PENSIONS AND LIFETIME SAVINGS ASSOCIATION

October 2015

BETTER DATA, BETTER GOVERNANCE

MADE SIMPLE GUIDE

ACKNOWLEDGEMENTS

We would like to thank J.P. Morgan for its help producing and sponsoring this guide.

J.P.Morgan

This guide is for information only. It is not investment advice.
Published by the Pensions and Lifetime Savings Association 2015 ©
First published: October 2015

CONTENTS

1 INTRODUCTION

THIS GUIDE WILL EXPLAIN HOW DATA MANAGEMENT SYSTEMS WORK. IT WILL EXPLORE THE IMPORTANCE OF MANAGING DATA FOR TRUSTEES, AND WILL HELP THEM TO UNDERSTAND THE DIFFERENT OPTIONS FOR HOLDING AND MANAGING THEIR SCHEME'S DATA.

UK pension trustees are required by law to hold accurate data on their members, and The Pensions Regulator continues to highlight the importance of good data management and record keeping. Trustees are expected to test the quality and completeness of data at least annually; and failure to maintain accurate and complete member data, investment records and liability information can result in financial risk and a risk to members' benefits. Yet a number of studies and surveys have established that trustees have a limited grasp of data issues.

Data management and record keeping is an area which trustees must take seriously. As the volume and breadth of required data continues to increase, trustees need to have a holistic data and reporting strategy that can satisfy record keeping, regulatory, risk management and reporting requirements.

Illustration of the key data categories that need to be considered when building a holistic data and reporting strategy for a pension scheme.

As a pensions professional, whether or not you are involved directly in housing and querying data, you still should understand the principles. Whatever analytical tools and reports you use, awareness of the data behind the scenes will help you to interpret the results you see. It will inform your decisions and help distinguish real issues from red herrings.

TRUSTEES OFTEN DO NOT APPEAR TO HAVE A GRASP OF DATA ISSUES, NOR DO THEY FULLY UNDERSTAND THEIR RESPONSIBILITY TO RESOLVE THESE.

Mercer 2010 – Pension Protection Fund, Future of the Risk-based Levy – Reflecting good governance practices and risk management.
www.pensionprotectionfund.org.uk/levy/Documents/Mercer_future_levy.pdf

DATA QUALITY: DESPITE THE REGULATOR'S GUIDANCE, AS WELL AS THE THREAT OF MORE FORMAL REQUIREMENTS IF SCHEMES FAIL TO COMPLY WITH IT ON A VOLUNTARY BASIS, OUR SURVEY FOUND THAT NEARLY 30 PER CENT OF RESPONDENTS HAVE YET TO AGREE TARGETS FOR STANDARDS OF DATA AND A DEADLINE FOR ACHIEVING THEM. WORRYINGLY, OVER 11 PER CENT 'DON'T KNOW' WHETHER SUCH TARGETS HAVE BEEN AGREED.

JLT Benefit Solutions Limited – Survey Report 2011 – The relationship between pension scheme administration and effective scheme governance.
www.pensions-pmi.org.uk/documents/jlt-survey-report-final/survey-report-final.pdf

2
A BRIEF HISTORY OF DATA
FROM 0s AND 1s TO MUSIC, VIDEO AND 'BIG DATA'

Before jumping into our topic, it is worth taking a quick look at how we got here. Once upon a time, all data was essentially stored in very long lists. If you wanted to find something at the end of the list, the only way to get there was to start at the beginning and read forward. Advances in both hardware (moving from tape drives to disk) and in software allowed for indexing of data and querying only the specific rows of data you need. The advent of relational databases enabled easy joining across many kinds of data, synchronised updates and consistent labelling of data elements. And the big data revolution has added new dimensions to what is possible. The timeline below summarises this history.

