Analysis by ETL …

November 1, 2009 datamgmt No Comments

Performing source-system analysis by writing ETL is just about the most expensive way you can do it; but many organisations do exactly that. Why? Because they "can't afford" to do their analysis the cheaper way. No, this does not make sense.

This is how it works…

In most data warehousing projects, business analysts are separated from data profilers, data modellers, technical designers, ETL developers, and every other possible sub-division of the task you could imagine.

We sometimes find brilliant business analysts, but sadly, they are rare. More usually the business analysts don't like to get their hands dirty, and present the designers with works of elaborate fiction which have some resemblance to how the system might work in an idealised world. Usually their documents are at a "high level" (actually meaning vague and inaccurate).

Maybe we get some help from data profilers, who run the data profiling tool on a tiny sample of the data. (One tool, in the top right corner of Gartner's Magic Quadrant, limits samples to 999,999 records. I suggest that this is completely useless for a database table of a quarter of a billion rows, and such scale is usual in the sort of enterprise that would feel the need for a data warehouse.) The data profilers dutifully print out the verbose reports from the data profiling tool and the technical designers ignore them (not through malice, or even laziness, but because they know that these reports will not be much use).

The designers are more likely to add a further layer of invention to produce "Mapping Specifications" based on the way the system must work (surely!).

Then the ETL developers build their ETL jobs, generally pretty faithful to the mapping specifications, but often with some elaboration to take care of exceptional cases (most of which never happen in the real data) and with some "corrections" to the mapping specifications which (surely!) must be wrong on these points.

Now we could do with some test data. It is, of course, too expensive to build test data to cover all the test cases (which have not been defined until now) so work stops while negotiations begin to get extracts of real data from the operational systems.

Eventually the data arrives. The people providing it have carefully selected it to represent the cleanest view of the source system. Testing commences.

Actually, what happens now is that analysis commences. The ETL crashes into an idealised version of the real world and… crashes. The data is very unlike what was defined and it just doesn't work. We don't get an end-to-end run because the data formats are not accurate and even the extracts fail. We patch up the extracts, one defect at a time, until – eventually – the data gets as far as the staging area. Unfortunately, the project is now beginning to run late, so we don't have time to correct the mapping specifications. As for the source systems analysis – well those guys are long gone.

So we've stumbled through the extract and now we can start on the transformation. Well, if we couldn't get the source data formats right what chance have we got with the relationships which exist in the operational systems (which have been patched and enhanced over the years, with no regard to good database design, object-oriented principles, user interface good practice, or any discipline which might have given us a nice database to work with)?

Many months later, after many late nights and weekends of work by the ETL team, we finally have something which appears to work.

But this is on the sample data. When the system goes live, the ETL is exposed to the data quality errors, missing reference data, old test data, clever bits of encoding within fields (which the users have invented to overcome shortcomings in the system) and real data volumes.

Everything fails again, there are recriminations, more late nights and weekends. Many months later the system finally goes live and users start running reports. By this time the users have no confidence in the data warehouse – it has got a bit of a reputation – so they question everything which does not match what they had before, even if the new reports happen to be more accurate.

work and where data quality and design flaws exist. But the ETL developers have finished their work, so their contracts end and they walk out of the

door taking the knowledge with them. No-one is going to pay for them to write it down.

This all sounds terribly cynical – doesn't it? But sadly it happens, over and over again. The tool vendors encourage this approach by the lie that their tool is the Silver Bullet (see The Mythical Man-Month by F.P. Brooks) that will solve all these ETL problems. The IT analysis and advice companies are accessories to the crime, re-enforcing the myths from the vendors.

It doesn't have to be this way. The application of realistic, sound, rigorous, good practices can overcome all of these problems, but this does require that managers understand how such projects work and don't just believe that pretending to be Alan Sugar or Gordon Ramsay is the best way to get results.

So what makes it better. First of all, we need people with skills across the range required for data warehousing projects. We want ETL developers who can talk to users, we want business analysts who understand the principles of good database design.

Then we carefully dig for real requirements. We don't just gather them. Quantity is not our goal. We don't want a spreadsheet with a list of 1,500 fields that appear on today's reports. We want to know what the users really need, what will save them the most time, what will improve the quality of the information they use. In the terminology of Agile development we want "stories" – the users' definition of what they need, why they want it, how important it is to them and the "acceptance criteria" which will enable us to know when we have delivered what they want.

The people doing the development must be involved at this stage. As they build working relationships with the users they can ask for clarification, they can show what they have discovered, and they can expose what they are building for the users to verify as they go along.

