# THE DATA MESH – THE NEW KID ON THE DATA ARCHITECTURE BLOCK.

Author: **Rick F. van der Lans**

INTENDA

**The data mesh is a new approach to designing and developing data architectures. Unlike a centralized and monolithic architecture based on a data warehouse or data lake, a data mesh is a highly decentralized data architecture.**

In this blog, we'll take a look at the data mesh and how data virtualization can help to develop one. Since it's almost impossible to describe and do justice to data mesh in a single, short blog , I'll limit myself to focusing on the basic essentials. The majority of the data architectures designed over the past thirty years have been designed to integrate data from a large number of source systems, allowing a wide range of users to exploit that integrated data. For example, data warehouse architectures are designed to support reports and dashboards for which data must be merged from all kinds of source systems.

But there are also data lakes being developed to allow data scientists to analyze data from multiple sources. The approach in all such data architectures is to extract data from different source systems and copy it to a centralized, monolithic data store.

**Although these centralized data architectures have served countless companies well, they have the following structural drawbacks:**

• Data engineers working on source systems are typically single-domain experts (examples of data engineers are DBAs, ETL developers, system administrators, data security experts, data warehouse designers, and data administrators). They tend to be very knowledgeable about the domains for which the source systems were developed. They know exactly how the data is stored in these source systems, what that data means, and how well the quality is guaranteed.

On the other hand, data engineers working on centralized data warehouse architectures are multi-domain experts. Evidently, they need to be experts in the tools they use to extract, process, and copy the data, but they also need to understand all the ins and outs of the data they process. This implies that they need to be multi-domain experts. This is almost impossible, especially in large companies, because they really have to understand all the details of all that data. If communication between the two groups of data engineers is not perfect, data may be misunderstood, leading to misunderstanding of the data across the organisation.

• When source systems engineers change something, they need to inform the other group of engineers. If they don't, the systems of the latter group will fail, or worse, will process the data incorrectly, unbeknown to the users.

• Data warehouse and data lake engineers have to develop solutions for extracting data from the source systems. This has to be done in such a way that there is minimal or no measurable interference on those source systems. Regardless of how this is implemented, it requires close collaboration between the engineers. This leads to an unwelcome, tight coupling of the source systems and the centralized data architecture.

• Another structural drawback of these centralized data architectures is data quality. The main question is always, "Who is responsible for correcting data if it's incorrect or missing?"

**The data mesh is designed to avoid such problems altogether, rather than trying to solve them. A data mesh is a highly distributed data architecture.**

One of the most important differences is that source systems engineers are also responsible for developing interfaces for allowing all kinds of applications and users (including data scientists, self-service BI users, batch reports, etc.) to use the domain data. Note that these interfaces are not just service interfaces on top of source systems. Implementing an interface may involve a data warehouse, a data lake, a data mart, or all of these. So, the nodes of a data mesh are like source systems combined with miniature data architectures for data delivery. The interface will take care of data delivery in such a way that users can focus on using the data, without having to concern themselves with verifying data quality, security, and privacy, since these are taken care of by the nodes. Additionally, data mesh nodes should be able to provide all the relevant metadata to describe the data, so a single node in fact incorporates a source system/s plus its/their data delivery solutions, plus their interfaces.

***A data mesh is a highly distributed data architecture.***

Besides source system-based data mesh nodes, some nodes may support users who need data from multiple nodes. Such nodes, which may also contain databases that resemble data warehouses or data lakes, can also accessed by users through an interface.

Data engineers of a data mesh node remain single-domain experts, but for the entire node, not just for the original source system. This minimizes communication problems and misinterpretations of data. The interface of a data mesh node must support any form of data usage, from simple requests for a patient file or invoice, via straightforward dashboards and reports, to advanced forms of analytics, such as data science. Therefore, the technology used to develop these interfaces must support both record-oriented and set-oriented usage. This is where data virtualization comes into play. Data virtualization technology has been developed to create interfaces on almost any kind of system and makes it possible to access that data through a variety of interfaces, including record-oriented and set-oriented interfaces. Additionally, it enables the implementation of data security and privacy rules and provides users with metadata. Several technologies are available for developing those interfaces, and what data virtualization brings to the table is that it already supports most of the required technology, enabling quick development of data meshes.

The data mesh architecture is too interesting to ignore and I strongly recommend that architects study its principles and compare implementation technologies based on ease of development.

***The data mesh is designed to avoid such problems altogether, rather than trying to solve them.***