
# Data Virtualization in the Time of Big Data

## Making Big Data Easy

A Technical Whitepaper

Rick F. van der Lans
Independent Business Intelligence Analyst
R20/Consultancy

March 2022
Completely Revised Edition 2.0

Sponsored by

**About TIBCO**

TIBCO Software unlocks the potential of real-time data for making faster, smarter decisions. Our Connected Intelligence Platform seamlessly connects any application or data source; intelligently unifies data for greater access, trust, and control; and confidently predicts outcomes in real time and at scale. Learn how solutions to our customers' most critical business challenges are made possible by TIBCO at www.tibco.com.

# Table of Contents

# 1 Introduction

**Big Data Has Arrived** – The big data concept was introduced in the late 1990s and has since been embraced by all kinds of organizations. Data can be "big" because of its sheer volume, because it is not structured in the traditional way, or because it streams in with enormous quantities. The business advantages of big data systems are clear, they can improve, deepen, and strengthen an organization's analytical capabilities enabling them to improve their business and decision-making processes.

In the beginning, most SQL products were not optimized to handle big data workloads. Therefore, to store, process, and analyze big data, many organizations selected big data storage platforms that were designed and optimized to work with big data.

**Limitations of Big Data Platforms** – While the new platforms enable organizations to develop big data systems, they do have their omissions and restrictions, such as:

- Because most of these platforms do not support SQL, developers must learn and become familiar with their proprietary database languages, APIs, and database concepts.
- Many big data platforms are optimized to support a single use case, such as transactional processing, graph analytics, or time-series analytics. Using the same data for another use case is difficult and almost always involves copying the data to another storage platform optimized for the other use case.
- Popular BI and reporting tools cannot be used to access the data stored in those systems, because of the lack of SQL support.
- Many platforms do not understand the structure of the data. This requires that logic is developed within each and every application and analytical tool to transform the structure-less data into structure-rich data.
- Some big data is produced remotely and distributed and is too much to copy to a centralized environment to be used for analytics.
- Support for securing data in many big data platforms is still limited compared to other platforms.
- Most of the big data systems can deliver technical metadata, but not descriptive metadata.
- Using multiple big data systems supporting different languages complicates integration of all the data.

**Limitations of Big Data Platforms** – Data virtualization servers are designed and optimized to present data stored in all kinds of systems, databases, and platforms as a single, integrated logical database that can be accessed by the preferred language. This enables developers to use the familiar SQL language to access non-SQL systems and to integrate data from multiple systems as easily as if they are joining two tables from one database. All this functionality works without having to physically copy data: *zero-copy integration*.

**This Whitepaper** – This whitepaper describes how data virtualization can solve the forementioned omissions and restrictions of big data platforms. Data virtualization servers can act as smart gateways to all the big data platforms in conjunction with the more traditional enterprise data platforms allowing developers to use their preferred language, including SQL.

Accessing big data storage platforms through data virtualization extends and expands what organizations can do with their big data investments. It raises the business value of big data.

# 2 Unlocking Big Data for Frictionless Access

**Accessing Big Data Can Be Complicated** – Storing big data is only useful and valuable if it is accessible by all data consumers who need it. This applies to all types of data consumers, including business users working with simple BI tools, customers using Java apps running on mobile devices, data scientists using advanced analytical tools to create prescriptive models, and financial specialists using Excel.

Additionally, access to data should be as simple as possible. In other words, data consumers need *frictionless access* to big data. Unfortunately, many of the big data platforms are not always easy to access. This chapter describes three aspects that impede frictionless data access.

*Data consumers need frictionless access to data.*

**Absent SQL Interface** – Many big data platforms do not provide a native SQL interface. They support proprietary interfaces, languages, and database concepts. Therefore, many are classified as *NoSQL* products. Because many BI tools do expect data to be available through a SQL interface, a mismatch exists between these two tool categories.

Data virtualization servers have always enabled developers to use well-known and standardized interfaces, such as SQL, REST/JSON, SOAP/XML, and OData, to access a wide range of data sources using complex, technical, and mostly proprietary interfaces, such as NoSQL databases, Hadoop files, Excel spreadsheets, enterprise service busses, and mainframe-based database servers; see Figure 1. The benefits of using data virtualization are that developers do not have to learn new programming languages and database concepts, most reporting and analytical tools can be used to access big data, and big data is unlocked for a wide range of developers, including BI developers, data scientists, and IT specialists.

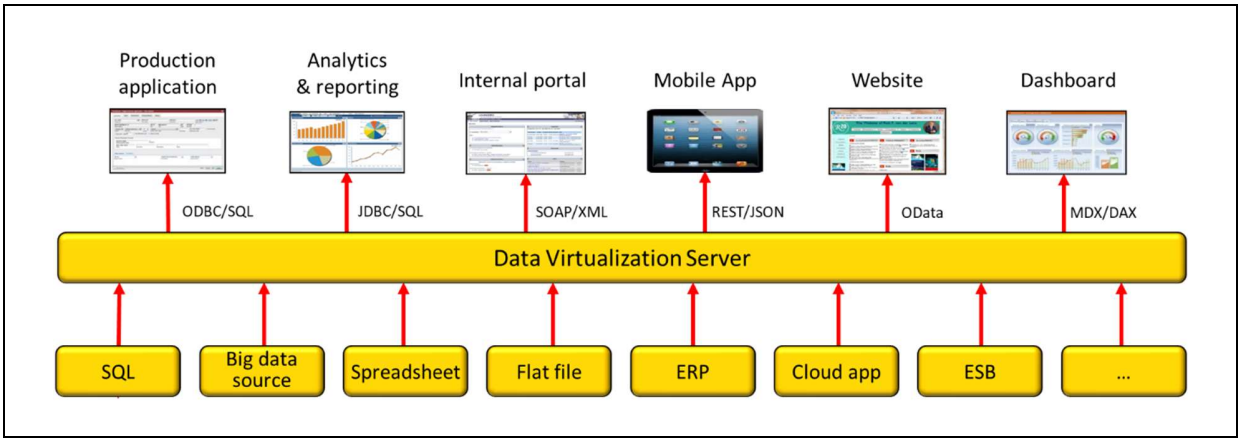


**Figure 1** *With data virtualization, developers can use the language they prefer to access any data source.*

**Polyglot Persistence** – Numerous organizations have invested not in one, but several big data platforms, all supporting different languages and interfaces. Persisting data in several systems using different languages is called *polyglot persistence*. Integrating data from those systems means having to deal with the complexity of this multitude of languages.

