by experts
for experts

White Paper

Experiences of the hidden complexities of data standards integration

Summary

Public data standards are sometimes viewed as a panacea, but the reality is somewhat different. There are hidden complexities that often delay the adoption of the standard, or which stall subsequent migration to future revisions. The purpose of this paper is to share our experience as data migration specialists working with PPDM. We will discuss the challenges of both moving to PPDM for the first time and transitioning between different versions of the PPDM model.

Drawing on real life experiences, we aim to review the lessons learned. This will cover the importance of managing complex data relationships and working with expanding data sets. We will also explain how reusable data mappings and models can overcome some of the hurdles usually associated with traditional methods.

The data standards paradox

Adopting a data standard throws up a paradox: the price of progress is complexity. If the data standard remains fixed and static, adoption is relatively easy. But to accommodate the needs of its community, the functional capabilities of the standard must develop. This will also drive adoption, but it is this constant change that creates the very complexity that causes standard initiatives to stall.

Unless this functional evolution is factored into a standards adoption strategy, the task becomes too onerous. The result is that adopters sometimes freeze at various versions of the standard and this creates an unintended and unwanted legacy system.

Drivers of change

In the PPDM context, we have noticed two factors that have been key drivers of change. These are data explosion and data interpretation lineage.

Data explosion

We operate in an era of unprecedented change. This means that new devices are being used in every phase of Exploration & Production (E&P), gathering more data with which better decisions can be made.

Timescales are also collapsing. Once, drilling and logging data were distinct activities separated by days, but now they happen simultaneously.

New devices return more terabytes of raw data; and this data is also more sophisticated than was previously possible.

Metadata (in the Dublin core and ISO 19115 sense) are becoming ever more important in providing context. This has a direct impact on proprietary database design and functionality. It is also being factored into PPDM model development and this is why PPDM will and must evolve to promote adoption and use. The price of progress is growing data complexity.

Interpretation

There is also something unique about E&P, in that the assets remain hidden deep underground. There is no certainty that a prospect is viable before it is drilled, and massive investment decisions are made on interpretations of current and historic data.

Professional interpreters work with raw data to derive increasingly valuable knowledge that underpins how decisions are made. This is intrinsically complex because most interpretation is based on previous

interpretation, which can be wrong. To complicate matters further, theories of interpretation also change. So PPDM must not just store raw data, it must also store the accumulated knowledge.

Tracking the history of previous interpretations is also essential and we must record:

- When decisions were made;
- What information was used in making decisions;
- How were decisions made;
- Who made them and why.

Data governance, MDM initiatives and chains of custody all want answers to the 'when, what, how, who, why' question. This can identify inconsistencies and indicate where errors of judgment or poor data were used.

To provide such answers, data models develop and a time series data ownership graph becomes crucial. If we don't understand this process, complexity is buried; and we can't fix problems we can't see.

A PPDM example of complexity driven by change

This example is based on recent experiences building a PPDM data connector for a major client in the Oil & Gas industry, while investigating migration from earlier versions of PPDM. We will use the example of check shot surveys and how the data is represented in PPDM.

- PPDM 3.2 represents check shot surveys in a simple way with little data.
- PPDM 3.3 is far more functional. The WELL_CHECKSHOT and WELL_CHECKSHT_SRVY tables are renamed and they expand. For each point, the new WELL_CHECKSHOT_DETAIL now provides thirteen in place of seven attributes; the header record WELL_CHECKSHOT_SURVEY has thirty in place of six. What's more, seven of those are now foreign keys to various reference tables where yet more data can be stored.
- PPDM 3.4 has limited development, although some tables are renamed and the model starts to mature.
- PPDM 3.5 matures further and there are subtle semantic changes. Whereas 3.4 provided a ROW_CHANGED_DATE and ROW_CHANGED_BY, 3.5 now provides both a space for ROW_CREATED date and owner, as well as a ROW_CHANGED pair. This is typical of the kind of subtle change we'd expect to support data governance, but we must understand it. In 3.5, the implied semantics of ROW_CHANGED are different. If we missed this, let this complexity remain buried and continued to fill the changed date in a data loader, we could create serious issues. A later interpreter may believe that items have been updated, massaged or cleaned, whereas all we had done was load raw data.
- PPDM 3.6 is a big step-change that could cause all kinds of challenges. There is now no table called CHECKSHOT, or anything like it. The whole well check shot concept is entirely remodelled. Now, logically, check shot is handled as seismic data, which is what it is and all seismic is uniformly classified and stored.
- PPDM 3.7 applies four new attributes, ACTIVE_IND, EFFECTIVE_DATE, EXPIRY_DATE and ROW_QUALITY, to almost everything. Using these, we can record the lifecycle of these objects. It is the kind of complexity which would be hidden if we didn't understand that this is all about answering those 'when, what, how, who, why' questions.
- PPDM 3.8 continues with some potentially far-reaching changes, which need to be caught as early as possible to minimise the cost of their impact. The R_COUNTRY, R_COUNTY and R_PROVINCE_STATE tables are now deprecated and renamed with a Z_ prefix, to encourage use of the more flexible AREA tables. The PARENT_UWI and PARENT_RELATIONSHIP_TYPE columns have been removed from the WELL table, and well/borehole relationships must now be modelled using the WELL_XREF table. Some numeric values now have a higher precision, which could result, for example, in any tests using these columns to fail.

Managing the complexity

Now imagine that we had built PPDM data loaders for all these versions with the standard tools of the geologist-turned-data manager: Perl, Excel and SQL.