1960s | 1970s | 1980s | 1990s | 2000s | 2010s

DIRECT ACCESS
- Alphabet and numeric data
- Stored on magnetic tape: a user starts at the beginning to find the desired data point (single step)
- Strong at batch processing where sequence does not matter, such as calculating amortization on one million mortgages in succession
- Weak on user interface / query tools to an individual mortgage schedule, for example

RANDOM ACCESS
- Alphabet and numeric data
- Stored on disk or cylinder of multiple disks, creating ability to jump to the desired data. Index tables were developed to store the physical address of every data record, plus the data's associated key values, letting users randomly access each set of data
- Enables the user to use one field – e.g. taxpayer id number – to navigate into a tax table to find out how much an individual taxpayer paid

RELATIONAL DATABASE
- Random access data based on indexed keys
- Common-language name for every piece of data
- Navigation from table to table, based on common fields found within the tables
- Updates managed across tables into 'transactions' so that data is never partially updated
- Stored statistics about the data itself allows optimiser programs to efficiently access data
- Able to manage vast numbers of simultaneous queries through queues and caches (in-memory storage)

RELATIONAL DATABASE
- Explosion of data types, unstructured data and social media
- Data expands to include images, sound and video – requiring vastly more storage space and new ways of storing and searching

'Big data' refers to the challenge of organising unstructured data, which cannot be meaningfully organised according to its relation to other data. (e.g. searches of financial news, scans of social media for investment ideas, scanning regulations or identifying high-risk traders or clients).

BIG DATA VERSUS RELATIONAL DATABASES

Some of us have some exposure to the relational databases: Oracle, DB2 and Sql-Server are all examples of relational databases, and the vast majority of financial applications depend on relational databases.

Today's buzzword, though, is 'big data'. But what exactly does that mean, and where does it fit in? Big data means more than just lots of terabytes. It also implies that the data is unstructured – which in turn means it does not fit into a relational database.

To see where big data comes in, imagine that you wanted to find all of the news articles about companies that had some particular attribute or had been through some particular event. Clearly, it would be impossible to store in a relational database every news article about every company, and create some near-infinite set of indexes to support any conceivable kind of search. And yet, if you type such a query into a search engine – 'corporate acquisitions', for example – you'd likely get a pretty reasonable result.

This then is the challenge of unstructured data – data that cannot be precisely organised according to its relation to other data. While this type of data is still less common in the financial world, it comes up in searches of financial news, scans of social media for investment ideas, and has potential application in everything from aiding legal departments in scanning regulations to identifying rogue traders or high-risk clients. Because unstructured data often requires vast amounts of storage, and mining it efficiently is a very major challenge, these sorts of applications are generally referred to, collectively, as big data.

Overwhelmingly, however, financial data continues to be highly structured and relational databases remain the means by which it is stored. And so we return our focus to the familiar, relational world.

WHAT IS MEANT BY A 'NORMALISED' DATABASE, AND WHY IS THAT IMPORTANT?

Early on, there tended to be a pretty rigid view about the 'right' way to organise data within a database. While this view has evolved and changed as theory has butted up against practical reality, it is worth taking a look at the traditional way data is modeled, the challenges it presents, and how thinking has evolved.

The traditional theory has been guided by a relatively simple concept: each piece of data should only appear once in your database. Consider a table that stores a pension plan's holdings. Data related to this imaginary Portfolio Holdings table include:

PORTFOLIO HOLDINGS
Portfolio manager name
Portfolio ID
Month end date
Security identifier
Number of shares
Market value
Security name
Security type
Industry

Now imagine that the plan includes 30 portfolios that all hold AOL Corporation, and you have 10 years' (or 120 month-ends') worth of history for each portfolio. So what happens when AOL acquires Time Warner (as it did in 2000)? The Security Name and Industry for this security both change, and, given this table structure, data needs to be updated in 30 portfolios x 120 time periods = 3,600 rows of data.

What if the portfolio manager name were to change? In that case, every row for every security in the portfolio would need to be updated – perhaps a million or more rows of data.

