Relational is the New Black—Uniting Data and Context

OCTOBER 2019

THOUGHTPOINT 2 OF A 5-PART SERIES BY
DR. BARRY DEVLIN, 9SIGHT CONSULTING
BARRY@9SIGHT.COM

Hadoop's early premise that big data stores should be largely schemaless and interpreted only at time of use is faulty and must be revisited to ensure valid and appropriate use of the entire information assets of a digital business.

Unicorns. And unstructured data. What do they have in common? Since the beginning of the new millennium, the IT industry has been enthralled by both, yet neither exists. Let's (reluctantly) forget unicorns and instead show that *unstructured data* is also a mythical beast. Check out this data fragment: {06Harrym601gngr35m119052018}. It may seem unstructured, but readers can likely see a name and maybe a date in there; it's clearly structured. Mix up the letters and numbers, but the result is still not fully unstructured data because each character has a detectable binary structure. Data must have structure; otherwise it would just be noise. *Unstructured data* is an oxymoron.

Such pattern recognition in data shows that structure—or ***schema***—is the basis of meaning. Here's a second fragment: {99, Meghan, f, 507, dkbr, 38, m2, 19052018}, with added structure via CSV formatting. Royalists can now identify the context and maybe guess more fields[1]. Context is key to meaning. As seen in my book, *Business unIntelligence*[2], context is the difference between data that computers crunch and information that humans use. Without a viable data schema, finding context and meaning in data is well-nigh impossible, putting insight discovery, decision making, and action taking in digital business at high risk or error and ultimate failure.

A data schema is more than a structure; it is the key to understanding the meaning of its content, especially when used in a digital business.

Schema-on-read—what's that all about?

In the first decade of this century, the processing challenges of volume, velocity and variety of externally sourced big data drove a new, structure-lite view of data storage and management, seen in both the Hadoop and NoSQL approaches. By 2010, a new breed of data professionals declared this an exciting, novel concept: ***schema-on-read***. It proposed that big data should, in preference, be stored in whatever format in which it arrived and that all definition and interpretation of its structure, context and meaning should be postponed until someone needed to use it for some business purpose.

The rationale was that upfront structuring of incoming data was too onerous or in some cases even impossible because:

SPONSORED BY
teradata.
www.teradata.com

1. The data volumes were too large and its arrival velocity too fast to allow the structuring and processing required to load it into traditional, relational databases

2. Data structures were variable and rapidly changeable over time, making relational databases largely incompatible with such data and, more important, that it was extremely costly to adjust fixed table schemas to changes in incoming data structure

3. Maintaining the data in its raw form and original silos would ensure nothing of interest was lost, and allow the maximum flexibility in analytics and other business uses

Like the proverbial road to hell, this one is also paved with good intentions. There is some truth—to varying degrees—in each of the above arguments. Technological evolution has weakened some of the original rationale. Since the schema-on-read approach was formulated, many organizations have built data lakes designed to avoid the problems and benefit from the opportunities listed above. Many have fallen foul of the hidden traps that occur when meaning, context and structure are overlooked as the following story demonstrates.

Schema-on-read, seen in Hadoop data lakes, was conceived to ease the challenges of the three Vs of big data.

The parable of the truckloads of data

Trucoeur is an imaginary French truck manufacturer that sells vehicles across the EU. A key competitive goal is to reduce operating costs for its customers. Unplanned downtime and maintenance are expensive; having a truck off the road can cost more than €1,000 per day, excluding parts and labor. When big data emerged in the early 2010s, Trucoeur saw an opportunity to move from scheduled to preventative maintenance by tracking and analyzing dozens of data points in near real-time from onboard mechanical sensors, historical warranty, and parts inventory information, as well as third party data sources—such as weather, geolocation, vehicle usage, and traffic patterns. Predicting high-risk part failures would allow maintenance to be planned around truck schedules, locations, parts availability and more. Savings of more than 25% were anticipated.

The plan was to gather the necessary data in its raw form from over two dozen different feeds into a Hadoop data lake and allow data scientists to access and analyze it to create models of failure modes and predicted timing based on sensor and other externally sourced data. The results would be merged with internal warranty and inventory data. Maintenance plans—what to repair or replace where and when to minimize truck, and even driver, downtime—would then be sent to fleet operators.

