

## White Paper

# Federated Data - The Big Vision Statement

Adrian Miley

---

### **Synopsis**

One of the things that any architecture should address is not just the current business requirements but also develop a vision of the long term technology trends and show how the architecture helps to adapt to the emerging trends in order to future proof the companies assets as much as possible. This article presents our vision for Distributed Data and Service-Oriented environments going forward into the future.

One of the things that any architecture should address is not just the current business requirements as stated in the various business proposals put forward by the stakeholders but also develop a vision of the long term business requirements and technology trends and try to show how the architecture both helps to meet those requirements and adapt to the emerging trends in order to future proof the companies assets as much as possible.

In the area of large scale, high volume, and dynamic data distribution the significant trends are:

- Federated Datasets where data is transparently distributed across a number of environments and possibly different database platforms.

Unfortunately there are a lot of limitations with this approach (as anyone who has ever tried to join two databases from different vendor's will attest) such as:

- They are generally read-only –read access is always the first problem to be addressed but will eventually need to cover all types of data manipulation.
- Still work at the physical level i.e. they use SQL statements executed against a particular database schema definition.
- Requires a static schema to be declared.
- Dynamic Message Structures (Internet style distribution technologies such as XML).
- Distributed Component-based Computing and Grid Computing where the workload for carrying out a particular task may be spread across multiple nodes in a network.

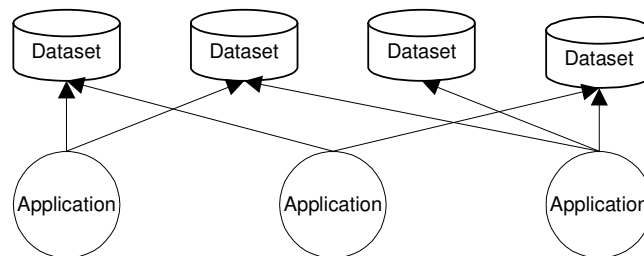
Merging these trends together it is easy to envisage a data distribution environment that is dynamically reconfigurable in all its aspects and where the technical infrastructure can automatically adjust to changes in definition, distribution and storage of Data with minimal, and usually even no, changes to the underlying software or technical infrastructure.

In recent years there has been a massive explosion in the amount of data that can be gathered and made available to customers. Generally the two approaches to managing this data have been either:

- Build a single very large database. This is usually the approach where the dataset being managed has few entities with large number of records for each entity.
- Build a distributed database where there are large numbers of entities that can be partitioned into logically separate, vertically partitioned data-sets.

Large data-centric organisations generally have data with the latter characteristics and consequently have built a set of separate datasets that are lightly coupled together and rely on the software processes to "stitch" the data together into a logical whole.

It gives us something that, at the simplest level, looks like the following:



However Applications designed around distributed datasets have undesirable limitations, in particular:

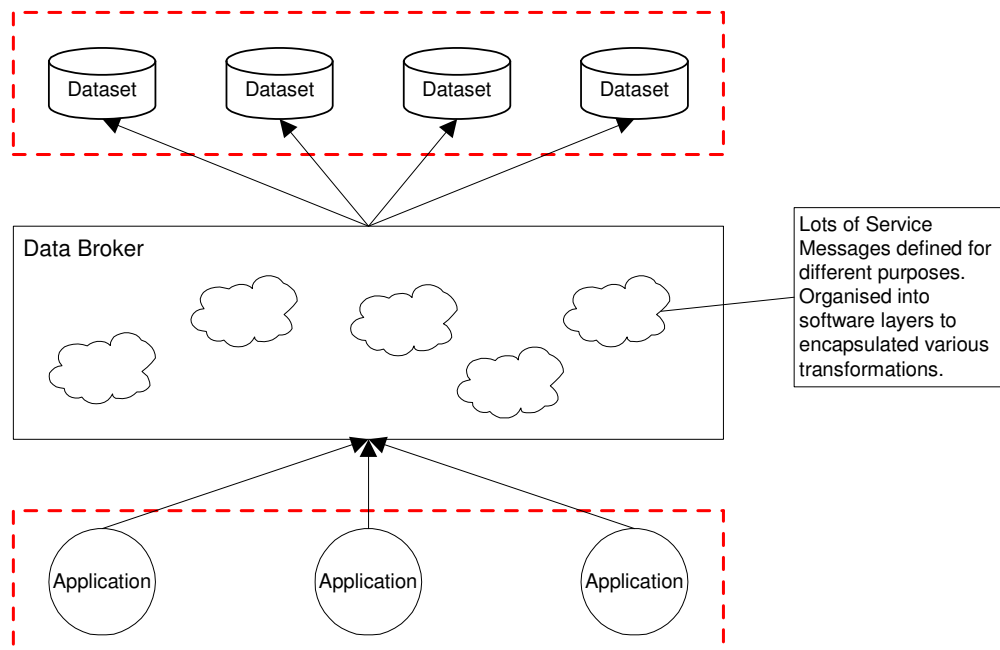
- The Application needs to know where the data is stored in order to access it.
- The retrieval operations are usually tied to the physical structure of the database containing the data.
- When data is migrated from one location to another location the distributed application needs to be modified.

In a slow moving static environment these limitations are not a problem because applications may exist for years (if not decades) without any significant changes being made to the system. However (as we've found a number of times in the past) in an ever changing environment this structure incurs a high maintenance cost as systems are continually upgraded, businesses are reorganised and data-sets consolidated or split to match strategic business requirements.

The answer to the problem is creating a Federated Dataset.

The main feature of a Federated environment is that the application is not aware of the actual location of the data it is manipulating but instead uses a data broker to identify and fetch any requested data and perform the transformations from the Private Form that the data is stored in and the Public Form that the application uses.

It is the data-broker that understands where data is located and how to assemble the requested entities / classes from the underlying physical structures. So we get an environment that looks something like this:



Note: This is a Request / Response Pull model where Applications request data to be pulled from the dataset. By inverting the diagram and renaming Retrieval Services" to "Update Services" we get a Push model where Applications update the datasets by pushing requests into the brokering system.

This is pretty much the architecture that service-oriented environments are moving to (even if we're not aware of it) where the Data Broker service is being implemented at each interface between the architectural layers.

However, although, from a configuration management perspective, this is a far better approach than the distributed approach, because it has localised the Application to Database interface into a single layer and also encapsulates the independence of data storage and process, it is still not fully federated because the Data Broker, which is assembling the Public Form from the Private Form, still needs to know how to retrieve the data and perform the transformations.

Basically we have localised the problem but not eliminated it. When data is migrated from one location to another we still have to amend the Retrieval Services to reflect this.

The answer to this is migrating from a static data structure environment (where messages are pre-defined and compiled into the brokering mechanism) to a dynamic data structure environment (where the structure of messages are defined at the time the message is used).

There are two main forms of dynamic messaging mechanisms, which are:

- Self-defining Structures where the definition of the content is part of the message itself- very flexible and adapts easily to new requirements but verbose and with high processing overhead.
- Externally defined Structures (such as XML) – very flexible but requires access to externally defined structure / schema definition, usually stored as flat-files, to hold the structural definition of the data. Also have a significant overhead, compared to self-defining structures, in parsing and accessing the data.

