Business Intelligence—NoSQL… No Problem

May 2012

A White Paper by

Dr. Barry Devlin, 9sight Consulting
barry@9sight.com

Business Intelligence (BI) is traditionally associated with the relational model and SQL as a query language. This linkage is understandable from an historical perspective. It is also at best restrictive and in some cases downright misleading. SQL queries and relational databases are, without doubt, at the heart of core business data integration and consistency across the enterprise. But the emergence of big data demands that we expand our perspective.

Modern business requires flexibility and agility in its use of information. It is no longer possible in all cases to know the long-term direction and outcome of any initiative. But, data must still be managed. NoSQL databases, and the document-oriented variety, in particular, promise this desirable combination of features. In this paper, we examine MongoDB, one of the leading document-oriented databases, to understand how this emerging technology can be applied to BI. And we explore the positioning of such technology in the new architectural vision of pervasive support for decision making.

Two current customer examples round out the paper, showing what is currently possible, both in Web-centric businesses and in a more traditional telecommunications environment. The conclusion is not only that NoSQL has a role in BI; rather, that in some key areas of business need and data processing, document-oriented NoSQL offers a powerful alternative and extension to the traditional relational approach.

Contents

Sponsored by:
10gen, Inc. / MongoDB
www.10gen.com

BI—business needs and relational roots

Business intelligence (BI) originated (as data warehousing) in the mid-1980s. As I described[1] in 1988, the primary driver for data warehousing was the creation of an integrated, consistent and reliable repository of historical information for decision support and reporting. The architecture proposed as a solution a *"Business Data Warehouse (BDW)… [a] single logical storehouse of all the information used to report on the business"*. The reconciled, cleansed and largely normalized BDW data was copied and transformed from the operational environment using extract, transform and load (ETL) technology. The envisioned platform for BDW implementation was the relational database management systems (RDBMS) becoming mainstream at that time.

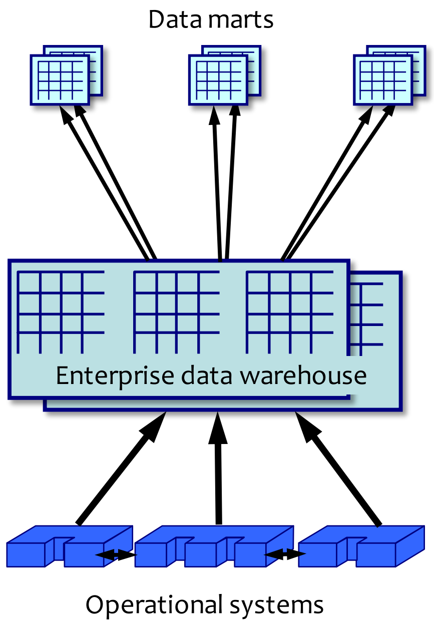
While a single logical—never mind physical—storehouse of decision support information has seldom been achieved, some key principles have remained constant over the intervening two and a half decades. First, information for use by decision makers has continued to be extracted from its operational sources and served from a separate, independent environment. This principle arises from the desire of business users for a stable (over time) and well-integrated view of key business metrics. It has also long suited IT as a way to ensure the performance, stability and security of operational systems.

Second, BI information has generally been carefully modeled and structured before making it available to business users. This approach arises from the need to integrate and cleanse data from sometimes suspect sources and the desire to simplify the information provided to the business.

Third, with some variations and extensions, the relational model and its implementation in RDBMSs continue to be used as the basis for decision support and reporting systems. This stems from the observation that business users are very comfortable with the concept of data in rows and columns—as evidenced by the undying popularity of spreadsheets—although the notion of keys and, in particular, joins remains foreign even today to most users.

Thus, the architectural model[2] still in vogue after all these years remains as shown in figure 1. While somewhat idealized, we continue to see implementations based on a layered structure of operational sources feeding an enterprise data warehouse (EDW) , which, in turn, feeds data marts as the principal sources of decision support for the business. In practice, of course, we find multiple EDWs, data marts fed directly from the operational environment as well as a multitude of other aberrations. But, the principle has remained largely intact… until the advent of big data, and all it entails.