The business goals were excellent. The technology budget—based on commodity hardware and open source software—looked very affordable. The project team was staffed and appropriately skilled with Hadoop programmers, experienced Unix systems administrators, and a mix of experienced and newly minted data scientists who knew R and understood and could model truck maintenance.

With hardware and software installed, the data center began to hum quietly as the first data was easily ingested to a schema-on-read model. What could possibly go wrong?

Schema-on-read offers welcome agility in the building and initial loading of a data lake.

In short, data could—and did—go wrong. Very wrong. As the fourth and fifth feeds were connected, alarm bells began to ring, albeit quietly, but ring, nonetheless. The first traffic data from the UK was subtly different from the mainland data already loaded and began to throw the models off track. Of course, the miles vs. kilometers difference was known

and accounted for, but traffic-based predictions became unreliable. The problem was traced to another factor that was well-known but not included in the earlier data: the UK drives on the opposite side of the road and UK trucks have the steering wheel on the right, variables that had to be retrofitted to all other data and models.

"No problem," said the data scientists. "Adding new fields is easy in schema-on-read." So, they did. And when the next data problem arose, they added yet more fields. Sometimes they had to exclude or reinterpret existing fields in specific cases. Soon, data in different files and stores was becoming incompatible in subtle but challenging ways.

Then the new variant of the PQ-Plus truck was released. The engineers had added some newly requested sensor data. No problem with schema-on-read. What turned out to be more of a problem was that the engineers had also subtly redesigned some of the existing sensor data outputs for speed and efficiency. That took some time and luck to discover when the number of unpredicted truck breakdowns began to creep up again.

As additional sources are ingested, schema-on-read may lead to the degradation of a data lake into a data swamp.

While schema-on-read is good at addressing the three problems and opportunities listed at the top of page 2, it also brings its own set of challenges, potentially turning a data lake into a data swamp. Let's take a look at the opposite of schema-on-read.

Schema-on-write—what's right with this?

Proponents of schema-on-read contrast it to the traditional "schema-on-write" approach. This latter term was seldom if ever seen prior to the emergence of schema-on-read, because it was almost universally accepted that data should be well-structured by design. The weight of expert opinion was strongly in favor of designing data storage according to the relational model introduced by Dr. E.F. Codd in 1970[3] and Dr. Peter Chen's 1976 seminal paper on entity-relationship modelling[4].

Schema-on-write demands that you model and structure your data and storage before gathering data. Data modelling is, in simplistic terms, the process of refining the rather messy reality of real-world information into something that is suitable for the neat and tidy—and definitely naïve—mindset of a digital computer. Modelling is only a process of rationalization and documentation. To be useful in a computer, it must lead to a schema for the data that implements the model and instantiates the metadata—or, as I prefer to call it, *context-setting information*—that defines its meaning.

This leads us right back to Harry and Megan and the question of how best to build data structures that incorporate context and meaning, preferably in a form that is easily understood by people and performs well for reading, writing, and computation by computers. We already have such an approach: the relational model as instantiated in relational database management systems (RDBMSs).

Schema-on-write—just classical modelling and database design—has long been central to data management in setting context and defining meaning for business.

The RDBMS is a tried and tested technology with forty years of experience embedded. It was true that it did not handle the volumes, velocity and variety of big data well when schema-on-read was gestating. However, relational technology has improved and expanded in scope since then in modern RDBMS environments, such as Teradata Vantage. Furthermore, the challenges and opportunities of big data have also evolved in the interim. Relational is the new black—not just fashionable, but stylish, hardwearing and suitable for all weathers.

Extended relational solves modern big data challenges

When schema-on-read was devised as a solution to the three Vs of big data almost a decade ago, for most businesses, big data was a largely separate and distinct area of data processing, isolated from the traditional day-to-day computing that ran their operations and decision-making support activities. In today's digital business, the distinction has completely disappeared. Big data and traditional data—both externally and internally sourced—are intermixed and used together in multiple business processes. It is no longer realistic to treat them as independent processing environments.