Dealing with multiple different language polyglot persistence is simplified with data virtualization; see Figure 2. All the different languages can be shielded from the developers. Developers do not have to work with all these different languages, interfaces, and database concepts. Instead, all the big data sources are presented as one integrated database accessible through one language. Likewise, it also reduces the risks for an organization to deploy another specialized big data platform for a new big data system.

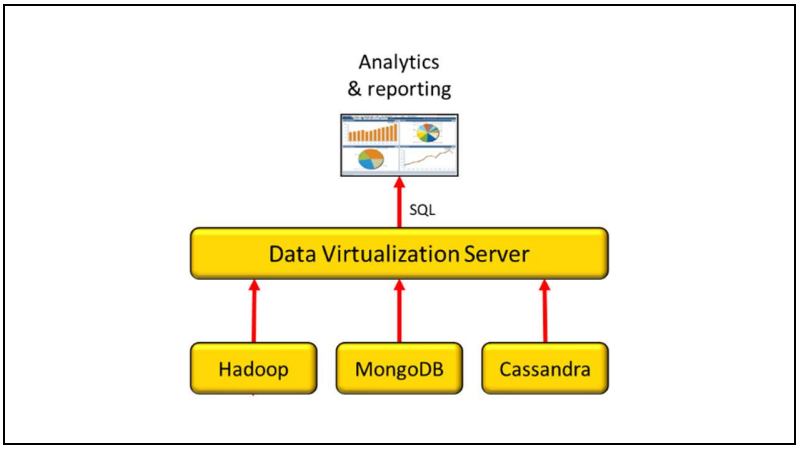


**Figure 2** *Data virtualization simplifies dealing with polyglot persistence.*

**Schema-On-Read** – With the introduction of Hadoop and cloud platforms, it has become common to store data using an approach called *schema-on-read*. Schema-on-read means that the file or system that contains the data, does not understand the structure of the data. An example of schema-on-read is when data is stored in Hadoop files and the values in the records are separated by commas or tabs. Schema-on-read can also mean that the records in one file can have different sets of values.

*Logic is required to transform structure-less into structure-rich data.*

The consequence of schema-on-read, where the structure of the data is not known to the storage platform, that applications need to know how to interpret the data. In other words, the *schema* (structure) of this *structure-less* data must be determined by the applications when they read the data, hence the term schema-on-*read*. Applications must include logic to transform structure-less data into meaningful, *structure-rich* data. In fact, each application that accesses that same data must include logic that understands the data. And if the structure of the data changes, for example, a new column with values is added, the logic in each application accessing the data must be changed accordingly. This can be quite an exercise, especially if it is not documented properly which applications access this data. The more applications that use this schema-on-read data, the greater the maintenance effort will be.

To avoid this replication of logic to transform the structure-less data into structure-rich data, the logic can be defined in the virtual tables of a data virtualization server; see Figure 3. Offering structure-rich data to IT developers, data scientists, and business users saves them time. It also simplifies maintenance, because if the structure of the data changes, it only needs to be changed once in the data virtualization server and not in all the applications.

Note that from a data storage perspective, storing structure-less data offers certain benefits, the most important one being data storage flexibility. If the structure of the original data changes, the definition of the data structure does not need to be changed. Data already loaded, which has the old data structure, remains unchanged.

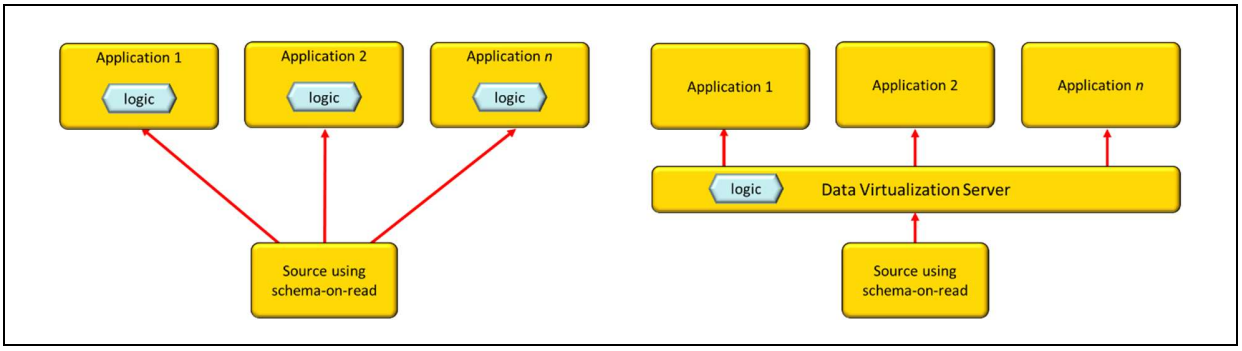


**Figure 3** *The logic to transform a schema-on-read data source into a structure-rich one can be centralized in a data virtualization server.*

## 3 Centralizing Data Processing Logic on Big Data

To become consumable, big and traditional data may need to be filtered, aggregated, integrated, transformed, and calculated. This *data processing logic* may need to be implemented in many applications; see Figure 4 (left-hand diagram). Defining this logic within the applications themselves, results replication of this logic. This decreases productivity, complicates maintenance, and introduces the risk that the logic is implemented inconsistently. This is especially true when the logic is complex and implemented with different languages and tools.

All this can be simplified by defining this data processing logic centrally in a data virtualization server; see Figure 4 (right-hand diagram). In this case the logic is defined once and used many times, which improves productivity, maintenance, and consistency.

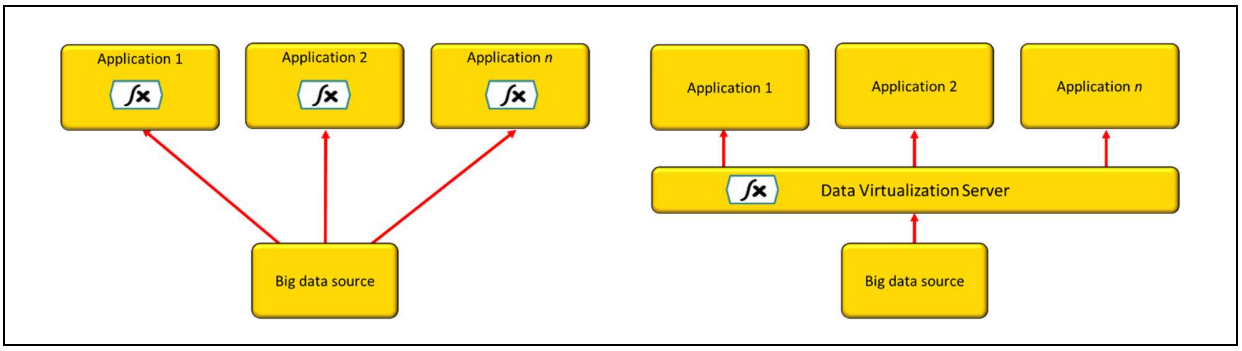


**Figure 4** *Data processing logic can be defined centrally in a data virtualization server.*

## 4 Accessing Remote Big Data

**Big Data Too Big to Move** – Some factories, oil platforms, and manufacturing plants, produce massive amounts of sensor data that qualifies as big data. This sensor data can be very valuable for analytical purposes. Combining sensor data from multiple machines in particular can be insightful. To analyze all that sensor data produced in multiple, remote places, it must be combined. Many centralized datalakes and datahubs are developed to store all that remote sensor data in one location for analytical purposes. Sometimes, though, big data is simply too big to move or copy, because of bandwidth shortage and

inadequate load speeds. In this case, instead of moving the data to the point where it can be analyzed, the analyses must be pushed to the location where the data is produced and stored.

