The Fusion of Distributed Data Lakes

Developing Modern Data Lakes

A Technical Whitepaper

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

February 2019

Sponsored by

About TIBCO
TIBCO fuels digital business by enabling better decisions and faster, smarter actions through the TIBCO Connected Intelligence Cloud. From APIs and systems to devices and people, we interconnect everything, capture data in real time wherever it is, and augment the intelligence of your business through analytical insights. Thousands of customers around the globe rely on us to build compelling experiences, energize operations, and propel innovation. Learn how TIBCO makes digital smarter at www.tibco.com.

Table of Contents

1 Introduction

The Data Lake – The data lake concept was introduced a few years ago and has been popular from its inception. Data lakes are introduced to assist with investigating, exploring, experimenting, and refining data, in addition with archiving data. Most data lakes are developed to ease the work of data scientists when developing analytical models.

Data Lakes Have Changed – Although the concept is still relatively new, the nature of the data lake has already changed since its inception. In fact, it has undergone three major changes:

- **From a single-purpose data lake to a multi-purpose data lake:** Currently, the data lake is more and more used as a data repository for all the source data in its original form and format to be used for multiple purposes, not only data science anymore.
- **From a centralized data lake to a distributed data lake:** The intention was for an organization to have one central data lake. Currently, numerous organizations have developed multiple data lakes instead.
- **From a complete solution to a partial solution:** Because the data lakes don't contain all the data, data scientists have to combine data residing in data lakes with data residing in other data sources such as data warehouses.

The Fused Data Lake – Currently, a modern data lake is not a single data platform anymore, but a heterogeneous and distributed set of data platforms. 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 users an integrated view of all the data required for analytics to form a *fused data lake.* Generally, there are three different solutions to fuse data residing in multiple data lakes:

A modern data lake is a fusion of multiple data lakes.

- Integration by data science tool
- Integration by data replication
- Integration by data virtualization

The Whitepaper – This whitepaper describes in detail these three solutions and compares them, although the focus is on the third one: data integration by data virtualization. It explains how data virtualization technology can be used to fuse a modern data lake environment consisting of a set of distributed data lakes. Data virtualization simplifies access to such a new architecture and accelerates the performance of analytics allowing data scientists to study more alternative analytical models.

Using data virtualization to develop a fused data lake.

2 A Modern Data Lake is a Distributed Data Platform

The Evolution of the Data Lake – The *data lake* is the new kid on the block in the world of data warehouses, data marts, and data staging areas. It is a relatively new repository of data. One of the first popular definitions of data lake comes from James Serra[1]: "A data lake is a storage repository [...] that holds a vast amount of raw data in its native format until it is needed." In that same article Serra continues with describing the usage of the data lake: "It's a great place for investigating, exploring, experimenting, and refining data, in addition to archiving data." In other words, data lakes are developed for data scientists and other investigative users. Initially, without data lakes, the data that these users needed was copied from their sources to a somewhat non-organized environment. These environments are developed for the data scientists to run their analytics in an isolated environment. A data lake should streamline and manage this process of copying, storing, and managing the data for analytical purposes.

Currently, most data lakes are developed with data platforms such as an Hadoop cluster or an analytical SQL database server, such as Google BigQuery or SnowflakeDB.

Although the concept is still relatively new, the nature of the data lake has already changed since its inception. In fact, it has undergone three major changes. First, the target audience of the data lake changed. Initially, the data lake was meant for data scientists to store all the data they need for their search for patterns, trends, and models. Currently, the data lake is more and more used as a data repository for all the source data in its original form and format to be used for multiple purposes, not only data science anymore. The effect was that the data lake changed from a *single-purpose data lake* to a *multi-purpose data lake*.

Change 1: From a single-purpose to a multi-purpose data lake.

From Centralized to Distributed Data Lakes — The second change the data lake underwent relates to its centralization. The intention was for an organization to have one central data lake. Currently, numerous organizations have developed multiple data lakes instead. For example, they may have one Hadoop cluster running on premise and another one in the Microsoft Azure cloud. Or, they may have two geographically distributed data lakes to support different parts of the organization. They may even have separate data lakes for separate divisions of the organization.

Change 2: From a centralized to a distributed data lake.

Reasons Why Multiple Data Lakes are Developed — There can be organizational, regulatory, and technical reasons for developing multiple data lakes:

- **Different use cases:** Data lakes may have been developed for different use cases. For example, one for data scientists, another one for offloading cold data, and the third one purely for cheap, alternate data storage.
- **Decentralized organization:** Organizations may have a decentralized character leading to separate data lakes for different divisions or departments.
- **Acquisition:** Through the acquisition of another organization another data lake may enter the organization. This data lake is possibly developed with different technology and on a different technical platform.
- **Geographically distributed users:** Organizations may have their data scientists spread across the globe. Having one centralized data lake may result in unacceptable network delays when analyzing the data. A physically decentralized data lake may be the answer.

[1] J. Serra, *What is a Data Lake?*, April 2015; see http://www.jamesserra.com/archive/2015/04/what-is-a-data-lake

- **Platform migration:** An organization may be in the middle of a migration from one platform to another. In the meantime, the data may partly reside in the new and partly still in the old platform.
- **Workload distribution:** Especially when a multipurpose data lake is developed, the query and analytical workload may be too much to handle for one centralized data lake. Splitting them in multiple, possibly overlapping data lakes allows for spreading the workload.
- **Prohibiting regulations:** Regulations may be the reason for developing multiple data lakes. For example, in some industries regulations prohibit the storage of certain types of data together in one data platform, and in some countries regulations prohibit data to leave the country.

