

# Importance of Data Abstraction, Data Virtualization, and Data Services

---

David S. Linthicum

David S. Linthicum LLC



The management of data is core to successful IT. However, few enterprises have a strategy for the use of data assets, and thus waste money and opportunity. What's more, the exploding use of both cloud computing and virtualization has made the management of data assets that much more important. The value of a well-defined strategy and approach, as well as core enabling data abstraction and virtualization technology is key.

In addition, the continued use of SOA as an architectural pattern means the ability to address data as services is a critical success factor. Indeed, in virtualized applications, perhaps those that leverage cloud computing assets, SOA continues to be the best practice. SOA requires that data be addressed using common service-based interfaces, providing both a physical and logical view of the data.

Considering the shifts in the way we do IT and the underlying need to address the issues of data, we are considering approaches that provide both a systemic change in the way we manage data assets, and ultimately the ability to actually increase the value of data. Thus we have the approach of data abstraction, or the ability to address very complex physical instances of heterogeneous data using common schemas, development, and mechanisms.

While there are numerous applications for data abstraction, the core use cases today surround:

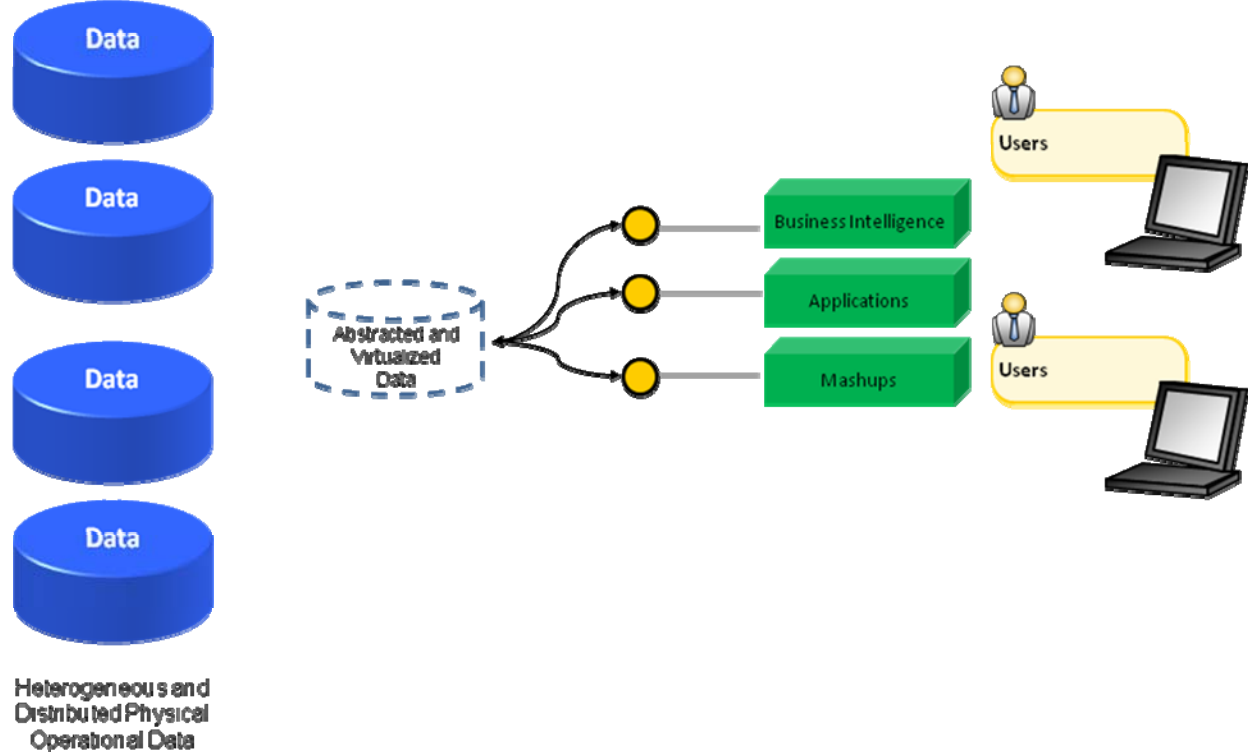
- Business Intelligence
- Applications
- Mashups

**Business intelligence**, simply put, includes approaches to understand operational data so that intelligent decisions can be made around the business. Thus, the ability to place data into a structure that is a better representative of the way we want to look at data for decision purposes is core to business intelligence. While first looked at as the process of rolling up information from many physical databases to one, we now understand how to leverage more cost effective and architecturally elegant solutions, such as data abstraction and virtualization.

