Newly Emerging Best Practices for Big Data

A Kimball Group White Paper
By Ralph Kimball

KIMBALL GROUP
Consulting | Kimball University

The big data revolution is well under way. We know that the sheer bigness of the data is not what is interesting. Rather, big data also departs severely from the familiar text and number data that we have stored in relational databases and analyzed with SQL for more than 20 years. The format and content of big data ranges from unstructured free text to highly structured relational, vectors, matrices, images, and collections of name-value pairs.

The first big shock to the system is that standard relational databases and SQL simply can't store or process big data, and are reaching fundamental capacity and scaling limits. Not only are the data formats outside the scope of relational databases, but much of the processing requires iterative logic, complex branching, and special analytic algorithms. SQL is a declarative language with a powerful but fixed syntax. Big data generally needs procedural languages and the ability to program arbitrary new logic.

The second big shock to the system is the shift away from slice and dice reporting based on simple filters and aggregations to analytics. Reports, dashboards and ad hoc queries will always be important, but big data is best exploited by combing across huge unfiltered datasets assembled by combining both historical and real time data.

Finally, the third big shock to the system is the recognition that the value of big data increases sharply as latency decreases and the data is delivered faster. Tenfold and hundredfold performance improvements result in qualitatively different analysis opportunities, often translating into increased revenue and profit.

All of this has made for a very dynamic, technology-driven marketplace with two main development threads: extended relational databases and Hadoop. I described these architectures in depth in my "Evolving Role of the Enterprise Data Warehouse in the Era of Big Data Analytics" white paper.

The big data marketplace is far from mature, but we now have several years of accumulated experience with a number of best practices specific to big data. This white paper captures these best practices, steering a middle ground between high level motherhood admonitions versus down in the weeds technical minutia specific to a single tool.

It's important to recognize we have a well-tested set of best practices developed for relationally-based enterprise data warehouses (EDWs) that big data efforts should leverage. We list them briefly:

- Drive the choice of data sources feeding the EDW from business needs
- Focus incessantly on user interface simplicity and performance

The following lists EDW best practices especially relevant to big data:

- Think dimensionally: divide the world into dimensions and facts
- Integrate separate data sources with conformed dimensions
- Track time variance with slowly changing dimensions (SCDs)
- Anchor all dimensions with durable surrogate keys

In the remainder of this paper, we divide big data best practices into four categories: data management, data architecture, data modeling, and data governance.

Management Best Practices for Big Data

The following best practices apply to the overall management of a big data environment.

Structure big data environments around analytics, not ad hoc querying or standard reporting. Every step in the data pathway from original source to analyst's screen must support complex analytic routines implemented as user defined functions (UDFs) or via a metadata driven development environment that can be programmed for each type of analysis. This includes loaders, cleansers, integrators, user interfaces, and finally BI tools. This best practice does not recommend repudiating your existing environment, but rather extending it to support the new demands of analytics. See the architectural best practices section below.

Do not attempt to build a legacy big data environment at this time. The big data environment is changing too rapidly at this time to consider building a long-lasting legacy foundation. Rather, plan for disruptive changes coming from every direction: new data types, competitive challenges, programming approaches, hardware, networking technology, and services offered by literally hundreds of new big data providers. For the foreseeable future, maintain a balance among several implementation approaches including Hadoop, traditional grid computing, pushdown optimization in an RDBMS, on-premise computing, cloud computing, and even your mainframe. No one of these approaches will be the single winner in the long run. Platform as a service (PaaS) providers offer an attractive option that can help you assemble a compatible set of tools. Similarly, much of the system architecture and programming can be specified in a layer above the specific deployment choices, a distinct advantage of metadata-driven development environments.

> Example: Use HCatalog in the Hadoop environment to provide a layer of abstraction above the specific storage location and data format. This allows Pig scripts, for instance, to remain unchanged when locations and formats have been altered.
>
> Example: Think of Hadoop as a flexible, general purpose environment for many forms of ETL processing, where the goal is to add sufficient structure and context to big data so that it can be loaded into an RDBMS. The same data in Hadoop can be accessed and transformed with Hive, Pig, HBase, and MapReduce code written in a variety of languages, even simultaneously.