IN PRACTICE, MANY DATABASE DESIGNERS HAVE MOVED AWAY FROM THE TEXTBOOK

In order to avoid these high-volume and complex updates, the notion of a 'normalised' database was developed. Each data element would be stored only once. Instead of repeating the data for AOL on every row of the PORTFOLIO_HOLDINGS table, data elements about a single security would be moved off to a separate table: a Security Master. Instead of repeating the portfolio manager name over and over for every holding, move it to a table of static data about the portfolio itself. Thus, our table on page 6, in a normalised format, would be split into three tables:

PORTFOLIO_DATA	PORTFOLIO_HOLDINGS	SECURITY_MASTER
Portfolio_id	Portfolio_id	Security_id
Portfolio_manager_ name	Month_end_date	Security_type
Portfolio_ open_date	Security_identifier	Security_name
	Shares	Industry
	Market_value	
...other static reference data	...other historical values	...other static reference data
Static reference data	Historical values	Static reference data

With this design, the AOL acquisition requires an update to only one row on SECURITY_MASTER. The change of manager name requires an update to only one row on PORTFOLIO_DATA. A relatively simple query can navigate from PORTFOLIO_DATA to PORTFOLIO_HOLDINGS to SECURITY_MASTER, pulling whatever data is necessary.

Success! Each piece of data resides once and only once in the database, creating a perfectly normalised database.

SO IS THERE A DOWNSIDE TO A NORMALISED DATABASE?

The normalised data structure is the classic, textbook solution. But in practice, many database designers have moved away from the textbook. To understand why, imagine that instead of three tables, when you modelled your complete database you wound up with three *hundred* tables. And further imagine that your simple query also needs additional data. Let's say it needs P/E (Price / Earnings) Ratio, Parent Company Name, Country of Incorporation, etc. And because you did a good job in designing a highly normalised database, all of these attributes have landed in different tables. Suddenly, your simple query is not so simple anymore. It has grown to join together perhaps 20 tables instead of three. And each of these joins winds up being expensive in terms of system resources. Your query speed has suddenly declined and you are unable to get your data out of the database fast enough.

It turns out that while a fully normalised database has advantages in limiting the scope of updates, and is therefore a good design for an operational data store (for example, a database that is connected to a user interface where data is being rapidly inserted or altered), it's a pretty awful design for data extraction and reporting. In reporting, the fact that data is spread out across lots of little tables just slows everything down.

REPORTING DATABASES:

It was this challenge that caused some designers to propose storing the same data in two, physically separate databases:

- Store it in a 'normalised' operational data store
- Replicate it over to a 'denormalised' reporting database

And what is the format of the denormalised reporting database? In a reporting database, the data is once again 'flattened out', or 'denormalised'. There are fewer tables with more attributes on them. This means queries require fewer joins and data retrieval is faster. It also means data is stored redundantly as a convenience for reporting. For example, if we look again at the PORTFOLIO_HOLDINGS table on page 9, in a reporting database, we might choose to add security_type and security_name back into the holdings table, since many reports need these attributes, and this will avoid having to join to SECURITY_MASTER to get them.

Basically, a reporting database involves a partial unravelling of the normalisation process that was used to create the operational database.

WHERE DO DATA WAREHOUSES FIT IN?

In many businesses, the data itself is a lot more than a live, constantly updating record of what is happening today. It is also an asset. It contains a huge amount of information that can be analysed to yield valuable insights. Let's say your fixed income portfolio has moved from a 5 to a 15 % investment in below-investment-grade securities over time. You know this from reports you have received, but you are trying to understand why this has happened.

- Did the investment manager sell high-grade securities and buy low-grade securities?
- Were several previously high-grade holdings downgraded over time?
- Did a new credit ratings provider cause a shift?
- Were previously unrated securities rated at some point by existing credit agencies, shifting the overall weighted-average credit rating of your portfolio?

In fact, any combination of these is possible.

To fully analyse the trend, you need to know all of the portfolio's holdings for every time period, plus how your security master appeared at each point in time. You would also want to see how data came into your system from the ratings agencies. You need clear audit control on all of your tables, so you know exactly when each change to the data was made.

A data warehouse can provide this.

A warehouse is generally designed in a flattened out format like a reporting database, because updates to the data are relatively infrequent, and the sheer size of the database means you need to worry about query speed. The difference is that, while a reporting database may store only current versions of data – especially relatively stable data such as a security master – the data warehouse stores every version of every piece of data, as the data evolves. And a good warehouse provides traceability – so you always know where a data element came from, when an update occurred, and what caused the update.

