

## White Paper

# Building Synchronised Distributed Data-Sets – Some Thoughts

Adrian Miley

---

### Synopsis

It is very likely that within a large-scale multi-site enterprise there will exist many data-sets, each with their own suite of data maintenance applications, that appear to be disparate but actually contain a significant percentage of overlapping or duplicated data.

This article is a preamble to the "*Managing Synchronised Distributed Data-Sets*" presentation at the **DAMA Europe 2005** conference which will explore some of the issues mentioned in more detail.

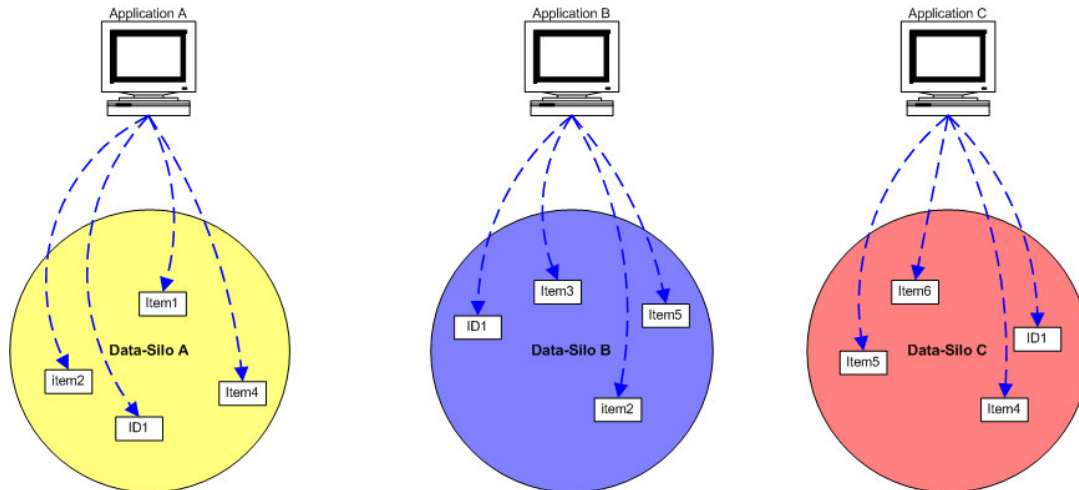
---

This document contains information proprietary to Miley Watts & Associates Ltd, and may not be reproduced, disclosed or used in whole or part without express written permission of Miley Watts & Associates Ltd.

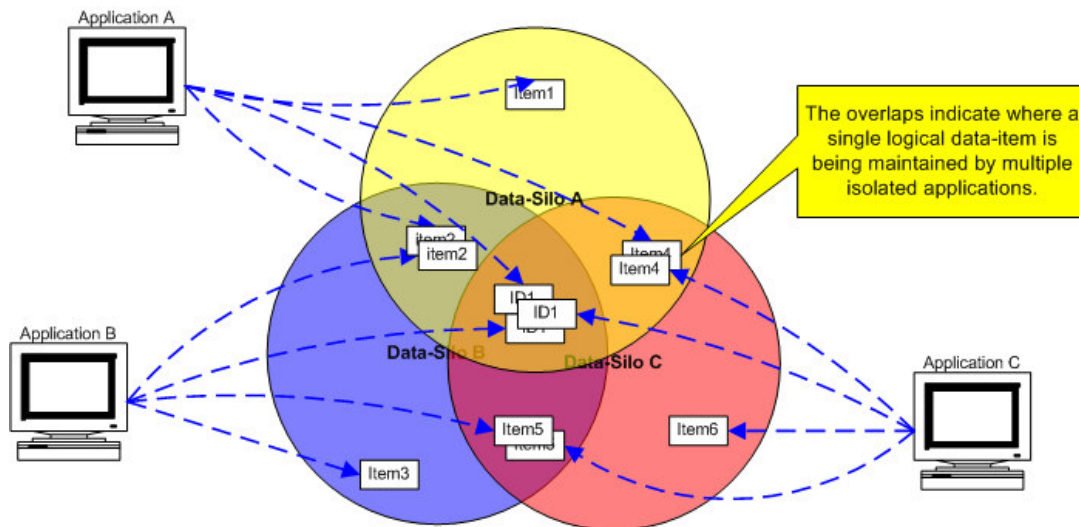
If you would like to discuss any of the ideas described in this document in more detail then please contact us at [enquiries@mileywatts.com](mailto:enquiries@mileywatts.com) and we will be happy to have a conversation.

It is very likely that within a large-scale multi-site enterprise there will exist many data-sets, each with their own suite of data maintenance applications, that appear to be disparate but actually contain a significant percentage of overlapping or duplicated data.