Figure 1: *Classic data warehouse architecture*



Why big data changes the BI landscape

Big data presents a definitional nightmare. Its size, by which it is defined, is a movable feast. Big data, according to Wikipedia[3], "*…is a term applied to data sets whose size is beyond the ability of commonly used software tools to capture, manage and process the data within a tolerable elapsed time. Big data sizes are a constantly moving target currently ranging from a few dozen terabytes to many petabytes in a single data set.*" Its other characteristics—structure, content, usage and more—are equally as variable. And yet, big data is forcing a deep rethink of many aspects of computing, particularly business intelligence. So, what is going on?

From a business perspective, big data offers new insights and new ways of doing business:

- ***Build on demand:*** Big data volumes enable accurate prediction of demand for products
- ***Market on steroids:*** Direct and (near) real-time access to what customers and prospects think about the brand, products and more from social media data has reinvented marketing
- ***Revamp the supply chain:*** React to market changes and emerging customer behavior, and propagate market events through the supply chain

- ***Service in anticipation:*** Usage records from clickstreams to power usage accelerate service actions and reactions to increase customer satisfaction and reduce waiting times
- ***Reinvent the business:*** Sensor data sent by everything from consumers' automobile engines to cell phones is enabling businesses to create totally new business models

For entirely Web-based businesses, the above business directions may be business-as-usual in many cases; for them, it was a case of inventing—rather than reinventing—the business. However, more traditional businesses are being forced to change by the ubiquity of the Web. Some few are actually choosing to change. In either case, given that the Internet and big data are, by their nature, technical drivers, the involvement of IT is mandatory. In fact, in the most successful of cases, it is rather more than involvement. The relationship between business and IT becomes symbiotic, with both sides benefiting immensely from the partnership. However, for IT in traditional or transitional businesses, big data poses challenges to the *status quo*:

1. ***Flexibility of usage:*** New business approaches and ongoing changes in data sources demand increased agility in managing and processing data, and integrating new, continuously changing sources of data, requiring more advanced analytics to process
2. ***Elimination of copies:*** The volumes and sources of big data make it increasingly difficult to pursue the old approach of making copies of source data for every need
3. ***Speed of delivery:*** With users demanding ever-closer to real-time data for certain types of decision making, moving data through a layered architecture becomes more difficult
4. ***Operational decision making:*** The ultimate goal for delivery speed in some cases is moving decision making completely inside the operational processes

The consequences are twofold. First, the architecture shown in figure 1 can no longer serve as the only approach to delivering BI. Points 2-4 above strongly suggest that the layered structure be eliminated or dramatically simplified. Second, the hegemony of the RDBMS is broken. Points 1 and 4 beg for new technologies to provide agility and the ability to handle different types of workloads in a single technology. Which leads us to NoSQL, discussed in the next section. These consequences also lead to an emerging data warehouse architecture, shown in figure 2, consisting more of data pillars rather than data layers. Each pillar is defined by the type of data and the analytical processing required. The pillars range from streamed data that never lands on disk to flat files, and every kind of data structure in between. Data from the operational environment is fed in parallel into one or more pillars and also flows between pillars as needed. The environment is supported by extensive metadata, both business and technical. Users have a virtual, business-oriented view of the data, wherever and in whatever structure it resides, both in the warehouse and in the operational systems below.