In other words, for a variety of reasons the data lake concept changed from a *centralized data lake* to a *distributed data lake*.

Change 3: Extending data lakes with data warehouses.

Combining Data Lakes with Data Warehouses – The third change relates to the data stored in data warehouses and data marts that can be very valuable to analytics. Commonly, the data in these data stores is cleansed and processed and, most importantly, they contain historic data missing in many operational systems and also missing in data lakes. It may be this historic data that data scientists need to find certain patterns to develop their forecasting models. Therefore, data scientists have to combine data from data lakes with data residing in data warehouses.

A Modern Data Lake is a Fused Data Lake – Conclusion, a *modern data lake* is not a single data platform anymore, but a heterogeneous and distributed set of data platforms. These data platforms must be integrated to present to data scientists and other users a fused view of all the data required for analytics to form a *fused data lake*; see Figure 1.

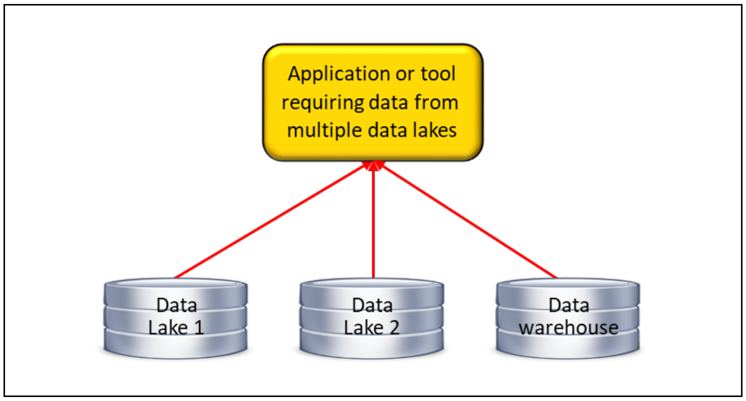

Figure 1 *A modern data lake allows data scientists to access a heterogeneous and distributed set of data platforms that is integrated to form a fused data lake.*

3 Three Alternative Solutions for Developing Fused Data Lakes

Data Lakes for Simplifying Data Access – As indicated, initially data lakes were introduced to have one data storage environment that would ease data access and analytics. The distributed nature of a modern data lake diminishes this advantage somewhat, because it makes it harder for data scientists to locate and retrieve all the data they need, because they have to work with multiple data lakes. Because the data resides in multiple data platforms somehow it must be integrated. All those data platforms must be fused together to offer data scientists easy data access and data usage.

Three Solutions for Fusing Data Lakes – Generally, there are three different solutions to fuse data residing in multiple data lakes and data warehouses:

- Integration by data science tool
- Integration by data replication
- Integration by data virtualization

These three solutions can be compared on various aspects, such as productivity, reusability of specifications, data replication, data latency, and performance. Such a comparison is included in Section 5. This section focuses on two key comparison aspects. Both aspects are related to the so-called *transformation specifications*. Examples of transformation specifications are extract, filter, aggregate, and integrate data. These specifications are required to get the right data at the right aggregation level out of the data platforms and to fuse it in the right way. Such specifications may be defined using proprietary languages offered by ETL tools, data preparation tools, data science tools, or in SQL, or in more traditional programming languages, such as Java, R, and Python.

Transformation specifications are required to integrate data from multiple data lakes.

The three solutions differ in (1) where the transformation specifications are defined and kept, and in (2) where the processing of the specifications take place. All the other comparison aspects are directly or indirectly dependent on these two. For example, processing the specifications as close to the where the data is stored commonly improves performance, and defining specifications centrally allows for a higher level of reuse.

Solution 1: Data Lake Integration by Data Science Tool – Most tools used by data scientists have the ability to extract and integrate data from multiple data platforms. The analytical and statistical tools may support this functionality, or a data preparation or wrangling tool that data scientists use may support this feature. Most of these tools apply an approach that involves pulling all the data from the required data platforms and integrating them “inside” the tool; see Figure 2. In this figure the column on the left indicates where the transformation specifications are defined. The number of black dots represents the split of the transformation specifications. In this architecture all the specifications are defined within the data science tool. In the column on the right black dots with a similar meaning are used to indicate where relatively most of the specifications are executed and processed. In this architecture most of the processing is done by the data science tools and some by the lake platforms themselves.

There are two key drawbacks of this solution. First, specifications are defined with a specific tool. Those specifications can, therefore, not be reused by other tools. This negatively impacts productivity and maintenance. Second, the data processing power of the data platforms themselves are barely used and this negatively influences performance.

Data replication to fuse data lakes leads to extra data stores.

Solution 2: Data Lake Integration by Data Replication – Another solution for integrating data lakes is by copying or replicating the required data from the individual data lakes and data warehouses to a separate data platform, one developed specifically for analyzing the data models; see Figure 3. This approach resembles the classic data warehouse architecture in which data is copied to data marts. This extra data store can be called a *data science mart*. Some of the transformation specifications are defined within the data science tool but most of them are defined in the middle layer of the architecture. Commonly, ETL tools are used to develop this solution. The Extraction & Integration layer does most of the processing.

This data science mart is accessed by the data scientists. This data store is like a temporary and personal data lake that is removed when the data scientists have developed their analytical models.

