What is Data Virtualization?

by

Rick F. van der Lans, R20/Consultancy

August 2011

Introduction

Data virtualization is receiving more and more attention in the IT industry, especially from those interested in data management and business intelligence. And that increased interest is well-deserved, because data virtualization has a unifying impact on how applications manage, handle, and access the many disparate data sources in an organization. But what exactly is it, and why should we be interested?

Virtualization itself is not a new concept in the IT industry. Probably the first application of virtualization was in the 1960s when IBM used this concept to split mainframes into separate virtual machines, which made it possible for one machine to run multiple applications concurrently. Nowadays, almost everything can be virtualized, including processors, storage, networks, data centers, and operating systems. In general, virtualization means that applications can use a resource without concern for where it resides, what the technical interface is, how it has been implemented, which platform it uses, how large it is, how much of it is available, and so on. The virtualization solution encapsulates the resource in such a way that all those technical details become hidden, and the application sees a simpler interface.

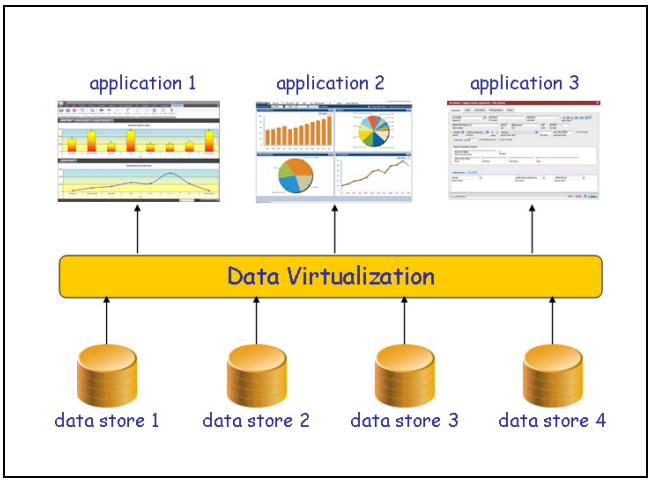

Figure 1 *When data virtualization is applied all the data sources are presented as one integrated data source*

Data virtualization is one of those virtualization forms. The encapsulated resource is, as the term indicates, data. In a nutshell, when data virtualization is applied, an abstraction layer is provided that hides for the applications most of the technical aspects of how and where data is stored; see Figure 1. Because of that abstraction layer, applications don't need to know where all the data is

stored physically, where the database servers run, what the required API is, which database language to use, and so on. To every application it will feel as if it accesses one large database.

Defining Data Virtualization

Before we define data virtualization, we introduce two new terms: *data consumer* and *data store*. A data virtualization layer can be accessed by any type of application, for example, it can be an online data entry application, a reporting application, a statistical model, an internet application, a batch application, and an RFID sensor. In this book we will use the neutral term data consumer for all these different types of applications. In other words, a data consumer can be any application retrieving or manipulating data. Likewise, the data being accessed through a data virtualization can be anything, for example, a table in a SQL database, a spreadsheet, and a sequential file. In this book we will use the neutral term data store to indicate any source of data.

The definition of data virtualization is:

> *Data virtualization is the process of offering data consumers a data access interface that hides the technical aspects of data stores, such as location, storage structure, API, access language, and storage technology.*

Advantages of Data Virtualization

But what are the advantages of data virtualization? Why should we use it? For an application, accessing a data store directly is relatively easy. In most cases a user id, password, the name of a database or file, and some other technical details must be supplied, and before you know the application has access to the data stored in the tables. Database access statements written in SQL, XQuery or any other language, can be executed to deliver the right data.

So if it's so easy, why is a data virtualization layer needed? Accessing data via a data virtualization layer offers many practical advantages. First, we indicate the advantages of data virtualization if only one data store is accessed:

- The right database language and API: It could be that the database language and API offered by a data store is not ideal for the application, or maybe it's a language not supported by the application. Maybe the data store supports SQL through the JDBC API while the application would prefer to use MDX, XQuery, CQL, or a set of Java classes. A data virtualization layer should be able to convert the language and API supported by the data store to the proper database language and API.

- Simplified table structure: The table structure implemented by the data store might be complex for the application. This could lead to a large number of database queries for retrieving data and might complicate application development unnecessarily. A data virtualization layer could present a simpler and more appropriate table structure, leading to a simplification of application development and to a reduction of application maintenance.

- Minimize interference: An application might cause interference on the data store it's accessing. Its queries might be so resource intensive that other applications experience performance degradation. If a data virtualization layer supports some kind of caching feature, the application will access the data in the cache instead of data in the data store, hence minimizing interference.

