedad de fuentes de datos diferentes.

Con este enfoque, se puede acceder a cualquier tipo de datos de SAP a través de los marcos de estilo SQL o MapReduce. Además, los datos de los nodos HDFS locales se pueden cargar en los nodos locales de SAP HANA SOE. Las reglas se pueden definir qué datos se deben cargar y cómo se configura el mecanismo de actualización. Hay tres formas de integrar estos datos: Primero, como ya se describió, los datos se pueden cargar a través de un lector de archivos estándar. En segundo lugar, la integración se realiza en el marco de Spark como objetos RDD mediante el uso de SAP HANA SOE para operaciones relevantes como unión, filtros, agregación, etc. Al envolver SAP HANA SOE en objetos RDD, los clientes aún pueden usar todas las funcionalidades de Spark, como Spark SQL o MLib. Por lo tanto, los clientes pueden combinar las capacidades del marco de Spark con las capacidades de velocidad e integración de datos de HANA. La tercera integración se planifica utilizando las funciones de escalado horizontal de la base de datos de la opción de escalado horizontal de HANA. Las declaraciones SQL distribuidas se pueden enviar a HANA, que crea un único plan de ejecución para consultar datos HDFS y SAP HANA en un mecanismo de ejecución de base de datos que incluye un optimizador.

Con SAP HANA SOE, todas las capacidades descritas anteriormente se pueden usar en datos HDFS y se pueden combinar de manera flexible. Los datos se pueden almacenar en HDFS de tres maneras diferentes: la primera es mediante mecanismos HDFS estándar como se usa hoy en día. En segundo lugar, HDFS se utiliza como un almacén antiguo para HANA, donde los datos antiguos (para envejecimiento, véase III) se almacenan en un mecanismo de almacenamiento barato. El tercero es que HDFS se usa como una persistencia de registro, se almacena como un registro para SAP HANA SOE. Como se describió anteriormente, implementamos una versión del registro distribuido en la parte superior de HDFS. La API de SAP HANA SOE, la API de registro distribuido o el lector de archivos HDFS pueden consumir los datos de este registro.

[[](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/7109453/7113253/7113374/7113374-fig-3-source-large.gif)](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/7109453/7113253/7113374/7113374-fig-3-source-large.gif)

**Figura 3.** Componentes de Architectur de SAP HANA SOE

[[](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/7109453/7113253/7113374/7113374-fig-4-source-large.gif)](https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/7109453/7113253/7113374/7113374-fig-4-source-large.gif)

**Figura 4.** Ecosistema SAP HANA

**SECCION V.**

## **Escenarios para un ecosistema de gestión de datos**

Para completar el panorama general del ecosistema de gestión de datos de SAP HANA previsto, describimos algunos casos y escenarios de uso del mundo real para motivar nuestras declaraciones y decisiones arquitectónicas anteriores. En pocas palabras, vemos una clara demanda de infraestructuras de gestión de datos polifónicos. Cada vez más, los paisajes de gestión de datos comprenderán múltiples sistemas con sistemas altamente especializados. Los valores agregados del ecosistema de SAP es (a) cubrir el espectro completo que abarca desde el tradicional ACID a través de sistemas de transmisión hasta sistemas de escalamiento masivo que admiten una gran variedad de funcionalidades más allá del modelo relacional. Desde una perspectiva no funcional, el ecosistema de gestión de datos de SAP HANA se esfuerza (b) por proporcionar un entorno operativo único y coherente: un repositorio central para objetos comerciales con procedimientos de implementación consistentes en todos los sistemas SAP, migración sin problemas desde el desarrollo a través de pruebas a sistemas activos, interfaz única para una administración central de todos los componentes. Como se muestra a continuación, se requiere la profundidad funcional y la respiración no funcional para hacer frente a los futuros desafíos de gestión de datos en escenarios empresariales del mundo real.