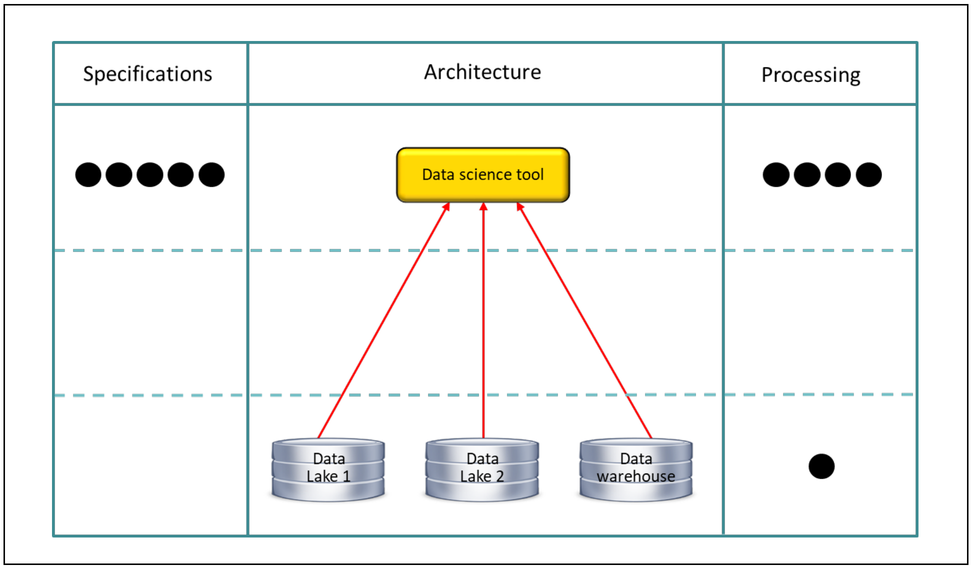

Figure 2 *Fusion of data lakes by the data science tool.*

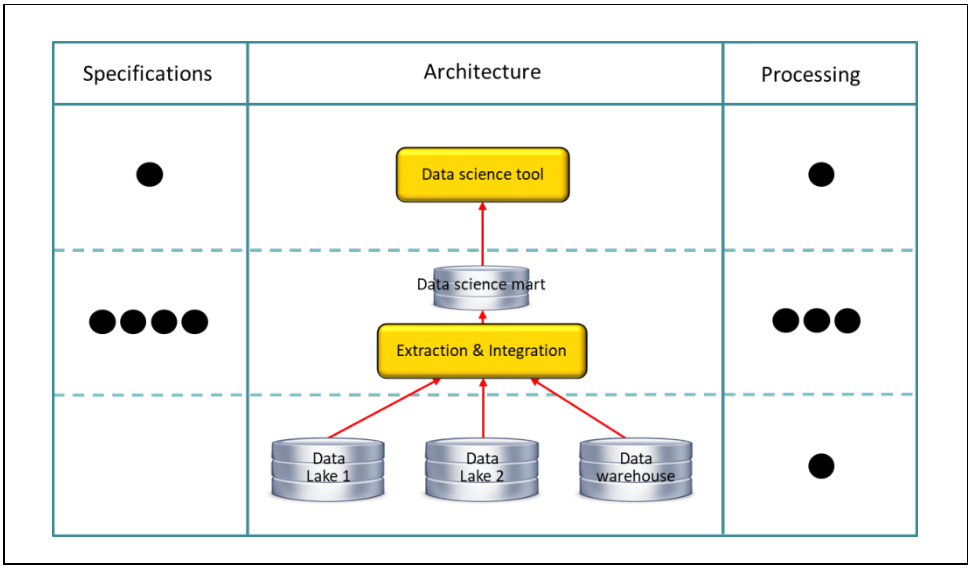

Figure 3 *Fusion of data lakes by data replication.*

The advantages of this solution are, first, that data is processed closer to where it is stored and second, as the specifications are defined in the middle layer, they can be reused by different data science tools. The two key disadvantages of this solution both stem from copying the data. First, due to the copying process the latency of the data accessed may be high. Second, storing the data again lowers the manageability of the entire environment.

Solution 3: Data Lake Integration by Data Virtualization – With data virtualization all the data lakes, data warehouses, and other data sources are presented as one logical database; see Figure 4. Data is *not* physically copied to an extra database for integration purposes. Data is fused by the data virtualization server when the data scientist asks for it.

Data virtualization to fuse data lakes requires no extra data stores.

Data virtualization technology was introduced to shield users from the fact that they are technically accessing multiple data platforms. Data virtualizations servers operate as a software layer on top of multiple data platforms and presenting all the data to the users as one integrated data platform.

