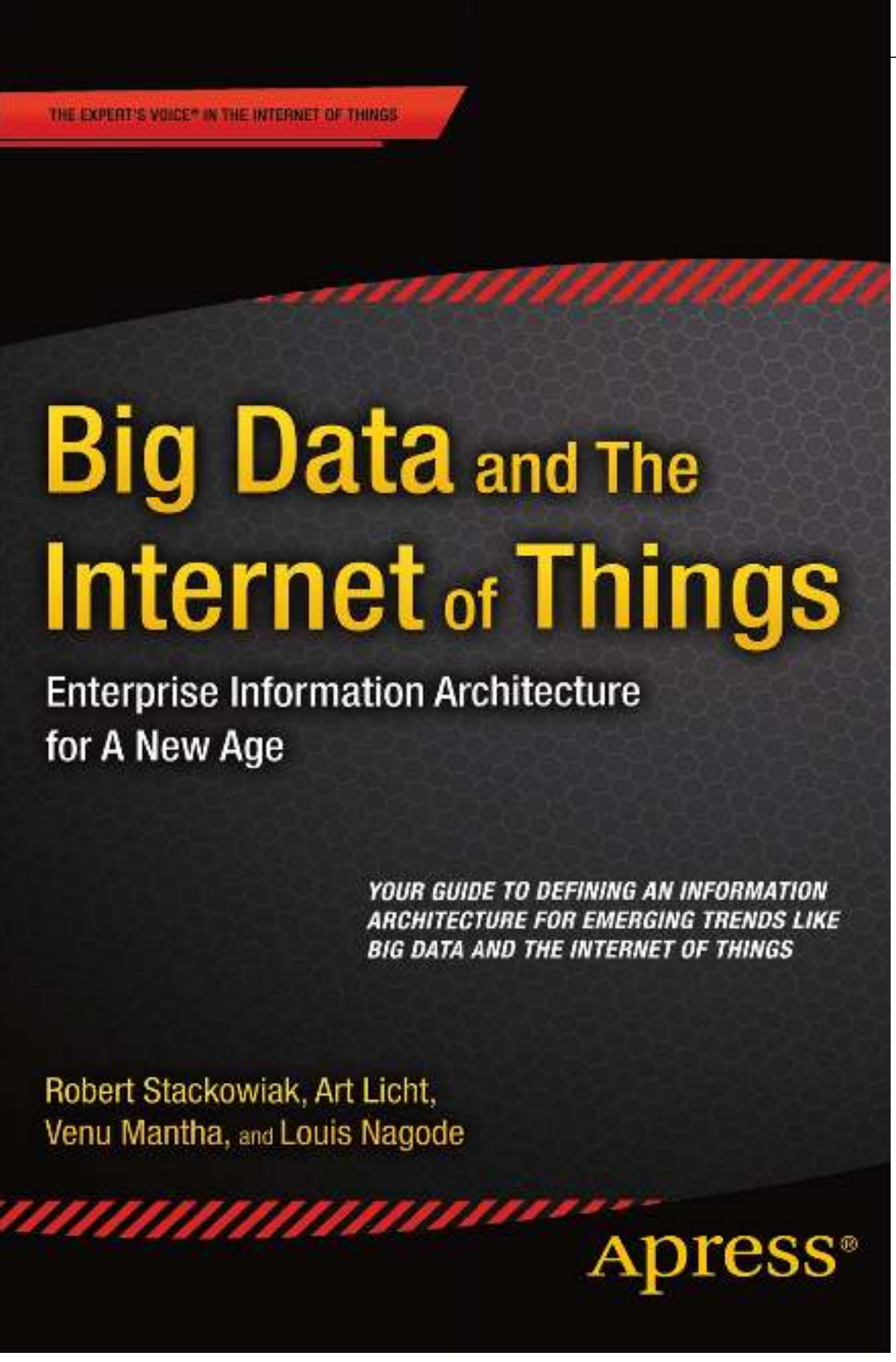
THE EXPERT'S VOICE® IN THE INTERNET OF THINGS
Big Data and The Internet of Things
Enterprise Information Architecture for A New Age
YOUR GUIDE TO DEFINING AN INFORMATION ARCHITECTURE FOR EMERGING TRENDS LIKE BIG DATA AND THE INTERNET OF THINGS
Robert Stackowiak, Art Licht, Venu Mantha, and Louis Nagode
Apress®

Contents at a Glance

Introduction

The genesis of this book began in 2012. Hadoop was being explored in mainstream organizations, and we believed that information architecture was about to be transformed. For many years, business intelligence and analytics solutions had centered on the enterprise data warehouse and data marts, and on the best practices for defining, populating, and analyzing the data in them. Optimal relational database design for structured data and managing the database had become the focus of many of these efforts. However, we saw that focus was changing.