Above all, this demands flexibility. Assume you will reprogram and re-host all your big data applications within two years. Choose approaches that can be reprogrammed and re-hosted. Consider using a metadata-driven codeless development environment to increase productivity and help insulate you from underlying technology changes.

Embrace sandbox silos and build a practice of productionizing sandbox results. Allow data scientists to construct their data experiments and prototypes using their preferred languages and programming environments. Then, after proof of concept, systematically reprogram and/or reconfigure these implementations with an "IT turn-over team."

> Example: Your production environment for custom analytic programming might be MatLab within PostgreSQL or SAS within a Teradata RDBMS, but your data scientists might be building their proofs of concept in a wide variety of preferred languages and architectures. The key insight here: IT must be uncharacteristically tolerant of the range of technologies the data scientists use, and be prepared in many cases to re-implement the data scientists' work in a standard set of technologies that can be supported over the long haul.

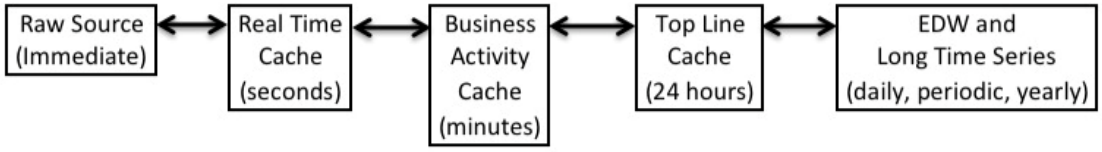
> Example: Your sandbox development environment might be a combination of ETL transformations and custom R code directly accessing Hadoop, but controlled by Informatica PowerCenter. Then when the data scientist is ready to hand over the proof of concept, much of the logic could immediately be redeployed under PowerCenter to run in a grid computing environment that is scalable, highly available, and secure.

Put your toe in the water with a simple big data application: backup and archiving. While getting started with your big data program and as you are searching for valuable business use cases with limited risk and assembling the requisite big data skills, consider using Hadoop as a low cost, flexible backup and archiving technology. Hadoop can store and retrieve data in the full range of formats from totally unstructured to highly structured specialized formats. This approach may also let you address the "sunsetting" challenge where original applications may not be available in the distant future (perhaps because of licensing restrictions), but you may dump data from those applications into your own documented format. Finally, remember that even Hadoop consumes resources and cost – so anytime data gets stored in Hadoop, data retention should be considered in advance such that HDFS folders and data sets can be purged or archived out of HDFS easily to even lower cost storage when the retention period expires.

Architecture Best Practices for Big Data

The following best practices affect the overall structure and organization of your big data environment.

Plan for a logical "data highway" with multiple caches of increasing latency. Physically implement only those caches appropriate for your environment. The data highway can have as many as five caches of increasing data latency, each with its distinct analytic advantages and tradeoffs:



* Raw source applications: credit card fraud detection, immediate complex event processing (CEP) including network stability and cyber attack detection.
* Real time applications: web page ad selection, personalized price promotions, online games monitoring, various forms of predictive and proactive monitoring.
* Business activity applications: low latency KPI dashboards pushed to users, trouble ticket tracking, process completion tracking, "fused" CEP reporting, customer service portals and dashboards, and mobile sales apps.
* Top line applications: tactical reporting, promotion tracking, mid course corrections based on social media buzz. "Top line" refers to the common practice by senior managers of seeing a quick top line review of what has happened in the enterprise over the past 24 hours.
* EDW and long time series applications: all forms of reporting, ad hoc querying, historical analysis, master data management, large scale temporal dynamics, Markov chain analysis.

Each cache that exists in a given environment is physical and distinct from the other caches. Data moves from the raw source down this highway through ETL processes. There may be multiple paths from the raw source to intermediate caches. For instance, data could go to the real time cache to drive a zero latency-style user interface, but at the same time be extracted directly into a daily top line cache that would look like a classic operational data store (ODS). Then the data from this ODS could feed the EDW. Data also flows in the reverse direction along the highway. See "implementing backflows" later in this section.