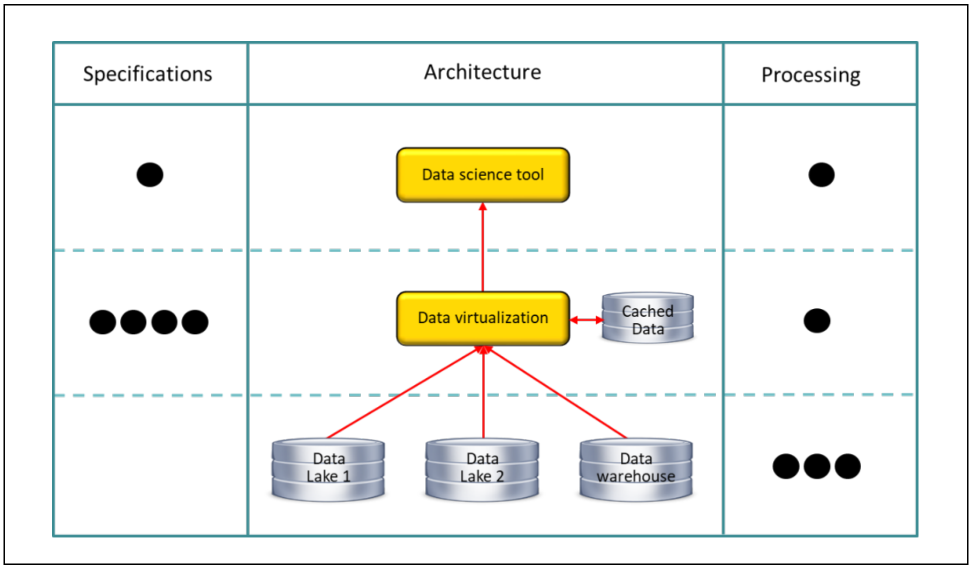

Figure 4 *Fusion of data lakes by data virtualization.*

The advantages of this solution are similar to the advantages of the data replication solution. First, data is processed closer to where it is stored thus leveraging the processing power of those sources and acceleration performance. Second, the transformation specifications can be reused by different data science tools. This solution does not have the disadvantage of replicating the data, meaning data latency can be low and no need exists to manage extra data stores.

In the next two sections many of these advantages and disadvantages are discussed in more detail.

4 Data Virtualization for Fusing Data Lakes

This section focuses on the third solution and specifically on the features offered by data virtualization servers to fuse a set of distributed data lakes and other data platforms. Due to the scope of this whitepaper primarily features are described related to how data is retrieved from data sources. For other features supported by these servers, such as lineage, business glossaries, and data security, we refer to their respective manuals.

4.1 Hiding the Distributed Nature of the Data Lake to Simplify Data Access

As indicated, in a modern data lake environment all the data is distributed across multiple data platforms. Data scientists should not be aware of this distribution of data. They should not have to deal with where which data is stored. One of the key features of data virtualization servers is that they hide the fact that data is distributed across multiple data platforms. To data scientists accessing a data virtualization server feels as accessing one integrated data lake. Network access details, authorization aspects, data security aspects that are all specific to the technologies used by the platforms are all hidden to the data scientists. Logically, the data from all the data lakes is presented as one data lake.

Data virtualization hides the fact that data is distributed across multiple data platforms.

4.2 Hiding Heterogeneous Data Platforms to Simplify Data Access

Data is distributed across multiple data lakes, possibly geographically spread across multiple clouds. However, each data lake may use a different data storage technology, one can be an on-premise Apache HDFS-based cluster, the other one a Teradata server, the third one an Amazon S3 cluster in the cloud, and the fourth one SnowflakeDB. Each platform potentially supports another language and interface or they may speak different dialects of the same language, for example SQL.

Data virtualization servers are developed to access data platforms with different APIs or languages. The data scientist's tool can use a language such as SQL to access all kinds of data storage technologies. The data virtualization servers translate the incoming request into a language supported by the data platform. Therefore, the data scientists don't have to understand all the technical details and peculiarities, APIs, and languages supported by the data platforms. The heterogeneous set of data lakes will look like one homogeneous data lake.

4.3 Query Pushdown to Exploit the Power of the Data Platforms

The Importance of Query Pushdown – The majority of data lakes are developed using Hadoop, Hadoop-like technologies, and analytical SQL database servers. Such technologies allows the query processing to be distributed across multiple nodes and processors. In other words, they support *parallel query processing*. Each technology accessing those platforms should exploit these capabilities. They must be able to delegate most or all of the query processing to the data platforms. Only then will the full power of the data platform be exploited. It would be a waste of the potential power of a data platform, if a solution retrieves all the data from the platform and does all the query processing itself. In other words, query processing must be pushed as close to where the data is stored as possible. This is called the *push-processing-down* or *query-pushdown* approach.

The pull-data-in-memory approach is not always ideal.

The Pull-data-in-memory Approach – As indicated, the worst solution thinkable is the one in which the solution "pulls" all the required data from those data platforms towards its own engine and runs the query "inside" the tool. The sheer amount of data transmission slows down the query enormously, plus the join executed inside the tool to bring together data from the data platforms is also a long winding process. Additionally, all the returned data must be kept in memory before processing can take place. What if the data pulled in is too much to load in memory? This *pull-data-in-memory approach* for running federated queries on multiple big data platforms is not always ideal. Retrieving too much data from the data platforms may cause massive data transmission delays. Data scientists need solutions that more intelligently and efficiently access this heterogeneous and distributed data lake environment.

The Push-processing-down Approach – Data virtualization servers support the *query-pushdown* approach and therefore exploit the full power of the data platforms, including their parallel query processing capabilities.

Query pushdown minimizes data transmission.

Another aspect of query pushdown is related to the amount of data transmitted from the data platform to the data virtualization server. The performance of a query as experienced by data scientists is not only determined by the speed with which the query is executed by the data platform. The performance is equal to the sum of the time needed for database processing, data virtualization processing, data transmission, data analytics, and data visualization by the BI tool; see Figure 5. There are two areas that may involve data transmission. First, when data is transmitted from the data platform to the data virtualization server and, second, when data is transmitted from the data virtualization server to the data science tool. By doing as much query pushdown as possible, less data is transmitted back to the data virtualization server. This minimizes network traffic. Note that when query pushdown is not executed properly, over 80% of the total response time may involve data transmission. For a detailed description of the concept of query pushdown; see the whitepaper "Data Virtualization in the Time of Big Data."[2]

