informatica

White Paper

The Vibe Virtual Data Machine

The Technical Foundation of a Modern Information Platform

This edition published June 2013

Table of Contents

Introduction

During the early days of business computing, computers were mainly used to automate back-office processes such as payroll and accounting. As the use of computers expanded to automate other back-office functions such as enterprise resource planning (ERP), businesses encountered problems with data fragmentation and inconsistency as they tried to understand data across disparate systems. The problem was further exacerbated as automation moved to front-office functions such as customer relationship management and sales force automation.

The advent of business intelligence and advanced analytics to aid analysts and business users with strategic decision making gave rise to a new class of analytical database systems that were optimized to address these requirements. These analytic use cases increased the need to cleanse, standardize, transform, and correlate data originating from different operational systems. As businesses have become more data-driven in their decision making, a comprehensive information platform enabling data integration and management has become a critical business requirement.

As we enter the age of big data, the problem of data management has become vastly more complicated. With the advent of mobile, cloud, and social media technologies, the sheer volume of data an enterprise can and must leverage for intelligent decision making has exploded. These data sources are geographically dispersed and may span the public Web, cloud-hosted software as a service (SaaS) applications, as well as on-premise enterprise applications.

Unlike the early days of data management, when data tended to be structured and accompanied by meaningful metadata, data from these modern systems comes in a variety of forms, such as unstructured documents and blogs, or semistructured files. The need to efficiently handle and persist these datasets has given rise to a new generation of so-called NoSQL databases. These datasets also require advanced processing techniques such as natural language processing to extract relevant information.

A final dimension to the big data problem is the increasing velocity of decision making. Modern information management platforms need to lower the latency between when data is generated and when decisions are made or actions are taken.

The Early Days: Hand Coding Data Integration

Early attempts at data integration were mostly homegrown and tended to be ad hoc and error prone. These data integration systems typically consisted of source systems that created a periodic file dump with metadata about the file format. The metadata could be specified in a location such as a COBOL copybook, embedded as part of the file as header rows, or encoded into the file name and its locations. A series of programs or jobs would pick up these files, parse and process the incoming records, and deposit them in another location. The data would then be further processed or loaded into a target system using yet another script.

As you can imagine, such systems were very fragile. Any change or failure in a job required significant manual intervention. Even if developers could overcome the operational challenges, such systems had serious drawbacks. Many of the programs that parsed, shaped, and cleansed the data were written in 3GL programming languages. Leveraging advances in hardware and software such as symmetric multiprocessing (SMP) and multithreading to increase performance was time consuming. Such performance tuning also required a very high degree of skill that was scarce. Moreover, when the underlying technology platforms inevitably advanced, developers were forced to rewrite the code to leverage these advancements.

Finally, these early systems did not provide a way for developers to understand the provenance of their data—that is, where the data originated and what computations or transformations were executed on the data. It was difficult to understand the impact of a change. For example, if a field in an underlying data source changed, most often the only way to understand the impact was to look for failures downstream.

Rise of 4GL Mapping

Seeing the limitations of custom-coded data integration, vendors introduced data integration tools in the 1990s. However, early data integration tools did not provide abstractions such as data flows and source definitions that would permit users to express their specification independently of data sources and execution platform. As a result, developers were forced to work with tools that were not well-suited for the task at hand.

Informatica's PowerCenter® product, introduced in 1997, pioneered the use of a 4GL visual data flow language to specify data movement across disparate systems. Informatica® tools allow users to specify the data integration logic as a series of graphical transformations chained together in a "mapping": a directed acyclic graph that models how users think about data movement. The visual mapping depicts a series of steps that make it very clear how data is accessed, transformed, and moved across systems.

Figure 1 shows an Informatica mapping that joins sales transaction data with an employee database and filters out smaller transactions to identify the top sales employees. The resulting data is stored in another table to be used in downstream applications—for example, to compute bonus awards. This visual mapping is significantly simpler to comprehend and change than a series of Perl or SQL scripts.