Much of the data along this highway must remain in non-relational formats ranging from unstructured text to complex multi-structured data such as images, arrays, graphs, links, matrices, and sets of name-value pairs.

Use big data analytics as a "fact extractor" to move data to the next cache. For example, the analysis of unstructured text tweets can produce a whole set of numerical, trendable sentiment measures including share of voice, audience engagement, conversation reach, active advocates, advocate influence, advocacy impact, resolution rate, resolution time, satisfaction score, topic trends, sentiment ratio, and idea impact. Also, see Splunk, a technology for extracting features from and indexing many forms of unstructured machine data; Kapow, a technology for extracting many forms of web-based data from blogs, discussion forums, websites, and portals; and of course, Informatica's HParser which can extract facts and dimensions from unstructured text documents, multi-structured XML documents and web logs, as well as industry standard structures such as market data, SWIFT, FIX, CDR, HL7, HIPAA, and many more.

Use big data integration to build comprehensive ecosystems that integrate conventional structured RDMS data, paper based documents, emails, and in-house business-oriented social networking. One of the potent messages from big data is the ability to integrate disparate data sources of very different modalities. We are getting streams of data from new data producing channels like social networks, mobile devices, and automated alert processes. Imagine a big financial institution handling millions of accounts, tens of millions of associated paper documents, and thousands of professionals both within the organization but in the field as partners or customers. Now set up a secure "social network" of all the trusted parties to communicate as business is being conducted. Much of this communication is significant and should be saved in a queryable way. Capture all this information in Hadoop, dimensionalize it (see the modeling best practices below), use it in the course of business, and then back it up and archive it.

Plan for data quality to be better further along the data highway. This is the classic tradeoff of latency versus quality. Analysts and business users must accept the reality that very low latency (i.e., immediate) data is unavoidably dirty because there are limits to how much cleansing and diagnosing can be done in very short time intervals. Tests and corrections on individual field contents can be performed at the fastest data transfer rates. Tests and corrections on structural relationships among fields and across data sources are necessarily slower. Tests and corrections involving complex business rules range from being instantaneous (such as a set of dates being in a certain order) to taking arbitrarily long times (such as waiting to see if a threshold of unusual events has been exceeded). And finally, slower ETL processes, such as those feeding the daily top line cache, often are built on fundamentally more complete data, for example where incomplete transaction sets and repudiated transactions have been eliminated. In these cases, the instantaneous data feeds simply do not have the correct information.

Apply filtering, cleansing, pruning, conforming, matching, joining, and diagnosing at the earliest touch points possible. This is a corollary of the previous best practice. Each step on the data highway provides more time to add value to the data. *Filtering, cleansing and pruning* reduces the amount of data transferred to the next cache and eliminates irrelevant or corrupted data. To be fair, there is a school of thought that applies cleansing logic only at analysis run time, because cleansing might delete

"interesting outliers." *Conforming* takes the active step of placing highly administered enterprise attributes into major entities like customer, product, and date. The existence of these conformed attributes allows *high value joins* to be made across separate application domains. A shorter name for this step is "integration!" *Diagnosing* allows many interesting attributes to be added to data, including special confidence tags and textual identifiers representing behavior clusters identified by a data mining professional. Data discovery and profiling aids in the identification of data domains, relationships, metadata tags useful for search, sensitive data, and data quality issues.

Implement backflows, especially from the EDW, to earlier caches on the data highway. The highly administered master dimensions in the EDW, such as customer, product, and date, should be connected back to data in earlier caches. Ideally, all that is needed is unique durable keys for these entities in all the caches. The corollary here is that Job One in each ETL step from one cache to the next is to replace idiosyncratic proprietary keys with the unique durable keys so that analysis in each cache can take advantage of the rich upstream content with a simple join on the unique durable key. Can this ETL step be performed even when transferring raw source data into the real time cache in less than a second? Maybe…

Dimension data is not the only data to be transferred back down the highway toward the source. Derived data from fact tables, such as historical summaries and complex data mining findings, can be packaged as simple indicators or grand totals - and then transferred to earlier caches on the data highway. And finally, reference links such as useful keys or codes can be embedded in the low latency data caches in order to allow an analyst to link to other relevant data with a single click.