**Applications** are any information resources that serve the business, such as those created by IT in support of the business. While in the past, applications have been single monolithic systems with native behaviors, processes, and data, today we have applications that span many different heterogeneous and distributed data sets. Applications may be custom developed, or perhaps prebuilt in packaged.

**Mashups** are instances of applications, or composite applications to be exact, where applications, or the mashup, are created through the combination of many different resources, including physical databases, abstracted databases, and APIs (on premise or Web-delivered). The power of mashups is that these applications may be quickly created, since they leverage many resources that already exist, binding those resources programmatically to form the applications.

Common to business intelligence, applications, and mashups is the need to leverage data in specific formats that transcend the physical structure. Thus, we have the need for data abstraction to create this virtual structure to support business intelligence, applications, and mashups (see Figure 1). This allows enterprises access to the information they need that resides in operational and physical data stores as abstracted and virtualized data, using the data structure that best fits the use case.



**Figure 1: Abstracted data provides business intelligence, applications, and mashups access to information enterprises need using the structure they require.**

## Approaches

Now that we understand the value, let's look at the approaches. There are three relevant patterns: Data Abstraction, Data Virtualization and Federation, and Data Services.

### Abstraction

Data abstraction, as related to data, is the handling of data in meaningful ways, or the ability to re-represent the data using structures that are more meaningful to the users of the data. Data abstraction enforces a separation between the abstract properties of a data type and the concrete details of its physical instance.

Therefore, the abstract properties are only visible to the consumer of the abstract data type, while the physical instance of the data is unseen and private. The power of this methodology is that you can change the physical instance many times while not impacting the way the data is leveraged.

For instance, one could define an abstract data type such as a lookup table. This lookup table may be implemented in specific ways, perhaps as a binary search tree or a hash table. Those who leverage this abstract data type are not concerned with how a lookup table is implemented since they are abstracted for what is occurring behind the scene.

The power of this concept is apparent since we're able to support diverse users of data, such as the examples of business intelligence, applications, and mashups, and then abstract the physical database from them. Thus, the database may change over time in structure and semantics, and it does not force changes by those who leverage the data. This provides a huge cost savings and strategic advantage. Moreover, abstraction allows data consumers to leverage standards, including the adoption of XML industry standards such as XBRL, ACORD, MIMOSA, PIDX, CIDX, etc., for the commercial sector, and NIEM, ICDL, MIEM, and IC-ISM for the public sector.

### Data Virtualization and Federation

Linked to data abstraction, data virtualization is used to integrate data from multiple disparate sources that may exist within the enterprises, outside the enterprise, or cloud computing-based, to provide a unified virtualized view of the data for use by any number of data consumers. This provides those who leverage data virtualization with the ability to fully leverage the database resources where they exist, and leverage them as if they were one physical database, without having to drive changes to the physical data.

The use of a virtual database has many advantages when you consider that most or all of the physical databases that currently drive the business can continue to leverage their structure and semantics. The virtual database is able to map the differences between the virtual data and the physical instances of the data, representing the databases through the use of a single virtual database. Thus, this virtual database accounts for any differences in native database formats and protocols for all relevant physical databases accessed through the data virtualization middleware.

Virtual database systems need to address the following:

- Heterogeneity
- Schema Mapping
- Autonomy

**Heterogeneity** refers to the fact that we leverage databases via data virtualization that uses different database technology, platforms, and connection mechanisms. Thus, the data virtualization technology needs to account for the differences in how the information is persisted, consumed, and updated, and the communication between the data virtualization software and the database that are in the federation.

**Schema mapping** refers to dealing with incompatible data types or query syntax that may vary greatly from database to database, or database to view. There needs to be a mechanism to resolve the differences between any number of databases and schemas, dealing with differing native semantics and physical structures. A schema mapping tool is a requirement of data virtualization to map the schemas from the physical to the abstracted and virtualized views.