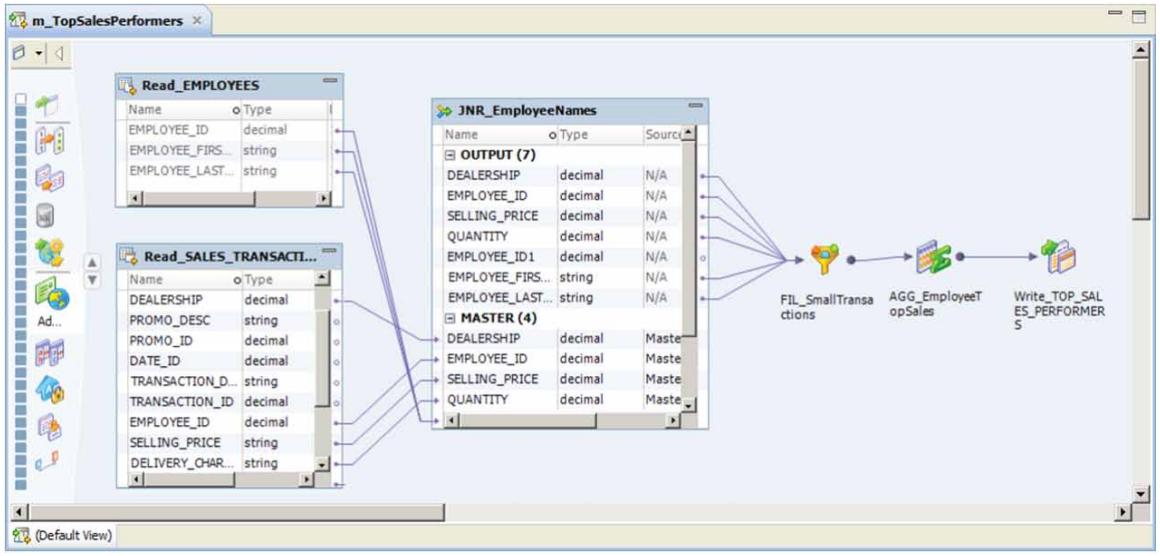

Figure 1. An example of an Informatica mapping

Informatica's easy-to-use interface lowers the skill barrier for data integration tasks. The visual data flow language separates the specification of an integration task from its run-time implementation. This separation allows data integration developers to stay focused on the business logic without worrying about the characteristics of the underlying run-time platform. In addition, the visual data flow language allows developers to easily share and reuse their work across projects.

Metadata-Driven Development

Metadata driven development is a development paradigm where business logic and processes are expressed using metadata that typically represents a higher level of abstraction as opposed to using a coding/scripting language. Because Informatica enables its developers to utilize metadata-driven development, they can easily understand and manage how information and processes are derived. Using Informatica's tools, developers can analyze and understand the impact of changes on the underlying data systems. Developers can also review the origin of data, as well as the rules or changes that have been applied to the source data.

Figure 2 presents an Informatica data lineage diagram, which depicts the origin of the data, describes the transformation path, and shows how it arrives at the target system.