Implement streaming data analytics in selected data flows. An interesting angle on low latency data is the desire to begin serious analysis on the data as it streams in, but possibly far before the data transfer process terminates. There is significant interest in streaming analysis systems which allow SQL-like queries to process the data as it flows into the system. In some use cases, when the results of a streaming query surpass a threshold, the analysis can be halted without running the job to the bitter end. An academic effort, known as continuous query language (CQL), has made impressive progress in defining the requirements for streaming data processing including clever semantics for dynamically moving time windows on the streaming data. Look for CQL language extensions and streaming data query capabilities in the load programs for both RDBMSs and HDFS deployed data sets. An ideal implementation would allow streaming data analysis to take place while the data is being loaded at gigabytes per second.

Implement far limits on scalability to avoid a "boundary crash." In the early days of computer programming, when machines had pathetically small hard drives and real memories, boundary crashes were common and were the bane of applications development. When the application runs out of disk space or real memory, the developer must resort to elaborate measures, usually requiring significant programming that adds nothing to the main content of the application. Boundary crashes for normal database applications have more or less been eliminated, but big

data raises this issue once again. Hadoop is an architecture that dramatically reduces programming scalability concerns because one can, for the most part, add commodity hardware indefinitely. Of course, even commodity hardware must be provisioned, plugged in and have high bandwidth network connections. The lesson is to plan very far ahead for scaling out to huge volumes and throughputs.

Perform big data prototyping on a public cloud and then move to a private cloud. The advantage of a public cloud is that it can be provisioned and scaled up instantly. Examples include Amazon EMR and Google BigQuery. In those cases where the sensitivity of the data allows quick in-and-out prototyping, this can be very effective. Just remember not to leave a huge data set on-line with the public cloud provider over the weekend when the programmers have gone home! However, keep in mind that in some cases where you are trying to exploit data locality with rack-aware MapReduce processes, you may not be able to use a public cloud service because they may not give you the data storage control you need.

Search for and expect 10x to 100x performance improvements over time, recognizing the paradigm shift for analysis at very high speeds. The openness of the big data marketplace has encouraged hundreds of special purpose tightly-coded solutions for specific kinds of analysis. This is a giant blessing and a curse. Once free from being controlled by a big vendor's RDBMS optimizer and inner loop, smart developers can implement spot solutions that are truly 100 times as fast as standard techniques. For instance, some impressive progress has been made on the infamous "big join" problem where a billion row dimension is joined to a trillion row fact table. For example, see Yahoo's approach for handling sparse joins on huge data sets as well as Google's Dremel and BigQuery projects. The curse is that these individual spot solutions are not yet part of a unified single architecture.

One very visible big data theme is visualization of data sets. "Flying around" a petabyte of data requires spectacular performance! Visualization of big data is an exciting new area of development that allows both analysis and discovery of unexpected features and data profiling.

Another exciting application that imposes huge performance demands is semantic zooming without pre-aggregations, in which the analyst descends from a highly aggregated level to progressively more detailed levels in unstructured or semi-structured data, analogous to zooming in on a map.

The important lesson behind this best practice is that revolutionary advances in our power to consume and analyze big data will result from 10x to 100x performance gains, and we have to be prepared to add these developments to our suite of tools.

Separate big data analytic workloads from the conventional EDW to preserve EDW service level agreements. If your big data is hosted in Hadoop, it probably doesn't compete for resources with your conventional RDBMS-based EDW. However, be cautious if your big data analytics run on the EDW machine since big data requirements change rapidly and inevitably in the direction of requiring more compute resources.

Exploit unique capabilities of in-database analytics. The major RDBMS players all invest significantly in in-database analytics. Once you pay the price of loading data into relational tables, SQL can be combined with analytic extensions in extremely powerful ways. Recent notable in-database developments include IBM's acquisition of Netezza and SPSS, Teradata and Greenplum's embedding of SAS, Oracle's Exadata R Enterprise, and PostgreSQL's syntax for programming analytics and other arbitrary functions with the database inner loop. All of these options make available tested libraries of hundreds of analytic routines. Some data integration platforms provide pushdown optimization to leverage in-database analytics as part of a data flow or ELT process.