**Autonomy** refers to the data virtualization technology's ability to deal with design autonomy, communication autonomy, and association autonomy. Design autonomy refers to the ability to select a database design without regard to the physical differences in how the data is stored and accessed. Communication autonomy refers the operation of the database and the ability to communicate or not communicate with other databases. Association autonomy gives power to a single database to remove itself from the federation, if needed.

## Data Services

Data services leverage both abstraction and virtualization, and provide a method of leveraging data through the use of service-based interfaces such as Web services. By using data abstraction and virtualization you are further able to customize the data schemas that are externalized to the service consumer, and abstract the data from the data service.

These services provide behavior, or, something that's done. The behavior, which is accessible by the consumer of the service, is typically designed to produce specific data, such as sales and customer information, or information in the context of behavior. The advantage of leveraging data services is that they may be easily leveraged by any software system that can invoke services, such as Web services. Thus the integration is easy. Moreover, they are the core component of a Service Oriented Architecture, and allow data to play a key role and be dealt with as a set of services.

## Challenges

Moving toward data virtualization requires some self-assessment of your existing IT issues, and a good understanding of the data assets that may be placed into a virtualized environment. The core issue is that systems that touch most of the critical business data are often considered risky, when nothing could be further from the truth when considering data virtualization and abstraction. Indeed, the risk is to continue down the path of leveraging heterogeneous data by binding the applications directly to the physical databases, thus embracing years of database and application changes that will increase costs exponentially.

However, there are cultural and technical barriers to overcome, including accelerating adoption of data virtualization which involves dealing with the users and looking at development paradigms. As with many positive changes that are made to the underlying IT architecture, you need to manage the existing resources, both technological and human.

## How to Accelerate Adoption

Adoption of data virtualization requires that you clearly state the value of this approach and technology. This means defining the business value up front to the stake holders in the organization. You do this by first looking at the existing inefficiencies within the existing approach to data management, typically with all applications, mashups, and business intelligence data consumers going directly against existing physical databases which don't support the schemas and content required.

Here are three key value points to consider here:

First, the value of eliminating costs to change the existing physical databases to support the existing data consumers. Changes to the database typically drive changes to the existing applications that leverage the data, which is risky and expensive. Leveraging data virtualization removes much of the cost and the risk.

Second, there is the value of agility around access to data. Customized, virtual views are changeable without driving redevelopment. By leveraging data virtualization, you're able to create virtual and abstracted database schemas that meet the needs of the data consumer exactly. Moreover, since the schema is changeable without forcing changes to the existing data consumers, the added agility allows you to better meet the needs of the business.

Third, there is the value of easier data sharing through the use of industry and other standards. Data services that conform physical data to standards can be easily built and widely deployed using data virtualization. This saves both time and cost as new consumers are added.

## Approaching the Users

Key to the success of leveraging data virtualization is getting the existing players on board with both the strategy and the technology. In the mix are two types of technology users: The data people and application development people. Both have unique views and experiences with the data, and thus you

have to consider different approaches when looking to promote the use of data abstractions and virtualization.

The **data people** control the existing data assets, including databases that serve data to many consumers, and databases that are bound to applications, such as ERP systems. The motivation around leveraging data abstraction and virtualization for the data people would be the value of avoiding changes to the physical database to accommodate frequent changes to the way consumers want to leverage data.

The **application development people** control the existing consumers of the data, applications, mashups, etc., and thus are looking to leverage the data in specific ways to support the application development efforts. Typically, they experience a lot of latency in getting the databases changed to support their development efforts. Thus, the use of data abstraction and virtualization will allow them to obtain the data they need, customized for the use cases as supported by the application.

## Development Paradigms

Beyond the way in which you approach people and their motivations, you need to consider the development paradigms as well. There are two basic approaches: transaction-oriented and data-oriented.

**Transaction-oriented** development refers to application development and services development where information or data is bound with the behavior of the transaction, such as sales order entry. Transactions provide isolation between programs that concurrently access a database, and information processing that is divided into individual, indivisible operations. Data abstraction and virtualization is valuable here considering the way the transactions must access the data by using specific abstracted schemas. Moreover, data abstraction and virtualization proves valuable in the management of data from the distributed heterogeneous origins, to the transactional application, and back again.