- Data transformation: Specific data values might have a format not suitable for the application. Imagine an application wants to see telephone number values as pure digits and not in the form where the area code is separated from the subscriber number by a dash. The data virtualization layer could handle this transformation for the application.

- Data cleansing: Some data values in a data store might not be correct. For example, the column Gender contains three different values to indicate Female. In this case, the application has to include code to transform the incorrect values to correct ones. It would be better if this is handled by a data virtulaization layer that only shows correct values to the applications.

Although all the above advantages already justify the use of a data virtualization layer, the major advantages come when applications need to integrate data from multiple data stores, especially when it's a heterogeneous set of data stores. When data virtualization is applied, this layer will present one logical database to the applications, which leads to the following additional advantages:

- Unified data access: Different data stores might be using different storage formats. For example, some of the data might be stored in a SQL database, some in Excel spreadsheets, some in index sequential files, some in databases supporting other database languages than SQL, some in XML documents, and some of the data might even be hidden in HTML-based webpages. A data virtualization layer might be able to offer one unified API and database language for accessing all those different storage formats, therefore simplifying data access for the applications. They will only have to support one language and API.

- No replication of data integration logic: If multiple applications must access multiple data stores, each and every data consumer has to include code responsible for integrating those data stores. This leads to a lot of replication of data integration solutions; see Figure 2. A data virtualization layer centralizes that integration code and all applications will share it.

- Consistent results: If each application includes its own integration solution, how do we guarantee that they all integrate data in the same way and according to the same rules? If this is not guaranteed, there is a chance that data consumers will deliver different and inconsistent results. If all the integration solutions are handled by a data virtualization layer, results will be increasingly consistent.

- Efficient distributed data access: When data from multiple data stores is joined, a performance question is always where and how the join is processed: is all the data first shipped to the application where it's joined, or should the data from one data store be shipped to another, and the latter will process the join? And other strategies exist. An application developer should not be concerned with such an issue. Therefore, this is a

task of coming up with an efficient distributed data access strategy that is handled by the data virtualization layer, and not by the developer.

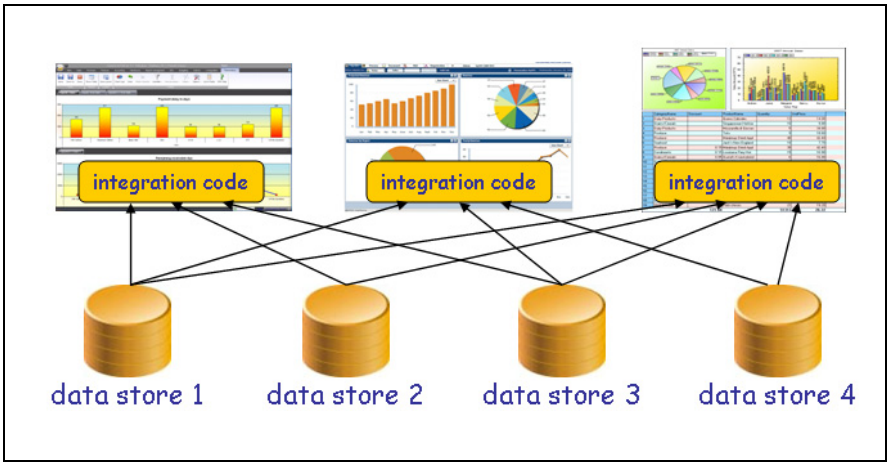

Figure 2 *Integrating data stores without a data virtualization server leads to duplication of integration code*

To summarize, data virtualization simplifies application development because it reduces the amount of code required for accessing the necessary data in the right way and form; see Figure 3. And if multiple applications use the same data virtualization layer, they share the same specifications; see Figure 4.

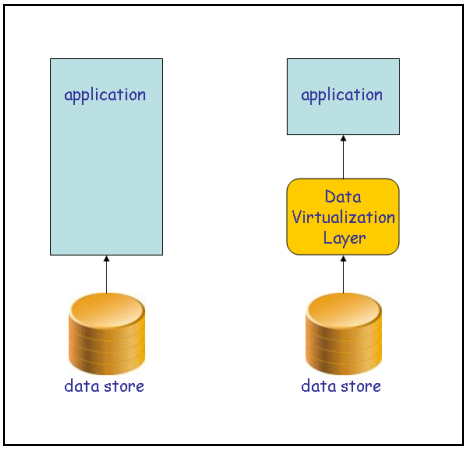

Figure 3 *Data virtualization reduces the amount of application code required for accessing data*