That is, the enterprise may look like this:



But in reality looks like this:



Even though there is only one copy for the majority of the data-items – of course we cannot be absolutely sure about that because we don't have an integrated data model to prove it – the existence of duplicated data-items can raise a number of issues and concerns, such as:

Why is the data inconsistent? Inconsistent data potentially identifies a problem with the existing business processes – both manual and automated – resulting in wasting either time or money.

What are the implications on the business of having inconsistent data? With legislation, such as Sarbanes-Oxley, publishing inconsistent or misleading information may leave the company open to litigation.

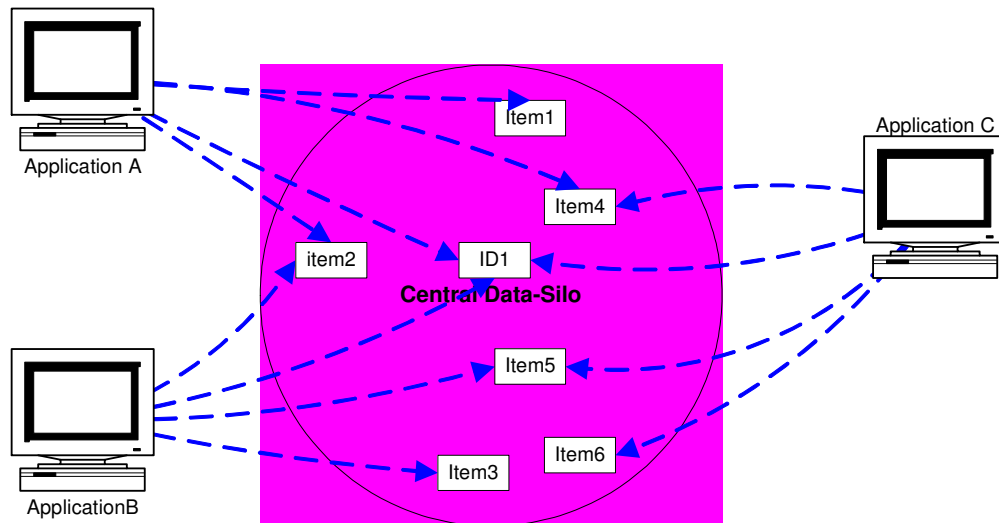
If the data is published to an external organisation then how is the publisher's reputation affected by publishing inconsistent data?

How trustworthy is the data? In the event that two copies of the same data-item have different values then which of the copies is regarded as the authoritative copy - the Data-Item of Record? Not having a clearly defined Data-Item of Record can damage trust in the data and lead to hesitant decision making.

What is the cost of managing multiple copies of the same data? All business activity, no matter how trivial, has a cost associated with it so the long-term cost of unnecessary duplication may be significant. Removal of duplication is one of the best methods of removing cost from a business.

Where significant concern exists a business may decide that it needs to enforce consistency across the data-sets. To do this there are three main approaches that may be taken:

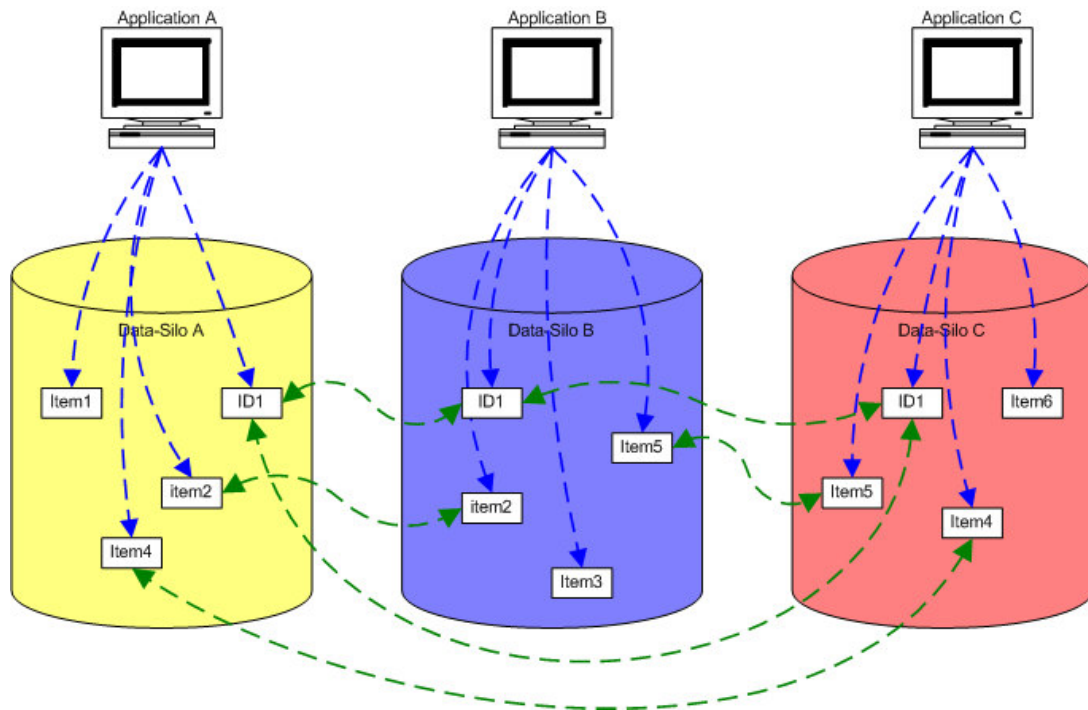
1. Build a central data repository, migrate all the data into the repository and re-engineer the data processing applications to work with the new environment. This was the traditional solution to any complex data integration problem when centrally managed systems were all the rage.