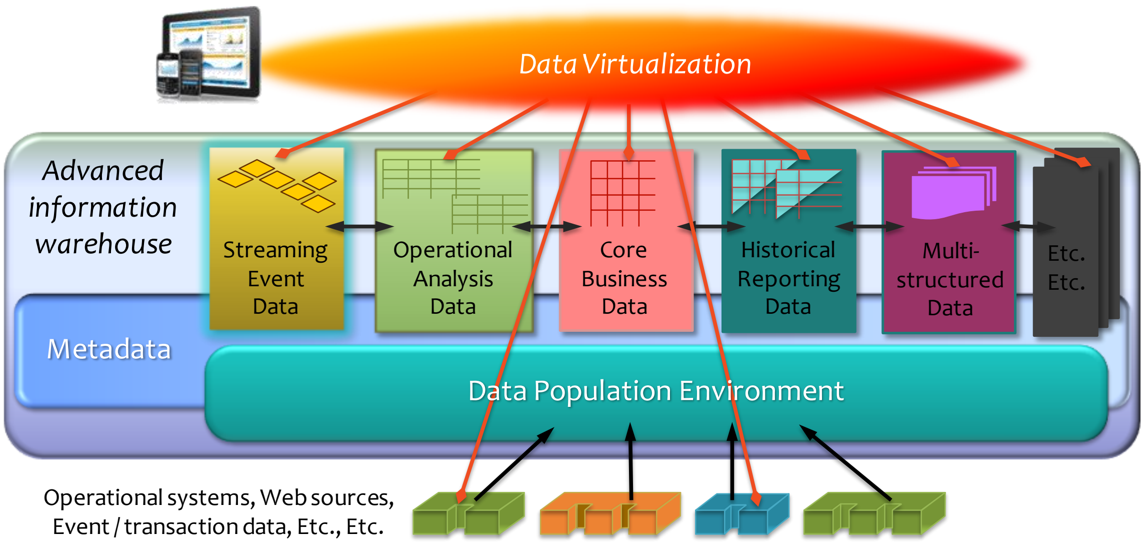

Data Virtualization
Advanced information warehouse
Streaming Event Data
Operational Analysis Data
Core Business Data
Historical Reporting Data
Multi-structured Data
Etc. Etc.
Metadata
Data Population Environment
Operational systems, Web sources, Event / transaction data, Etc., Etc.

Figure 2: *Emerging data warehouse architecture*

NoSQL—the whys and the wherefores

NoSQL, like big data, is a term that means different things to different people. At the most basic level, there is even the question of whether it stands for "no SQL" or "not only SQL". A typical definition refers to database management systems (DBMS) that differ from the classic relational database in one or more of the following: depart from the relational model as defined by Ted Codd[4], are not fully ACID (atomicity, consistency, isolation, durability) compliant, use something other than SQL as an interface language or do not have a fixed data schema. This is clearly a broad church, and has led to many existing graph, object-oriented, XML and other non-relational databases being rebranded as NoSQL. For the purposes of this paper, we focus on a database approach called document-oriented databases, a specific subset of the NoSQL world.

But first, for the relationally-oriented reader, some brief background description is in order. By the mid-2000s, the larger Web vendors were struggling with the ability of relational databases to handle the volumes of transactions received, distribution over multiple servers and flexibility to changing data needs. They began to develop new approaches based on a simpler key-value model (e.g. Google Bigtable and Amazon Dynamo), with less database management overhead. In fact, both actually avoided calling their work databases. The key-value model simply means that there exists one indexed field that is known to the DBMS as a key. The remainder of the record—the value—is opaque to the DBMS and its content managed entirely by the application. This is so basic that database gurus may debate whether this is truly a database, but let's call it a DBMS for simplicity. As data management needs grew, developers began to introduce and expose to the DBMS more structure within the value entity and additional keys, leading to more elaborate databases, of which document-oriented (e.g. MongoDB and CouchDB) are most directly relevant to BI because of their more flexible and extensive search and retrieval functionality.

We now focus on MongoDB as an indicative example. First, note that document-oriented has nothing to do with typical textual documents! "Document" in this case refers to JavaScript Object Notation (JSON) documents, a text-based open standard that can represent simple data structures. The following is a document storing a blog post I might write about this White Paper, stored in JSON:

```
{       author: {
                first_name: "Barry",
                last_name: "Devlin"  },
        title: "Business Intelligence - NoSQL… No Problem",
        text: "Business intelligence (BI) originated in the mid-1980s... ",
        keywords: [ "nosql", "database", "data warehousing" ] }
```

We see here the main characteristics. The objects are name : value pairs, which can be nested, as shown in the author entry. We also can have arrays, such as the keywords object above. The structure as shown above is a rather inefficient storage mechanism in some respects. First, it's text, and we need numbers and Boolean as values, which can be stored more efficiently. Second, every document stores the name of every value, as well as the value itself. MongoDB uses binary-encoded JSON (BSON), which addresses the first point. On the positive side, the fact that no storage is allocated for fields that do not exist in a particular document, means that in very sparse data sets, significant savings in storage in comparison to relational databases, where space must be reserved in every row for every field whether populated or null.