For the first time, streaming data sources were seen as potentially important in solving business problems. Attempts were made to explore such data experimentally in hope of finding hidden value. Unfortunately, many efforts were going nowhere. The authors were acutely aware of this as we were called into many organizations to provide advice.

We did find some organizations that were successful in analyzing the new data sources. When we took a step back, we saw a common pattern emerging that was leading to their success. Prior to starting Big Data initiatives, the organizations' stakeholders had developed theories about how the new data would improve business decisions. When building prototypes, they were able to prove or disprove these theories quickly.

This successful approach was not completely new. In fact, many used the same strategy when developing successful data warehouses, business intelligence, and advanced analytics solutions that became critical to running their businesses. We describe this phased approach as a methodology for success in this book. We walk through the phases of the methodology in each chapter and describe how they apply to Big Data and Internet of Things projects.

Back in 2012, we started to document the methodology and assemble artifacts that would prove useful when advising our clients, regardless of their technology footprint. We then worked with the Oracle Enterprise Architecture community, systems integrators, and our clients in testing and refining the approach.

At times, the approach led us to recommend traditional technology footprints. However, new data sources often introduced a need for Hadoop and NoSQL database solutions. Increasingly, we saw Internet of Things applications also driving new footprints. So, we let the data sources and business problems to be solved drive the architecture.

About two years into running our workshops, we noticed that though many books described the technical components behind Big Data and Internet of Things projects, they rarely touched on how to evaluate and recommend solutions aligned to the information architecture or business requirements in an organization. Fortunately, our friends at Apress saw a similar need for the book we had in mind.

This book does not replace the technical references you likely have on your bookshelf describing in detail the components that can be part of the future state information architecture. That is not the intent of this book. (We sometimes ask enterprise architects what components are relevant, and the number quickly grows into the hundreds.)

Our intent is to provide you with a solid grounding as to how and why the components should be brought together in your future state information architecture. We take you through a methodology that establishes a vision of that future footprint; gathers business requirements, data, and analysis requirements; assesses skills; determines information architecture changes needed; and defines a roadmap. Finally, we provide you with some guidance as to things to consider during the implementation.

We believe that this book will provide value to enterprise architects where much of the book's content is directed. But we also think that it will be a valuable resource for others in IT and the lines of business who seek success in these projects.

Helping you succeed is our primary goal. We hope that you find the book helps you reach your goals.

CHAPTER 1

Big Data Solutions and the Internet of Things

This book begins with a chapter title that contains two of the most hyped technology concepts in information architecture today: *Big Data* and the *Internet of Things*. Since this book is intended for enterprise architects and information architects, as well as anyone tasked with designing and building these solutions or concerned about the ultimate success of such projects, we will avoid the hype. Instead, we will provide a solid grounding on how to get these projects started and ultimately succeed in their delivery. To do that, we first review how and why these concepts emerged, what preceded them, and how they might fit into your emerging architecture.

The authors believe that Big Data and the Internet of Things are important evolutionary steps and are increasingly relevant when defining new information architecture projects. Obviously, you think the technologies that make up these solutions could have an important role to play in your organization's information architecture as you are reading this book. Because we believe these steps are evolutionary, we also believe that many of the lessons learned previously in developing and deploying information architecture projects can and should be applied in Big Data and Internet of Things projects.

Enterprise architects will continue to find value in applying agile methodologies and development processes that move the organization's vision forward and take into account business context, governance, and the evolution of the current state architecture into a desired future state. A critical milestone is the creation of a roadmap that lays out the prioritized project implementation phases that must take place for a project to succeed.

Organizations already successful in defining and building these next generation solutions have followed these best practices, building upon previous experience they had gained when they created and deployed earlier generations of information architecture. We will review some of these methodologies in this chapter.

On the other hand, organizations that have approached Big Data and the Internet of Things as unique technology initiatives, experiments, or resume building exercises often struggle finding value in such efforts and in the technology itself. Many never gain a connection to the business requirements within their company or organization. When such projects remain designated as purely technical research efforts, they usually reach a point where they are either deemed optional for future funding or declared outright failures. This is unfortunate, but it is not without precedence.