Data virtualization can help with this. Virtual tables can be defined that link to the remote big data sources; see Figure 5. To analyze sensor data from multiple remote sites, data consumers only need to join the virtual tables. To them, it feels as if all the data is stored in one place.

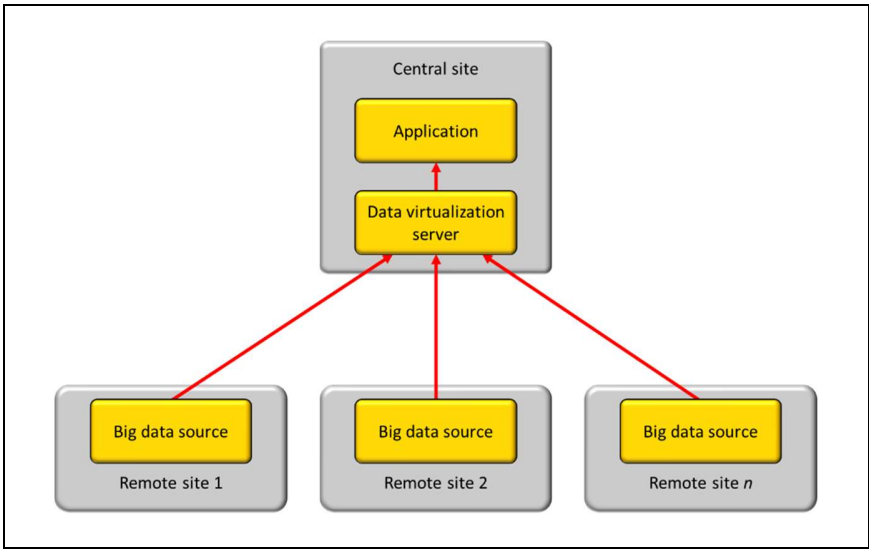


**Figure 5** *Data virtualization can simplify access of remote data sources.*

**Query Pushdown** – The data virtualization server uses *query pushdown* to prevent that all the remote data is copied when the join is executed. This means that as much of the required query processing as possible is executed by the big data platforms on the remote sites. For example, if only sensor data of a specific type and day needs to be analyzed, a filter is pushed down to make sure that only relevant data is transmitted to the data virtualization server. Or, if only aggregated results are needed, the aggregation operation is pushed down to the remote sites, and only aggregated results are returned.

*Query pushdown is used by data virtualization servers to minimize the amount of data transmitted from remote sites.*

Query pushdown works if the big data platform running on the remote sites is intelligent enough to execute those queries. However, if the remote data is stored in a simple file, no query pushdown is possible, and the amount of data transmitted across the network is probably very high. This can be reduced by installing a data virtualization server on each of the remote sites as well can; see Figure 6. For example, if a manufacturing plant collects all its sensor data in a simple file that is accessed by a data virtualization server running on a central server, a huge amount of network traffic is generated, slowing down queries. For example, imagine that an analyst only needs to know the average temperature of the last two months measured by a specific heat sensor. This is a relatively simple SQL query. However, because no parts of the query can be pushed down, data virtualization servers can only retrieve all the rows from the file and execute the query on the central server. Retrieving all the rows across the network makes the query slow.

A more efficient solution is to install a second data virtualization server on the remote site. Then, the central data virtualization server pushes the entire query down to the remote data virtualization server. That data virtualization server accesses the entire file locally involving no network traffic, executes the query, and transmits only the query result back to the central data virtualization server. This involves less network traffic and improves query performance. All this remains shielded from the data consumers.

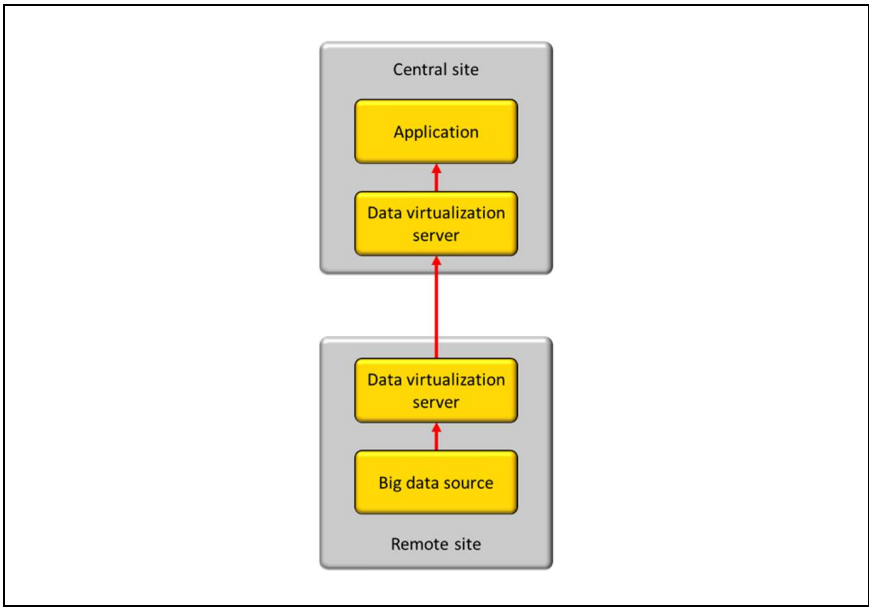


**Figure 6** *A second data virtualization server can minimize network traffic when big data on remote sites is accessed.*

## 5 Caching Big Data to Accelerate Queries

*Many NoSQL platforms do not support strong analytical features.*

**Query Pushdown** – Some big data platforms, including many of the popular NoSQL platforms, are optimized to run massive amounts of transactions. However, being optimized for a single use case comes at a price. Because they are optimized for transactional processing, their query and analytical capabilities are weak and limited. For example, some do not even support joins or aggregations. Those operations are expected to be executed by the applications themselves. In general, queries on such big data platforms are slow and can significantly interfere with the transactional workload.

