Unlocking Machine-Generated Data

Bridging the Structure Chasm between Hadoop and Relational

September 2013

A White Paper by

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

Machine-generated data is the coming wave of big data. It will dwarf the current experience in both volume and velocity. Furthermore, in terms of content and structure, as well as its business use, it is very different to the social media big data most prevalent today. This paper describes how to unlock the value of machine-generated data, inspired by the approach taken by one of its bigger users, eBay.

The two defining characteristics of machine-generated data are its variability during both definition and production, and its semi-structured nature. These characteristics lead directly to name-value pairs (NVPs) as the most appropriate and useful format for such data. The first half of this paper explores the characteristics of name-value pair data by way of examples from three industries and positions it within the broader scope of all information—including big data—used by business today.

We then explore the approach taken by eBay to storing and processing name-value pair data. This shows the justification for choosing relational database technology as the foundation and includes sample SQL snippets that demonstrate how sensor data is easily transformed into analytics-friendly relational tables using Teradata v14.0 functionality.

Finally, we position machine-generated and other data types within Teradata's Unified Data Architecture as an overarching vision of how information and data will be increasingly situated in the future.

Contents

Sponsored by:
Teradata Corporation
www.teradata.com

We seldom think about why we store data the way we do, why it's structured the way it is and how it got to be that way. The simple answer is because computers can easily store and process data of a known structure. Named fields defined as Integer, Floating Point or Double Precision tell the CPU exactly how to handle the data. A highly structured database groups like items together for easier and faster manipulation. We can assign variables and write programs—`sell_price = cost_price + margin_uplift`—to describe what the computer should do for us. This, of course, is Computer Science 101.

But there is another answer to those questions seldom considered by traditional developers: *because we know in advance the fundamental "shape" of the data*. Determining that shape and the best structure was the goal of data modeling. Although such definition was once universally possible for business computing, it is no longer always the case that we know the data "shape" in advance. Sometimes, today, the data isn't ours. Often, it originates from machines. Sometimes the business doesn't know how they want to use it. In such cases, the data "shape" is indeterminate and we may store it as-is until we need to use it, an approach often referred to as *late binding*.

Up until the late 1990s, almost all the data used in business originated internally from operational applications built by IT according to *predefined* business requirements. The business declared that it wanted to automate order entry, for example. The users thought carefully about the information they needed from the customers. IT and business sat together and (sometimes) harmoniously modeled the data needs. Modeling defines which data logically belongs together, which data can be used to identify unique items, and which fields must be physically stored together to ensure that valid meaning is captured, performance assured, errors minimized, and so on. None of this is new, of course. But it's often missed that modeling and designing operational applications implies that the business knows its requirements beforehand and that IT has control of the shape and content of the data. Operational applications can be built only if the data to be recorded has well-known and well-defined characteristics. A traditional data warehouse can be populated only from data with clear meanings and relationships. Such data is the family jewels of the business, defining the truth of the state of the enterprise. Its creation is a task of extracting the meaning from reality and assigning it to the safety of the records—key and fields—of a (usually) relational database.

Machine-generated data is among the most important and fastest growth areas in big data.

Today, much of the information used by business no longer originates internally. Some of it comes from social media sites, e-mail, or voice transcriptions. Of primary interest in this paper is ***machine-generated data*** that originates from devices of varying levels of complexity, from simple sensors, through on-board controllers to computer servers. I will introduce a classification of information/data types later, but for now, note that the above sources have one thing in common: the business receiving them controls neither their structure nor content. Furthermore, as we shall see in the following examples, the structure of such information can vary over time in ways that are either unpredictable or unplanned when the application is first conceived. The analysis and use of such semi- or multi-structured data is tricky. As we shall see, an important class of such data can be handled—in many cases, most efficiently—in a relational database. This data, consisting of ***name-value pairs***[1] **(NVP)**, originates only from machines and is rapidly growing in importance as the Internet of Things becomes fully pervasive. It comes in a couple of forms, but as the term implies, each data item carries its name with it, from the moment it's created by some machine, on whatever its journey until its value—both physical and financial—is extracted by the business. A name-value pair thus carries an "identity card" with it everywhere it goes, so that anybody who needs to can find out what it truly is—without having to resort to some master catalog.

Name-value pairs—a history by example

Server web logs and their analysis

Since the creation of the World Wide Web in the early 1990s, web servers have stored "weblogs" of the pages visited by users. These logs contain information including client IP address, request date/time, resource requested, HTTP return code, etc. Our interest lies in the resource requested, which takes the form of a URL (uniform resource locator). In simpler days or cleaner implementations, this is a page address, such as `http://www.9sight.com/resources.htm`. However, the URL specification also allows a query string, which is where things get interesting. A query string typically contains text that a visitor entered into a search field and/or contextual information from the web page. A query string looks like `?field1=valueA&field2=valueB &field3=valueC` or some more complex variant. It consists of a simple text string of arbitrary "name=value" pairs defined according to the needs of the web page designer. Thus, we might see `?country=us&lang=english`, which specifies the attributes of the browser the client uses. Thus we see name value pairs: country is a name with "us" as the value paired to it; lang is the name with "english" as the value.