In this book, we consider Big Data initiatives that commonly include traditional data warehouses built with relational database management system (RDBMS) technology, Hadoop clusters, NoSQL databases, and other emerging data management solutions. We extend the description of initiatives driving the adoption of the extended information architecture to include the Internet of Things where sensors and devices with intelligent controllers are deployed. These sensors and devices are linked to the infrastructure to enable analysis of data that is gathered. Intelligent sensors and controllers on the devices are designed to trigger immediate actions when needed.

So, we begin this chapter by describing how Big Data and the Internet of Things became part of the long history of evolution in information processing and architecture. We start our description of this history at a time long before such initiatives were imagined. Figure 1-1 illustrates the timeline that we will quickly proceed through.

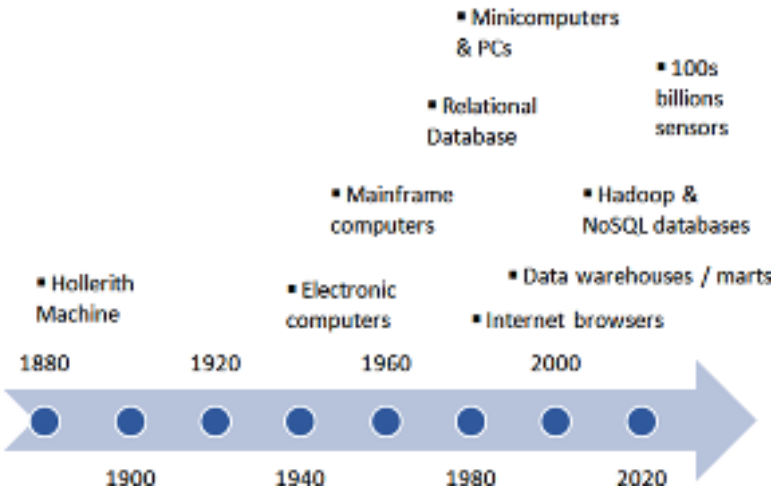

Figure 1-1. *Evolution in modern computing timeline*

From Punched Cards to Decision Support

There are many opinions as to when modern computing began. Our historical description starts at a time when computing moved beyond mechanical calculators. We begin with the creation of data processing solutions focused on providing specific information. Many believe that an important early data processing solution that set the table for what was to follow was based on punched cards and equipment invented by Herman Hollerith.

The business problem this invention first addressed was tabulating and reporting on data collected during the US census. The concept of a census certainly wasn't new in the 1880s when Hollerith presented his solution. For many centuries, governments had manually collected data about how many people lived in their territories. Along the way, an expanding array of data items became desirable for collection such as citizen name, address, sex, age, household size, urban vs. rural address, place of birth,

level of education, and more. The desire for more of these key performance indicators (KPIs) combined with population growth drove the need for a more automated approach to data collection and processing. Hollerith's punched card solution addressed these needs. By the 1930s, the technology had become widely popular for other kinds of data processing applications such as providing the footprint for accounting systems in large businesses.

The 1940s and the World War II introduced the need to solve complex military problems at a faster pace, including the deciphering of messages hidden by encryption and calculating the optimal trajectories for massive guns that fired shells. The need for rapid and incremental problem solving drove the development of early electronic computing devices consisting of switches, vacuum tubes, and wiring in racks that filled entire rooms. After the war, research in creating faster computers for military initiatives continued and the technology made its way into commercial businesses for financial accounting and other uses.

The following decades saw the introduction of modern software operating systems and programming languages (to make applications development easier and faster) and databases for rapid and simpler retrieval of data. Databases evolved from being hierarchical in nature to the more flexible relational model where data was stored in tables consisting of rows and columns. The tables were linked by foreign keys between common columns within them. The Structured Query Language (SQL) soon became the standard means of accessing the relational database.

Throughout the early 1970s, application development focused on processing and reporting on frequently updated data and came to be known as online transaction processing (OLTP). Software development was predicated on a need to capture and report on specific KPIs that the business or organization needed. Though transistors and integrated circuits greatly increased the capabilities of these systems and started to bring down the cost of computing, mainframes and software were still too expensive to do much experimentation.

All of that changed with the introduction of lower cost minicomputers and then personal computers during the late 1970s and early 1980s. Spreadsheets and relational databases enabled more flexible analysis of data in what initially were described as decision support systems. But as time went on and data became more distributed, there was a growing realization that inconsistent approaches to data gathering led to questionable analysis results and business conclusions. The time was right to define new approaches to information architecture.