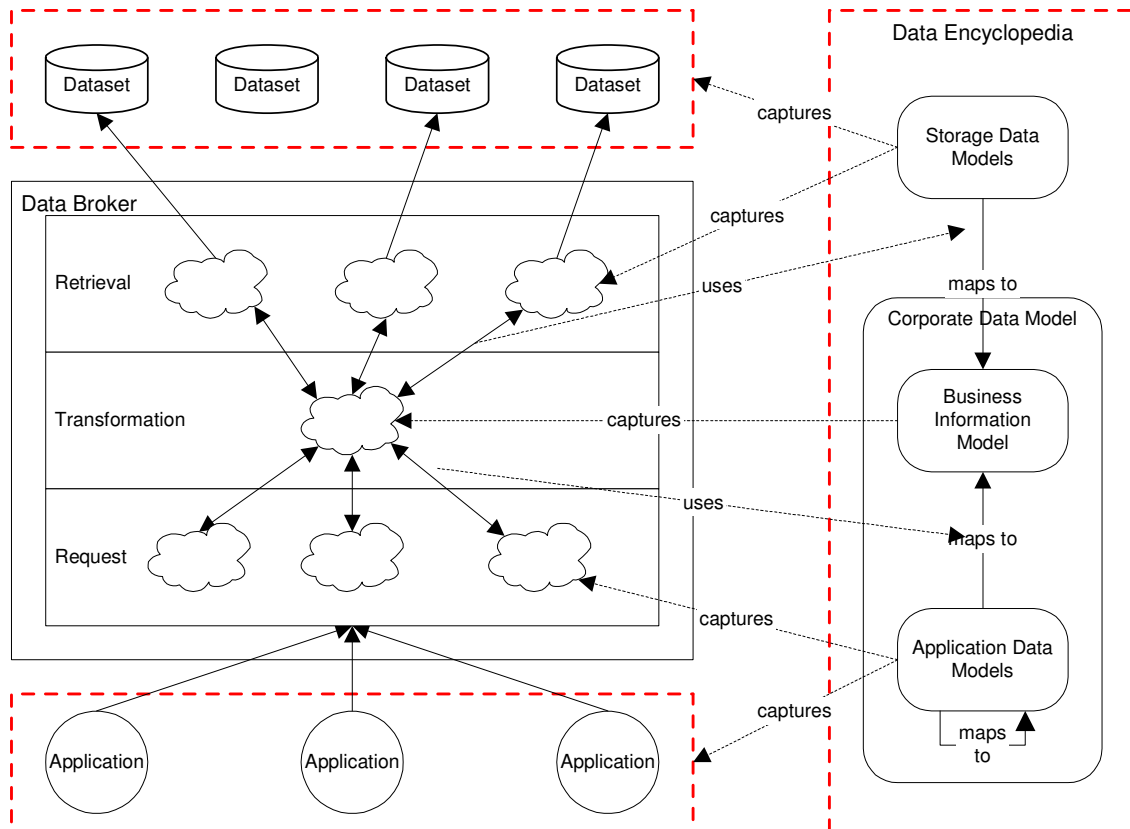
In order to completely hide the underlying physical structures and any technology specific optimisations that may be used we need to abstract the Data Broker one stage further and produce a Dynamic Data Exchange framework

If we combine the high-level concepts together we essentially want an architecture with the following characteristics:

- Federated data access providing location independence of the actual data.
- Dynamic Data Structures that allow the definition of an Entity and its properties to be altered "on the fly".
- A business rules environment that allow validation, events and actions to be integrated into the environment without software changes being necessary.
- To protect ourselves from underlying changes in data management technology we should be using logical entity definitions in the interfaces.

In essence we need a metadata driven environment that is driven by a Data Encyclopedia providing declarative statements that are read and interpreted on demand.

This gives us a configuration that looks something like this:



The key concept is that the Data Broker serves up objects defined at the conceptual level based on Business Entities defined within the Business Information Model. The Business Information Model acts as the link between the Application Data Models and the Storage Data Models and so enables the dynamic mapping between those models.

Hence we not only achieve complete separation between the Application and Storage but also, through controlling the transformations via a Data Encyclopedia, achieve a dynamic messaging environment where messages are built and populated on demand rather than being pre-defined at the software level.

This is the big vision for Distributed Data and Service-Oriented environments going forward into the future.