Web developers use this approach extensively, but the structure presents a challenge for business intelligence (BI) and any in-depth web analytics. The incoming data contains well-defined fields such as request date/time, requested resource, as we've seen, that will be perfectly at home in any BI environment. But it also contains the query string, the structure and meaning of which are arbitrary and not easily supported in a relational database. Furthermore, there can be an arbitrary number of name-value pairs, in any order, and they can change frequently on a web page. Such structures were rare before the advent of the Web; applications were designed with foreknowledge of the fields required and their interrelationships. This is not to imply that this new structure is unreasonable. The name-value pair construct offers a degree of design flexibility that is highly useful and perhaps mandatory in rapidly changing or unpredictable environments. The designer of the front-end application or web page has the freedom to quickly make any and all changes demanded by the market or the business, irrespective of downstream data modeling needs. There is minimal governance to observe beyond ensuring that the name space is managed to ensure uniqueness or define intended usage. Only basic metadata rules such as documenting the meaning of the name and allowed values need be enforced. The query string itself shrinks, expands and morphs as required. Great for customer-facing innovation, but pity those tasked to deliver reporting and analysis. Retail, banking, eCommerce, and social websites use it extensively, and they definitely use analytics too. How do they do it? We will return the solution detail later, but first, it is instructive to look at some other cases, showing how this approach is applicable across different industries.

Name-value pair data is a highly flexible, semi-structured data format used extensively by web sites to record user actions and site responses.

Sensors in your automobile… and everywhere else

Automobile manufacturers have added increasing numbers of sensors to engines, gearboxes and more over the past decade. Tachometers, temperature sensors and fuel gauges have been joined by accelerometers, pressure gauges, vibration and stress sensors, to name but a few. On-board diagnostics (OBD) is the term used for a vehicle's self-diagnostic and reporting capabilities, which give the vehicle owner or repair technician access to the status of these sensors. The first systems were introduced in the early 1980s and would simply illuminate an indicator light if a problem was detected. More recent OBD implementations communicate via standardized digital codes to provide both historical and real-time data to allow rapid identification and repair of vehicle malfunctions. And now it's not just the mechanical aspects. GPS allows location and speed to be recorded. Weather and road

conditions can also be logged. Within the vehicle, all this data is assembled together by one or more dedicated computers for use in engine tuning and maintenance. Connect the onboard computer to a central location via cell phone or wireless Internet and a host of uses emerge.

Automobile telematics is rapidly evolving with new sensors being added as costs drop and new applications emerge. As a result, the data transmitted by the onboard computers changes often as new autos are introduced or onboard computers are upgraded. However, existing data sets from previous automobile models must continue being usable side-by-side with the new NVPs. The messages used by OBD are name-value pairs, transmitted in this case in binary format. A current list of many of the standard names and values in OBD II can be found on Wikipedia[2]. As in the case of the web log query string, we can see the flexibility to change that the name-value pair format offers. New measurements and even types of measurement can be added as sensors become available, obsolete names simply stop occurring in the messages and any required changes—for example, in unit of measure—can be simply accommodated by adding new codes or other approaches. The messages are compact, carrying only the data needed. So-called "missing values"—data that hasn't been collected because a sensor failed, for example—do not require any form of "value=null" indication. The format is thus ideal for transmission to dealer warranty and service departments via SMS/Txt where costs are related to volumes of data transferred.

When such sensor data is readily available and its content easily expanded and simply described, new business applications become possible, from fleet management to pay-as-you-drive insurance. Telematics is already a big source of data in some industries. Airplane and jet engine manufacturers have been collecting such information for years for use in maintenance programs. Trucking and other transportation companies are using it increasingly for fleet management, first at a simple technical level and then for logistics planning and tracking. A number of insurance companies have used telematics to pioneer pay-as-you-drive automobile insurance. In such an application, sensor data about speed, sudden accelerations and decelerations, as well as location, create a profile of driver behavior that is used to set premium payments and, even, decide to offer or decline cover. The approach is also seen as a way to "nudge[3]" driver behavior. And, as manufactured goods from fridges to television set-top boxes, from water meters to security systems are instrumented in the expanding Internet of Things, the possibility of ever more powerful applications grows. Hospitals are also collecting sensor data from patients to improve treatments and reduce costs. The growth in numbers of such messages is expected to be enormous, with data volumes well exceeding those experienced in social media applications today. In addition, a very rapid rate of change can also be anticipated as new uses emerge and existing application change. This implies that the messages to be transmitted will be huge in number and in a constant state of flux over the coming years.

Name-value pair data is very appropriate for transmitting and recording data from sensors and other machine sources.

Online gaming is not only for fun

As a final example, another area with large volumes of relatively small but highly variable messages is online gaming, from single-player console or shooter games all the way to massively multiplayer online role-playing games (MMORPGs). The size of this market is often underestimated by those not involved in the area. *World of Warcraft*, a popular MMORPG, for example, has more than 8 million subscribers as of March 2013. The MMORPG US market in 2012 is roughly $8 billion.

In this environment, games applications on consoles, browsers, mobile devices or PCs all send and receive messages concurrently with game play. Such messages may provision new weapons or rewards, offer players opportunities to buy merchandise, and so on. With the ongoing evolution of these games, the ability to handle additional and changed messages in a name-value format is of particular interest.