The Data Warehouse

Bill Inmon is often described as the person who provided the first early definition of the role of these new data stores as "data warehouses". He described the data warehouse as "a subject oriented, integrated, non-volatile, and time variant collection of data in support of management's decisions". In the early 1990s, he further refined the concept of an enterprise data warehouse (EDW). The EDW was proposed as the single repository of all historic data for a company. It was described as containing a data model in third normal form where all of the attributes are atomic and contain unique values, similar to the schema in OLTP databases.

Figure 1-2 illustrates a very small portion of an imaginary third normal form model for an airline ticketing data warehouse. As shown, it could be used to analyze individual airline passenger transactions, airliner seats that are ticketed, flight segments, ticket fares sold, and promotions / frequent flyer awards.

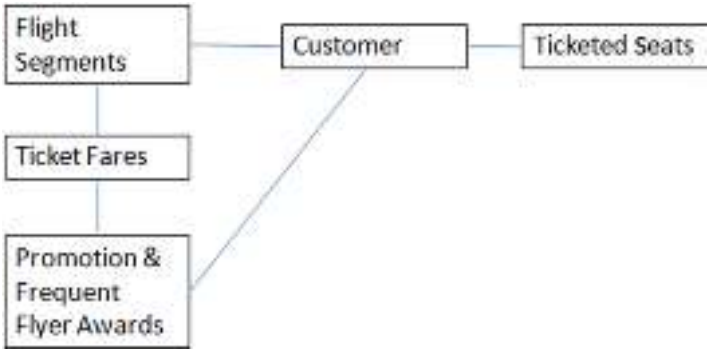

Figure 1-2. *Simple third normal form (3NF) schema*

The EDW is loaded with data extracted from OLTP tables in the source systems. Transformations are used to gain consistency in data definitions when extracting data from a variety of sources and for implementation of data quality rules and standards. When data warehouses were first developed, the extraction, transformation, and load (ETL) processing between sources and targets was often performed on a weekly or monthly basis in batch mode. However, business demands for near real-time data analysis continued to push toward more frequent loading of the data warehouse. Today, data loading is often a continuous trickle feed, and any time delay in loading is usually due to the complexity of transformations the data must go through. Many organizations have discovered that the only way to reduce latency caused by data transformations is to place more stringent rules on how data is populated initially in the OLTP systems, thus ensuring quality and consistency at the sources and lessoning the need for transformations.

Many early practitioners initially focused on gathering all of the data they could in the data warehouse, figuring that business analysts would determine what to do with it later. This “build it and they will come” approach often led to stalled projects when business analysts couldn’t easily manipulate the data that was needed to answer their business questions. Many business analysts simply downloaded data out of the EDW and into spreadsheets by using a variety of extractions they created themselves. They sometimes augmented that data with data from other sources that they had access to. Arguments ensued as to where the single version of the truth existed. This led to many early EDWs being declared as failures, so their designs came under reevaluation.

■ **Note** If the EDW "build and they will come" approach sounds similar to approaches being attempted in IT-led Hadoop and NoSQL database projects today, the authors believe this is not a coincidence. As any architect knows, form should follow function. The reverse notion, on the other hand, is not the proper way to design solutions. Unfortunately, we are seeing history repeating itself in many of these Big Data projects, and the consequences could be similarly dismal until the lessons of the past are relearned.

As debates were taking place about the usefulness of the EDW within lines of business at many companies and organizations, Ralph Kimball introduced an approach that appeared to enable business analysts to perform ad hoc queries in a more intuitive way. His star schema design featured a large fact table surrounded by dimension tables (sometimes called look-up tables) and containing hierarchies. This schema was popularly deployed in data marts, often defined as line of business subject-oriented data warehouses.

To illustrate its usefulness, we have a very simple airline data mart illustrated in Figure 1-3. We wish to determine the customers who took flights from the United States to Mexico in July 2014. As illustrated in this star schema, customer transactions are in held in the fact table. The originating and destination dimension tables contain geographic drill-down information (continent, country, state or province, city, and airport identifier). The time dimension enables drill down to specific time periods (year, month, week, day, hour of day).