A combined strategy and convergent architecture are required. This is not to say that a single technology base can answer all requirements. Rather, one technology must be chosen as the core—the *primum inter pares*—of the diverse set of technologies required to support all modern information and data management needs. The mandatory and obvious business requirement for pervasive context and omnipresent meaning points clearly toward schema-on-write, instantiated in relational database technology, as the only viable approach to storing and managing the core information of the business. I describe this strategy in detail in the IDEAL (conceptual) and REAL (logical) architectures of *Business unIntelligence*[2].

An extended relational environment, such as Teradata Vantage, offers the core data storage and connectivity to meet the complex data needs of digital business today.

An extended relational environment, such as Teradata Vantage, supports this strategic approach by providing:

- Full support in the relational model for a significant range of volumes and velocities of data ingestion and storage
- The ability to easily change existing database schemas to support data variety and later changes to defined schemas
- Ingestion, storage and management of data in non-relational formats, such as CSV, JSON, XML and more within the RDBMS
- Direct access via SQL, R, Python and a wide array of analytics functions to all data stored in the RDBMS and to remote, distributed data stores, including Hadoop and object stores, such as Amazon S3 and Azure Blob
- Separation of compute and storage, and implementation of both independently on premises and/or in the cloud

Taken together, these features favor a schema-on-write approach to data management, while not precluding the use of schema-on-read where needed and appropriate.

Integrating data and context—done or redone

Digital business is a "big data" world where an enormous percentage of data comes into the enterprise from external—and often poorly constructed and managed—sources. It is vital that data scientists and businesspeople can use it correctly and validly in decision making and action taking. To do so, the structure, context and meaning of this data must be made and kept fully clear from its earliest arrival in the enterprise until the last moment it is used in the digital business value chain. Schema-on-write based on the relational model and exemplified by Teradata Vantage is the most appropriate approach to achieving this goal.

Teradata Vantage combines all required function to give business the ability to use data correctly and validly in decision making and action taking.

While some long-standing data management professionals may see this as no more than a return to old wisdom, the reality is much more. The extended relational approach differs from traditional data warehousing by allowing data to reside outside the RDBMS, while—in contrast to data lakes—mandating that such diverse data is governed according to best data management principles from the relational environment.

The extended relational approach differs from data warehousing by allowing data to reside outside the RDBMS but governed to best data management principles.

This is the second article in a series of five ThoughtPoints on "Rethinking Hadoop for Modern Analytics." The complete series of articles is:

1. *Hadoop—Spreadsheets on Steroids http://bit.ly/2N59ZCO*
2. *Relational is the New Black—Uniting Data and Context http://bit.ly/2CSpV6t*
3. *AI and Analytics—All Gold Taps but No Plumbing http://bit.ly/2DCKXqe*
4. *The Joy of ASAP—Analytics by a Single Access Point http://bit.ly/2S2vjga*
5. *The Right Vantage Point Offers Advanced SQL Views http://bit.ly/2TZ1Epr*

An omnibus edition of all five articles is also available at http://bit.ly/36IWy95

*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. His book, **"Business unIntelligence—Insight and Innovation Beyond Analytics and Big Data"** was published in October 2013.*

Barry is founder and principal of 9sight Consulting. He specializes in the human, organizational and technological implications of deep business insight solutions combining all aspects of internally and externally sourced information, analytics, and artificial intelligence. A regular contributor to Twitter (@BarryDevlin), TDWI Upside, and more, Barry is based in Bristol, UK, and operates worldwide.

[1] The fields I concocted in the data fragment are: Royal succession, Name, Sex, Height (feet and inches), Hair color, Age, Marriage number, Date of last marriage

[2] Barry Devlin, "Business unIntelligence", 2013, Technics Publications, New Jersey, http://bit.ly/BunI-TP2

[3] Edgar F. Codd, A Relational Model of Data for Large Shared Data Banks. 1970, *Communications of the ACM*, 13(6), pp. 377-387

[4] Peter P. Chen, The Entity-Relationship Model—Toward a Unified View of Data. March1976, *ACM Transactions on Database Systems*, 1(1), pp. 9-36, http://bit.ly/35NrQGH

SPONSORED BY

www.teradata.com

teradata.