While the real time nature of online games is fairly intense, it's also true that there is both a selling opportunity as well as a game experience improvement opportunity. If game players use a specific magic spell in a Middle Earth cave and then die 85% of the time, it's a good time to notify the game developers to change it or risk losing customers. Or it's a good time to remind the players to buy the "new improved magic spells" kit for $1.99. Half a million buyers times $2 is worth some R&D. Either way, analyzing name-value pairs from games seems mandatory.

Name-value pair data as a component of big data

The topics covered in the previous section are often rolled into discussions of "big data". Unfortunately, so is social media—from Tweets to YouTube videos—and large files of transaction and event data that have been processed for years in large businesses. The problem is that the big data topic has become highly confused, mostly by marketing and press hype. To alleviate the confusion and to focus in on the category of name-value pair data, we first step back from technology issues and see how all the information and processes used by a business interrelate. This leads to a new vision of the information landscape, with three distinct, deeply interrelated domains:

1. **Human-sourced information**[*]: All information ultimately originates from people, an artifact of the human mind. This information is the highly subjective record of human experiences, previously recorded in books and works of art, and later in photographs, audio and video. Human-sourced information is now almost entirely digitized and electronically stored everywhere from tweets to movies. Loosely structured and often ungoverned, this information may not reliably represent for the business what has happened in the real world. Structuring and standardization (e.g. modeling, validation in operational systems and cleansing as data moves to BI) allows the business to convert human-sourced information to more reliable process-mediated data.

2. **Process-mediated data**: Business processes are at the heart of running and managing every business. These processes record and monitor business events of interest, such as registering a customer, manufacturing a product, taking an order, etc. The process-mediated data thus collected is highly structured and includes transactions, reference tables and relationships, as well as the metadata that sets its context. Process-mediated data has long been the vast majority of what IT managed and processed, in both operational and BI systems. Its highly structured and regulated form is well suited to promoting information management and data quality, as well as for storage and manipulation in relational database systems.

3. **Machine-generated data**: All the name-value pair data in the previous examples falls into this category. Its source is the sensors, machines and computers used to measure and record the events and situations in the physical world. From simple sensor records to complex web logs, machine-generated data is well-structured and usually considered to be highly reliable, provided that the occurrence of faulty sensors and missing data is accounted for. As sensors proliferate and eCommerce becomes pervasive, driving ever larger data volumes, machine-generated data is becoming an increasingly important component of the information stored and processed by many businesses. Its well-structured nature is amenable to computer processing. It is sometimes claimed that its size and speed is beyond the abilities of traditional RDBMS, mandating NoSQL data stores. However, high-performance relational databases are regularly used for such data.

[*] Here, "data" is well-structured and/or modeled; "information" is more loosely structured and human-centric.

Figure 1 shows the relationship between these three domains and the functionality that surrounds and transforms them. Human-sourced information and machine-generated data are the original sources of all process-mediated data, which has long been the focus of IT efforts. These sources are more flexible and precede traditional process-mediated data. In most cases, only a small, well-defined subset moves through the traditional business process layer that creates process-mediated data. The goal is to ensure the quality and consistency of the resulting data, but the side effects are often reduced flexibility and delayed availability of data.

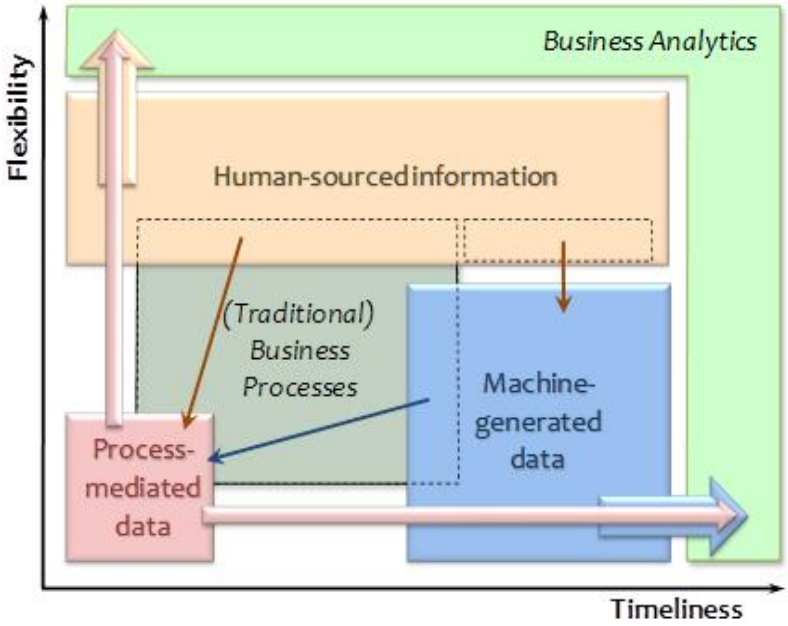


Figure 1: The tri-domain information model

The relative sizes and perceived importance of these three domains has shifted over the past decade and will shift further. Process-mediated data was the dominant, and almost exclusive, domain since the introduction of business computing in the 1960s. Digitized human-sourced information and machine-generated data were small in volume and value in comparison to the well-managed data in operational and informational systems. The big data explosion of the last decade consists mostly of human-sourced social media information. In 2012, responses to the EMA / 9sight big data survey[4] showed that human-sourced information accounts for nearly half of the sources of big data, with process-mediated data still outstripping the machine-generated variety by a small margin. In the coming years, rapid growth of the Internet of Things will likely promote machine-generated data to the highest levels of volume and importance.