In data virtualization servers, virtual tables can be defined that point to such big data platforms. Unfortunately, because these platforms are limited in their query capabilities, almost no query pushdown can be performed by the data virtualization servers. And executing queries interferes with the transactional workload.

**Caching Virtual Tables** – The *caching* mechanism of a data virtualization server may solve this problem. Any virtual table can be *cached,* which means that the virtual content of a virtual table is determined and physically stored. Developers can determine in which data platform these cached virtual tables are stored. From then on queries executed on that virtual table are processed by running the query on the cached version of the virtual table. If these cached tables are stored in a fast, analytical SQL database server, query execution will be much faster than on the original big data platform. In fact, instead of pushing down small pieces of the query processing to the big data platform, large parts of the query are now pushed down to the fast SQL server. Caching virtual tables pointing to big data platforms also solves the problem of interference, as the queries are no longer executed on the platform, but on the SQL server. Caching virtual tables is especially useful for frequently recurring queries.

Cached tables are managed by the data virtualization server. For example, when the definition of the virtual table is changed, the contents of the cached tables is automatically refreshed.

A drawback of accessing cached data is that it is not real-time data. The refresh rate of the cached tables determines how high the data latency will be. The lower the refresh rate, the lower the data latency, and the higher the interference.

## 6 Seamlessly Integrating Big Data with Enterprise Data

Business users may be interested in combining data from a big data store with data stored in transactional systems or data warehouses. For example, a retail company might store all the individual sales transactions from all their stores in a big data platform that can handle a massive real-time, ingest workload, and the data on customers, products, and stores is stored in a more traditional data warehouse. In this case, business users need access to both environments.

*Data virtualization simplifies integration of big data with enterprise data.*

Data virtualization simplifies integration of big data with enterprise data stored in data warehouses or transactional databases developed with traditional database technology. In fact, the entire set of data sources, including the big data stores, can be presented to developers as one integrated database. For example, historical data (stored in a SQL-based data warehouse) can be integrated with streaming data stored in Hadoop, and descriptive data stored in data marts can be integrated with sensor data kept in a NoSQL database. In any of these cases, developers will not even know that some data is and some is not stored in the big data platforms.

## 7 Securing Big Data

In most, if not all environments, it is unlikely that every data consumer should have access to all the data. The aspect of security called a*uthorization* deals with defining who is allowed to do what with which data elements. Unfortunately, not all the technologies used to store big data support sophisticated and fine-grained features for securing access to data. Some only allow administrators to specify whether someone is allowed to use an entire file. Especially if the data is stored in files, security features can be minimal.

By forcing data consumers to access big data through a data virtualization server, detailed and fine-grained security rules can be defined and enforced. Access privileges can be defined on the table, column, record, and individual value level, and user groups and roles can be defined.

If data consumers are only allowed to see aggregated or derived data, they are granted access only to virtual tables that contain aggregated data. Such virtual tables can even be extended with logic that removes aggregations that are not sufficiently hiding individual objects. Imagine that all the sales transactions are aggregated by city and one city contains only one customer. By looking at the aggregated virtual contents of the virtual table, it is clear to everyone who that customer is, especially if users also have access to the customer data. Logic can be added to the virtual tables that excludes all cities that do not have enough customers.

To comply with specific privacy regulations, such as HIPAA and GDPR, masking rules can be defined to hide personally identifiable information. For example, some users are not allowed to see the credit card information of customers, while others are only allowed to see the last four digits.

# 8 Supplying Meta Data for Big Data

**The Importance of Metadata** – Data consumers need access to *metadata*. Business users need metadata that explains what data means, data scientists need metadata to understand how to interpret the data they work with, and developers need access to metadata for similar reasons. Data without metadata is useless.

Numerous tools are available that can organize metadata and make it available to data consumers. However, most of these tools are isolated systems. For data consumers, this means that they need to access a separate system to find out what the data means and to locate the right tables. It is always a challenge to keep the real systems that provide the data consistent with the metadata system. It is easy to forget to update the metadata system when the real system needs an update urgently.

**Enriching Big Data with Metadata** – Data virtualization servers can be used to define and describe the data accessible through virtual tables. A definition and description of each virtual table and column can be added. Metadata is stored in the data virtualization's own repository; see Figure 7. Because the metadata is linked to the virtual tables, when one is deleted, the metadata is deleted as well, keeping the virtual tables and their metadata consistent.

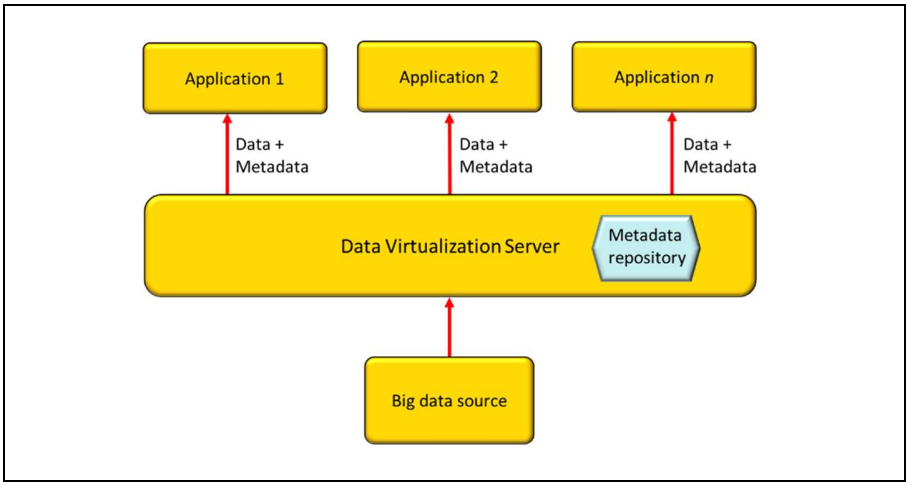


**Figure 7** *Besides delivering data, data virtualization servers can deliver metadata.*

Besides definitions and descriptions, a data virtualization server can also present a data model of all the data (see Figure 8) and a lineage diagram showing the relationships between all the virtual tables and data sources (see Figure 9).