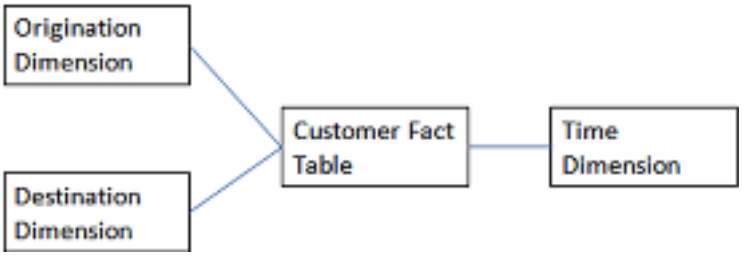

Figure 1-3. *Simple star schema*

Not all relational databases were initially adept at providing optimal query performance where a star schema was defined. These performance challenges led to the creation of multidimensional online analytics processing (MOLAP) engines especially designed to handle the hierarchies and star schema. MOLAP engines performed so well because these "cubes" consisted of pre-joined drill paths through the data. Figure 1-4 pictures a physical representation of a three-dimensional cube.

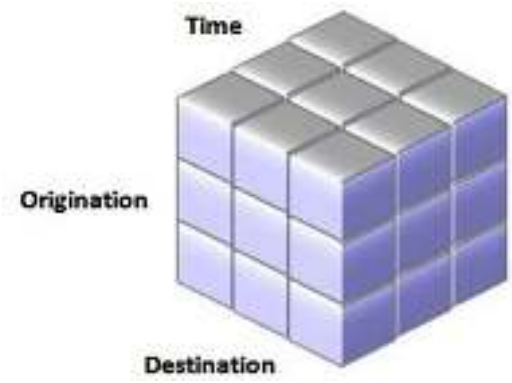

Figure 1-4. *Three-dimensional cube representation*

Later, as relational database optimizers matured, it became possible to achieve good query performance when deploying the star schema within the relational database management system. These became known as relational online analytical processing (ROLAP) implementations.

Independent vs. Dependent Data Marts

In the mid-1990s, there was much debate about the usefulness of the EDW when compared to data marts. When business analysts found the star schema was easier to navigate (and often deployed their own marts), some IT database programmers responded by creating views over the top of the data and schema in their EDW to overcome this objection. However, the programming and maintenance effort in building views was typically not timely enough to meet the growing demands of business analysts.

Another problem often arose. When individual data marts are defined and deployed independently of each other and don't follow data definition rules established within the EDW, inconsistent representation of the common data can call into question where the true data is housed. Figure 1-5 illustrates the complexity that can emerge when various lines of business build their own independent data marts and extract data directly from OLTP sources. In actual implementations, the complexity is sometimes greater than what is shown here as data might flow directly between data marts as well. Spreadsheets might also be added to this illustration serving as business intelligence tools tied to unique storage and representations of data. Organizations that deploy in this manner generally spend a great amount of time in business meetings arguing about who has the correct report representing the true state of the business, even if the reports are supposed to show the same KPIs.