Why is name-value pair data special?

As we've seen, all name-value pair data is machine-generated. However, not all machine-generated data exists in this form. Call detail records (CDRs) generated by telephone exchange servers are certainly machine-generated, but not stored or transmitted as name-value pairs; their record structure is laid down in advance and rarely changed under well-defined conditions. In the case of vehicle telematics, while we saw name-value pairs used between the on-board computer and the initial collection point, further communication of pre-processed data is likely to use XML-structured messages or files. In other cases, particularly in the case of communications between identical machines, data (without names) may be transferred in binary format. So, what makes name-vale pairs special compared to other types of machine-generated data?

Machine-generated data in its rawest form carries limited meaning or business value. It is just numbers—voltage levels, resistances, etc.—emitted by sensors. Initial processing in a simple computer associated with one or more sensors assigns meanings to these numbers—this voltage indicates a temperature of 15°C, that change in resistance shows a phone call started, this state change from a mouse indicates a user selection. In this way, meaning is added to machine-generated data before it is passed on into the wider world. The resulting message (or file) carries that meaning in one of two ways: positionally (the value for each characteristic always occurs in the same place in the record) or as a name (what it means) and a value (how much or what type). The positional approach is more efficient in cases where all characteristics are defined in advance and are present in most or all messages. The named approach is more flexible to changing characteristics and more efficient in sparse

data. The named approach occurs in two types. In a schema-less approach, names are assigned as required; it is, thus, more flexible to change, but the problem is that names may accumulate multiple meanings or different names are used for the same characteristic. In a schema-based approach (e.g. XML), names are set by agreement, which favors reduced ambivalence over flexibility.

The business value of machine-generated data emerges only when combined with process-mediated data.

Having assigned meaning and received the messages in a business process, we can look at business value. It is only when machine-generated data is joined with or converted into process-mediated data that it receives sufficient context to create substantial value for the business. CDRs, for example, as they arrive in the central processing servers of a Telecom company describe only the physical event of a call—origin, destination, time and duration. This data can be analyzed for network optimization, but only when it is combined with contract information, billing rates and so on can it be used to calculate or analyze the financial value to the business. In order to join or convert machine-generated data to/with process-mediated data, it is vital to know the data fields in the machine-generated data and their meaning with a high degree of certainty and accuracy. In the case of CDRs, this is guaranteed because the exchange servers and central processing are under the control of the same organization and the structure and content of the CDRs are well-defined and largely unchanging. The CDRs can thus be loaded into a relational database in a known and fixed relationship. There is little or no flexibility to change in this relationship, nor is it needed. However, in our earlier examples, the flexibility to handle change is vital. Such change arises from three distinct causes:

1. The business activity is naturally changeable—website marketing activities and online gaming are prime examples of this. Data structures and actions change often, unlike traditional applications.
2. The business model is still in the early stages of evolution or undergoing significant change—consumer automobile telematics is in this phase at present. We may expect the rate of change to decrease over time, at which stage a more fixed structure may offer better performance.
3. Machines recording the data and those receiving it are in different ecosystems and/or the two ecosystems are owned or controlled independently—a situation that is arising more often as the Internet of Things become simultaneously more pervasive and more fragmented.

The level of flexibility to change required in these cases makes name-value pairs vital and a schema-less approach most likely, given its compact nature in comparison to a schema-based, XML model.

Name-value pairs: enabling the Web, empowering the Internet of Things

The semi-structured and highly variable nature of machine-generated data demands flexible and agile ways of designing and using data. The name-value pair data structure meets these needs admirably. Up-front data modeling is minimized or, in some cases, eliminated. Data standardization occurs just prior to use, a data management approach well-suited to data that is externally sourced. This late-binding approach, coupled with the minimalistic structure of name-value pairs, encourages flexibility in the initial design and allows agile change once in production. In contrast to other more formal semi-structured models, such as XML, name-value pairs favor flexibility over enforcing consistency.

The characteristics of name-value pair machine-generated data are a blend of those associated with process-mediated data and human-sourced information. The former has long been implemented most successfully in relational databases; the latter is widely considered to be the forte of Hadoop or NoSQL solutions. The question therefore arises: how do we best store and process name-value pair data when it arrives in our environment? Should we choose relational, Hadoop or NoSQL? And why?

To answer these questions, we turn next to eBay's experience in addressing the needs of analytic users of web log data query strings from its online store.

Using and processing name-value pair data

The key question to ask is: what do we want to achieve in business terms with this name-value pair data? Of course, the detailed answer varies depending on the content of the data and the industry in which you operate. However, in general we can say that name-value pair data most often provides environmental or background information for a recognized business entity such as a customer or transaction. Standalone analysis or use of name-value pairs is therefore of less interest than in the context of core business data. The query string in the web log information from a retail website may contain data about what I searched for, how I got there and what was displayed, but useful business analysis requires that this data be linked through some unique userid to my profile and transaction history. In a pay-as-you-drive application, sensor information in the name-value pair data stream must be associated with a unique car and, more interestingly, with a particular driver. Furthermore, the name-value pair data is highly detailed and production oriented. Its value to the business ranges from pure operations to operational analysis to tactical BI. More generally, incoming machine-generated, name-value pair data is useful to the business only when combined with existing process-mediated data and when integrated in a production analytics environment.