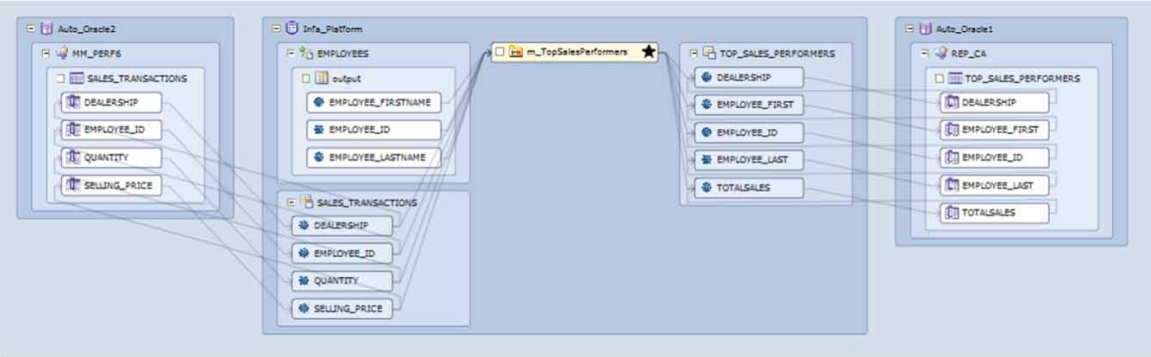

Figure 2. A lineage diagram for a mapping

The Vibe Virtual Data Machine

Following in Informatica's footsteps, a number of vendors implemented a 4GL graphical design approach to data integration in the late 1990s. However, a serious challenge remained for most of these solutions. Even if you could graphically define the transformation and mapping, what would happen if you had to point to a different target database, or if you chose a new distributed computing platform? Typically, this type of change required a rewrite of the graphical mapping. In short, adopting new technologies meant redoing everything. The reason is that most of these products were graphical code generators. Even now, the architecture of graphical code generators means that entirely different tools are required to support different run-time environments, such as cloud-based versus on-premise deployments or SMP versus NoSQL architectures.

In contrast, Informatica solved this problem from the very beginning by architecting PowerCenter on a virtual data machine. Informatica's Vibe™ virtual data machine (VDM) is the key enabling technology that frees data integration developers from the vagaries of the underlying platforms and technologies. Vibe is analogous to a computing machine—just as a computing machine accepts and executes a set of instructions, Vibe accepts a set of data-centric instructions. Vibe's instruction set consists of the collection of data operators it supports. A mapping is a valid sequence of instructions that Vibe interprets and executes. Just as a virtual computing machine can be run on multiple physical platforms, Vibe's ability to deploy to different run-time environments such as SMP, grid, analytical database appliances, and Hadoop makes it a virtual data machine.

Another unique capability of Vibe is that it can run in different modalities. Vibe can run in batch mode, typically needed for extract, transform, and load (ETL) use cases where the focus is to achieve maximum throughput in a given time. Vibe can also run in a request/response mode where the focus is to reduce the response time that is needed, such as data virtualization use cases in which data is accessed via SQL or SOAP/REST Web services.

Figure 3 depicts Vibe's high-level architecture. Vibe has two layers, a logical layer that enables the virtualization aspect of Vibe and a physical layer that executes the instructions and performs the heavy lifting of data access and transformation. The optimizer and executor components enable the virtualization phase whereas the transformation and connectors are part of the execution phase.