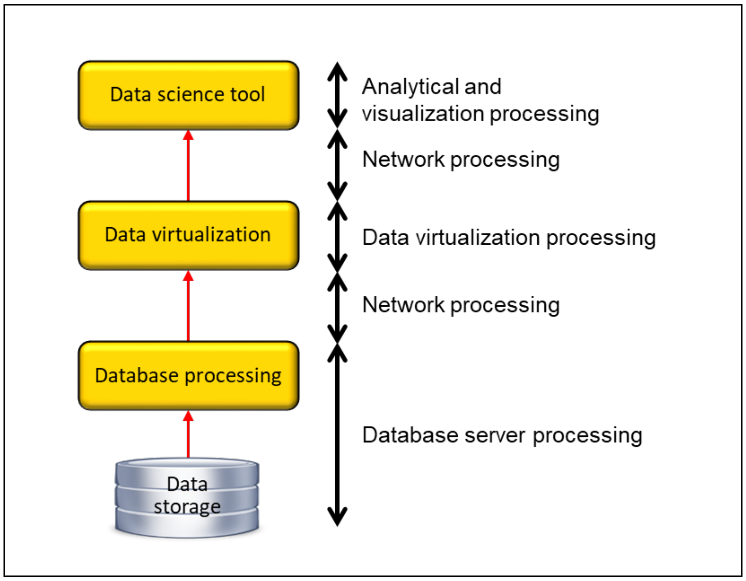

Figure 5 *The performance of a query is equal to the sum of the time needed for database processing, data virtualization processing, data transmission, data analytics, and data visualization.*

[2] R.F. van der Lans, *Data Virtualization in the Time of Big Data*, December 2017; see https://www.tibco.com/resources/whitepaper/data-virtualization-time-big-data

4.4 Parallel Query Processing to Accelerate Big Queries

Data Virtualization and Internal Parallel Processing – If data scientists request data for analytical purposes, it travels a long path; see Figure 6. Data is retrieved from disk, next it is processed by the database server, this result is transmitted to the data virtualization server, which processes the data even further, and finally, the result is transmitted to the data science tool for analytical processing. Every module in this path is a potential performance bottleneck; the performance of the entire path is determined by the slowest module. If possible, all modules making up this path must be able to process data in parallel to exploit *massively parallel hardware architectures* (MMP). The diagram on the left hand side of Figure 6 shows a solution in which the data virtualization server does not support parallelization. It makes this server the potential bottleneck in this solution. To avoid this, data virtualization servers should also be able to distribute their processing across multiple nodes. In other words, they should also support parallel query processing internally in a similar way as database servers support parallel processing to exploit MPP hardware architectures; see the diagram on the right hand side of Figure 6.

Ideally, all modules involved in query processing must support parallel processing.

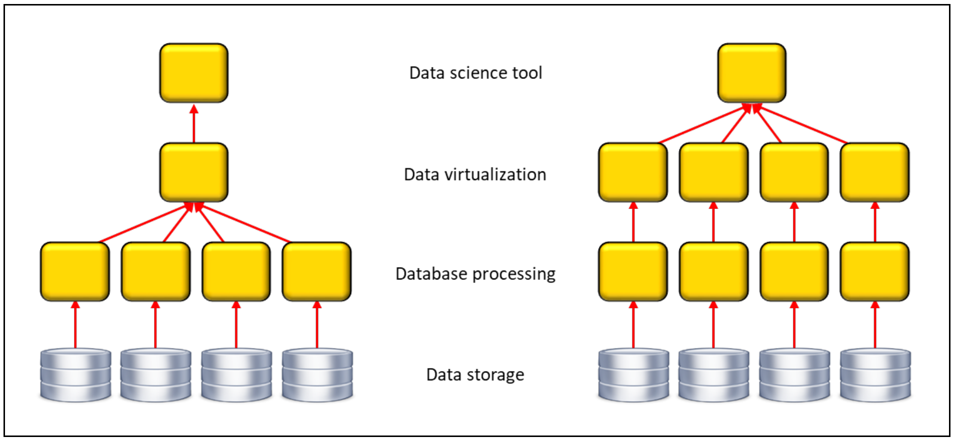

Figure 6 *Data virtualization servers must be able to distribute query processing across multiple nodes to avoid becoming the performance bottleneck in a big data environment.*

Data Virtualization and External Parallel Processing – Data virtualization servers with an architecture as shown on the right hand side of Figure 6 support parallel processing *internally*. An alternative approach to parallelize query processing is what can be called *external parallel query processing*. Here, the data virtualization layer copies the requested data first from one data platform to another that supports parallel query processing and subsequently asks that platform to execute the query; see Figure 7. With external parallel query processing the data is lifted and shifted from the source to the other data platform. Next the data is stored in a partitioned fashion, and finally query pushdown is used and the query is executed in parallel. The query optimizer of such a data virtualization server decides whether this on-demand lift and shift approach makes sense. It has to balance out the extra time needed to lift and shift the data versus the performance improvement due to the parallel processing. After the query is executed the data is removed from the second data platform. This approach works well for ad-hoc queries. For recurring queries another approach is preferred, where the data is not repeatedly lifted and shifted, but where the result is kept for some time and reused several times.

External parallel query processing works well for ad-hoc queries.