Although Hadoop or NoSQL are often considered first for processing name-value pair data (because it is seen as part of the big data phenomenon) they are not necessarily the best options.

These business considerations allow us to propose a preferred processing environment for such data. There exist three high-level choices: (i) Hadoop, (ii) NoSQL data store and (iii) relational database.

- Hadoop is widely perceived as providing highly flexible, cost-effective platform on commodity servers for parallel processing of all types of big data. This is true within certain bounds. While there has been some moves towards providing a real-time, database platform within it, Hadoop remains a batch-oriented, programmatic environment best suited to exploration and discovery type activities performed by programmers on a one-off basis. Furthermore, core business data will have to be copied into the Hadoop environment to support the join processing required. These characteristics are a poor match for the requirements above.
- NoSQL data stores are valued for their ability to handle big data in loosely defined or highly changeable data schemata in real-time with varying degrees of ACID-compliance. They also operate and scale well on large clusters of commodity hardware. These characteristics are a good match for name-value pair data alone. However, the requirement to join such data with existing core data is difficult to satisfy, would demand both copying that data into the NoSQL environment and restructuring it to the NoSQL model. While less programmer-centric than Hadoop, NoSQL data stores do require some level of technical skill to use them. In many cases, analytic processing can prove challenging. Unless your core data is already in a NoSQL environment, this is a poor match for the business needs.
- At first glance, relational databases seem like a poor match too. They are not renowned for their support of either flexible schemata or text strings. Nonetheless, they are at the heart of all operational and analytic production work today and core business information already exists there, in structures optimized for operational and analytic work. In addition, power business users are often familiar with SQL or other tools based on it, allowing them to use name-value pair data more easily if it could be made available in a relational format. Fortunately, that possibility now exists and using it is the topic of the next section.

Handling name-value pair data in the Teradata database at eBay

By 2008, eBay was facing a challenge posed by the growth rate in web log records. The value of such data was already proven. IT could and did deliver it using an ETL-based approach. But, now, business users wanted to analyze and play with that data themselves in tools they already knew, rather than have IT create batch analytic jobs or reports on their behalf. IT needed a way to make the data available for more people, to scale to petabytes and to improve flexibility to changing data. A number of alternative approaches were investigated, but none offered the overwhelming benefits that could be achieved if the name-value pair data from query strings could be processed effectively in a relational database and offered to end users via standard SQL.

Two typical relational approaches were initially considered. The first was to extract each unique name-value pair into a specific column based on the name in the pair. The resulting table would, however, be extremely sparse and some tens of thousands of columns wide. As new name-value pairs were created, the table and load process would have to be altered on an ongoing basis. The ongoing maintenance and storage requirements led to the rejection of this tactic. The second approach considered was based on two tables, table 1 with the log message key and any fixed, well-defined attributes; and table 2 being a long table with a row for each name-value pair, with one column containing names and another the values, all linked via a foreign key relationship to table 1. This approach was rejected due to ETL workload, query complexity and database performance concerns.

As one the oldest—founded in 1995—and largest marketplaces on the Web, eBay lives or dies by its ability to understand what its buyers are looking for and its success in linking prospective buyers to sellers. eBay continuously and actively improves the appearance and behavior of its website in order to make the most appropriate offers to prospective buyers and to give the best service to advertisers. This process is driven by data collected about search terms entered, result sets shown and eventual purchases. The data thus collected is used by business planners to improve page layouts, product placements, and drive higher sales.

eBay needed to get this extensive and rapidly changing data into the hands of its business analysts in a timely and highly operationally reliable manner, and in a form that they could easily manipulate and query. With large-scale, production data warehousing already on a Teradata platform, these requirements strongly suggested that a relational solution would be the optimal approach—if they could process query strings effectively and enable analysis via standard SQL.

The final solution is much simpler, has better storage and performance characteristics and maintains the flexibility of the name-value pair method. The query string is simply stored as-is in a Varchar(4096) field and specialized SQL functions have been included in the database (generally available in Teradada v14.0) to parse out specific name-value pairs at query time. Thus, for example, applying the SQL function "`Select nvp(query_string,'vo') from logtable;`" to a query string containing "`?h=c8&g=739d69c81&c=1&vo=117933&rnm=0…`" returns the value 117933. In addition to the Varchar string, the table contains standard key and timestamp columns, as well as other fixed fields from the web log. To further improve usability and performance at query time, common name-value pairs that seldom if ever change can be extracted to their own columns at load time and used directly by business users.

Teradata v14.0 provides three functions for processing name-value pair data, based on the pioneering work carried out at eBay:

1. NVP: extract name-value pair (with the possibility to specify delimiters other than & and =)

A more complex form of name-value pair data consists of a delimited list of values for a single name, such as "`?price=243.00,234.00,239.00,254.00`", which might represent four airline ticket prices returned in a search. The following two functions can be applied:

2. STRTOK: extract a single item from a delimited list
3. STRTOK_SPLIT_TO_TABLE: transpose or pivot a delimited list into multiple rows of a column

A code example of handling name-value pair data

The following is a more detailed example of the above functions applied to automotive sensor data.