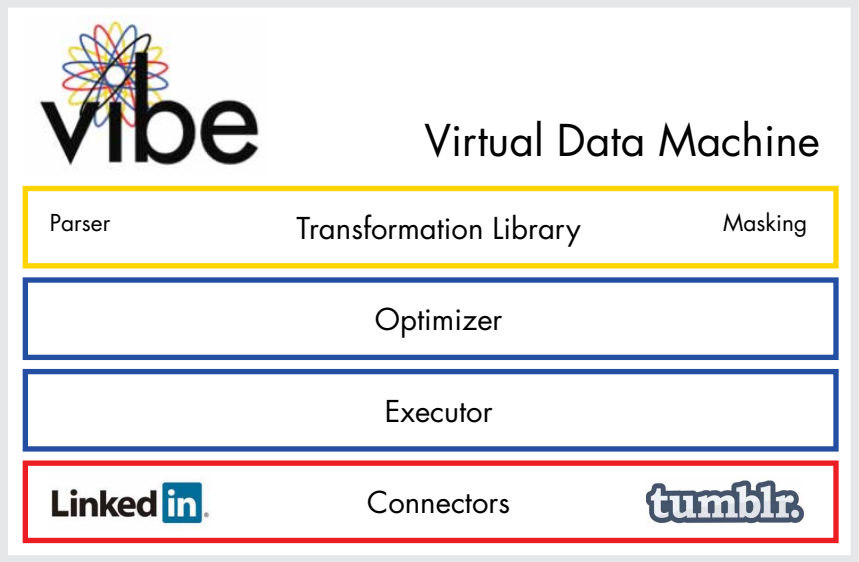

Figure 3. The high-level architecture of Vibe

Transformation Library

The Vibe VDM includes a library of prebuilt transformations or operators that perform specific functions such as filtering, cleansing, matching, and masking data. A transformation typically accepts one or more input rows, manipulates them, and outputs one or more rows. The transformation library also includes a very advanced parser component that can parse a very wide variety of complex file structures such as XML, EDI, and HL7, as well as any log file format.

Informatica also provides a set of application programming interfaces (APIs) that third parties can implement to plug in custom transformation logic. For example, a third-party developer could develop an "Internet service provider (ISP) identifier" transform that takes in an Internet protocol (IP) address, does an IP lookup, and returns information about the ISP that is providing Internet access. Such a transform could be useful in systems that process Web logs.

Optimizer

The optimizer compiles data processing logic into an internal representation and analyzes the resulting data flow to find the optimal execution plan. The goal of these optimizations is to reduce the volume of data that needs to be moved and processed and to optimize the use of computing resources. Here are some examples of optimizations performed by this component:

- If the source system can handle SQL predicates, the optimizer translates fragments of the internal mapping into SQL and passes the SQL to the source. This step can substantially reduce the amount of data that needs to be retrieved from the source systems.
- Cost-based optimization evaluates a mapping, then generates semantically equivalent mappings. For example, it may reorder the joins based on cardinality information about the sources.
- Semi-join optimization attempts to reduce the amount of data extracted from the source by modifying join operations in the mapping.

Executor

The executor component of Vibe is responsible for translating the operators into their corresponding run-time constructs for the target platform. There is an executor instance associated with every type of execution platform that Vibe supports. When a mapping is submitted for execution, it is associated with an execution context that identifies the target platform. Vibe picks the appropriate executor component to perform the work of mapping translation. For example, the Hive executor translates the mapping into a series of HiveQL tasks and creates a task flow with appropriate dependencies between these tasks. The executor is also responsible for the orchestration of these task flows.

Connectors

Vibe provides connectivity to a variety of data sources out of the box. These include all the standard relational database systems, analytical appliances, legacy mainframe systems, and ERP applications. Using the connectors, developers can access data scattered across cloud-based SaaS technologies, as well as social media data sources such as Twitter and Facebook. For real-time use cases, Vibe can connect to and process data from messaging systems such as JMS and WebSphere MQ. Vibe provides change data capture connectors that can capture changes in operational systems and propagate them downstream. Finally, Informatica supplies APIs that third-party developers can use to design and add new connectors to access data through Vibe.

Map Once. Deploy Anywhere.

The Vibe architecture allows programs written to the Vibe instruction set to run on a variety of platforms. As a result, developers that use Vibe can reuse their work seamlessly as they move from one computing platform to another. Mappings developed to run on a desktop using PowerCenter Express can run and scale seamlessly on an SMP box. They can also be deployed to a massively parallel Hadoop cluster, without recoding.