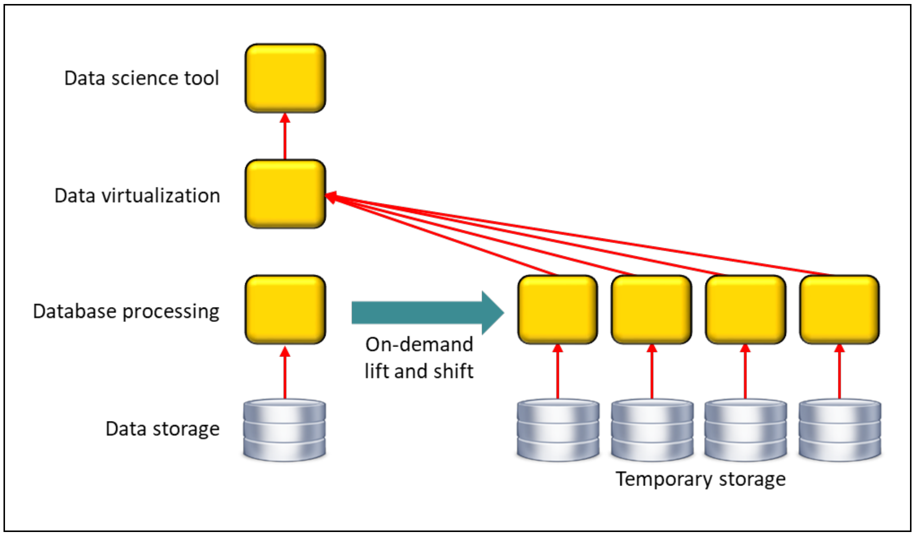

Figure 7 *Data virtualization servers using internally a lift-and-shift approach for the data to exploit parallel query processing.*

4.5 Distributed Query Optimization to Efficiently Access Multiple Data Lakes

Optimizing Federated Queries – Every data platform optimizes incoming queries to ensure that the most efficient execution strategy to retrieve the data is deployed. But when queries span multiple data platforms, the so called *federated queries,* query optimization has to be handled by the module that invokes the data platforms. This module can be, for example, a data science tool or a data virtualization server. In Figure 8 the module that executes federated queries is a data science tool.

The optimizer of the querying module tries to improve query response times. For a federated query this means it must come up with an execution strategy that minimizes the amount of data transmitted from the data platforms. Transmitting as less data as possible from the data platforms minimizes network delay and, furthermore, the less data is received by the querying module, the less extra transformation specifications it has to process. Both influence the response time of queries positively. Optimization techniques, such as injection joins, ship joins, pruning, and pushdown operations, for federated queries are needed to make sure that the amount of data transmission is minimized.

Many data science tools do not support many optimization techniques for federated queries. As indicated, they apply the pull-data-in-memory approach. This may result in transmission of massive amounts of data.

Data Virtualization and Distributed Query Optimization – The ability of a data virtualization server to use parallel processing and query pushdown is crucial here to accelerate the federated queries; see Figure 9. This is and has always been one of the key issues of data virtualization servers. A wide range of query optimization techniques are supported to optimize federated queries.

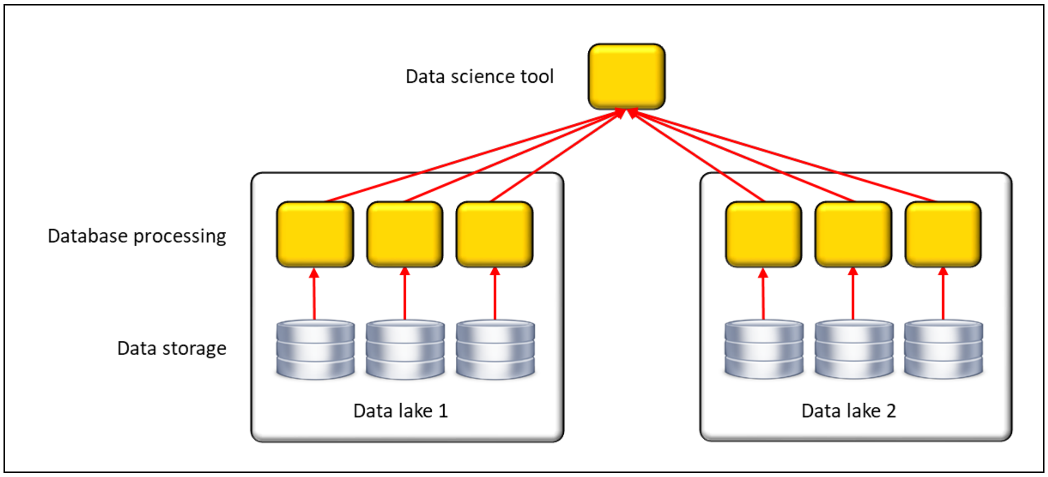

Figure 8 *Pull-data-in-memory is not the most efficient approach for executing federated queries when multiple data lakes are accessed.*