**Data-oriented** development is more data-focused, such as sales reporting, and concentrates on externalizing the data to the end users. Data abstraction and virtualization is most useful here, considering that many different databases and data stores may be used for data-oriented applications, and thus you have the ability to create custom virtualized views and manage data consumption.

## Leveraging Data Abstraction and Virtualization for the Government

Core to the issues around the use of information technology that the US government deals with these days is the use and management of existing data assets. Over the years, and through many procurement cycles, many government agencies have accumulated a very impressive list of heterogeneous data assets that use different technologies, approaches, and schemas, built over the last 30 years. Thus, there needs to be a data abstraction and virtualization strategy. This strategy should include the selection of key enabling technology to support data abstraction and virtualization to many management entities and optimization of as many of these data assets possible.

The government is also moving toward architectural mandates and standards, including the use of the DODAF (Department of Defense Architecture Framework) for enterprise architectural guidance, and the adherence to the FEA (Federal Enterprise Architecture) and other guidelines that those within Federal IT need to consider. This further complicates the data architecture issues, and makes the need for data abstraction and virtualization technology, which provides flexibility and agility around the use of data, that much more important.

For example, the Data Reference Model (DRM), which is part of the FEA, describes the data and information that support government program and business line operations. Specifically, this model allows agencies to describe the types of interaction and exchanges that occur between the Federal Government and those who interface with the government. The DRM organizes government data into greater levels of detail, and leverages a common data model to facilitate the information exchange processes intra-government and between government and external stakeholders.

Approaching data architecture for the federal government has many challenges, including the ability to leverage the existing data assets using a common data model when many of those assets can't change to accommodate the common model. Thus, the use of data abstraction and virtualization means that the government data architect is able to place a layer of middleware between the existing heterogeneous data assets, no matter what technology is employed, and map those assets to a customized representation of the data using whatever schema and view is required.

This use of data abstraction and virtualization technology provides the government with the flexibility to leverage assets already in place, and use those assets to support applications such as business intelligence and new application development, without requiring expensive and risky modifications. This has the effect of saving the government millions in IT dollars that would be spent in modifying the existing data assets, and allows the government to move quickly to an optimized state of data management where there are no limitations on what can be done with the existing information.

## **Leveraging Data Abstraction and Virtualization for the Corporation**

While the government needs data abstraction and virtualization technology, as we discussed previously, there is also need in the commercial sector. Like the government, many Global 2000 corporations have data architectures that are overly complex and less than efficient. In many instances, productivity is hindered by inability to access the information required for both transactional and business intelligence systems. The physical database layers are just too distributed, and leverage many types of database technology and database models that were acquired over the years. In short, the data is not in logical order, which makes it useful only to the applications that are already tightly coupled to the physical database.

New compliance regulations and requirements around vertical markets drive corporate IT back to the existing databases for the required information. Data abstraction and virtualization technology allows this access by placing a logical layer between new applications and the existing data and physical

databases. Using this technology, corporate IT avoids expensive and risky modifications to the physical databases, but they can still access the operational or business intelligence information, as they need it, and for any number of purposes.