All of the data of the blog post is in a single structure, stored together. Fans of normalization will immediately note that if I have written many posts, it might be a good idea to keep all author information separately and replace the author field above with a pointer. A good database designer will also recognize the tradeoffs between the two approaches: put simply—storage vs. processing and structure vs. agility. Keeping all related data together in a single structure is attractive for big data volumes spread across multiple, dispersed servers. Distributed update, with all its overheads to maintain data consistency, is avoided. Reading the data occurs in one place, avoiding relational joins.

Furthermore, if at some later date, the blog provider decides that storing a Twitter name for authors is a good idea, they simply add another name : value pair into the author object. Future entries can store that information; past entries don't have it, but can be updated with it, if required. Compared to a typical relational database, where a schema change is needed to do this and the database must be rebuilt, this approach gives added flexibility to the developers. The database doesn't see the change, although the application program must, of course.

With the decision to structure the value field in JSON, it now becomes possible to index on any of the name : value pairs, or on any combination of them. As with all indexing, the tradeoff is write vs. search performance.

Along with the database, there exists a language for inserting, finding, updating, etc. data. Of course, by definition, the language is not SQL. In this case, it is a procedural, Java-like data manipulation language. However, one can write the equivalent of many SQL statements in this language[5].

Given that we're working with big data, it makes sense that document-oriented databases provide distributed, replicated deployment models. Asynchronously updated replicas of a primary, updatable data image can provide high availability and scalability for client reads, provided eventual consistency of information is acceptable—as is the case in many social media and similar applications. Large document sets can also be split (or sharded) over multiple servers and automatically redistributed when additional servers are added for additional scalability.

So, having understood a little of the technology, let's examine the implications for BI.

NoSQL in the land of the Business Intelligents

In order to position a NoSQL, document-oriented database like MongoDB in Business Intelligence, let's return to figure 2. NoSQL emerged first in response to high-volume, adaptive operational demands. Clearly, therefore, it fits in the lowest layer. With the modern multi-format approach of the advanced data warehouse, it is also possible to position NoSQL in any pillar where its strengths are needed, provided that the population, metadata, data sharing and virtualized access needs are met. These two placements map nicely to a pair of very different modern BI needs.

Operational Analytics and BI—access to live data

Allowing BI access to live operational data has traditionally been strongly discouraged for a number of reasons such as performance impact on operational users, restricted or low-quality data in the operational systems and possible security concerns. While such issues still exist to varying degrees in many environments, two factors are combining to make such direct access more urgent: exploding data volumes and users' demands for ever closer to real-time data. Increasingly, these two factors make the old solution of copying data into a data warehouse or even an operational data store technically difficult and expensive.

An operational NoSQL document-oriented database can be opened up to BI use in two ways that mitigate the performance concerns mentioned above, because the horizontal scalability of the database allows the easy addition of servers to handle the extra processing load. The first approach is simply to shard the database over additional servers, spreading both the operational (read/write) and informational (read-only) load and providing real-time data access for BI needs. As with all access to live operational data, queries should be carefully controlled—restricted to those that use indexes to find and deliver relatively small subsets of the data. The second approach is to create database-managed replicas of the operational database, used only for BI needs. In this case, because replication is asynchronous, there may be some delay before updates are propagated, so this approach supports near real-time BI needs. With the use of data virtualization technology, data from the operational environment can be joined with data in the informational environment.

The use of NoSQL does not change data quality or security concerns, of course. However, an added bonus is that that agility of the database in supporting changes to operational data structures is passed on directly to the informational environment. Fields added in the operational environment are immediately and automatically available in the informational area without any need for added design work or schema change.