IN MANY BUSINESSES, THE DATA ITSELF IS A LOT MORE THAN A LIVE, CONSTANTLY UPDATING RECORD OF WHAT IS HAPPENING TODAY

WAREHOUSES AND DATA MINING

The other key function of a warehouse is data mining. The above example is based on researching a particular anomaly. Data mining might be thought of as the process of *looking for* anomalies, for interesting trends or conducting empirical research. For example, an investment manager looking for trading opportunities might want to know, “How do stocks perform in the business days immediately after ex-date?” Or, “Historically, if a stock has gone up two days in a row, what are the odds that it will go up the following day?” These are the sorts of questions asked of a warehouse, or research database, in a data mining exercise.

DOES THAT MEAN WE NEED THREE SEPARATE DATA STORES – OPERATIONAL, REPORTING AND WAREHOUSE?

Definitely not. In practice, databases are seldom divided quite so neatly by function as these examples might suggest. Some operational data stores can facilitate reporting. Many reporting databases also serve as warehouses. And both reporting databases and warehouses can wind up having calculation components added on to them – meaning they become hybrids serving a warehouse function, reporting and a bit of additional processing. This may not be a purist solution, but for many problems it serves the need. Moreover, replicating data from one database to the next can itself be a cost that must be weighed against the advantage.

As with most other aspects of systems design, the right solution should ultimately be dictated by the specific requirements, cost, benefit, legacy environment, and so on. There is clearly no one right solution (although of course there are many wrong ones!).

SO WHERE DOES METADATA COME IN?

Metadata is not a new term or a new concept (its first use was in a computer science article in 1968) but it is certainly newly in common usage (and misusage) with the public.

Simply put, metadata is data that describes other data. Market value is data. The fact that it is numeric, and that it should be displayed to two decimal places, are pieces of metadata. The timestamp of when the data was inserted is another important piece of metadata. With social media, big data, and so on, metadata has seen an increased focus, as the need to find new ways to tag data, and to organise and access massive amounts of unstructured data, has required the capturing of a great deal of metadata. In most financial applications however, it has long served a pretty unglamorous, behind-the-scenes role.

3 DATA GOVERNANCE

NOW THAT WE HAVE REVIEWED SOME DATABASE BASICS, IT IS TIME TO TURN OUR ATTENTION TO ONE OF THE KEY CHALLENGES OF TODAY: DATA GOVERNANCE.

WHY DID DATA GOVERNANCE SUDDENLY BECOME SO IMPORTANT?

While good data governance has always been important, there are some drivers that have made it more imperative than ever:

- Complexity within the financial world has grown exponentially, both from a regulatory perspective and in terms of investment types. With that complexity come all of the challenges around clear, precisely labelled data.
- Enterprise-wide data warehouses have many benefits, but also present challenges. They require joining across applications and content areas. Often, expertise across the data no longer resides with a single individual, and requires a governance team.
- Business intelligence tools have made it possible for business users to interact more directly with the underlying data. Users can now develop analyses and presentations without involving technology teams. This placed a greater burden on the data itself, since there is no custom software application that extracts and prepares it.
- In general, organisations are more dependent than ever on complete, timely, accurate information.

DATA GOVERNANCE OBJECTIVES

Most are familiar with the proverbial question posed by many a software developer:

> **DO YOU WANT YOUR DATA DELIVERED ON TIME, OR DO YOU WANT IT TO BE CORRECT?**

If database design and administration are needed to ensure the former, data governance is needed to ensure the latter. It is only through both good design and good governance that the business can respond, with confidence, "I would like both please, plus a few performance metrics."

Moreover, it is impossible to entirely decouple design and governance. Without a good place on the database to house information regarding data sources, it is difficult to analyse what data might be stale or missing.

Here are six tried-and-true governance principles that will aid in maintaining the quality of both the database and the data it houses:

1. Clear taxonomy – Every attribute needs a complete name and description – or to put it another way, every piece of data needs good metadata that describes it. This is especially difficult, and therefore especially critical, in large warehouses where few individuals are experts in the entire set of data. For example, "Market_value" is a potentially ambiguous data name. Is this denominated in the fund's base currency or the traded currency? Does it include accrual? Is it based on a stale price? It may not always be possible to include all of this information in a data name, but this is where a full data dictionary comes into play.