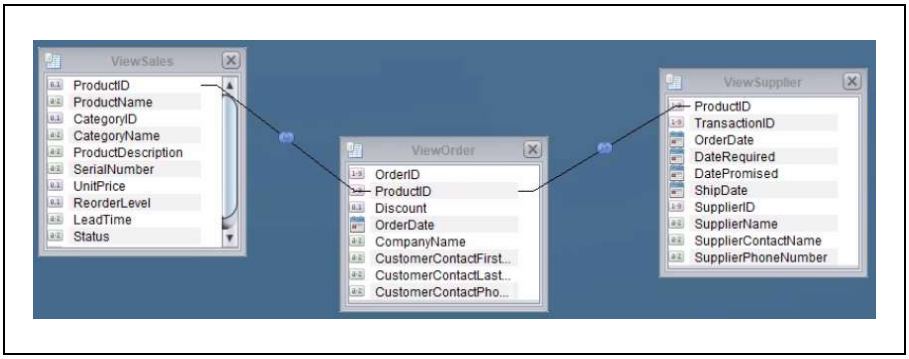


**Figure 8** *Data virtualization servers can show the data model.*

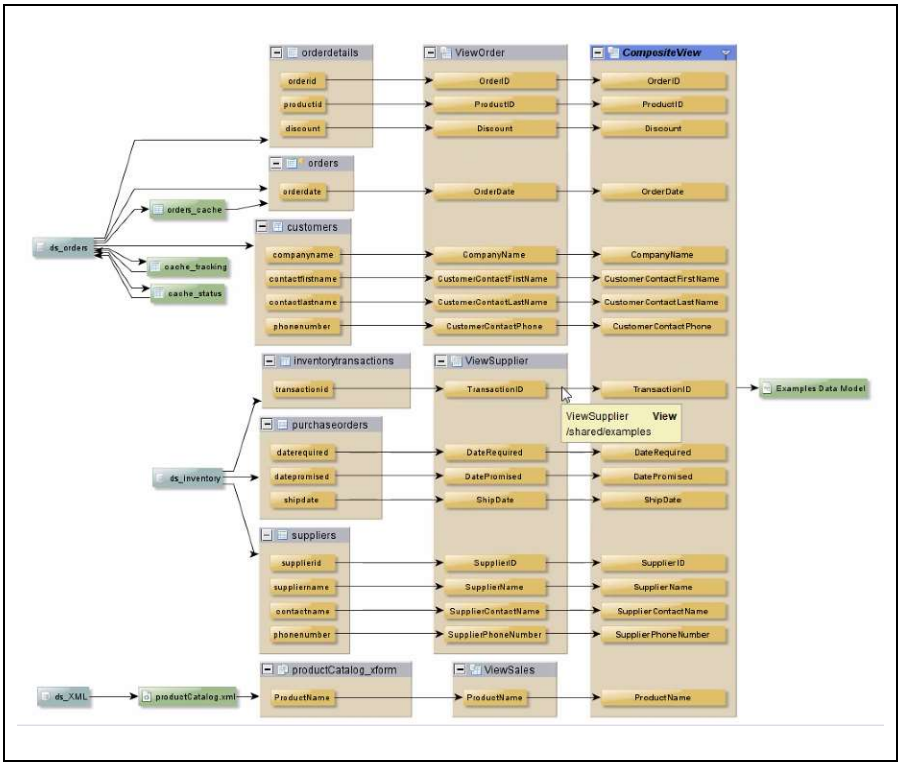


**Figure 9** *Data virtualization servers can present a lineage diagram that shows the relationships between virtual tables and their sources.*

All metadata is available to the data consumers. It can be used by users to understand the data they are looking at and it can also be used by developers and data scientists and other investigative users to look for the right virtual tables to create their dashboards or data science models. The metadata can also be accessed through an API.

Data virtualization servers make metadata accessible and just as frictionless as data itself.

# 9 Parallel Processing on Big Data Platforms

**Big Data Demands Parallel Processing** – Big data platforms support several techniques to accelerate requests on large data sets, an important one is *parallel processing*. When all the data of a file is stored on one disk and only one process can access the file, access to that data is *serial*. Two records from that same file are never processed simultaneously, only serially. Serial processing of real big data sets takes too long. The first technique to improve the performance of requests on big data was introduced in database servers years ago and is called *file* or *data partitioning* and is the foundation for *parallel processing*. This technique has also been implemented in big data platforms.

Parallel processing involves data partitioning and an internal architecture that consists of one *master* (or a few) and several *worker* processes; see Figure 10. Application requests are received by the master process. The master knows that the data is partitioned and divides the request into a set of *subrequests*. These subrequests are transmitted to the workers on the nodes that contain the data. Next, each worker processes its own partition of the data. In other words, the master starts multiple processes to process the partitions in *parallel*. Finished workers return the intermediate results to the master. Lastly, the master combines all the results of the workers and returns the final result to the application. Note that applications are not aware of this parallel processing.

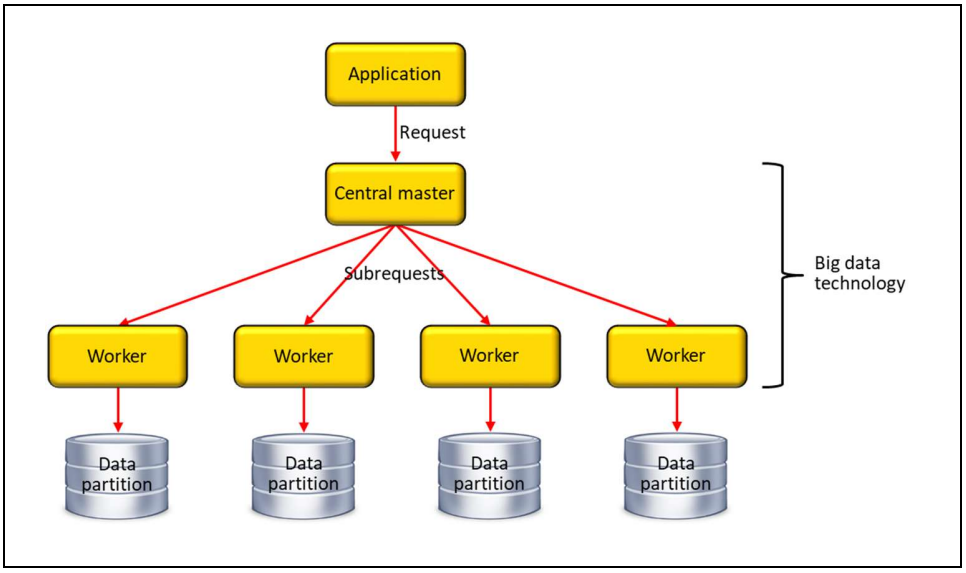


**Figure 10** *Big data technology is able to process requests in parallel by dividing them in subrequests and sending them to the workers for parallel processing.*

**Massive Parallel Processing** – For a long time, data partitioning and parallel processing features provided sufficient performance improvements, until big data came along. The sheer volume of big data alone requires a level of parallelization higher than usual. Unfortunately, most database servers could only efficiently parallelize processing across a limited number of nodes. The Hadoop platform and NoSQL products, on the other hand, were specifically designed to support *massive parallel processing*. They can distribute data and processing across hundreds of disks and nodes and allow a similar number of workers to run in parallel. Their internal architectures make these systems *big data-ready*.