Data Modeling Best Practices for Big Data

The following best practices affect the logical and physical structures of the data.

Think dimensionally: divide the world into dimensions and facts. Business users find the concept of dimensions to be natural and obvious. No matter what the format of the data, the basic associated entities such as customer, product, service, location, or time can always be found. In the following best practice we will see how, with a little discipline, dimensions can be used to integrate data sources. But before we can get to the integration finish line, we must identify the dimensions in each data source and attach them to every low level atomic data observation. This process of dimensionalization is a good application for big data analytics. For example, a single Twitter tweet "Wow! That is awesome!" may not seem to contain anything worth dimensionalizing, but with some analysis we often can get customer (or citizen or patient), location, product (or service or contract or event), marketplace condition, provider, weather, cohort group (or demographic cluster), session, triggering prior event, final outcome, and the list goes on. Some form of automated dimensionalizing is required to stay ahead of the high velocity streams of data. As we point out in a subsequent best practice, incoming data should be fully dimensionalized at the earliest extraction step.

Integrate separate data sources with conformed dimensions. Conformed dimensions are the glue that holds together separate data sources, and allow them to be combined in a single analysis. Conformed dimensions are perhaps the most powerful best practice from the conventional EDW world that should be inherited by big data.

The basic idea behind conformed dimensions is the presence of one or more *enterprise attributes* (fields) in the versions of dimensions associated with separate data sources. For instance, every customer-facing process in an enterprise will have some variation of a customer dimension. These variations of the customer dimension may have different keys, different field definitions, and even different granularity. But even in the worst cases of incompatible data, one or more enterprise attributes can be defined that can be embedded in all the customer dimension variations. For instance, a customer demographic category is a plausible choice. Such a descriptor could be attached to nearly every customer dimension, even those at higher levels of aggregation. Once this has been done, analyses of that group on this customer demographic category can cross every participating data source with a simple sort-

merge process after separate queries are run against the different data sources. Best of all, the step of introducing the enterprise attributes into the separate databases can be done in an incremental, agile, and non-disruptive way as described in detail in my "Essential Steps for the Integrated EDW" white paper on this subject. All existing analysis applications will continue to run as the conformed dimension content is rolled out.

Anchor all dimensions with durable surrogate keys. If there is one lesson we have learned in the EDW world, it is not to anchor your major entities such as customer, product, and time with the "natural keys" defined by a specific application. These natural keys turn out to be a snare and a delusion in the real world. They are incompatible across applications and are poorly administered. The first step in every data source is to augment the natural key coming from a source with an enterprise-wide durable surrogate key. *Durable* mean that there is no business rule that can change the key. The durable key belongs to IT, not to the data source. *Surrogate* means that the keys themselves are simple integers either assigned in sequence or generated by a robust hashing algorithm that guarantees uniqueness. An isolated surrogate key has no applications content. It is just an identifier.

The big data world is filled with obvious dimensions that must possess durable surrogate keys. Earlier in this paper when we proposed pushing data backwards down the data highway, we relied on the presence of the durable surrogate keys to make this process work. We also stated that Job One on every data extraction from a raw source was to embed the durable surrogate keys in the appropriate dimensions.

Expect to integrate structured and unstructured data. Big data considerably broadens the integration challenge. Much big data will never end up in a relational database; rather it will stay in Hadoop or a grid. But once we are armed with conformed dimensions and durable surrogate keys, all the forms of data can be combined in single analyses. For example, a medical study can select a group of patients with certain demographic and health status attributes, and then combine their conventional EDW-style data with image data (photographs, x-rays, EKGs), free form text data (physician's notes), social media sentiments (opinions of treatment), and cohort group linkages (patients with similar situations).

Architecturally, this integration step needs to take place at query time, not at data load and structure time. The most flexible way to perform this integration is through data virtualization, where the integrated data sets appear to be physical tables but are actually specifications similar to relational views where the separate data sources are joined at query time. If data virtualization is not used, then the final BI layer must accomplish this integration step.