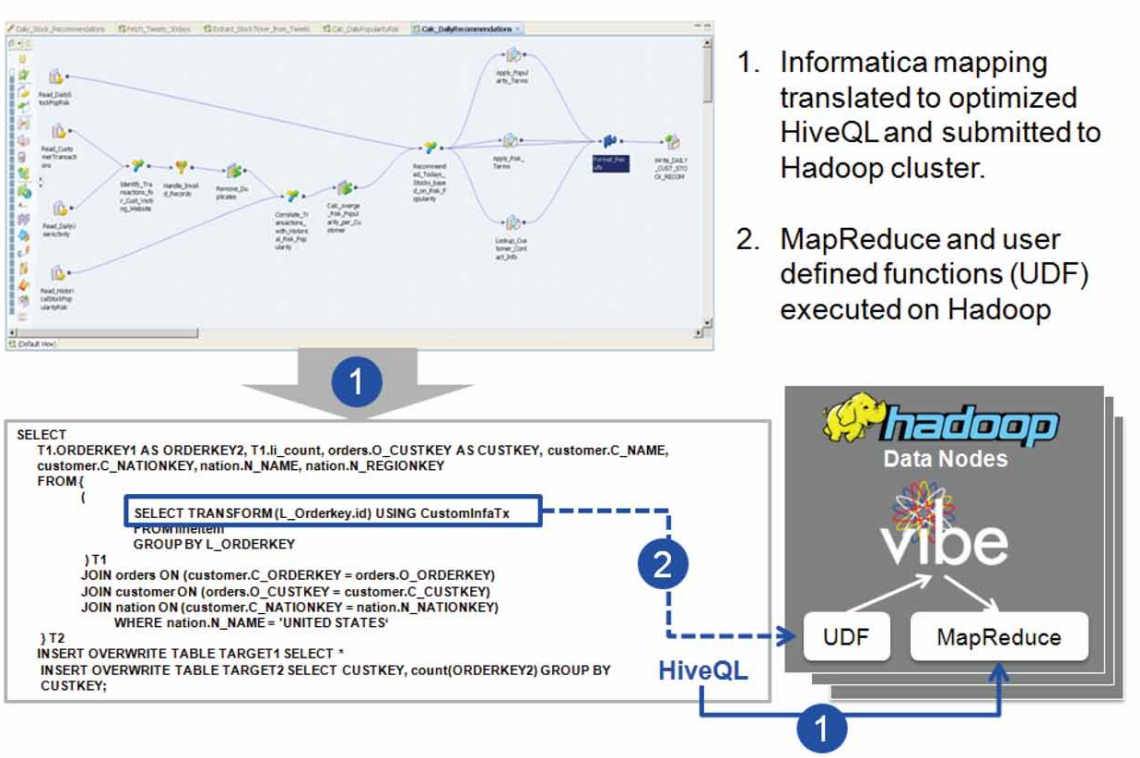

Figure 4. Mapping deployed to Hadoop cluster

The portability of mapping artifacts across platforms can be further illustrated by two examples. As described earlier, when a mapping is deployed to Hadoop, the executor component of Vibe translates the mapping into a series of Hive scripts. During the conversion phase, Vibe tries to map all the transformations to their HiveQL equivalent. However, Informatica provides a rich variety of data transformation constructs that do not have an equivalent in HiveQL. Rather than disallow these constructs or invalidate these mappings, Vibe uses Hive's extension mechanism to invoke transforms that run inside an instance of Vibe's native engine running on the corresponding Hadoop data node. Figure 4 illustrates this translation and execution process.

Vibe's unique ability to run in different modalities such as batch and request/response further increases the scope of metadata reuse across multiple use cases. Just as Vibe can scale up for big data use cases, it can scale down for data virtualization scenarios. The Informatica Data Services (IDS) product allows virtualized access to data across different systems via standard protocols such as JDBC, ODBC, SOAP, and REST. An Informatica developer can create virtual tables using the same mapping language and transformations used in batch ETL use cases. This capability of Vibe enables the same mapping to be deployed for both request/response and batch use cases.

With Vibe, developers can "Map Once. Deploy Anywhere.™" because of the VDM's portability across a wide variety of systems that vary in response time, throughput requirements, and scaling characteristics. Developers do not need to learn new frameworks or languages such as MapReduce and Hive to leverage the power of Hadoop. They also do not need to adopt a new set of tools to provide virtualized access to data. Vibe enables not just metadata artifacts but also developer skills to be portable to a wide variety of use cases and emerging computing platforms.