Undoubtedly, more techniques will be implemented in big data platforms in the coming years to achieve even higher levels of parallelization and support even bigger workloads.

**Parallel Processing and Data Virtualization** – The risk of accessing a big data platform using a data virtualization server is that the latter will act as a funnel in which most of the work is still done serially, not in parallel. It is as if the data virtualization server acts like a single toll booth on a six-lane highway and all cars must join one lane to get through that single toll booth.

To prevent becoming a bottleneck, a data virtualization server, such as the TIBCO® Data Virtualization Server, also supports parallel processing. It offers an internal architecture similar to the one in Figure 11. Incoming requests are studied by the master process of the data virtualization server which decides whether queries can be executed in parallel. If so, the requests are transformed into subrequests that are executed by several data virtualization worker processes. In this way, the processing of operations is pushed down to the data virtualization workers, which are then responsible for pushing down as much of the processing as possible to the big data platform.

*A data virtualization server, such as TIBCO® Data Virtualization Server, is big data-ready.*

In other words, TIBCO Data Virtualization software exploits the full parallel processing capabilities of the underlying big data platform and is not a bottleneck.

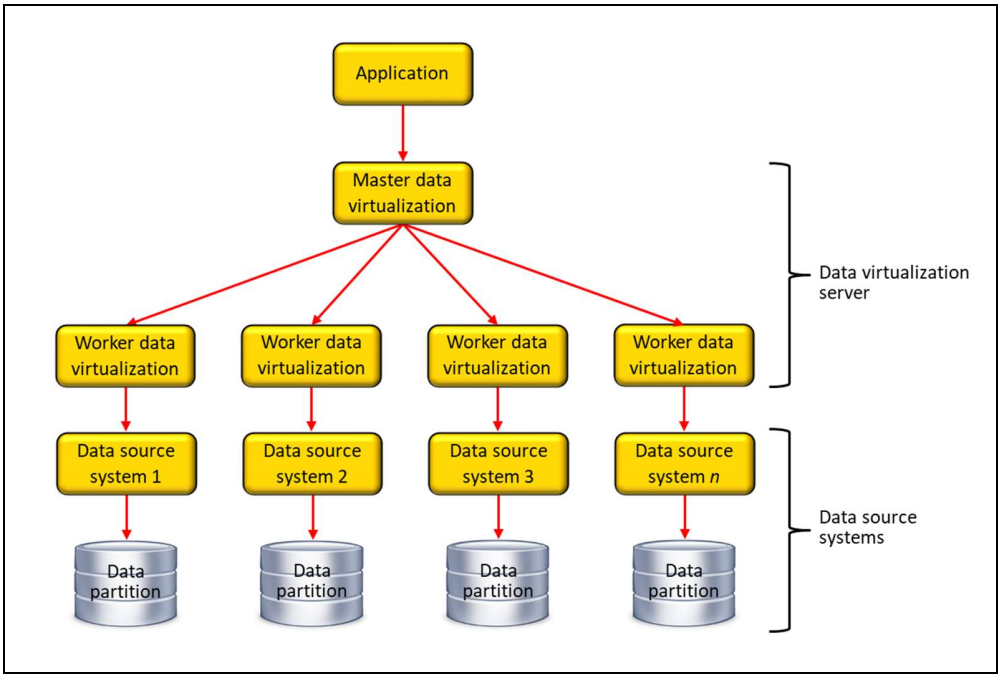


**Figure 11** *TIBCO Data Virtualization also supports parallel processing.*

## 10 Developing Virtual Data Lakes for Data Scientists

**Data Scientists and Raw Data** – Data scientists, statisticians, quants, and other investigative users often claim that they need access to raw, original, non-processed data. Therefore, data is copied to data lakes without any form of processing. If an Extract-Transform-Load (ETL) tool is used to copy data to a data lake, only the letters E and L are used. The consequence is that data scientists still need to develop most of the processing logic themselves to transform the raw data to more consumable data. In other words, they are responsible for programming the letter T of the word ETL, which involves much data engineering work.

Characteristically, data scientists have sufficient technical knowhow to develop the letter T. They know how to unravel and process data stored in their original formats, and their powerful analytical tools support features to transform such files to data structures that are easy to analyze. They know how to cleanse, transform, and integrate data. But not all data scientists are technically savvy. Some of them may not be familiar with these raw native formats and how to process them, and they probably are not interested in performing all that data engineering work. Still, these data scientists need access to data lakes.

**The Zones of a Data Lake** – More recently, a data lake architecture consisting of zones has been proposed; see Figure 12. In such a data lake, the landing zone contains the raw data, which compares to the original data lake in which the data has not been processed and stored in a native format. The second zone, the *curated zone*, contains copied data from the landing zone. During copying, data is slightly processed, cleansed, potentially integrated, and so on. Next, the data is copied to the *production zone*. At each step, the data is slightly processed. It is like an assembly line where with each step the data becomes a more consumable product. In this proposed architecture each zone is still a physical data storage layer.

In this architecture, different user groups can access different landing zones. Many data scientists may use the data in the landing zone to have access to the raw data, while others may use other zones, depending on their information needs.

The drawbacks of this architecture all relate to the copying of data, such as ETL programs that copy the data from one zone to another must be developed, maintained, managed, and scheduled, and all the copies of the data must be secured. It is a very data storage intensive architecture.

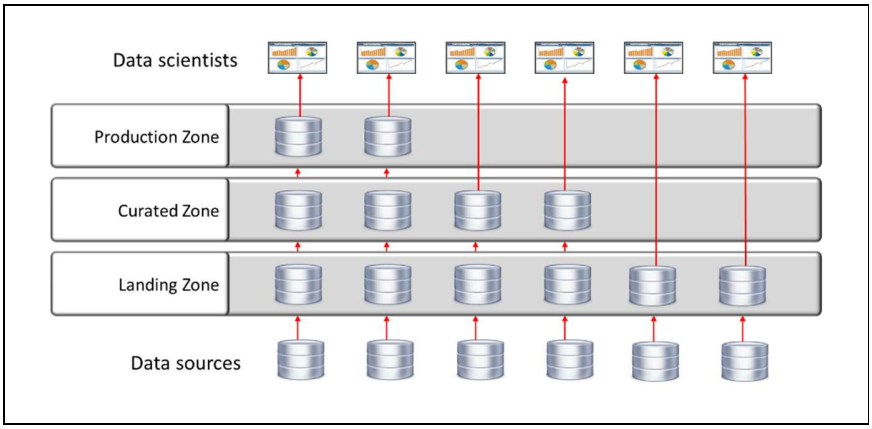


**Figure 12** *A data lake architecture consisting of three zones and where different user groups can access different data lake zones.*

Note: In the literature, alternative names for these zones are used, such as *bronze zone* and *raw tier* for the landing zone; *refinery zone, silver zone,* and *work tier* for the curated zone; and *lagoon, gold tier,* and *access tier* for the production zone.