Define table and insert two sample log data lines:

```
create volatile table car_data (log_ts timestamp(6), vin_nbr bigint,
sensor_name varchar(128), payload varchar(4096) character set unicode)
on commit preserve rows;
Insert into car_data('2012-01-01
14:12:31.000000',123,'Fuel','temperature=85&pressure=3&level=0.8');
Insert into car_data('2012-01-01
14:12:31.000000',789,'Cylinders','cyl_list=2,5,6&error=12345');
```

Extract the temperature and pressure for fuel:

```
select log_ts,vin_nbr,nvp(payload,'temperature') as
Temperature,nvp(payload,'pressure') as Pressure from car_data where
sensor_name = 'Fuel';
```

	log_ts	vin_nbr	Temperature	Pressure
1	2012-01-01 14:12:31.000000	123	85	3

Extract the first cylinder with a malfunction:

```
select log_ts,vin_nbr,strtok(nvp(payload,'cyl_list'),',',1) as
firstBadCyl from car_data where sensor_name = 'Cylinders'
```

	log_ts	vin_nbr	firstBadCyl
1	2012-01-01 14:12:31.000000	789	2

Extract the cylinder ids with a malfunction and return as a set:

```
with bad_cyl_data (vin_nbr,cyl_list)
as (select vin_nbr,nvp(payload,'cyl_list') from car_data where
sensor_name = 'Cylinders')
select * from table
(strtok_split_to_table(bad_cyl_data.vin_nbr,bad_cyl_data.cyl_list,',')
returns (vin_nbr int, result_position int, result_item varchar(20)
character set unicode)) as split_data
```

	vin_nbr	result_position	result_item
1	21	1	2
2	21	2	5
3	21	3	6

The advantages of this approach are:

- The flexibility inherent in the name-value pair approach is maintained throughout the ETL, data processing and storage. Only at the last moment, at query time, is the string parsed out.
- A production-strength relational database is used, benefiting from all the reliability, performance, availability and management features found there. In contrast to a Hadoop solution, which could certainly be coded, the relational approach keeps all data securely controlled and preserved.
- No additional storage or processing overhead is added to the relational database to handle name-value pairs.
- Block level compression of the data exceeding 80% can be achieved.
- Business users gain direct access to the query string data and can manipulate and analyze it as they need. Only common SQL skills are required; as opposed to MapReduce programming skills needed in Hadoop.

A relational database approach often offers the best combination of features for supporting name-value pair data.

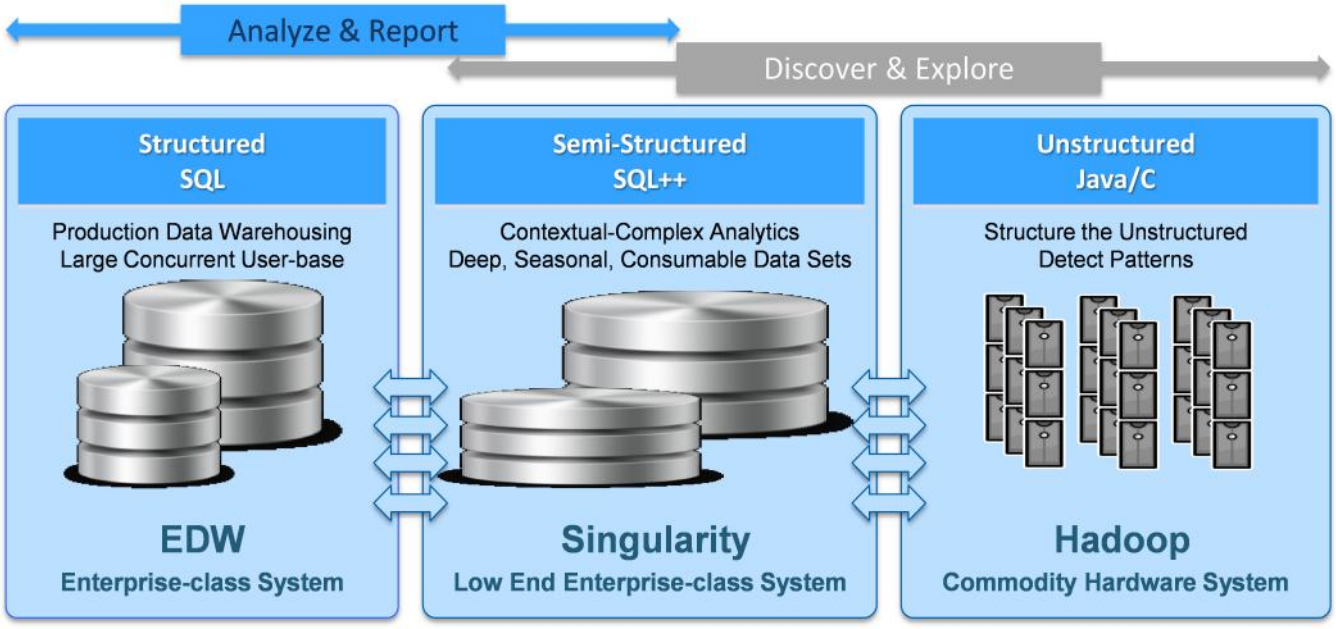
The overall architecture at eBay, with a Singularity at its heart

The above technology was integrated into eBay's overall processing environment as a system called Singularity in 2009, running on a second Teradata instance alongside the existing data warehouse.