Embedding Vibe

So far we have discussed the reuse, sharing, and resulting agility data integration that developers gain by adopting the Vibe VDM. Application developers looking to build custom applications can derive the same benefit of portability to different platforms by embedding Vibe in their applications. With Vibe, application developers do not need to worry about connectivity to new and emerging data sources.

The Informatica Data Explorer (IDE) product that embeds Vibe illustrates the power of this ability. IDE is a data profiling product used to discover the content, quality, and structure of data sources. For example, a data analyst can use IDE to look for personally identifiable information such as taxpayer identification numbers to ensure that an organization is compliant with data privacy policies. IDE users work with "Profile", an abstraction that they can easily understand, which is executed to look for data patterns. Profiles are translated to mappings that are handed to the embedded Vibe VDM for execution. To detect patterns and latent structure in data, IDE needs custom pattern matching capabilities and transformations that are not built into the core VDM. IDE uses Vibe's extension points to implement these custom pattern matching functions. Embedding Vibe allows IDE to leverage the VDM's broad connectivity, thus allowing its users to explore a wide variety of sources. Vibe also provides transparent scaling for IDE, which can be seamlessly deployed to Hadoop.

Vibe can be embedded inside a variety of applications due to its ability to scale up as well as scale down, depending on the nature of a workload. This is exemplified by the Informatica Cloud Service (ICS). In ICS, Vibe is embedded within the downloadable secure agent that can execute on laptop and low-end desktop systems. The local agent receives instructions from the Informatica Cloud, but processes data locally. In the future, it will be feasible to embed a low-footprint Vibe into gateway devices in machine-to-machine networks.

Vibe: Future Directions

Informatica has been investing in Vibe for 15 years and will continue to innovate to ensure that Vibe remains at the forefront of modern information management. Areas where Informatica plans further investment include the following three:

SDK

All the advantages that Vibe confers to data integration developers are just as relevant to independent software vendors (ISVs) building data-centric applications. To enable application developers to take advantage of Vibe, Informatica provides APIs that enable them to embed Vibe into their application. By embedding Vibe, the applications can be deployed to and scale seamlessly across compute platforms. Vibe also enables applications to be agnostic to data sources, thus adding another dimension of application portability.

The Vibe software development kit (SDK) currently furnishes a way to parameterize and run mappings that are authored using Informatica's design tools. Going forward, we plan to externalize the APIs for programmatically generating the mappings. This approach will allow third-party developers to create purpose-built tools and applications that cater to domain experts. It will also enable ISVs to write applications for "schema on read" systems where the data schema is not known a priori.

Composite Types

Vibe currently supports the relational data model with primitive datatypes. The transformation library supports transforms that let developers parse complex files into XML and map between XML structures. Going forward, Informatica plans to support composite types as a first-class notion in the data pipeline. Figure 5 illustrates a complex type within a mapping. "GeoCoordinate" and "Polygon" are complex types that represent a point location and an area with its perimeters, respectively. "GeoFencing" is a custom transform that understands these complex types and computes whether a given point lies within the specified polygon. With composite types as first-class objects, developers will no longer have to shoehorn these types as binary or string types. This will result in increased productivity due to type safety, ease of use, and sharing.