Popular opinion says that the best business solution is to consolidate the data into a single centralised database, and re-engineer all data-processing applications to work against the new repository. This has indeed been the traditional solution of choice because it has the obvious overriding benefit that there is only a single copy of each data-item and, consequently, we are guaranteed to only have one version of the truth.

However, in a large scale enterprise the significant problems with this approach are:

- Expensive – every existing application needs re-testing against the new platform plus possible re-engineering to remove incompatibilities.
  - Physical Data Models are frequently incompatible with each other so cannot be easily merged – especially true when more than one DBMS is involved because they all adhere to different proprietary standards.
  - Disruptive to the organisations operations with massive retesting of applications – whilst the migration is taking place new data sets are being designed and implemented.
  - Difficult to get agreement from all stakeholders as to what the integrated platform looks like – vested interests dictate that most stakeholders want the integrated platform to look most like their current sub-set in order to avoid impact of the above two problems.
  - Problem reoccurs whenever a new dataset – as happens whenever a merger or acquisition is carried out – needs to be integrated.
2. Build a Federated Data Environment and enforce synchronisation of the data through distributed updates at the point that the data is updated.



This has a number of advantages over migrating to a Centralised Data Environment, such as:

- ✓ Leaves Data-Silos in situ therefore all existing applications continue to work against the same local data-set and require minimum modification.
- ✓ It is extensible – the framework, if designed to a sufficiently generic pattern, can be extended to incorporate other data-sets as the business needs arise e.g. when an acquisition occurs.
- ✓ Once the technical infrastructure has been built then it allows for a gradual roll-out of any solution. Existing data-sets can be integrated independently of each other.

However to ensure consistency across the domain we replicate each relevant update – as show by the green arrows – across data-sets.

3. Manually synchronise the data through defined business processes where the company employees carry out the tasks of individually updating each data-item. This however is really just a very inefficient and labour intensive variation of option (2) that is generally taken when neither of the above is acceptable.

Both Option 1 and Option 2 have advantages and disadvantages depending on the environment but overriding all technical considerations are of course the restrictions imposed by the corporate environment, such as:

- The geographic availability of data, especially when distributing data over inter-continental distances, may mean that a central data-centre may not have the same “opening hours” as the local user.
- Data Ownership may need to be retained by an Operating Subsidiary especially if there is a likelihood that an Operating Subsidiary make be sold off in the near future. In this case integrating the data into a central data repository is not acceptable because of the cost of disentangling the data if the Operating Subsidiary is actually disposed of.
- The sheer number of data-sets and data maintenance applications makes it unfeasible, due to cost, to merge them all into a single environment within a responsible timescale.
- The Organisations Operating Model means that the environment is not stable enough to allow the required “freeze” on system changes. For example, the Organisation is constantly updating its technical infrastructure or acquiring new subsidiaries or developing new products.

For these and many other reasons, an Organisation may decide to retain its Distributed Data Environment and evolve it into a Federated Data Environment.

So, having taken the decision to create a Federated Data Environment, we finally arrive at the point of this article which is to discuss the not insignificant task of making the Federated Data Environment happen and answering the following key question

*What do we need to do to synchronise updates to multiple data-items across the Distributed Data Environment?*

It doesn't look like a very big question but in resolving we touch on every area of the Federated Data Environment and it raises secondary issues in both the Business Requirements and the Data Architecture & Design (i.e. implementation) areas.

Business Stakeholder Issues that should be addressed include:

- Producing an Integrated Data Model containing a core Business Data Model of all of the data-items covered by the Distributed Data Environment and mappings to all the Physical Data Models where each data-item is stored and to all the Application Data Models that use the data-item.

The Integrated Data Model can be built up as each data-set is integrated into the Federated Data Environment but the core of the model – identifying the Primary Business Entities – should be in place before any implementation work is carried

out because it is imperative to know what is stored where in order to be able to orchestrate updates.

A consequence of not developing the Integrated Data Model before the technical framework is developed is the significant risk that model incompatibilities – including many showstoppers – may not be identified until after much time and money has been spent on implementation.