Special purpose data marts

Figure 2 shows a representative sample of the types of data and function supported by the different pillars of the Advanced Data Warehouse (ADW). Of these, core business data is the successor of the traditional EDW layer, containing the consistent, quality-assured data that is paramount to the overall governance and reporting of the organization. Given the highly structured and modeled nature of such data, a NoSQL database is not a viable choice. Streaming event data is, by definition, not a database. The other pillars named above correspond to traditional data marts whose technological implementation and structure are chosen to meet the specific needs of their users. In this area, NoSQL databases can find a role.

Multi-structured data is the most obvious candidate for implementation in a NoSQL document-oriented database. Multi-structured data refers to data that contains a mixture of different basic data types such as numeric, text, image and so on. XML is multi-structural data. So is a JSON document. Any analytic need for which the JSON document structure is suitable is thus a candidate for storage in a NoSQL document-oriented database. For a purely logical perspective, the creation of replicas of operational databases described in the previous section is an example of a special purpose data mart, where data population is managed by the DBMS rather than an ETL tool.

As mentioned above, some of the data manipulation language in NoSQL is more procedural than the declarative approach familiar to relational users. This was a design choices that allows developers to reason more easily about performance aspects of distributed databases, but has two implications for use. First, we can see that the separation of logical data meaning and physical storage that enables business users (with some training) to use relational databases for *ad hoc* querying or as a basis for generic reporting tools is missing. The result is that relatively easily-understood SQL queries such as:

```
select customer_id, sum(price) from orders group by customer_id
```

require more programmatic thinking:

```
db.orders.aggregate( [
{ $group: { _id: “$customer_id”, total: { $sum:“$price”}}} ])
```

Such data marts are thus more suited to specialized, production analytics rather than general business-user access. This is analogous to the trade-off made in the relational BI environment: direct-access data marts are built on denormalized tables for a combination of speed and simplicity of access, whereas the EDW and specialized data marts often have more highly normalized structures. The second implication is related to the production use of such marts. With such a powerful procedural approach, together with rich APIs, we can implement tighter integration of the operational and informational systems. This is important in the modern environment where the outcome of analysis must be automatically fed back into operational environments such as web pages, ad servers, decision engines and so on. We see this explicitly in the customer examples that follow.

A tale of two customers

Traackr—business intelligence on Web influence

Traackr might be described as a model Web company. Small, but perfectly formed, its business springs entirely from life on the Web, information gleaned from there is its raw material and its products provide the intelligence demanded by modern businesses on their image on the Web. Traackr aims to answer a seemingly simply question: how can I measure and (hopefully) improve my brand or product image on the Web? Traackr’s philosophy is that image stems not from Web pages, but from the people who populate them—the influencers who create and spread that image.

Clients—often marketing groups and agencies—who sign up for Traackr’s subscription service create sets of keywords that characterize the subject of their interest, such as a brand, product or concept.

Traackr searches for these keywords across a wide variety of sources, such as blog posts, Twitter, Facebook, Flickr and YouTube (tags and descriptions) and more. Rather than returning these entries, Traackr groups them by author and then rates the influence of each author by a statistical algorithm based on reach (the size of the author's audience), resonance (how much the author's material is rebroadcast) and relevance (a combination of how often an author posts on the keyword topics and how often and how recently he/she has posted). The result returned to the user is a ranked list of typically 50 influencers that can be tracked over time. Clients use this information to decide which influencers they should reach out to, how they could use influencers to change brand image and to track if their efforts are bearing fruit.

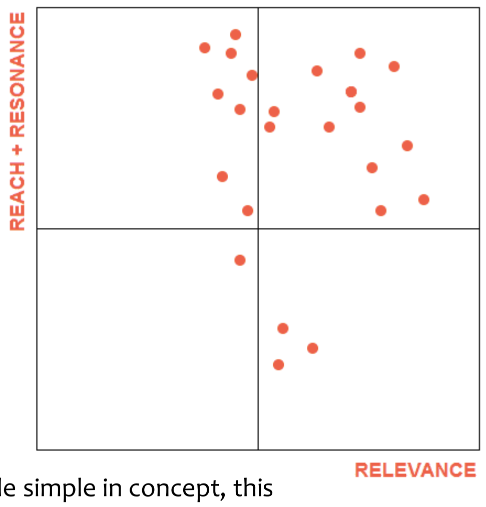


Figure 3: *An example of Traackr's influencer analytics*

From the point of view of its clients, Traackr is simply a BI system to measure and track who are the most influential people on the Web in their area of interest. Figure 3 shows an example of the type of analysis provided. While simple in concept, this requires a powerful operational environment to collect and process the source Web data. The result is an environment that must simultaneously meet both operational and analytical needs. Furthermore, this is an emerging area—the analytical model that Traackr uses to score, measure and predict is undergoing continuous research and development. In this sense, Traackr also uses the data it collects in it own BI process to investigate how best to measure influence and thus serve its customers.