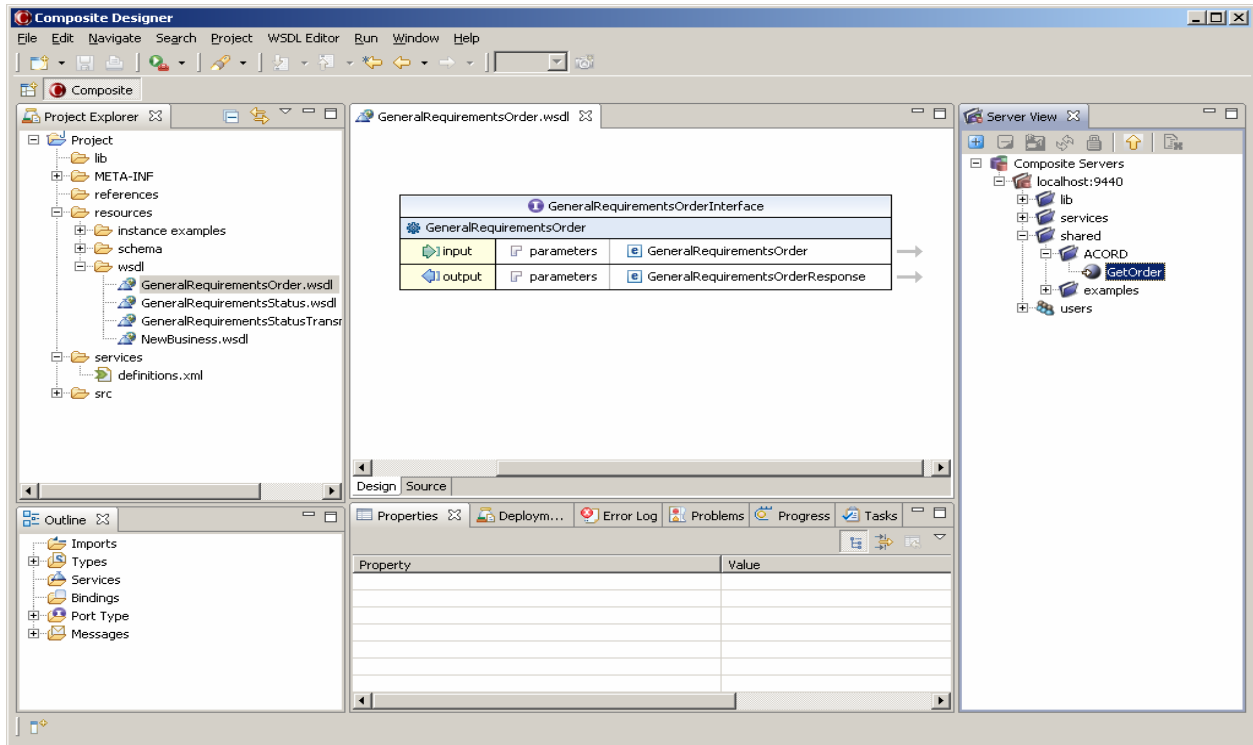
Moreover, the need to modernize data architectures in order to reduce costs within corporations will drive many in corporate IT toward SOA and cloud computing. The need to access data in a logical manner will be on the critical path as we look to abstract the data for use within services, and as services themselves. Data abstraction and virtualization will provide a foundation to make those efforts a success, as well as reduce risk and costs, with benefits going right to the bottom line.

## **Native Development Environment**

When considering data abstraction and virtualization, you also need to consider how things will be built, including support for the existing native development environment. Composite Software supports development through two paths, the Eclipse IDE and traditional relational data modeling. Each provides different ways to get to the same data abstraction and virtualization objectives.

### **Eclipse IDE**

The Eclipse IDE is a powerful IDE framework with strong support from the open source community. Composite Software's Designer supports the Eclipse environment for XML/SOA and Java centric developers (see Figure 2). Moreover, Composite Designer enables top-down or contract-first approach to development of standards-based Web services as well as a range of enhanced graphical editors for XML, XSLT, XQuery, XSD, WSDL, etc., development.



**Figure 2: By supporting the Eclipse IDE, Composite Software brings data abstraction and virtualization to the developer using familiar tools.**

## Relational Modeling

Typically data people prefer relational modeling oriented development tools when building virtualized views and lower level data services. Composite Software's Studio is the environment of choice for SQL centric developers.

## Leveraging Composite Software

Clearly, there is a core need for data abstraction and virtualization when considering how the data will be consumed going forward. However, the data abstraction and virtualization path requires that you leverage the right technology. Composite Software provides proven technology to address any number of physical source databases, allowing any number of standards-based and customized schemas, for consumption within business intelligence, applications, and mashups.

The Composite Information Server (See Figure 3) is a Java-based server that accesses existing data non-invasively, federates disparate data, abstracts and simplifies complex data, and delivers it as virtualized data services or relational views. Featuring patent-pending query optimization technology, Composite Information Server delivers the highest performance in the industry. Dual development environments optimize productivity: Studio for the traditional database-centric developers, and Designer for the services-centric application developers who are comfortable with an Eclipse-based environment.