1. Combinación de datos de bases de datos con sistemas de álgebra numérica: en entornos empresariales, los analistas de datos a menudo cargan datos de una base de datos relacional en un sistema de álgebra numérica para realizar su análisis mediante cálculos complejos utilizando métodos de álgebra lineal. Por ejemplo, los analistas financieros que almacenan datos de precios de acciones dentro de un RDBMS requieren, por un lado, el contexto comercial de los valores de las acciones, por ejemplo, un extracto de noticias recientes o un historial completo de cifras económicas de las diferentes compañías. Por otro lado, los analistas utilizan algoritmos estadísticos, por ejemplo, para identificar correlaciones de acciones y derivados. Como consecuencia, el ecosistema tiene que proporcionar una integración transparente de sistemas externos y organizar un procesamiento de consultas distribuidas, asumiendo la carga del intercambio explícito de datos entre los sistemas por parte del usuario.
2. Una institución cliente recopila datos de sensores masivos dentro de una gran instalación de Hadoop midiendo una gran cantidad de parámetros de grandes infraestructuras de producción. Además, el sistema ERP del cliente muestra el estado de la producción actual, así como los problemas de producción (es decir, la degradación inesperada de la producción). El desafío general ahora es correlacionar los datos del sensor con los eventos en el proceso de producción para analizar y predecir fallas de la máquina o activar actividades de mantenimiento proactivamente. Como consecuencia, el ecosistema tiene que proporcionar una integración perfecta de las infraestructuras de Hadoop y los sistemas de aplicación ERP tradicionales.
3. Un productor de jabón para baños quiere planificar las rutas para que sus equipos de servicio llenen los dispensadores. Los sensores en cada dispensador miden el grado de llenado e indican la necesidad de una recarga. Además, la compañía quiere llenarlos de manera proactiva incluso sin una necesidad inmediata, si tienen aviso (reunido de diferentes sitios web) de un gran evento en la ubicación específica. El enrutamiento comprende identificar rutas óptimas a diferentes ubicaciones, así como dar instrucciones dentro de ubicaciones enormes para encontrar los baños correctos. Con base en los patrones de uso del baño, la compañía ahora ofrece un servicio para que los equipos de las instalaciones locales bloqueen (y abran) automáticamente algunos baños, si el número de personas que visitan un baño para un evento es significativamente menor o mayor de lo esperado para reducir el costos de mantenimiento. De nuevo, los datos del sensor se almacenan en un sistema Hadoop, los datos de ubicación se almacenan en el sistema de información SIG. El sistema ERP contiene los datos maestros de la compañía y realiza la planificación de recursos, la planificación de rutas, así como la facturación y otras funciones comerciales. Los diferentes conjuntos de datos, así como las invocaciones de servicios empresariales para optimizar rutas y calcular modelos que predicen el uso de baños, deben combinarse desde una perspectiva única para la aplicación.
4. Una compañía de seguros quiere calcular sus tarifas de seguro en función de las probabilidades de huracanes y la ruta de los huracanes. Han almacenado la gran cantidad de datos sobre los huracanes pasados ​​en un almacenamiento similar a Hadoop. Sus clientes actuales y sus tarifas actuales se almacenan en su sistema ERP y las ubicaciones de los clientes se mantienen en un almacenamiento geoespacial. El objetivo es generar un modelo de predicción de futuros huracanes y asignarlo a las ubicaciones de los clientes para generar un perfil de riesgo para las diferentes ubicaciones. Los modelos computados deben volver al ERP para su consumo.
5. Un cliente es responsable de un gasoducto que se almacena como un gran gráfico. Además de la perspectiva lógica de la tubería, se almacena la información de ubicación para el gráfico. Uno de los muchos casos de uso para el cliente es el desarrollo de un plan de evacuación en tiempo real si se detecta una fuga en la tubería de gas.

Como se puede ver, muchas instalaciones de aplicaciones modernas de uso intensivo de datos requieren fusionar de manera transparente los sistemas ERP tradicionales y los conjuntos de datos relacionales clásicos con series gráficas estructuradas, geoespaciales o temporales que potencialmente residen en diferentes sistemas con diferentes requisitos de rendimiento y TCO. Mover la lógica a los datos que residen en un motor diseñado específicamente y fusionar de manera transparente los conjuntos de resultados es una capacidad central de un ecosistema de gestión de datos.

**SECCION VI.**

## **Resumen**

Dentro del documento, hemos esbozado los pilares fundamentales de un ecosistema de gestión de datos a escala web. Argumentamos que dicho entorno (1) exhibirá un rico conjunto de funcionalidades específicas representadas en estructuras de datos especializadas con algoritmos sofisticados que potencialmente residen en motores de gestión de datos específicos. Además, (2) vemos un enorme potencial en el conocimiento de aplicaciones comunicadas a la capa de gestión de datos. Cuanto más sepa el sistema de la base de datos sobre oportunidades y / o restricciones, se podrán generar mejores planes de ejecución y se podrán utilizar estructuras de datos más eficientes. Finalmente, argumentamos que (3) se necesita una orquestación poderosa para proporcionar un único punto de entrada, así como una comprensión semántica única para las aplicaciones comerciales modernas. El ecosistema de gestión de datos de SAP HANA se esfuerza por lograr exactamente estos requisitos.

### Reconocimiento

Nos gustaría agradecer al equipo de Investigación y Desarrollo de HANA por construir los motores centrales de HANA, así como el ecosistema general con la infraestructura necesaria para proporcionar una base de gestión de datos a escala web para aplicaciones empresariales modernas. Como siempre, solo trabajar en equipo puede generar una historia tan exitosa.