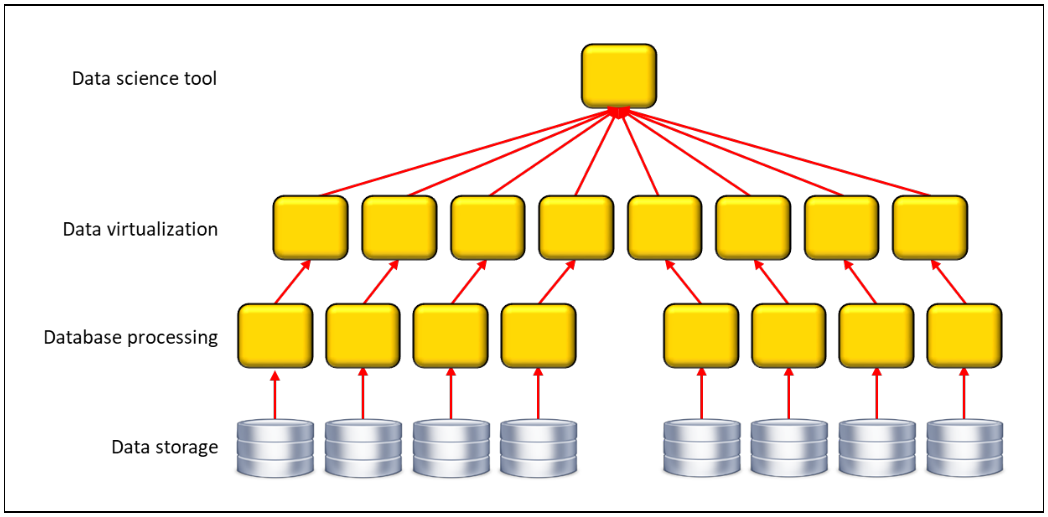

Figure 9 *Data virtualization servers support parallel processing and query pushdown to efficiently execute federated queries on multiple data lakes.*

Again, all this query optimization is hidden from the data scientists. They don't have to know they access multiple data lakes that are fused together by the data virtualization server to form one logical and integrated data lake.

Not All Queries are a Challenge – Not every query executed on a data lake is a federated query optimization challenge, because not every query that accesses multiple big data platforms processes massive amounts of data. It depends on the amount of data that needs to be accessed (the input) and on how much data is being returned by the query (the output); see Figure 10.

The simplest queries are the ones that don't use much data as input and don't generate much output data either. An example of such a query is "Get the address of customer 15." Even if the address data has to come from multiple data platforms, it is still a fast query. These queries are commonly referred to as ***point queries*** and are easy to optimize.

Another category is formed by queries that need to process a lot of data, but their results measured in bytes are small. An example is "Get the average value of a list of numeric values." Even though the data platform has to do a lot of processing, the amount of data returned is very limited and doesn't need much data transmission. These are called ***summarization queries***.

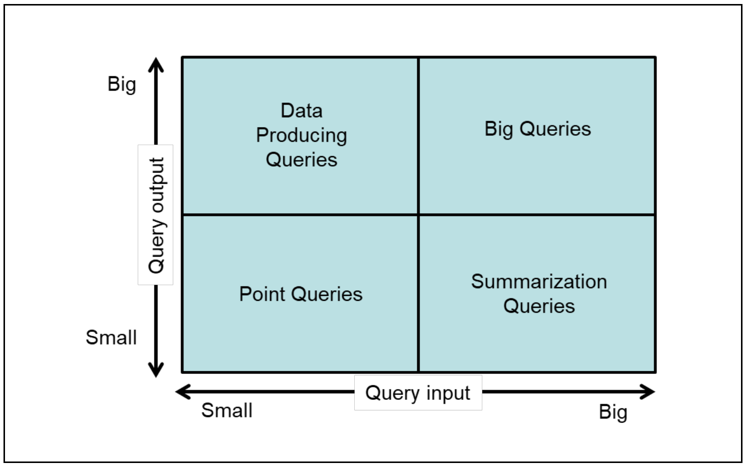

Figure 10 *Queries can be classified based on the amount of data they need to access (the input) and the amount of data they return (the output). Queries executed by data scientists on data lakes are considered to belong to the big query category.*

A rare category of queries are the ones that do not access much data, but return a lot of data. In other words, these are queries with a small input and big output. Somehow these queries generate new data out of existing data. This can be done through, for example, a Cartesian product or a recursive query. The amount of data processed is minimal while data transmission can be massive. As indicated, these ***data producing queries*** are only used sporadically.

The last category are the ones that have a lot of data as input and also produce a lot of data. These are called ***big queries*** and they require considerable data processing and data transmission. Many queries executed by data scientists and other investigative users belong to this category. Big queries and especially the *federated big queries* are the query optimization challenge. The other query categories are easier to optimize.

Big queries have a lot of data as input and as output.

4.6 Data Caching to Temporarily Store Data

Data Caching in a Nutshell – In a data virtualization server *virtual tables* are defined that give access to the source systems. When virtual tables are queried by data scientists, the real data is accessed. Virtual tables have virtual content which is determined when they are queried.

Every virtual table can be *cached*. Caching means that the virtual content of a virtual table is determined and physically stored. From then on queries on that virtual table are processed by running the query on the stored cache. This can speed up queries considerably. Users can determine in which data platform the cached virtual tables are stored. It doesn't have to be one of the source platforms. Caching virtual tables is useful for recurring queries.

Note that caching is invisible to the users of the virtual table. The data virtualization server hides this concept. No code has to be changed. Caches can be removed when they become obsolete.

Caching Data to a Parallel Processing Platform – Not all the data lakes are developed using Hadoop or another technology that supports parallel query processing. Some are developed with more traditional database technologies that support no or minimal parallel processing. If such a database holds massive amounts of data, running big queries can be a performance challenge.

Caching can be used to move data to a fast parallel data platform.

Through data virtualization this can be solved by caching a virtual table defined on the non-parallel technology. The cached data can be stored on a high-end data platform that does support parallel processing; see Figure 11. Queries on that virtual table are not processed by the simple data platform anymore, but by the platform supporting parallel query processing. This lift-and-shift approach of the data can be applied to all the data that the data scientists are interested in.