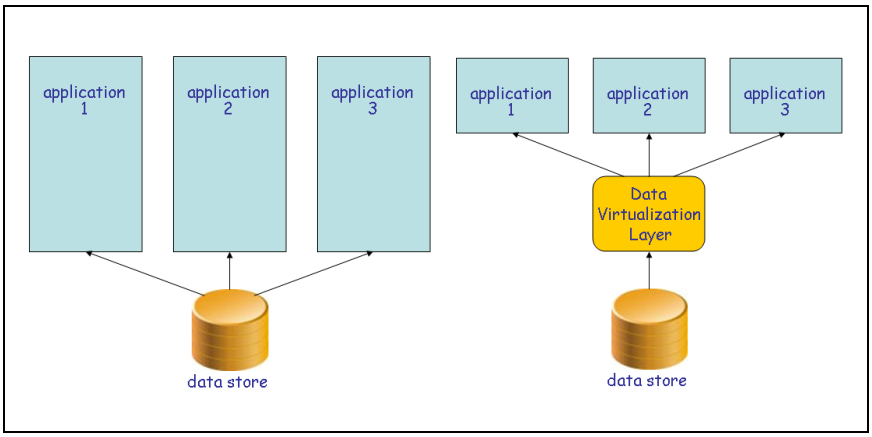

Figure 4 *Multiple applications share the same specifications if data virtualization is applied*

Implementations of Data Virtualization

Technically, data virtualization can be implemented in many different ways. Here are a few examples:

- With a *data virtualization server* multiple data stores can be made to look as one. The applications will see one large data store, while in fact the data is stored in several data stores. More on this in the next section.

- An Enterprise Service Bus (ESB) can be used to develop a layer of services that allows access to data. The applications invoking those services will not know where the data is stored, what the original source interface is, how the data is stored, and what its storage structure is. They will only see, for example, a SOAP or REST interface. In this case, the ESB is the abstraction layer.

- Placing data stores in the cloud is also a form of data virtualization. To access a data store, the applications will see the cloud API, but they have no idea where the data itself resides. Whether the data is stored and managed locally or remotely is transparent.

- In a way, building up a virtual database in memory with data loaded from data stored in physical databases can also be regarded as data virtualization. The storage structure, API, and location of the real data is transparent to the application accessing the in-memory database. In the BI industry this is now referred to as *in-memory analytics*.

- Object-Relational Mappers (ORM) are tools that convert data structures from data stores to the concepts used in object-oriented programming languages, such as Java and C#. For example, an ORM might be able to convert the flat table structures of a SQL database to the object-oriented concepts used in Java. The effect is that the Java programmers won't have to understand and deal with the characteristics of the SQL concepts, but purely with Java concepts.

- Organizations could also develop their own software-based abstraction layer that hides where and how the data is stored.

The data virtualization server is the first alternative listed here. A *data virtualization server* is a dedicated product designed specifically to present multiple heterogeneous data stores as one logical data store to data consumers. For the data consumers accessing the data virtualization server is very similar to logging on to a database server. Without knowing the data consumers join data coming from different data stores, even when different storage models and concepts are used.

Many data virtualization products are currently available; see the table below.

Data Virtualization Server	Vendor
Composite Information Server	Composite
Dataflux Federation Server	Dataflux
Denodo Platform	Denodo
Entropysoft Content Federation Server	Entropysoft
IBM InfoSphere Federation Server	IBM
Informatica Data Services	Informatica
Information Builders iWay Enterprise Information Integration suite	Information Builders
Queplix Virtual Data Manager	Queplix
RedHat JBoss Enterprise Data Services	RedHat
Software AG Information Integrator	Software AG

Summary

Organizations are becoming more and more aware of the value of the data they have been collecting in their IT systems. However, currently their data is scattered over many different systems and is stored in a plethora of different data store technologies, ranging from high-end SQL databases to simple spreadsheets. The need to have one integrated and unified view of all the data is rapidly growing. Organizations want to be able to do 360° reporting (show everything we know about a customer), they want to see one single version of the truth (consistent reporting), and so on. Data virtualization can play an active and central role in unifying data sources and presenting the data to applications and business users in an integrated fashion.

About the Author Rick F. van der Lans

Rick F. van der Lans is an independent analyst, author and lecturer specializing in data warehousing, business intelligence, service oriented architectures, and database technology. He is chairman of the annual European Data Warehouse and Business Intelligence Conference (organized in London), chairman of the BI event in The Netherlands. He writes for several magazines and websites including BeyeNetwork.com. He introduced the Data Delivery Platform in 2009 in a number of articles published on BeyeNetwork.com.

You can get in touch with Rick via:
Email: rick@r20.nl
Twitter: http://twitter.com/Rick_vanderlans
LinkedIn: http://www.linkedin.com/pub/rick-van-der-lans/9/207/223