The team may use data profiling tools to find out the extreme edges of the system, the biggest and smallest values, the unusual patterns, the missing reference data, the orphaned transactions. The users will be able to explain some of the apparent strange values, and will discover why they have not always got the results they expected in the past. With the users' involvement, data quality issues can be fixed in the source systems. The data warehousing project is starting to deliver benefits before a line of ETL code has been written.

We may find that some master data management is required. Typically we will find different versions of the product catalogue in different applications. A project to sort out a single product catalogue will also save money in the business before a line of ETL code has been written.

With a proper understanding of the source systems the team can build a data model for the data warehouse which really models the work of the enterprise, not a fictional idealisation of it, and not a mirror of each operational system, but a process neutral data model which will be a valuable investment for years to come.

So, at this stage we have a robust target model (the data warehouse) and a good understanding of the source systems. We can now define the source to target mapping accurately, and, at the same time, build test cases to prove that the mappings work correctly. Yes, we build test cases, we do not just grab some real(ish) data and bash the ETL into it to see what happens. It does take a little time, but we automate our test runs so that the test case we build today can be run over and over again to keep on proving that the system is running correctly. The data volumes needed for this kind of unit testing are tiny, so test runs are completed quickly and can be run every time a new ETL module is added or an existing one is changed. (The ETL tools do not give much help in this, despite their 6-figure price tags.)

When we get a part of the data warehouse built successfully, then we can try some volume tests. Real data is useful here, but if it is not available, we use test data generators to ensure that our ETL and our data warehouse can handle real-world volumes (and a bit more). We may need to do some tuning. This may be as simple as adding some indexes, or it may cause us to rewrite chunks of our ETL code. Either way, we run our automated test suite, which now contains hundreds, maybe thousands of tests, to prove that we have not broken anything. Now the cost of building the tests is really paying us back.

At this stage the users can start running reports on the first part of the data warehouse. We can work with them to reconcile the results back to the source systems and to determine where any discrepancies lie. If we have done the preceding steps well, the new data warehouse will be right more often than not. When it is wrong, we can fix it and run our test suite to make sure we have not broken anything else. The users' involvement here builds confidence in the new data warehouse and smoothes the path to implementation.

So, if the first part of this article was cynical; is the second part hopelessly optimistic? No, it is not. We have done this and made it work, many times. The keys to success are:

• people in the team with broad and deep skills

• involvement of the users from start to finish

• openness and honesty

The tools we use encourage these characteristics, but no tool is a silver bullet – without the buy-in of the team (users and developers) no tool will make a data warehousing project work.

To develop and manage documents, we use a wiki, and we do not lock down the security within the team (it is secure from competitors and any see the current content and the history. Mistakes can always be fixed because the wiki keeps a complete history of changes, but the need to revert

to history is very rare in our experience.

We use a tracking system for tasks, risks, issues, enhancements, test cases and defects. Again, everyone can see and comment on all of these. If a problem is brewing, then the team knows about it before it becomes a catastrophe.

We do not accept quick and dirty fixes. Technical debt is paid off as soon as possible, it is not allowed to accumulate. We are not afraid to "re-factor" to make our database and our code clean – we can easily prove that it is still working by running our automated tests.

Of course, we also use data modelling, data profiling, ETL, version control, and other tools, but we do not expect the tools to solve all our problems, and we choose free or low-cost tools where they do the job.

And, we do not do our analysis by writing ETL. We do analysis the much less expensive way, using our brains before we write all the code.

This article was originally published on BIonRails, another Data Management & Warehousing website

Editorial *Agile*, *Analysis*, *BI on Rails*, *Data Quality*, *ETL*, *Mythical Man Month*, *Process Neutral Data Model*, *Project Management*, *Technical Debt*

datamgmt

You Might Also Like

Sequent Values and Principles
November 25, 2013

Addressing Business Intelligence Data Quality
September 24, 2012

Inspirational Data Visualization
September 14, 2012

« The project wiki – a cost reduction tool

Creating Graphics for BI Reports in Ruby on Rails »

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

Post Comment

This site uses Akismet to reduce spam. Learn how your comment data is processed.

CONTACT US

Call:

Fixed: +44 (0) 1458 850 433

Mobile: +44 (0) 7990 594 372

Skype: datamgmt

Chat:

Skype: datamgmt

WhatsApp: +44 (0) 7990 594 372

Add our details to your contacts

Registered Office:

Manchester House, High Street,

Stalbridge, Sturminster Newton,

Dorset, DT10 2LL

United Kingdom

Company Details:

Registered in England and Wales

Registration Number: 3526504

VAT Registration: GB724448236

Directors: DM & HJ Walker