The data retrieved from Web is multi-structured and largely textual, varying widely in content across different sources. Consider, for example, the difference between a blog post and a tweet. A tweet is a very short message with limited tagging from a single sender, goes out to a known audience of followers and may be retweeted by some of those followers with minimal change. A blog post has a much larger message which may be part of a multi-author blog, supplemented by multiple tags or keywords that is broadcast and read by an unbounded set of readers, some of whom may comment on it, repost it, tweet about it and more. It is clear that these two entities have very different attributes, related in different ways to the underlying interest of measuring the influence of the writer.

Devising a relational database that efficiently stored and handled this variety of models and variable text structures would be a daunting task. A single table containing all types of content would require multiple variable-length text fields, many of which might be null in different rows depending on the content source. Normalizing the table into multiple tables based on content type would lead to a complex model with multiple joins. By Web standards, Traackr today stores and processes a relatively limited amount of data—a couple of terabytes or approx. 200 million posts which represents about 6 month worth of content for influencers in the database and typically add up to a million posts a day. However, it does require a distributed processing solution, which, especially with a multi-table model, further diminishes the attractiveness of a relational approach. Furthermore, as an emerging area, the data stored and its relationships is undergoing continuous change, which would result in regular reloads of the database.

On the other hand, the data and processing characteristics just described are well suited to the characteristics of a JSON-based document-oriented database. Traackr has chosen MongoDB as the solution to its operational storage and processing needs[6]. In addition, as a BI environment, the data must be searched in multiple ways to satisfy differing needs. The ability of the document-oriented DBMS to support multiple indexes is an advantage here. The comprehensive query language (although not SQL), along with MongoDB's aggregation framework provide a strong base for analytical work. At the present stage of market development, there is a dearth of BI tools that operate against JSON stores. Much of this work is currently done through Java programming or through extraction of the data to a spreadsheet for analysis—Traackr uses both approaches. More generic BI tools are beginning to emerge such as Pentaho's NoSQL solution[7], Nucleon Software's BI Studio[8] and more.

Traackr's use of MongoDB illustrates that a document-oriented database can support the combined operational and informational needs of a business based on the new Internet model with growing volumes of multi-structured and externally-sourced data.

O_2 UK—recreating an e-commerce platform

NoSQL databases are not only for totally Web-centric businesses (although, of course, few businesses today can survive without the Web). More traditional businesses such as retail, telecommunications, financial services and more can also benefit. The determining factors are the variability and multi-structured nature of the data involved and the flexibility of access or processing required. Such characteristics are often associated with data sourced from the Web, but other types of data and its use can also exhibit these qualities. For O_2, the data that drove their initial investigation into NoSQL databases as a platform for development was their product catalogue and its use across multiple channels in sales and marketing. In an IT environment traditionally more comfortable with large, proprietary relational databases, this was an innovative step…

With over 20 million customers, O_2 is the largest consumer brand and the second largest mobile (cell) phone operator in the UK and a subsidiary of Telefónica S.A. However, in a market where pretty much everybody who wants a mobile phone already has one, O_2 is looking for related business areas into which they can expand. These areas include home phone and broadband, finance and insurance, entertainment and more.