Figure 2 shows the overall structure, now consisting of three components:

1. eBay's original BI implementation was a traditional EDW environment supporting production reporting and querying. It has grown over the past 11 years to support thousands of users with a few petabytes of structured data. This system is optimized for balanced performance and availability, taking advantage of the sophisticated workload and data storage management tools provided by Teradata's flagship Active Enterprise Data Warehouse 132 node 6690 machine.
2. The second pillar, known as Singularity (named after Ray Kurzweil's "The Singularity is Near" book), supports hundreds of users. It provides the functionality for handling name-value pair data discussed earlier, with the largest table holding over 3 PB of event data and providing a few thousand EDW tables for additional context. With a focus more on data volume and complex analytics, this system is a Teradata 1650 Extreme Data Appliance with 256 nodes, optimized for analytics and handling some four times the volume of data held on the EDW.

Figure 2: The eBay architecture



3. In 2010 eBay introduced a Hadoop-based environment, consisting now of two clusters of over 2,000 nodes each and almost ten times the data volumes of the EDW. These systems hold all the semi-structured data collected as well as hundreds of EDW tables to provide context. They support a few hundred data scientists and developers in deep discovery and pattern detection applications that explore the edges of what can be found in the Web and other data.

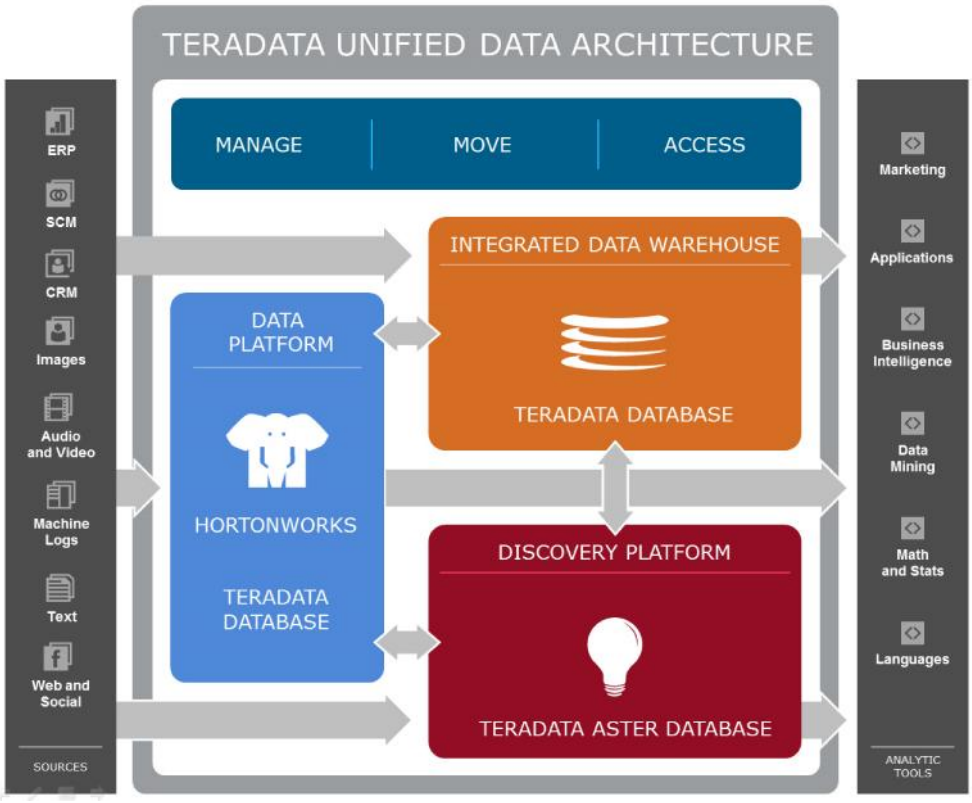
The infrastructure linking these three platforms has been built to easily move large amounts data quickly and on demand back and forth between them. In general, very large data sets are shared incrementally, while smaller data sets and user-created data sets are streamed. The design is optimized to keep the systems closely synchronized in near real-time and to avoid conflicts and race conditions between the machines.

It is also instructive to consider these three systems from right to left in figure 2, as the progression from exploration of ideas and possibilities on the right, through insight into production experiments in the center, to tracking and managing the business on the left. This represents a very logical movement from predictive analytics, through operational analytics, to reporting and tactical BI activities, each supported by a platform optimized for the processing and data needs of the different business processes.

A Unified Data Architecture

Teradata has traditionally focused on an exclusively enterprise data warehouse (EDW) point of view. That positioning has changed as the market changed over the past five years, with new requirements for operational analytics and the explosion of novel, mainly external data sources as we've already seen. Teradata began to adopt a much more inclusive approach to data management technology at least as far back as 2009, as can be seen in my white paper[5], *"Business Integrated Insight (BI²)—Reinventing enterprise information management"*, which they sponsored. The paper described how any modern environment for decision support would have to include a far broader range of data structures, types and levels of reliance. These characteristics correspond to many of the "v-words" later applied to big data. The 2011 acquisition of Aster, the subsequent Hortonworks collaboration, and the recent Teradata Unified Data Architecture (UDA) all confirm a more inclusive view.