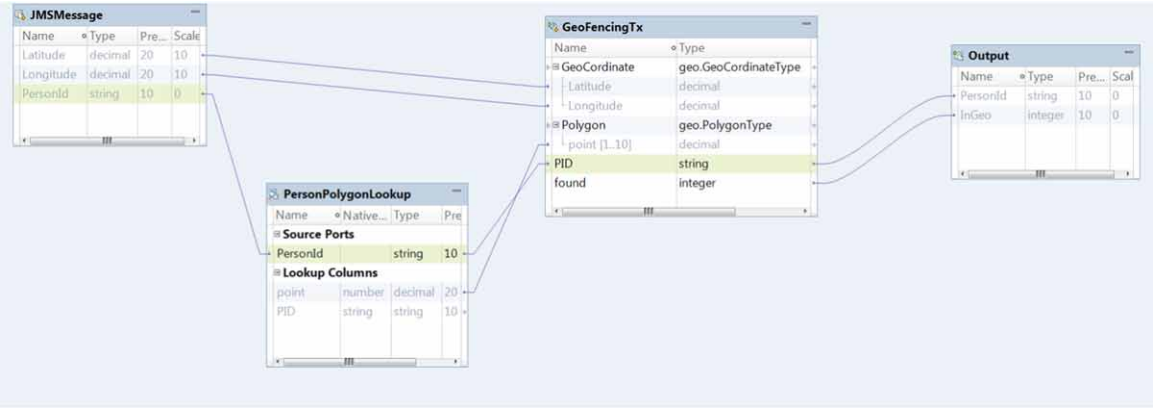

Figure 5. Composite type support

Templates

There are many situations where multiple mappings look and feel the same, with only minor variations, because they are implementing the same underlying integration pattern. Templates will allow developers to capture these patterns. Figure 6 shows the template concept.

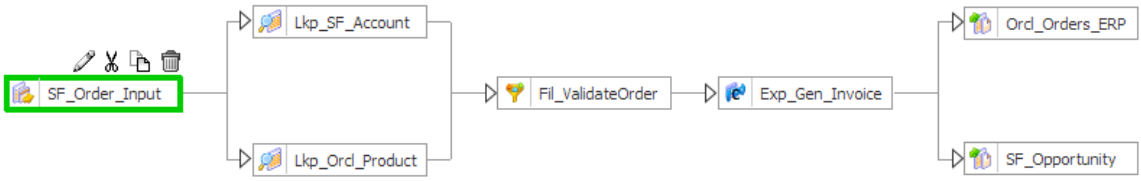

Figure 6. Rule-driven data flow template

Templates look and feel like mappings, but do not require developers to specify detail at a column level. The actual data columns to be propagated through the flow can be specified via a variety of rules. These can be metadata-based rules that can be bound at a later time such as deployment time, or even later at the time of mapping instantiation. Template-based data flow is a very powerful paradigm that will result in a dramatic increase in productivity.

Conclusion

We are in the early days of the big data era, and new technologies are constantly being invented to manage the resulting complexity. Just in the Hadoop world we have seen the emergence of languages such as Pig and Hive to shield developers from the complexity of the MapReduce programming paradigm. With the advent of pluggable computing frameworks in Hadoop YARN, we expect newer computing frameworks such as Apache Drill and Impala to emerge and challenge these frameworks. There are also systems that are trying to hybridize Hadoop HDFS and relational database systems. It is very likely that the big data landscape will see much more churn before consensus emerges on the right technology architecture.

The change is not limited to the realm of Hadoop. IT environments continue to hybridize, leveraging both cloud and on-premise platforms, leading to further data fragmentation. By adopting Vibe as their virtual data machine, data integration developers can insulate themselves from the uncertainties of all these emerging data-centric compute platforms and the challenges of data fragmentation.

All these new data technologies open up a world of possibilities for business to gain insight into its efforts and drive innovation. IT must not revert back to the hand coding or code generation paradigms of data integration. To unleash the potential locked up in these data systems, developers must embrace an agile approach to data integration that the Vibe virtual data machine enables.

About Informatica

Informatica Corporation (NASDAQ: INFA) is the world's number one independent provider of data integration software. Organizations around the world rely on Informatica for maximizing return on data to drive their top business imperatives. Worldwide, over 4,630 enterprises depend on Informatica to fully leverage their information assets residing on-premise, in the Cloud and across social networks.

Put potential to work.™

Worldwide Headquarters, 100 Cardinal Way, Redwood City, CA 94063, USA Phone: 650.385.5000 Fax: 650.385.5500
Toll-free in the US: 1.800.653.3871 informatica.com linkedin.com/company/informatica twitter.com/InformaticaCorp

IN09_0613_02474