2. Consistency – The same field cannot be called Accrued_market_value on one table and just market_value on another table. This leads to confusion. Confusion creates a lack of trust in the entire database. And in the end, this can lead to the failure of an entire initiative. Where data is being fed from multiple upstream systems, it may seem convenient to borrow labels from the systems upstream. But rationalising the data into a common set of fields and tables is crucial to laying a good foundation for a warehouse.

3. Data source retention – Traceability back to the original source is key. Your data may be correct but, in order to fully trust it, you need to be able to *prove* it is correct. For this, you need to know how, when and why it got there in the first place. You need breadcrumbs that you can follow backwards, upstream, all the way to the source.

4. Referential integrity – Ensure that every piece of data that *should* be able to join from table to table *does* join from table to table. If you have Currency Code on your holdings table, and you have another table that translates from the code to the description, you must, without exception, have a row in your translation table that matches every row in your holdings table. Similarly, you must have a row in your Portfolio table for every portfolio_id in your portfolio_holdings table. And so on. This is known as maintaining 'referential integrity' – ie the ability to reference properly between tables.

5. One version of 'the truth' – The phrase 'official book of record' has come into the lexicon simply because many enterprises have confusion about what the official book of record really is. As data proliferates, establishing and maintaining a single version of 'the truth' can avoid great amounts of time spent doing analysis, performing reconciliations, and generally debating policy and practice.

6. Accuracy –Data must be accurate but achieving that is not so simple. Since millions of rows of new data could be added daily, it is critical to identify and manage potential sources of data corruption. These can include:

- **a.** Human error keying in data
- **b.** Load jobs running out-of-sequence
- **c.** Bad upstream data source
- **d.** Software logic fails in a specific case
- **e.** Code-change introduced a defect

IMPLEMENTING GOOD GOVERNANCE

Now that we've defined the objectives of a data governance programme, how does one achieve these objectives? Let's look at what data governance is and, conversely, what it is not.

WHAT IT IS:

SET AND OVERSEE POLICIES AROUND DATA

- Contribute expertise and guide principles to database design
- Set policy around data validations and corrections
- Set policy around data retention and archiving
- Establish data security standards – who can see what data
- Review data quality and coverage metrics on a regular basis
- Review appropriateness of data sources and taxonomy
- Establish and maintain a roadmap of future enhancements

WHAT IT IS NOT:

MAINTAIN DATA

- Validate data
- Scrub, edit or cleanse data
- Detailed database design
- Regular database administration and optimisation

Like other assets across an enterprise, data requires oversight. While a governance committee doesn't scrub the data, it does review policy around corrections, the results of initiatives, and updating policy as needed. And while it does not own the detailed database design, it should frame all the requirements to guide design, and review the design for potential issues.

WHO SHOULD OWN DATA GOVERNANCE?

To be most effective, governance should:

- be managed by key stakeholders within technology and the business, working collaboratively and bringing together their respective areas of expertise;
- include individuals with direct experience and history with the relevant applications; and
- include those engaged at the beginning of a project, in order to ensure continuity.

WHEN DOES DATA GOVERNANCE BEGIN?

Governance should begin with the design phase of a project. There may not be any data yet inside the database, but the future governance team should already be defining high-level design. At project inception, governance should focus on:

1. Clearly articulated, quantifiable benefits from a data initiative
2. Oversight of concrete requirements
3. Review of overall design

Lack of clearly articulated expectations and requirements is one of the most common causes of project failure.

4
A CASE STUDY:
HOW TO BUILD AND MAINTAIN A REALLY BAD DATA WAREHOUSE

THEORY IS ALL WELL AND GOOD. BUT LET'S LOOK AT AN EXAMPLE WHERE THE PROCESS HAS BROKEN DOWN, AND SEE IF WE CAN IDENTIFY THE FAILURES AND THEIR SOLUTIONS:

THE SCENARIO

An equity manager would like to understand the performance of his technology sector. He uses his database to compare holdings and prices at the beginning and the end of a particular period of time.

Sum of: (ending price – beginning price) x shares

he expects to have his answer.

However, a quick review shows the result is incorrect.