**The Logical Data Lake** – An alternative data lake architecture that offers the same benefits as the zoned data lake architecture but without all the copying of data, is the *logical data lake*. Figure 13 contains a high-level overview of this architecture.

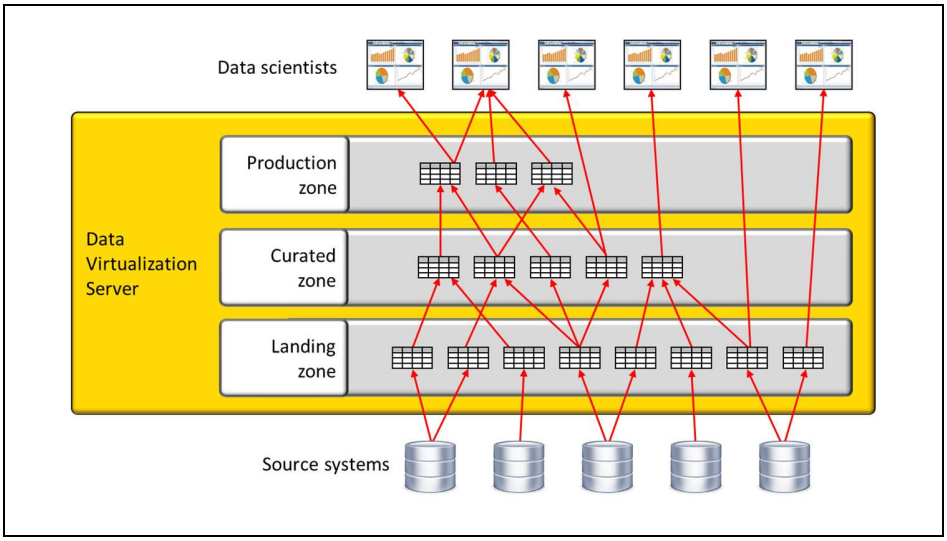


**Figure 13** *The logical data lake in which the zones are defined as layers with virtual tables.*

In logical data lakes, data scientists have access to all the data through a data virtualization server. The latter presents all the data to the data scientists as if it were stored centrally in one data storage platform. However, some of the data is indeed copied and stored centrally (the two data sources on the left), but some data is accessed remotely (the two data sources in the middle), and some is cached locally (the two on the right). Depending on what is required, possible, and feasible, one of the three approaches can be used to make data accessible for data

*Logical data lakes developed with data virtualization servers are more flexible than physical data lakes.*

scientists. Where copying data and storing it centrally is just one of the options in the original data lake architecture, it is not the only option in the logical data lake.

In the logical data lake architecture, the zones can be simulated as well by defining levels of virtual tables. Each level represents a zone. This architecture offers all the advantages of the data lake consisting of physical zones but without the disadvantages.

## 11 Fusing Multiple, Independent Data Lakes

Currently, a modern *data lake* is no longer a single data platform, but a heterogeneous and distributed set of (big) data platforms, each using a different data storage technology, such as an on-premise Apache HDFS-based cluster, a Teradata server, an Amazon S3 cluster in the cloud, and SnowflakeDB. Each platform may support a different language, interface, or SQL dialect. There can be organizational, regulatory, and technical reasons for developing multiple data lakes. Whatever the reason, these data platforms must be integrated together to present to data scientists and other investigative users an integrated view of all the data required for analytics to form a *fused data lake.*

Fused data lakes can be implemented using data virtualization. An integrated view of all the data can be presented easily. The distributed query optimizer optimizes the queries in which data from multiple data lakes is combined. Virtual tables that span multiple data lakes can be temporarily cached if certain data scientists need to access that virtual table frequently. The cached table can be stored locally to speed up performance.

A detailed description of how a fused data lake is developed using data virtualization can be found in the whitepaper “The Fusion of Distributed Data Lakes: Developing Modern Data Lakes”.[1]

## 12 Monitoring Big Data Usage

Data virtualization servers monitor all access to data. Each processed query is documented: when the query was executed, the execution time, the user who executed the query, the virtual tables accessed, and so on. This monitoring data can be analyzed providing administrators with answers to questions, such as:

- What is the frequency of queries?
- What is the frequency of use for each virtual table?
- Which queries are slow?
- Which queries are resource-intensive?

All these insights can help administrators to determine whether certain improvements or optimizations are needed. For example, can the caching of a virtual table be dropped, or should it be added, are some users constantly executing slow queries that need to be optimized, which virtual tables can be removed because they have not been used in recent months. Figure 14 contains a screenshot of some of the data being monitored.

[1] R.F. van der Lans, *The Fusion of Distributed Data Lakes: Developing Modern Data Lakes*, February 2019; see https://www.tibco.com/resources/whitepaper/fusion-distributed-data-lakes-developing-modern-data-lakes

