GETTING REAL

Peter Scott says companies have to be sure of the need before they insist on real-time business intelligence.

My car does real-time computing (or at least I hope the engine management, traction control and ABS systems are real time!). My mobile phone is real time, even some of my home entertainment gadgets are real time – but what about business intelligence systems?

In the early days of BI (or was it called decision support back then?) reporting data for the current month was an achievement. As techniques and technologies evolved, companies moved to report on the previous day but were still reliant on an out-of-hours batch process to move data from the source systems and store and aggregate it in their data warehouses and data marts.

But with 24-hour business days and the need to report across multiple time-zones, the traditional batch window is being squeezed into near non-existence. New data extraction and load paradigms have also been developed to trickle-feed reporting systems. But are these really real time – and does it really matter?

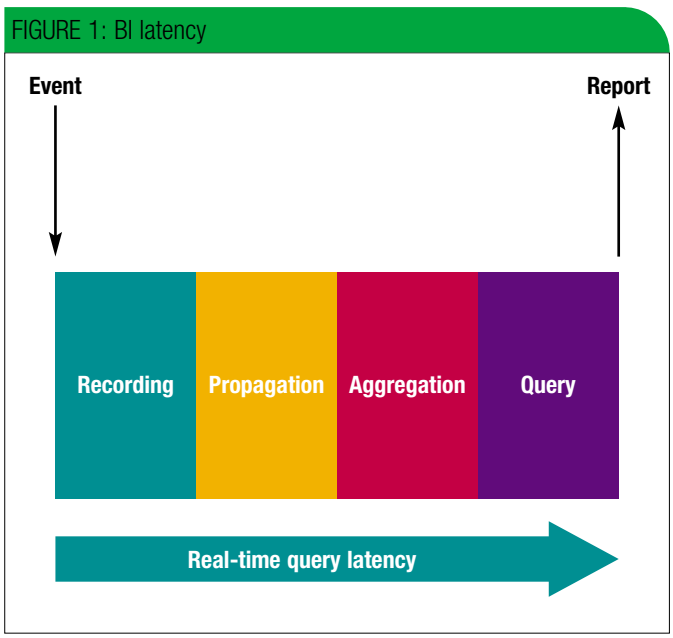
In reality, there is always going to be a degree of latency between an event happening and it being recorded on a transactional system, propagated to the reporting system, loaded, and finally aggregated on a data warehouse before being ready to query. Add to that the delay between the user issuing a query that accesses this new nugget of information and the result being returned to the user, and you find a discernible lag between event and observation.

True, there are things you can do to mitigate this lag but there will always be a couple of steps in the process where reducing one time period increases another – for example, you can shorten report execution time by aggregating the data to better answer the query, but this step adds more time to the data load and aggregate phase.

Furthermore, different types of incoming data can have differing latencies. For example, a new transaction may have a shorter latency than a change to a regional grouping of stores that may need to be propagated through many layers of historical aggregation tables (see Figure 1).

The division between data acquisition and data presentation presents a key question that any company needs to answer when contemplating moving to a real-time BI system: are you just interested in providing, as rapidly as possible, information based on already acquired data, or do you also need to report on newly added factual information?

FIGURE 1: BI latency



For example, a worker in a mobile phone company's call centre may like to see information about 'how good a customer is' and the likelihood that someone within that customer's demographic would switch to another supplier.

This information needs to be presented quickly (whilst they are speaking to them) and preferably within the context of the line-of-business application that manages the customer interaction. However it is unlikely that any of the metrics used by the call centre agent for customer profitability and predicted churn would be influenced by events of the last few hours.

Conversely, a system that is using embedded BI functionality to alert to potential fraud (stock markets and credit card companies are possible example users) may well need to know about up-to-the-moment activity.

Both examples require rapid access to information, but only the latter one also needs rapid acquisition.

Data acquisition

Chronologically, data acquisition comes before data presentation, so let's look at that first.

Here, the goal is to provide reportable data as quickly as possible. There is some truth in the saying that 'the fastest way to do something is to not do it at all and if you have to do it, do as little as needed'; in this case if you can avoid processes that extract, transform and load data, you implicitly avoid a large amount of moving data – and moving data is slow.