1. A couple of securities show a huge loss that doesn't seem like it could be right. Yet it matches with the prices on the price table.
2. The opening positions include a security that had been sold prior to the start of the period.
3. The report includes a security that is not even part of the technology sector!
4. The prices obtained from the database don't appear to match the prices used in official accounting statements.

HOW IS ALL OF THIS POSSIBLE? WHAT WENT WRONG?

All of these issues may occur due to common, easily-avoided mistakes that a governance team might have recognised and thwarted. Let's look at them one by one.

1. A couple of securities show a huge, mysterious loss.

The careful reader may have noticed a flaw in the report logic: price change does not always represent gain-loss. In the case of a stock split, there is a sudden price drop that should not be considered a loss. However, building this logic into a report is tricky. One would have to read backwards through a history of corporate actions, identify the split-factors, and then figure out how they fit in with trade timing in order to determine the effective price change of a particular holding. On the other hand, if this requirement had been considered upfront, adding split-adjusted prices to the pricing table would allow a simple query to obtain effective price change without building so much after-the-fact logic into the report.

2. The opening positions include a security that had been sold prior to the start of the period.

Typically, in order to minimise the number of updates against a warehouse, software developers will want to design 'delta refreshes' – updating only data that has changed. This reduces load on the database. But it also comes with a risk. Suppose the first time you receive data from an upstream system, it included ten holdings for a portfolio. The second time, you receive the same data, but it only includes nine holdings (because, for example, a late-posted trade indicated that one of the positions had actually been sold out). Your delta refresh will properly update the nine holdings that have changed. But will it be

IN EVERY CASE, THE PROBLEM COULD HAVE BEEN AVOIDED WITH STRONG GOVERNANCE AND A SOLID, UPFRONT DESIGN

smart enough to delete the tenth holding, which came from the previous version of the same data?

Since this sort of problem may not be immediately obvious, it could live on for some months before it is caught through some reconciliation. By that point, the 'fix' is no longer just a matter of addressing the defect in the loader code. The data clean-up exercise could be massive, as phantom holdings could have proliferated across the database.

3. **The report includes a security that is not even part of the technology sector.**

   A common design mistake is the failure to recognise the impact of changes to 'static' data. In this case, a security may have been considered 'technology' when the report was run, but was subsequently moved into a different sector based on information from a data provider. A good design would tag when the sector move occurred, and allow the end-user to easily decide whether to use beginning or end of period classifications.

   Importantly, if this information regarding changes to static data is not stored in the database from day one, it is almost impossible to reconstruct after the fact.

4. **The prices obtained from the database don't appear to match those used in official accounting statements.**

   With the proliferation of data, the lack of a solid data dictionary has become an increasing problem. Pricing is a typical example:

   - The same security can be priced at multiple exchanges and/or by multiple vendors.
   - Each vendor and exchange may offer several prices, including closing price, average price, etc.
   - Different portfolios may follow different pricing rules.

   A solid design should make it easier to navigate through the morass.

   Static data should explain clearly the pricing rule for each portfolio. Metadata should clearly define the meanings of every price. Building this into the data once will save from having to build it into multiple reports, eliminate potential for error, and generally create a sound structure that will stand the test of time.

In summary, the issues we encountered in this scenario include problems across:

- Taxonomy
- Data update logic
- Table structures and their relationships
- Tracking history of static data

In every case, the problem could have been avoided with strong governance and a solid, upfront design. And in every case, the sooner the issue is identified, the more easily it is remedied.

5
SOME CLASSIC DATA ISSUES IN THE INVESTMENT WORLD

CONSIDER SOME FAMILIAR ISSUES AND HOW THEY RELATE TO DATA AND DATA GOVERNANCE.

MANAGEMENT OF SHORT POSITIONS

Why do many systems seem to have a problem with handling short positions? It really has to do with the index structure of many holdings tables developed decades ago. Generally, they used primary keys that included a Portfolio_id, a Security_id and Date. So if you held both a long and a short position in the same security, you would produce a duplicate key – which most systems do not handle well.

Generally, the solution is to add a long-short indicator to the primary key of any holdings table, so two holdings of the same security may be stored without creating duplicate keys. But the cost of modernising or replacing older databases with this limitation is likely to be enormous.