Figure 3: Teradata Unified Data Architecture



The high-level picture[6] shown in figure 3 has been dubbed "marke-tecture" by some, but closer examination shows that there is deeper architectural

thinking behind it. Parts of the architecture have already been delivered, customers have working implementations, and further enhancements are planned. Teradata's diagram clearly identifies three distinct types of data use and processing, each supported by a different technological platform:

- ***Integrated data warehouse:*** The Teradata database, with its optimized hardware / software environment and its extensive tuning possibilities retains its role as the integrative heart of the new information environment. This is where "production" BI takes place—and much of today's BI most definitely supports the on-going production needs of the business.
- ***Discovery platform:*** The Aster database is positioned as a discovery platform, taking advantage of its columnar architecture and extensive ability to access and manipulate data locally and from the Hadoop environment to provide business users with a powerful and novel SQL-MapReduce analytics through a simple SQL interface. Previously supported by the sandboxing approach on the Teradata database (which, of course, is still possible), the Aster platform offers broader and deeper analytic possibilities to business users.
- ***Data platform:*** Hadoop and Teradata's Extreme Data Appliance complement each other in this role, handling different workloads. The Hadoop environment itself is seen both as a platform for storage of less well-structured data from external sources and as a staging and exploration environment for more programmatic analytics, at a level of complexity beyond that possible for most business users. Clearly, eBay's Singularity is fulfilling part of this role.

While the product-oriented depiction makes marketing sense, it partially obscures the fundamentally sound and well-founded architectural thinking informing the structure. This is essentially a modern, pillared, logical information architecture as described extensively in my new book[7]. Similarly, Gartner says Teradata is a leader in terms of vision for the logical data warehouse because of its UDA[8]. UDA is emerging as a reference architecture and maps directly to the eBay implementations described above.

Conclusions

Machine-generated data represents the next arriving tsunami of big data. What we've learned in the current wave will, of course, stand us in good stead. However, it would be a mistake to think that machine-generated data is the same as the human-sourced information that has, so far, constituted the majority of externally sourced big data. Nor, in many cases, is it very similar to internally sourced machine-generated data. Rather, this new data from the emerging Internet of Things has a combination of specific characteristics that distinguish it from previously considered data. First, it is loosely structured and subject to limited prior data modeling; we have little or no influence over the structure and content that arrives. Second, it is self-describing, although in a loosely managed way. Third, its structure and content is subject to change without notice at any time. Its structure, in most cases, corresponds to some form of name-value pairs, whether text or binary in nature.

The use and content of name-value pair data also differs from that of social media big data. Name-value pair data typically contains contextual or environmental information that makes sense only when tightly linked to more traditional operational/transactional data. Analytics are thus more operational in nature, and the data is more naturally used in the same environment as traditional BI work, as eBay's analysts made clear. Its semi-productional nature also mandates an environment with traditional reliability and management characteristics. In short, it is suited in the longer run to a relational rather than a Hadoop environment. Initial exploration and some specialized processing are still best done in Hadoop, but day-to-day use begs for the structure and control of a relational database environment. Teradata v14.0's built-in functionality, based on eBay's earlier work, enables the parsing and

reformatting of name-value pair data at the time of querying in a power-BI environment, allowing the inherent flexibility and compactness of the original format to be maintained as long as possible.

Machine-generated data is the third information domain, after traditional process-mediated data and the human-sourced information from social media sites that makes up much of today's big data. Handling these distinctly different data types requires an architecture structured in inter-communicating pillars, each residing on a platform optimized for specific and complementary processing and storage needs. Teradata's Unified Data Architecture provides a great example of how such an architecture can be crafted, at least in part, from an existing product base.

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 new book, ***"Business unIntelligence—Insight and Innovation Beyond Analytics and Big Data"*** *(http://bit.ly/BunI-Technics) is published in October 2013.*

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, TDWI and other publications, Barry is based in Cape Town, South Africa and operates worldwide.

---

[1] Also known as key-value, field-value or attribute-value pairs

[2] OBD-II PIDs. (2013, July 1). In *Wikipedia, The Free Encyclopedia*. Retrieved July 4, 2013, from http://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&oldid=562438124

[3] Thaler, R.H., Sunstein, C.R., *"Nudge: Improving Decisions About Health, Wealth, and Happiness"*, Yale University Press, (2008)

[4] Devlin, B., Rogers, S., Myers, J., *"Big Data comes of Age, EMA and 9sight Consulting Research Report"*, November 2012, http://bit.ly/Big_Data_Survey

[5] Devlin, B., *"Business Integrated Insight (BI2) - Reinventing enterprise information management"*, August 2009, http://bit.ly/BI2_White_Paper

[6] See http://www.teradata.com/products-and-services/unified-data-architecture

[7] Devlin, B., *"Business unIntelligence: Insight and Innovation beyond Analytics and Big Data"*, Technics Publications, (2013), http://bit.ly/BunI-Technics

[8] Gartner, Magic Quadrant for Data Warehouse Database Management Systems, January 2013