As anyone who has ever tried to research and choose a new mobile phone or change supplier can testify, the range of product choices and combinations available, from devices to tariffs to offers, some available to new customers and other only to existing contract holders can be exciting or overwhelming—depending on your point of view. All these potential offers must be represented in the product catalogue, a rather complex task given the variety of different types of data required for different devices, tariffs and offers. The challenge is exacerbated by the high pace of change in this market—new devices with diverse characteristics are released regularly but, more importantly, marketing require the ability to offer new tariff plans at a moment's notice. Add to this the additional business areas mentioned above and the products, tariffs and offers associated with these and we arrive at a highly complex product model requiring extensive agility. Furthermore, these products can be offered through a growing number of channels—the Web store, smart phones, physical stores, self-service kiosks and more—and from a customer perspective, any research or ordering done on one channel should be immediately available on any other.

It was against this background that O_2 decided in 2010 to upgrade their e-commerce platform, which—like the majority of such systems—had been designed as much as a decade earlier on a traditional Java / relational database platform. Their choice of MongoDB as the database component of the new platform was driven by factors mentioned earlier—its ability to efficiently handle highly variable data structures, support for multiple indexes and agility in changing the data model—in support of O_2's product catalogue and multi-channel sales. This program led to two independent commercial initiatives: Priority Moments, a loyalty program launched in 2011 and a revamped Website, which debuted this year. While Priority Moments is an innovative loyalty approach, allowing mobile phone customers to collect and redeem location-sensitive coupons for discounts at a wide variety of retailers, the important point to note here is that both Priority Moments and the e-commerce platform address classic retailing needs and require a complete set of both operational and informational function for optimal performance. Our interest here is in how business intelligence is delivered from the MongoDB platform.

Although it's still early days, especially in the case of the e-commerce function, O_2 is already extracting substantial insight from their implementation. This is as should be expected; both loyalty and e-commerce demand good BI to provide the expected business benefit. We'll focus here on the Priority Moments application, as the new Website has just gone into production. As a real-time, location-aware loyalty program, it is vital that both O_2 and the associated retailers have access to ongoing information on uptake of coupons, especially on selected high-value offers, of which there are limited numbers. Such real-time BI is obtained directly from the operational database, with queries and reports developed in Java by the development team. As we saw previously, using bespoke development rather than BI tools is a reflection of the lack of maturity of the application and the platform, but is seen as relatively simple in MongoDB's query language. However, the tools are becoming available and the development team focus is moving towards them for future BI.

Although millions of customers have signed up for Priority Moments and there are thousands of offers active on the system at any time, performance of these queries has been good and the impact on operational work minimal. Of particular interest is O_2's use—so far, as an independent research project—of the "tailable cursor" feature in MongoDB. This allows a query to be left open and as records arrive in the database, they automatically appear in the result, which can be routed to a live HTML5 graph[9] giving real-time feedback to monitor program activity.

The above work can be positioned as real-time BI aimed solely at the functional scope of the Priority Moments program. In the context of Figure 2, this is an example of directly accessing the operational environment for up to the minute data. However, O_2 is a large, complex organization that needs to track and integrate information for management across multiple functions. It thus has, as might be expected, a large traditional data warehouse environment, based in this case on the Teradata platform. Data is extracted from the MongoDB platform and loaded into the enterprise data warehouse for broader analysis. At this stage, data from MongoDB is normalized to the enterprise model of the warehouse, sacrificing the variability and flexibility of the NoSQL environment in favor of the consistency and integration required in an enterprise data warehouse. This is a valid trade-off, but one that must be chosen with increasing care as big data becomes more prevalent.

The O_2 implementation of MongoDB described shows the applicability of the database to a more traditional business problem in a larger, more conventional environment. The ability of the product to support multiple, varying data structures with good performance and agile implementation was the key driver in O_2's choice of tool. Its ability to support real-time BI and feed into the enterprise warehouse was an added bonus that is continuing to develop.

Conclusions

The NoSQL movement is rather unfortunately named. Its positioning as a technological opposite to standard relational databases may have echoed a revolutionary vision a few years ago; but, like all technical innovation, the value comes solely from how it can benefit business. In hindsight, a name reflecting more the advantages offered, such as *multi-structured databases* for example, might have allowed earlier, more widespread acceptance in more traditional business environments where the function they offer can also play a valuable role.