Nothing in such a methodology would help track those model improvements, nor would these technologies support any process for managing the changes. A PPDM loader developed using these tools would probably need to be rewritten for each new version.

And here's a bit of real complexity we don't want to hide. Building successive loads into successive versions of PPDM is not the entire task. What about data loaded into previous versions? We've loaded it, cleaned it, and it is now quality controlled data, so we don't want to have to load all this again. We need an upgrade path and a strategy that can build those upgrades. We need a plan to insert the relevant metadata that was missing in earlier versions. We need a consistent process of transform regeneration.

And this must ensure that the upgrade logic reflects the same business logic embedded in the data loaders. We need to ensure that the same data is loaded identically by different paths. The requirement is for a strategy that can simultaneously tackle initial and on-going model load and upgrade between versions. This is what we have developed.

A structured methodology

To make a robust and repeatable approach work, we use Transformation Manager, our data integration toolset. This is the approach we have adopted over many years in this industry:

1. Separate source and target data models and the logic which lies between them
This means that we can isolate the pure model structure and clearly see the elements, attributes and relationships in each model. We can also see detail such as database primary keys and comments. As exposing relationships is the key in handling PPDM and other highly normalized models, this is a critical step.

2. Separate the model from the mechanics of data storage
The mechanics define physical characteristics such as 'this is an Oracle database' or 'this flat file uses a particular delimiter or character set'. It is the model that tells us things like 'a well can have many bores', 'a wellbore many logs', and that 'log trace mnemonics' are catalogue controlled. At a stroke, this separation abolishes a whole category of complexity.

For both source and target we need a formal data model, because this enables us to read or write to database, XML, flat file, or any other data format.

3. Specify relationships between source and target
In all data integration projects, determining the rules for the data transfer is a fundamental requirement usually defined by analysts working in this field, often using spreadsheets.

But based on these or other forms of specification, we can create the integration components in Transformation Manager using its descriptive mapping language. This enables us to create a precisely defined description of the link between the two data models.

From this we can generate a runtime system which will execute the formal definitions. Even if we chose not to create an executable link, the formal definition of the mappings is still useful, because it shows where the complexity in the PPDM integration is and the formal syntax can be shared with others to verify our interpretation of their rules.

4. Error detection
To ensure that only good data is stored, Transformation Manager has a robust process of error detection

that operates like a series of filters. For each phase, we detect errors relevant to that phase and we don't send bad data to the next phase, where detection becomes even more complex.

We detect mechanical and logical errors separately. If the source is a flat file, a mechanical error could be malformed lines; logical errors could include dangling foreign key references or missing data values.

Next, we can detect errors at the mapping level, inconsistencies that are a consequence of the map itself. Here, for example, we could detect that we are trying to load production data for a source well which does not exist in the target.

Finally there are errors where the data is inconsistent with the target logical model. Here, simple tests (a string value is too long, a number is negative) can often be automatically constructed from the model. More complex tests (well bores cannot curve so sharply, these production figures are for an abandoned well) are built using the semantics of the model.

A staging store is very useful in providing an isolated area where we can disinfect the data before letting it out onto a master system. Staging stores were an integral part of the best practice data loaders we helped build for a major E&P company, and it is now common practice that these are stored until issues are resolved.

5. Execute a runtime link to generate the code required to generate the integration
This will generate integration components, in the form of Java code, which can reside anywhere in the architecture. This could be on the source, target or any other system to manage the integration between PPDM and non-PPDM data sources.

Managing PPDM change

Once we have mapped and executed non-PPDM to PPDM integration, what happens when a new version of the model is released? What is the process of migration and how difficult is it?

The answer depends on the degree of change but we approach it in the following way.

1. Load the new model
Transformation Manager will then generate a difference report that clearly identifies the elements that have been added, deleted, shortened and lengthened, subject to new constraints, described differently or any other change.

2. Replace the old model with the new model
This is a relatively simple task and we can then begin the process of generating a runnable transformation or integration component. Doing this will bring to our attention all those model features that have altered and, using the mapping capability, we can redesign the transformation.

Even in a worst case scenario (such as the type of complete redesign we saw for the check shot example earlier) we don't have to start from scratch and re-write everything. Our process of transformation regeneration means we can start from a solid platform, because the descriptive code previously written allows newcomers to get up to speed fast. It also means we can execute far quicker, since modification is only required for critical differences represented by the source and target changes.

When the transforms are created, Transformation Manager will also inform us if we are not writing all essential items to the target data store. This cuts down on errors and ensures a far more complete transformation. The software provides much greater accuracy, and a huge productivity advantage, over working with traditional hand-coded methods.

Using this approach, we simultaneously build both the new loader and the upgrade path from the previous version. This maximizes the investment already made in moving to PPDM, by re-using significant portions of logic and expertise.

Our experience is that this best practice approach exposes many of the complexities that otherwise will be a major obstacle to the on-going adoption of PPDM standards.

In conclusion

PPDM is an evolving standard and although this introduces complexity, with the right approach it can be simplified.

By creating industrial strength integration capability, we have shown that transformation regeneration is an excellent architecture for building a process to support on-going PPDM adoption and migration.

With care and foresight the complexities of data integration do not have to remain hidden. With the right approach they can be seen, controlled and addressed, enabling you to track the future standard and maximize the investment you make in moving to PPDM.

Contact us

ETL Solutions Ltd, Menai House, Parc Menai, Bangor, LL57 4HJ, UK
+44 (0)1248 675070
info@etlsolutions.com
www.etlsolutions.com