The table on page 15 depicts the type of information Trustees should be able to readily access in order to satisfy regulatory requirements.

THE COST OF MODERNISING OR REPLACING OLDER DATABASES IS LIKELY TO BE ENORMOUS

Derivative short / long exposure sample: the type of information required by Trustees to assist with the FRS 102 Investment Risk Disclosure requirements that came into effect for accounting periods beginning on or after January 1, 2015.

Account NAV: 581,452,475.24						
Security Description	Positive Market Value*	Negative Market Value*	Long Delta-Adjusted Notional	Short Delta-Adjusted Notional	Gross Notional	Gross Delta-Adjusted Notional
Exchange Traded Bond Index Future						
Total Bond lndex Future	1,380,178	(1,395,968)	155,182,344	(70,542,282)	225,724,626	225,724,626
Over-the-Counter Bilateral Credit Default Swap						
Total Credit Default Swap	817,281	(134,493)	48,009,824	0	48,009,824	48,009,824
Forward Currency Contract						
Total Forward Currency Contract	2,194,299	(2,235,537)	256,171,696	(148,461,107)	404,632,803	404,632,803
Interest Rate Swap						
Total Interest Rate Swap	41,673	(949,519)	216,829,858	0	216,829,858	216,829,858
Total Derivatives	676,193,724	(219,003,389)	895,197,113	895,197,113		
Total Exchange Traded	155,182,344	(70,542,282)	225,724,626	225,724,626		
Total Over-the-Counter Bilateral			521,011,379	(1 48,461, 107)	669,472,487	669,472,487
Total Over-the-Counter Cleared			0	0	0	0
Percentage of NAV			116%	(37%)	153%	153%

THE RESULT? THE SINGLE MOST EXPENSIVE DATA CLEAN-UP AND CODE FIX IN HISTORY

NON-STANDARD REPORTING PERIODS

It is generally assumed that reporting periods are either days or calendar months. And this assumption has been 'hardcoded' into the design of most financial databases. New requirements to show accounting, performance or risk by week, or by non-standard time periods prove extremely challenging. Most databases continue to have this limitation.

Y2K

In this famous data problem, systems around the world – particularly date-intensive financial systems – failed to carry enough information to uniquely identify every date. "651101" could mean November 1, 1965 or November 1, 2065. The result? The single most expensive data clean-up and code fix in history.

COUNTERPARTY DATA

Lack of industry-standard counterparty information became widely understood as an issue with the financial crisis of 2008. Not only did many systems fail to track counterparties, but there was also no single set of identifiers by which counterparties were tagged. Many financial institutions simply maintained long internally-managed lists of brokers and traders, with little understanding of when multiple names really represented the same legal entity, or were owned by the same legal entity. This made understanding true counterparty exposure a challenge. With reporting such as the sample below, institutions can gain a clear view of individual and aggregate counterparty exposure.

The type of OTC counterparty exposure reporting used to assess exposure at the group counterparty level.

OTC Counterparty Exposure

Counterparty	Negative Market Value	Positive Market Value	Potential Future Exposure (Notional Add-on)	Positive Net Market Value (After Netting + Potential Future Exposure)	Collateral Held	Net of Collateral OTC Counterparty Exposure (With Potential Future Exposure Add-on)
Total GREEN BANK	-7,336,722	2,794,328	6,500,250	1,957,856		1,957,856
Total ABC BANK	-40,688,354	11,061,145	20,600,202			
Total CBA GROUP	-21,918,952		13,195,710	10,010,240	1,286,998	1,286,998
Total XYZ BANK	-7,031,893	11,374,144	19,540,305	23,882,556	3,540,000	20,342,556
Grand Total	**-76,975,920**	**38,425,326**	**56,650,997**	**27,127,410**	**3,540,000**	**23,587,410**

6
LOOKING AHEAD

WHILE IT'S IMPOSSIBLE TO PREDICT WHAT THE FUTURE IMPACT OF TECHNOLOGY ON FINANCE WILL BE, A FEW KEY ISSUES AND TRENDS ARE WORTH NOTING:

REGULATORY ENVIRONMENT

Regulatory requirements will continue to be a major driver of many data governance principles – including data accuracy, coverage, labelling and depth.

