

Business Intelligence...

... all we need is a good query tool - right?

Actually no, that's not right. But it is a common mistake to make. A good query or OLAP tool is certainly an important part of a successful BI initiative. It's the piece that most business users are familiar with - however it is just the tip of the iceberg.

Unlike icebergs though, it's the *absence* of anything below the waterline that will sink your BI initiative.

Talk to any organization that has implemented a successful Business Intelligence initiative. Without exception they will all tell you that you need to first build a well architected data warehouse or data mart infrastructure. Read a book. Listen to the industry gurus. They all say the same thing.

But one of the major causes for long term failure of Business Intelligence projects is the failure to build a solid **architectural foundation**. So why is it that many organizations get it wrong? It's often because it is so easy to use a query tool to start delivering a few reports directly from your operational databases. Pretty soon, those few reports have grown to a few dozen. Business analysts are relying on them extensively and ask for additional reports. At some point this house of cards comes crashing down, usually for a number of reasons:

Data quality: Let's face it, there are *known* data quality issues in your operational data. What about the issues you *don't* yet know about? Bad or missing data can lead to bad business decisions. You know where that can lead to.

Complexity: Those first few reports were fairly simple. The new requests are not and may be impossible to produce. Data is stored in different databases, on different systems and in different formats.

Security: How can I give Mary from accounting access to summary payroll data without her seeing the detail?

Consistency: Why are my numbers different from yours?

Compliance: That GL report is going to be used for financial reporting. How does that fit in with Sarbanes-Oxley? Should I risk my neck?

Documentation: What reports are available? What do they mean? What data sources were used and how were calculations done?

On-going maintenance becomes a huge resource and cost burden.

Version Control: is non-existent. Keeping track of it all is impossible.

The problem though, is that it doesn't happen overnight. The signs are there from early on, but a few band-aid fixes keep things humming along nicely: build a summary file here, write a quick RPG program there. Get Jim to conjure up the complex SQL to join those 7 tables together, hire someone new to help with the report backlog.

Suddenly, this is a large, mission critical application that stops delivering value.

There are so many requirements that just can't be satisfied and it's a nightmare to maintain. You realize that you *have* built a data warehouse of sorts, or at least some data marts. But you did this without a plan and consequently it has no coherent architecture, no controls and very little documentation. You *really* hope Jim doesn't quit!



On top of all that, you've just been advised of an acquisition - and fairly soon you're going to need to merge their data with yours on the same reports! And later this year you're going to upgrade to a new version of your ERP software. All of your reports will need to be reviewed and probably revised. And the additional query workload on the system is slowing it down tremendously. Everyone's complaining and the trucks aren't getting out on time and...

It's about now that you realize you should have designed and built a proper data warehouse!

Building a Data Warehouse...

... it just requires some basic SQL - right?

Not quite! A data warehouse (or data mart) is much more than a collection of database tables.

It needs a lot of careful planning: Define the business requirements. Design the database. Identify and describe the data sources, transformations, validations and business rules. When you've finished this task, you'll probably find it took at least twice the resource you expected (time, people, and of course cost), so you're behind schedule. At least you now have a set of requirements, architecture and a plan. Now you need to implement it - quickly!

Building the database tables and indexes from the logical model isn't too hard. However, writing the Extract, Transform and Load (ETL) logic/code can be very difficult, especially if you expect to achieve all the requirements via the SQL language. SQL is great at pulling data OUT of a data warehouse. It's not so hot when it comes to putting data IN to a data warehouse.

There are many reasons for this - but let's just discuss two simple ones...

- **Validation.** We need to somehow identify and set aside the 'bad' data. An SQL statement can only generate one result set - either the good data OR the bad data. This means we need to process the same source data TWICE - once to identify and set aside the bad data and then to process and load the good data! What will that mean for performance? And even then we only have a set of bad data, with no indication of what is actually wrong with it!
- **Inserts vs updates.** You will undoubtedly need to handle both (hopefully automatically). SQL can perform one or the other, but not both in the same step. The common solution to this involves a multi-stage process, with intermediate staging tables. More inefficiency! And what about loading multiple tables (e.g. detail and summaries) from the same source data? You guessed it - more double handling!

There are many similar requirements that SQL has difficulty in handling, but which can be more easily achieved in a traditional programming language. So these are your choices from an ETL perspective:

- Go with SQL or SQL based tools and be very limited in capabilities, data validation, error reporting etc.
- Write your own program code - if you have the developer resources and lots of time to spare.
- Buy a full-function ETL tool.

There are a number of very good ETL tools available. However, when you research them, you'll find that most will not run natively on your System i. RODIN is the exception. **It is the only full-function ETL software suite that is designed from the ground up for System i, iSeries and AS/400 systems.**

But RODIN is much more than just an ETL tool...

Features such as parallel loading, metadata, change data capture, version control, change management, impact analysis and multiple environment support, coupled with rock solid reliability, superb performance and great product support have made RODIN the premier solution for ETL & Data Warehousing on IBM i

Allow us one hour of your time. We'll schedule a private web demonstration just for you and show you how RODIN will significantly reduce the effort in building and maintaining all aspects of a data warehouse.

Call 1-866-RODIN-DW to schedule your web demonstration. You owe it to yourself and your company to see why so many other organizations, including industry leaders such as HSBC Bank (worldwide), Fiserv, Office Depot, Discovery Channel, Caterpillar, Sysco and many others use RODIN to build and manage their data warehouses and ETL processes.

RODIN
Data Engineering
EXTRACT ⇨ TRANSFORM ⇨ LOAD ⇨ MANAGE

Developed by



www.thinkrodin.com