Track time variance with slowly changing dimensions (SCDs). Tracking time variance of dimensions is an old and venerable best practice from the EDW world. Basically, it makes good on the pledge we take to track history accurately. It is unacceptable to associate a current profile of a customer (or citizen, or patient, or student) with old history. In the worst case, the current profile is ridiculously wrong when applied to old history. Slowly changing dimension (SCD) processing comes in three flavors. The Type 1 SCD overwrites the profile when a change takes place, thereby losing history.

We may do this when we correct a data error. The Type 2 SCD is the most often used technique that generates a revised dimension record when a change takes place. The Type 2 SCD requires that when we generate the new dimension record, we retain the durable surrogate key as the glue that binds the new record to the old records, but we must also generate a unique primary key for the particular snapshot of the dimension member. Like conformed dimensions, this process has been described and vetted extensively. Finally the Type 3 SCD, which is not as common as the other two types, covers the situation where an "alternate reality" is defined that co-exists with the current reality. Please see the extensive coverage of SCDs in my books and on the kimballgroup.com website. But the point as far as big data is concerned is that it is just as important to associate the correct contemporary profile of a major entity with history as it has proven to be in the EDW world.

Get used to not declaring data structures until analysis time. One of the charms of big data is putting off declaring data structures at the time of loading into Hadoop or a data grid. This brings many advantages. The data structures may not be understood at load time. The data may have such variable content that a single data structure either makes no sense or forces you to modify the data to fit into a structure. If you can load data into Hadoop, for instance, without declaring its structure, you can avoid a resource intensive step. And finally, different analysts may legitimately see the same data in different ways. Of course, there is a penalty in some cases, because data without a declared structure may be difficult or impossible to index for rapid access as in an RDBMS. However, most big data analysis algorithms process entire data sets without expecting precise filtering of subsets of the data.

This best practice conflicts with traditional RDBMS methodologies, which puts a lot of emphasis on modeling the data carefully before loading. But this does not lead to a deadly conflict. For data destined for an RDBMS, the transfer from a Hadoop or data grid environment and from a name-value pair structure into RDBMS named columns can be thought of as a valuable ETL step.

Build technology around name-value pair data sources. Big data sources are filled with surprises. In many cases, you open the fire hose and discover unexpected or undocumented data content which you must nevertheless load at gigabytes per second. The escape from this problem is to load such data as simple name-value pairs. For example, if an applicant were to disclose their financial assets, they might declare something unexpected like "rare postage stamp = $10,000". In a name-value pair data set, this would be loaded gracefully even though you had never seen "rare postage stamp" and didn't know what to do with it at load time. Of course, this practice meshes nicely with the previous practice of deferring the declaration of data structures until past load time.

The MapReduce programming framework *requires* data to be presented as name-value pairs, which makes sense given the complete possible generality of big data.

Use data virtualization to allow rapid prototyping and schema alterations. Data virtualization is a powerful technique for declaring different logical data structures on underlying physical data. Standard view definitions in SQL are a good example of data virtualization. In theory, data virtualization can present a data source in any

format the analyst needs. But data virtualization trades off the cost of computing at run time with the cost of ETL to build physical tables before run time. Data virtualization is a powerful way to prototype data structures and make rapid alterations or provide distinct alternatives. The best data virtualization strategy is to expect to materialize the virtual schemas when they have been tested and vetted and the analysts want the performance improvements of actual physical tables.

Data Governance Best Practices for Big Data

The following best practices apply to managing your data as a valuable enterprise asset.

There is no such thing as big data governance. Now that we have your attention, the point is that data governance must be a comprehensive approach for your entire data ecosystem, not a spot solution for big data in isolation. Data governance for big data should be an extension of your approach for the governance of all your enterprise data. We have introduced a compelling case for how big data must be enhanced by integration with other existing forms of data, especially data from your EDW. But successful integration while establishing (or ignoring) data governance for big data in isolation leads to significant risk. At a minimum, data governance embraces privacy, security, compliance, data quality, metadata management, master data management, and the business glossary that exposes definitions and context to the business community. This is an imposing and daunting list of responsibilities and competencies, and IT should not attempt to define these without significant and sophisticated support from management - who must understand the scope of the effort, and support the cross-organizational cooperation required.