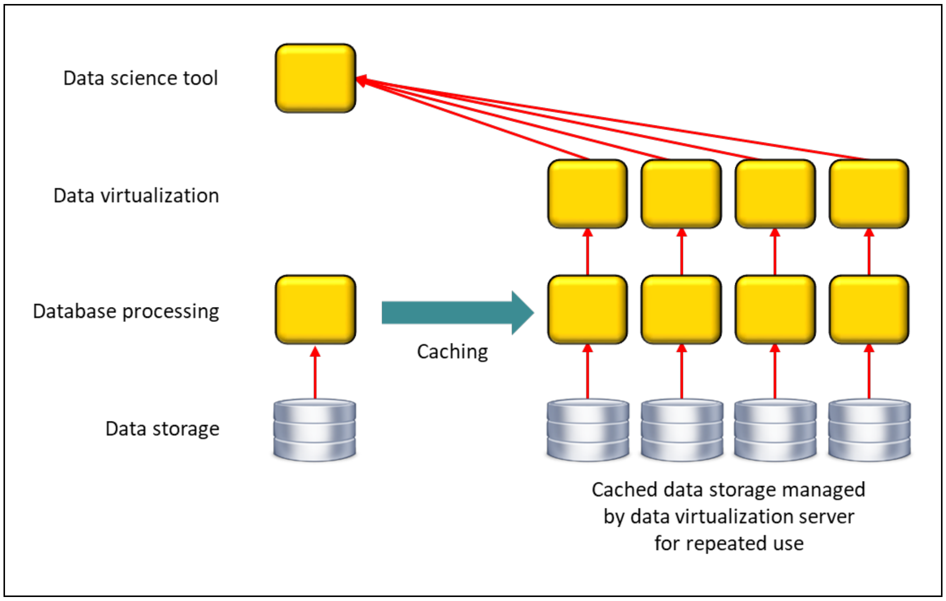
Data science tool
Data virtualization
Database processing
Data storage
Caching
Cached data storage managed by data virtualization server for repeated use

Figure 11 *Data virtualization servers can cache the source data in a data platform supporting parallel query processing*.

Caching Data on a Parallel Processing Platform – Despite all the advanced query optimization and parallelization techniques supported by data platforms, a federated big query can still be slow. In this case it may be useful to define the federation and join part of the query as a virtual table in the data virtualization server. This virtual table is then cached. Subsequently, the rest of the query is executed on the cached virtual table. This speeds up the data scientist’s queries because there is no need to execute the join anymore, nor does data have to be transmitted between data platforms. This approach is useful for recurring queries, but not for ad-hoc queries.

5 Comparison of Three Solutions for Developing Fused Data Lakes

The previous section compares the three solutions for fusing data lakes based on two characteristics: where are the transformation specifications defined and where are they processed. Table 1 contains a comparison of the three solutions on some extra characteristics. Most of these characteristics can be derived from the first two.

	Integration by data science	Integration by data replication	Integration by data virtualization
No additional software required to install and manage	Yes	No	No
No extra data storage	Yes	No	Yes
Development productivity of transformation specifications	Low	High	High
Data easy to understand and to integrate	No	Yes	Yes
Query performance	Low	High	High
Data science tool-independent specifications	No	Yes	Yes
Limitless amounts of data processable	No	Yes	Yes
Reusable transformation specifications	No	Yes	Yes
Latency of data	Low	Low	High

Table 1 *Comparison of three solutions for fusing data lakes.*

6 Closing Remarks

Data lakes have changed. Maybe in the beginning they were positioned as single data platforms using one data storage technology. Currently, most organizations have passed that stage and are now developing heterogeneous and distributed platforms. This is also due to the changed nature of the data lakes. It is not just data for data scientists only, they have become massive containers for all the enterprise and external data.

Data virtualization servers help to simplify access to these more complex architectures. They can deploy their heterogeneous data access capabilities and federated queries to present this complex architecture as one integrated data lake. Caching, query pushdown, and parallel query processing are all important features to improve the performance of big queries on such modern data lakes.

TIBCO's data virtualization server supports all the features described in Section 4. It is one of the few data virtualization servers supporting parallel query processing internally. Some of the other products don't support this feature and can end up being the performance bottleneck in a modern data lake that holds massive amounts of data spread across multiple data platforms.

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 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[3].

He has presented countless seminars, webinars, and keynotes at industry-leading conferences. For many years, he has served as the chairman of the annual *European Enterprise Data and Business Intelligence Conference* in London and the annual *Data Warehousing and Business Intelligence Summit*.

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 a number of articles[4] all published at B-eye-Network.com.

Over the years, Rick has written 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[5], entitled *Introduction to SQL*, which has been translated into several languages with more than 100,000 copies sold. More recently, he published his book[6] *Data Virtualization for Business Intelligence Systems*.

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).

About Tibco Software Inc.

TIBCO fuels digital business by enabling better decisions and faster, smarter actions through the TIBCO Connected Intelligence Cloud. From APIs and systems to devices and people, we interconnect everything, capture data in real time wherever it is, and augment the intelligence of your business through analytical insights. Thousands of customers around the globe rely on us to build compelling experiences, energize operations, and propel innovation. Learn how TIBCO makes digital smarter at www.tibco.com.

TIBCO Data Virtualization is data virtualization software that lets you integrate data at big data scale, with breakthrough speed and cost effectiveness. With TIBCO Data Virtualization, you can build and manage virtualized views and data services that access, transform and deliver the data your business requires to accelerate revenue, reduce costs, lessen risk, improve compliance and more.

For more information, please visit https://www.tibco.com/products/tibco-data-virtualization

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

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

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

[6] R.F. van der Lans, *Data Virtualization for Business Intelligence Systems*, Morgan Kaufmann Publishers, 2012.