EXPOSURE DATA

Demand for better understanding of portfolio exposures and risks will mean a need for ever-improved data about securities. For instance, understanding exposure to a country will require knowing not just where your holdings are domiciled, but what securities you hold that sell to a given country, and what percent of their revenues are tied to that country, etc.

The report below shows the sort of exposure analysis one should be able to perform with good classification data.

PRIVATE ASSETS

Asset owners and managers will demand robust data, including balance sheet information, about all of their holdings regardless of whether or not the securities are publicly traded.

ASSETS AND LIABILITIES AS ONE

In the past, pensions have generally managed assets and liabilities on independent platforms. But the need for liability-based investment analysis will likely drive more warehouses that contain both sets of data, and require easy joining across them.

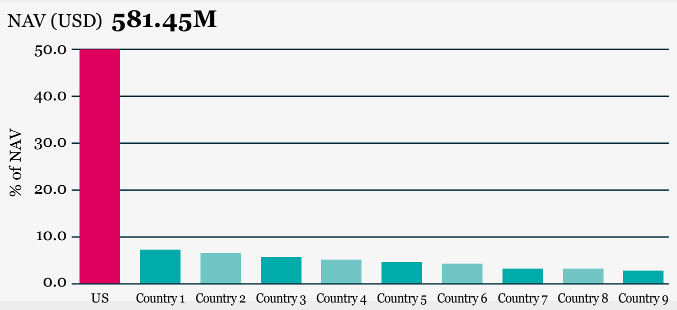
DATA GOVERNANCE BEYOND THE ENTERPRISE LEVEL

The counterparty issue described above shows the need for governance at the industry level (and this is in flight with initiatives such as Legal Entity Identifier). Some challenges cannot be solved separately within each firm. They must be solved collectively.

The type of exposure information used to assess risk and concentration across multiple risk dimensions such as asset, counterparty, country, currency, industry and issuer.

PORTFOLIO INSIGHT Demo Group Demo fund

Asset type	Forward Currency Contract	72.47%
Counterparty / Entity	JPMORGAN CHASE	0.23%
Country	UNITED STATES	49.45%
Currency	USD	54.87&
Industry	GOVERNMENT	37.78%
Issuer / Entity	JAPAN	5.05%
Issuer / OTC CTPY / Entity	JAPAN	5.05%



7
CONCLUSION

THE PENSIONS INDUSTRY FACES EVER-INCREASING NEEDS AROUND DATA AND DATA QUALITY. IN TODAY'S ENVIRONMENT, GOOD TRUSTEESHIP IMPLIES GOOD DATA GOVERNANCE. AND GOOD DATA GOVERNANCE, IN TURN, IS GREATLY ENHANCED THROUGH AN UNDERSTANDING OF SOME BASIC PRINCIPLES AND ISSUES AROUND FINANCIAL DATA.

SOME QUESTIONS HAVE NO SINGLE CORRECT ANSWER:

- what to store versus what to calculate on-the-fly?
- whether an enterprise needs a large warehouse, a reporting database, or some hybrid?
- can you refresh only the data that has changed, or is a full data reload required?

However, understanding the principles around data and data governance will help guide the fiduciary manager to the right decisions. Then, adhering to those principles will produce a secure foundation of complete, accurate, readily accessible information with which to govern.

UNDERSTANDING DATA WILL HELP GUIDE THE FIDUCIARY MANAGER TO THE RIGHT DECISIONS

DISCLAIMER

Material provided in this publication is meant as general information on matters of interest. This publication is not meant to give accounting, financial, consulting, investment, legal, or any other professional advice.

You should not take action based on this guide and you should speak to a professional adviser if you need such information or advice.

The publisher (The Pensions and Lifetime Savings Association) or sponsoring company cannot accept responsibility for any errors in this publication, or accept responsibility for any losses suffered by anyone who acts or fails to act as a result of any information given in this publication.

ISBN: 978-1-907612-34-3

Pensions and Lifetime Savings Association
Cheapside House,
138 Cheapside,
London EC2V 6AE

T: 020 7601 1700
E: plsa@plsa.co.uk

www.plsa.co.uk

October 2015