The above comment applies to all types of NoSQL database, but particularly to document-oriented databases, of which MongoDB is our focus in this paper. As probably the most sophisticated and flexible model of all the NoSQL varieties, document-oriented databases offer virtually unlimited flexibility in record format and agility in model extension as application needs change. The cost of this flexibility and agility is that the applications using the data must understand its structure and content to a much greater extent than is the case for traditional relational databases. In addition, while these characteristics encourage agile development, one must take care that data quality and consistency is not compromised. Together, these considerations suggest that NoSQL in general and documented-oriented databases in particular are best applied within well-bounded applications rather than enterprise-wide integration scenarios.

Another unfortunate aspect of the NoSQL moniker is that business intelligence experts have tended to ignore these tools, simply because BI is so closely associated with the relational database / SQL paradigm. The truth is, however, that the flexibility and agility offered by document-oriented databases can be of great benefit to BI. The same *caveats* apply: use it for well-bounded BI applications rather than enterprise data warehouses. A particular strength of document-oriented databases is in situations requiring operational BI or operational analytics, where direct access to the operational data for BI purposes can be offered with less fear of impacting the operational system itself. This is a result of the highly denormalized structure of document-oriented databases.

MongoDB demonstrates that document-oriented databases are coming of age and entering mainstream thinking for appropriate applications. The focus on production features such as performance, advanced indexing and batch processing, support for multiple programming languages as well as a query language have enabled a wide variety of companies in different industries and of varying sizes and maturity levels to implement applications on it. Furthermore, although open source, MongoDB benefits from a dedicated development and support company in 10gen since early 2010.

NoSQL—in its more recent "Not only SQL"—identity has accepted that relational databases will long continue to have a central role in IT in areas where data consistency and integrity are key concerns. It is now up to the more traditional database community to wake up to the potential benefits and uses of NoSQL technology and document-oriented stores in particular. When it comes to real-time combined operational / informational application needs using large volumes of multi-structured, sparse data, document-oriented databases have now reached a level of maturity where they can be and, indeed, are being considered for large-scale, mission-critical business needs and their associated BI.

For such classes of application, NoSQL is indeed no problem.

Dr. Barry Devlin is among the foremost authorities on business insight and one of the founders of data warehousing, having published the first architectural paper on the topic in 1988. With over 30 years of IT experience, including 20 years with IBM as a Distinguished Engineer, he is a widely respected analyst, consultant, lecturer and author of the seminal book, "Data Warehouse—from Architecture to Implementation" and numerous White Papers.

Barry is founder and principal of 9sight Consulting. He specializes in the human, organizational and IT implications of deep business insight solutions that combine operational, informational and collaborative environments. A regular contributor to BeyeNETWORK, Focus, SmartDataCollective and TDWI, Barry is based in Cape Town, South Africa and operates worldwide.

About 10gen Inc.

10gen is the initiator, contributor and continual sponsor of MongoDB, offering production support, training, consulting and online management services for the database. Thousands of leading Web 2.0 companies as well as Fortune 500 enterprises have adopted MongoDB. 10gen is funded by Flybridge Capital Partners, Sequoia Capital and Union Square Ventures.

10gen, Inc.
555 University Avenue
Palo Alto, CA 94301 www.10gen.com / www.mongodb.org

[1] Devlin, B. A. and Murphy, P. T., "*An architecture for a business and information system*", IBM Systems Journal, Vol 27, No 1, (1988) http://bit.ly/EBIS1988

[2] From Devlin, B., "*Data warehouse—From Architecture to Implementation*", Addison-Wesley, (1997)

[3] http://en.wikipedia.org/wiki/Big_data , accessed 13 March 2012

[4] Codd, E. F., "*A Relational Model of Data for Large Shared Data Banks*", Communications of the ACM, Vol 13, No 6, (1970)

[5] For examples, see: http://www.mongodb.org/display/DOCS/SQL+to+Mongo+Mapping+Chart

[6] The rationale for Traackr's choice of MongoDB: http://traackr.com/blog/2012/02/traackrs-migration-from-hbase-to-mongodb/

[7] See http://www.pentaho.com/big-data/nosql/

[8] See http://www.nucleonsoftware.com/BIStudio.aspx

[9] For background on this approach, see: http://hummingbirdstats.com/ and http://mbostock.github.com/d3/