- Plan the quality control requirements for data management according to the minimum needs of your business.

For example, is there a need to identify all the updates across the entire environment that resulted from a particular transaction? Do we want to measure the reliability of data changes against the source of the change?

- Defining requirements for Timeliness – how quickly an update will be seen by a consumer – in order to establish whether updates must be applied in real-time or whether a delay in synchronisation is permissible.

Real-Time updates require orchestration of updates at the time the update is being applied and implies that the update cannot be committed in one location without it being confirmed at all locations.

However, if a delay in synchronisation is permissible then the cost of implementation reduces because of the greater range of implementation options that are available and greater decoupling between data-sets. The Downside is that a greater delay increases the risk that one of the target data-sets will itself have been updated in the meantime.

- Ensuring that you have established scope for each data set and that you know where the responsibility for each Data Item of Record lies.
- Deciding on what should happen when things go wrong and how much resiliency is required in the update distribution mechanism - no matter how robust the infrastructure is it is absolutely certain that at some point something will go wrong and an update will go missing.

Even so some sort of reality check should be performed against the business requirements because it is possible to spend significant amounts of money building recovery mechanisms for a Federated Data Environment but a lot of money can be saved by asking for “just enough” resiliency

- Understanding how critical the data is to the running of the business

Most data-flows and computerised applications just make it easier for the business to run itself but some data applications, such as a Credit Card Payment system, incapacitate the business if not available or, in the case of a Financial Instrument Market Price feed, incapacitate customers if they do not deliver up-to-date and consistent data.

- Understand and define the ultimate end-state for building a Federated Data Environment. Putting the basic framework in place should be just the beginning, once complete the principles that underpin the framework must be constantly employed to ensure that new implementations integrate properly.

It's important to ensure these points are understood, and agreed, by the Business Stakeholders before implementation and that the "pain threshold" of any approach is acceptable.

As well as providing solutions to all of the requirements identified above additional and significant Data Architecture & Design Issues include:

- The production of an Integrated Data Model and establishing any data transformation rules that need to be applied when replicating data from one location to another. This is a pre-requisite in order to establish "Trust" between sub-domains. This is repeated just to make it clear how important one of these is!
- Deciding whether distributed updates are orchestrated by the Client or the Server side application. The Client application is the application that initiates the data maintenance event. The Server application is the Database Management System that manages the data that was modified.
- Deciding on how to satisfy Referential Integrity and Primary Key management  
Each data-set may well use its own System Generated Key to provide a Private Primary Key (private to the data-set) for each Business Entity instance so, when moving data across sub-domain boundaries it must be understood how to translate from one ID to another and the point at which the translation should take place.

Ideally a common Primary Key – called the Primary Business Identifier – should be mandated for each Business Entity and used for all messages that pass across the sub-domain boundary. If there is no common key then each source of an update and each target will have to carry the transformation from Private Primary Key to Primary Business Identifier and back again.

- Targeted Update Messages v Generic Update Messages  
In Targeted Updates the Update Originator knows in advance which targets may need a particular update and create a target specific update message for each recognised target. Each Update Listener only processes the messages that are specifically targeted at it.

With Generic Update Messages the Update Originator publishes a single message that is put onto the wire and any Update Listener can read the message and process it.

The trade-off between the approaches is that the Targeted Update Messages increase network traffic because multiple messages are produced for each update event but for Generic Update Messages each target has increased workload because they will receive messages that they may not want to process.

An analysis of the volumes and frequency of updates cross-referencing transaction volumes against target data-sets needs to be produced in order to model the data-flow and message processing workload.

- Record Locking and Data Concurrency – if different copies of the same data-item can be updated by more than one application then there is some probability that two updates to the same data-item may clash with each other.  
The higher the probability of this happening then the more critical it is to have a robust mechanism for locking a data-item for the period of an update and, indeed,

there are many well defined solutions to the problem each designed for a particular set of scenarios.

Once again the decision making requires detailed transaction analysis in order to identify potential clash points and how often they occur.

- Deciding on the Rollback / Undo mechanism to use – “Two-Phase Commit” or “Managed Undo”.

The knee-jerk reaction is usually “Two-Phase Commit” – where all data-items must be updated before the transaction can successfully terminate – but in a decoupled environment the “Managed Undo” approach is far less risky, but more cost-effective, to implement.

In a complex environment – probably any environment with more than five significant data-sets to integrate - any one of these issues could involve significant effort to resolve but can waste much time and effort in implementation if the least correct (nothing in I.T. is ever totally correct or totally incorrect – it’s why we’re always arguing! 😊) or least appropriate path is taken.

The tricky bit is acquiring the knowledge and experience regarding how to approach finding the right answer for a particular data environment and that, I suppose, is the reason that experts in data are used.