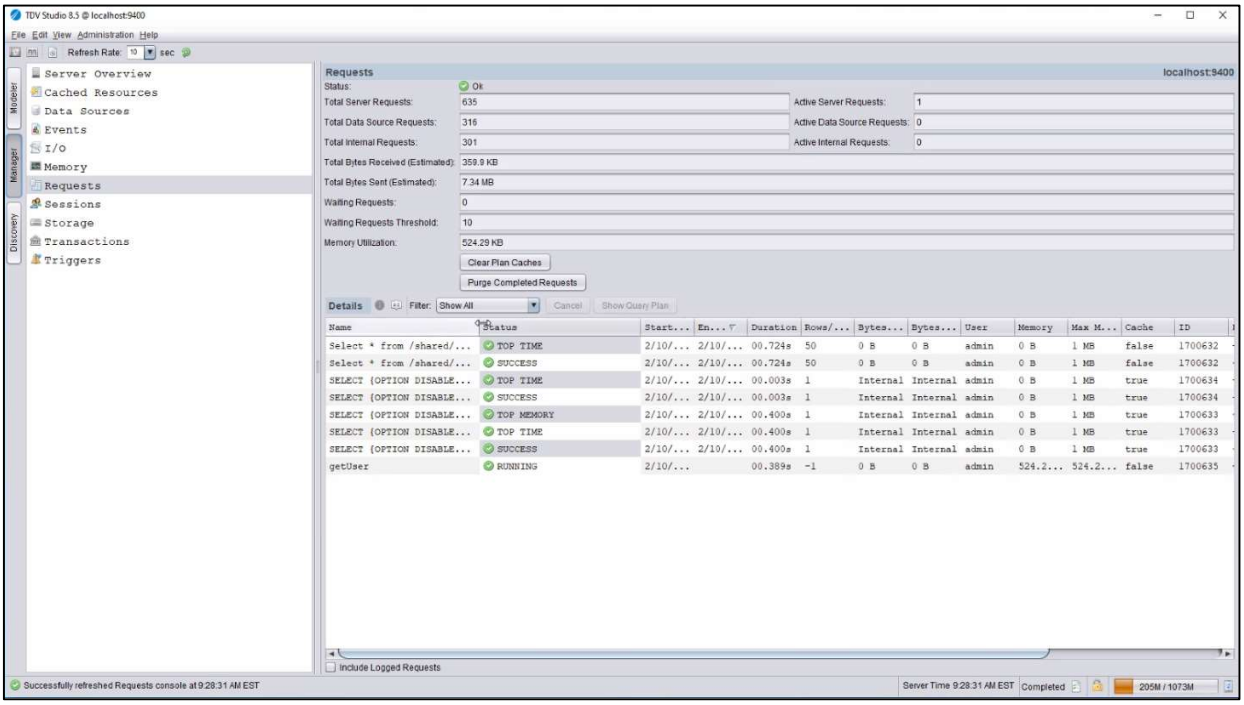


**Figure 14** *A screenshot showing some of the monitoring data created by a data virtualization server.*

This type of monitoring functionality is not provided by all the big data platforms. By accessing the big data platform through a data virtualization server, data usage is automatically monitored in detail and can be analyzed like any other type of data.

# 13 Closing Remarks

*Data virtualization extends the capabilities of big data platforms.*

Organizations are developing and operating big data systems. Many have selected non-SQL platforms to store big data. Due to these non-SQL interfaces, it can be difficult to access that big data easily. Also, because many platforms are developed to support a specific use case, its data cannot be deployed in other use cases and the data must be copied to another platform. Analytics on big data, the main reason organizations invest in it, can be particularly slow or is not supported.

Data virtualization extends the capabilities of big data platforms in the following ways:

- Enabling the use of the familiar SQL language on platforms that do not natively support SQL.
- Centrally transforming structure-less data (schema-on-read) to structure-rich data.
- Transforming a polyglot-persistent environment to an integrated monoglot environment.
- Enabling access to remote data if copying to a central environment is not an option.
- Accelerating queries on data platforms not optimized for analytics through caching.
- Seamlessly integrating big data with enterprise data.
- Extending the capabilities regarding data security and data privacy.
- Supplying descriptive metadata and lineage to all users.
- Parallel query processing on massive parallel platforms.

- Developing logical data lakes to shorten the data preparation phase of data science projects.
- Fusing multiple, independent data lakes into a single logical data lake.
- Monitoring data usage on big data platforms.

*Data virtualization can unlock big data for the entire organization.*

Big data only becomes valuable when it is used, especially when it is used for analytical purposes. This means that the more data consumers have access to it, the more it can be exploited. But that means that data should not just be available to IT developers, who are tech-savvy enough to understand their languages and APIs. Big data should be easily accessible to all data consumers, not just the lucky few. This is what data virtualization entails. In many ways, data virtualization can unlock big data for the entire organization and increases the ROI of big data projects

## About the Author Rick F. van der Lans

Rick van der Lans is a highly respected independent analyst, consultant, author, and internationally acclaimed lecturer specializing in data architecture, data warehousing, business intelligence, big data, database technology, and data virtualization. He works for R20/Consultancy (www.r20.nl), which he founded in 1987. In 2018, he was selected the sixth most influential BI analyst worldwide by onalytica.com[2]. He has presented countless seminars, webinars, and keynotes at industry-leading conferences.

Rick helps clients worldwide to design their data warehouse, big data, and business intelligence architectures and solutions and assists them with selecting the right products. He has been influential in introducing the new logical data warehouse architecture worldwide which helps organizations to develop more agile business intelligence systems. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in several articles[3], all published at B-eye-Network.com.

He is the author of several books on computing, including his new *Data Virtualization: Selected Writings*[4] and *Data Virtualization for Business Intelligence Systems*[5]. Some of these books are available in different languages. Books such as the popular *Introduction to SQL* are available in English, Dutch, Italian, Chinese, and German and are sold worldwide. Over the years, he has authored hundreds of articles and blogs for newspapers and websites and has authored many educational and popular white papers for a long list of vendors. He was the author of the first available book on SQL[6], entitled *Introduction to SQL*, which has been translated into several languages with more than 100,000 copies sold.

For more information, please visit www.r20.nl, or send an email to rick@r20.nl. You can also get in touch with him via LinkedIn and Twitter (@Rick_vanderlans).

**Ambassador of Axians Business Analytics Laren:** This consultancy company specializes in business intelligence, data management, big data, data warehousing, data virtualization, and analytics. In this part-time role, Rick works closely together with the consultants in many projects. Their joint experiences and insights are shared in seminars, webinars, blogs, and whitepapers.

## About TIBCO Software Inc.

TIBCO Software unlocks the potential of real-time data for making faster, smarter decisions. Our Connected Intelligence Platform seamlessly connects any application or data source; intelligently unifies data for greater access, trust, and control; and confidently predicts outcomes in real time and at scale. Learn how solutions to our customers' most critical business challenges are made possible by TIBCO at www.tibco.com.

TIBCO and the TIBCO logo are trademarks or registered trademarks of TIBCO Software Inc. or its subsidiaries in the United States and/or other countries.

---

[2] Onalytica.com, *Business Intelligence – Top Influencers, Brands and Publications*, June 2018; see http://www.onalytica.com/blog/posts/business-intelligence-top-influencers-brands-publications/

[3] See http://www.b-eye-network.com/channels/5087/view/12495

[4] R.F. van der Lans, *Data Virtualization: Selected Writings*, Lulu.com, September 2019; see http://www.r20.nl/DataVirtualizationBook.htm

[5] R.F. van der Lans, *Data Virtualization for Business Intelligence Systems*, Morgan Kaufmann Publishers, 2012; see https://www.r20.nl/DataVirtualization_V1.htm

[6] R.F. van der Lans, *Introduction to SQL; Mastering the Relational Database Language*, fourth edition, Addison-Wesley, 2007.