Although computer processors have become a lot faster over the years, the increase in network speed and the reduction in disk access time have not kept pace; pushing bytes around a system is a time-consuming process.

It is also conventional wisdom that transactional systems – the traditional source systems for BI applications – are optimised for processing single transactions quickly and not optimised for the sort of queries that require access to many individual transactions; this conventional wisdom also tells us that simultaneous reporting against transactional data sources is detrimental to transactional performance.

So, it appears there is a necessity to at least extract the data. However you can be clever about this.

Often transactional systems have standby systems ready for use in the event of a failure of the live system; these standby systems are kept in line with (or at least a known and acceptable amount behind) the live system and are a potential reporting source for recent transactional data without imposing a reporting burden on the production transactional system.

Unfortunately, not all vendors' databases are capable of permitting reporting against a standby database, but you can use similar replication techniques to build your own source database of recent transactions. This approach isolates your data from the transactional system so whatever queries you run against your copy will not impact the live system.

But, being a copy of the transactional data, it will not be optimally structured to answer queries such 'What volume of shares has customer X traded this fiscal year?'. Here you can cheat and change the question to 'What is the volume of shares that customer X has traded up to yesterday, added to their trades for today?' and look in two places for the answer – a pre-aggregated table in the data warehouse for the historical question and the transactional copy for a very narrow (one-day) slice of history.

This approach reduces the amount of information you need read to answer the query. Of course, at some stage you will need to load the data warehouse with the recent transactions and aggregate them, but that can happen later.

Splitting recent data from the historical data also allows you to be inventive with the way you store information. For example, you could store recent data on high-speed storage or even in memory and keep the older, potentially pre-aggregated, data on lower-cost but slower disk.

Although it is a good technique to capture database changes and store them in a replica database for querying, it is not always the best way to go about low-latency BI. Often you are dealing with distributed systems with messaging between components, and here you may be better served by making your real-time BI data source a message consumer in its own right; that is, build your high-performance query database as part of the distributed transactional system.

Real-time reporting

Whether or not you need to build real-time data acquisition into your BI system depends on circumstances, but real-time reporting is another matter – getting at the answers quickly is the whole purpose of real-time BI.

Traditionally, BI reporting has been delivered through a mixture of:

- Pre-built reports, often created as part of a batch process.
- Performance measurement dashboards.
- Dynamic reporting through standard desktop reporting tools, be they web-based or thick client-based.

In my view, none of these delivery methods are suitable for real-time business intelligence. Reports rapidly become obsolete and need to be refreshed; dashboards should only be used to provide a high-level view of the 'health' of an organisation or a process, and are not suited for rapidly changing detailed information; and query tools are just too cumbersome to use quickly.

Call centre agents do not want (or need) to slice-and-dice data information about a customer, or even create an *ad-hoc* report – they want to know the answer to specific questions that allow them to do their jobs better. And they probably want to get these answers without even knowing they are using a BI system.

The best way to present real-time BI is to embed it into the applications the agents use to do their jobs. That is, build query components to answer specific questions and link them to the business application by using whatever service oriented architecture, Web 2.0 or Software as a Service paradigm or phrase is appropriate to your organisation. Often these components will present textual information such as call centre scripting or suggested upselling opportunities and will not appear at all like traditional BI reports.

An important point to note about real-time BI reporting is that real-time response is only needed for a few specific queries; most queries will use the traditional reporting tools and dashboards though they may well be able to use real-time-acquired data if it is available.

In summary, real-time BI depends on acquiring the data as soon as possible after an event occurs and processing it so that it is available to report against, but there is always going to be a degree of latency due to the physics of moving data about.

You can be creative with how you move it and how much you move. You can also be creative about how you present the data – you can remove the ability of the users to change the question (which removes 'thinking time' from the process) and you can optimise the query to perform quickly.

But before you proceed down the path of squeezing the latency out from your end-to-end process, stop and think: does knowing something sooner – or even at all – give you a business benefit? That is, can you take an action based on this knowledge and do you gain more from taking the action than the cost of acquiring the knowledge?

● *Peter Scott is a consultant with UK-based business intelligence consultancy Rittman Mead Consulting. Email: peter.scott@rittmanmead.com.*

● *If you would like more information about this article or any of the products or companies mentioned in the article, please contact us at info@evaluationcentre.com.*