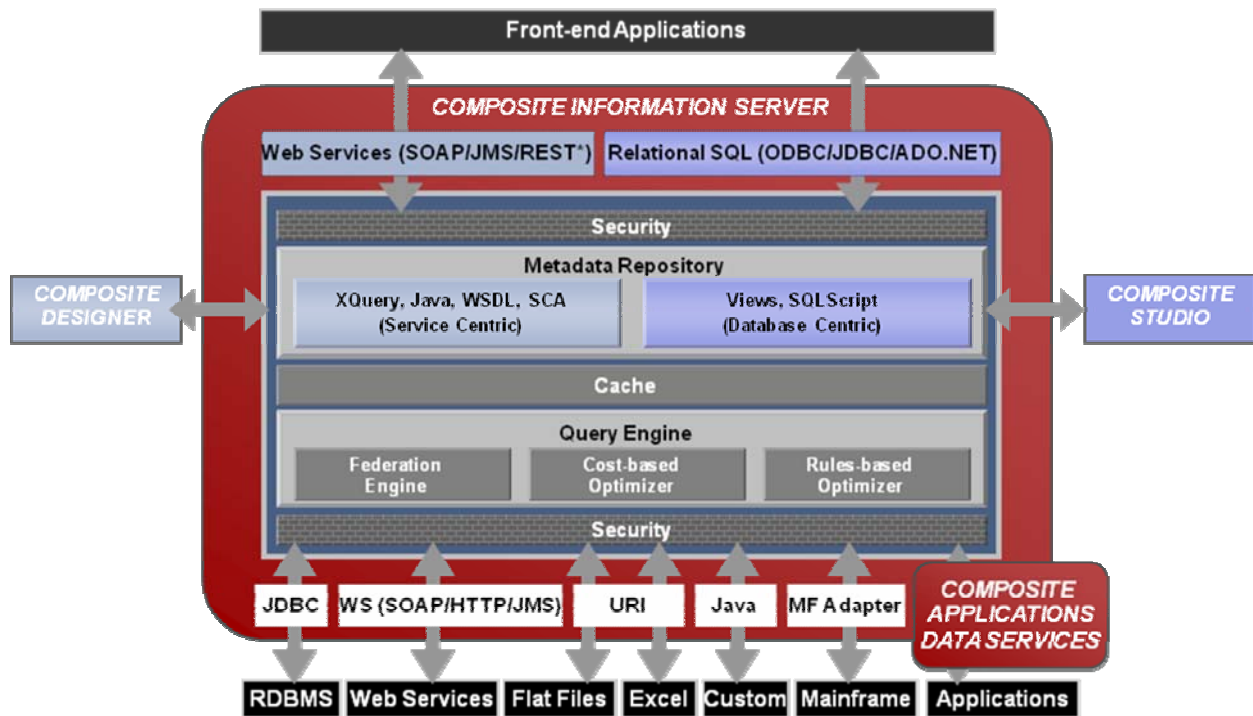


Figure 3: The architecture of the Composite Information Server.

As architectures grow more complex, and the need for agile data architectures continues to expand, the need for data abstraction and virtualization technology, such as the solution from Composite Software, will expand as well. What's important to keep in mind is the core value of this approach and technology, and when and where to leverage it to be most effective.

### ***About the Author***

David Linthicum (Dave) is an internationally known Enterprise Application Integration (EAI), Service Oriented Architecture (SOA), and cloud computing expert. In his career, Dave has formed or enhanced many of the ideas behind modern distributed computing including EAI, B2B Application Integration, and SOA, approaches and technologies in wide use today.

Currently, Dave is the founder of David S. Linthicum, LLC, a consulting organization dedicated to excellence in SOA product development, SOA implementation, corporate SOA strategy, and leveraging the next generation Web (Web 2.0). Dave is the former CEO of BRIDGEWERX, former CTO of Mercator Software, and has held key technology management roles with a number of organizations including CTO of SAGA Software, Mobil Oil, EDS, AT&T, and Ernst and Young. Dave is on the board of directors serving Bondmart.com, and provides advisory services for several venture capital organizations and key technology companies.

In addition, Dave was an associate professor of computer science for eight years, and continues to lecture at major technical colleges and universities including the University of Virginia, Arizona State University, and the University of Wisconsin. Dave keynotes at many leading technology conferences on application integration, SOA, Web 2.0, cloud computing, and enterprise architecture, and has appeared on a number of TV and radio shows as a computing expert.

David S. Linthicum, LLC  
[www.davidlinthicum.com](http://www.davidlinthicum.com)  
11654 Plaza America Drive, #103  
Reston, VA 20190  
[david@davidlinthicum.com](mailto:david@davidlinthicum.com)