Dimensionalize the data before applying governance. Here is an interesting challenge big data introduces: you must apply data governance principles even when you don't know what to expect from the content of the data. You will receive data arriving at up to gigabytes per second, often as name-value pairs with unexpected content. Your best chance at classifying data in ways that are important to your data governance responsibilities is to dimensionalize it as fully as possible at the earliest stage in your data pipeline. Parse it, match it, and apply identity resolution on the fly. We made this same point when arguing for the benefits of data integration, but here we advocate against using the data before this dimensionalizing step.

If analyzing data sets including identifying information about individuals or organizations, privacy is the most important governance perspective when working with any big data set incorporating information about individuals or organizations. Although every aspect of data governance looms as critically important, in these cases privacy carries the most responsibility and the most business risk. Egregious episodes of compromising the privacy of individuals or groups can damage your reputation, diminish marketplace trust, expose you to civil lawsuits, and get you in trouble with the law. These compromises can also be a barrier to sharing rich data sets between companies, institutions, third parties, and even within organizations severely limiting the power of big data in industries such as healthcare, education, and law enforcement. The flood of personal data to which we have access threatens

to dull our senses and lower our guards. At the very least, for most forms of analysis, personal details must be masked, and data aggregated enough to not allow identification of individuals. Note that at the time of this writing special attention must be paid when storing sensitive data in Hadoop since once data gets written to Hadoop, Hadoop doesn't manage updates very well – so data should either be masked or encrypted on write (i.e., persistent data masking) or data should be masked on read (i.e., dynamic data masking).

Don't put off data governance completely in your rush to use big data. Even for your exploratory big data prototype projects, maintain a checklist of issues to consider as you go forward. You don't want an ineffective bureaucracy, but maybe you can strive to deliver an agile bureaucracy! Your checklist, which you maintain discretely, should:

- Verify there is a vision and a business case that is providing direction and priorities.
- Identify people roles including data stewards, sponsors, program drivers and users.
- Verify organization buy-in and cross-organization steering committees and sponsorship to support escalations and prioritization.
- Qualify existing and required tools and architecture that will support the big data lifecycle being managed.
- Incorporate some notion of data usage policies and data quality standards.
- Embrace lightweight organizational change management for all other points on this list.
- Measure results, operational as well as business value ROI.
- Assess and influence dependent processes, upstream and downstream to minimize the ever-present garbage in/garage out dilemma.

Summary

Big data brings a host of changes and opportunities to IT and it is easy to think that a whole new set of rules must be created. But with the benefit of almost a decade of experience, many best practices have emerged. Many of these practices are recognizable extensions from the EDW/BI world, and admittedly quite a few are new and novel ways of thinking about data and the mission of IT. But the recognition that the mission has expanded is welcome and is in some ways overdue. The current explosion of data-collecting channels, new data types, and new analytic opportunities mean that the list of best practices will continue to grow in interesting ways.

Acknowledgements

I am indebted to the following big data experts whom I was privileged to talk with while preparing this white paper. They are listed alphabetically by organization:

EMC: Bill Schmarzo

Fannie Mae: Javed Zaidi

Federal Aviation Administration: Wayne Larson and colleagues

Informatica: John Haddad, Robert Karel

JP Morgan Chase: Tom Savarese

Kimball Group: Margy Ross

Merck: John Maslanski

Rackspace: Pete Peterson

VertiCloud: Raymie Stata

Yahoo: George Goldenberg

References

Informatica sponsored Kimball white paper on Big Data and Hadoop: http://vip.informatica.com/?elqPURLPage=8808

Informatica sponsored Kimball white paper on conformed dimensions and integration: http://vip.informatica.com/?elqPURLPage=807

Kimball introductory articles on slowly changing dimensions: http://www.kimballgroup.com/2008/08/21/slowly-changing-dimensions/

http://www.kimballgroup.com/2008/09/22/slowly-changing-dimensions-part-2/

Yahoo's approach to the big join challenge: http://www.slideshare.net/Hadoop_Summit/yahoo-display-advertising-attribution

Presentation on semantic zooming without pre-aggregations: http://www.slideshare.net/Hadoop_Summit/empowering-semantic-zooming-with-hadoop-and-hbase