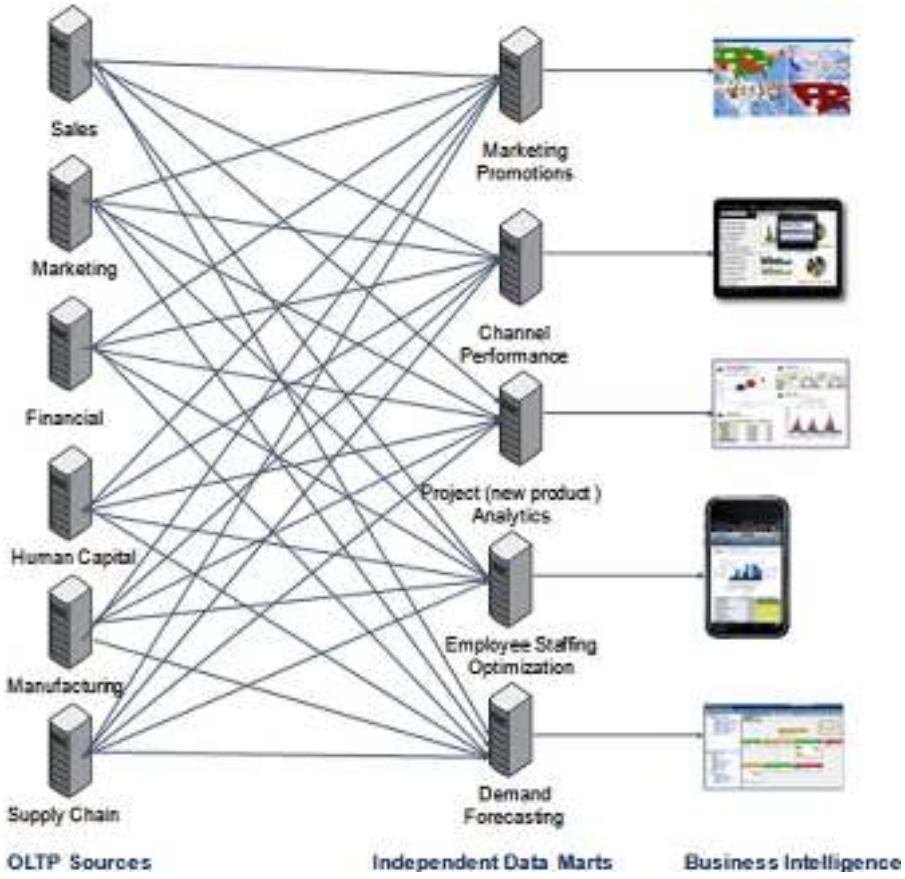

Figure 1-5. *Independent data marts with unique ETL between sources and target marts*

In the end, the debate should not have been about EDWs vs. data marts. There were solid reasons why both approaches had merit where the right architectural guidelines were applied. As many information and data warehouse architects began to realize this, a blended approach became the best practice. EDWs were implemented and extended incrementally as new sources of data were also required in the data marts. The data marts were made dependent upon the data definitions in the EDW. As the EDW remains the historic database of record, data fed into the marts is extracted from the EDW. The exception to using the EDW as the source of all data typically occurred when there was unique third-party data that was relevant to only a single line of business in an organization. Then that unique data was stored only in that line of business's data mart.

Figure 1-6 illustrates data marts dependent on the EDW. This approach often leads to defining conformed dimensions to establish consistency across data marts. When conformed dimensions are defined, it is possible to submit a single query that accesses data from multiple data marts (since the dimensions represent the same hierarchies in the various data marts).

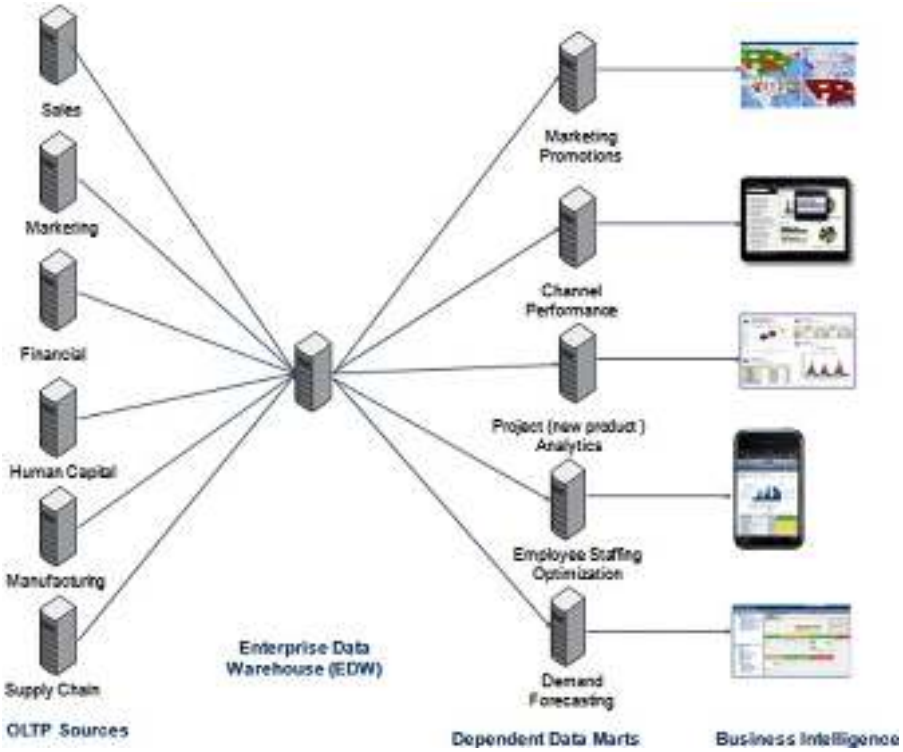

Figure 1-6. *Dependent data marts with ETL from the EDW, the trusted source of data*

Database data management platforms you are most likely to encounter as data warehouses and / or data mart engines include the following: Oracle (Database Enterprise Edition and Essbase), IBM (DB2 and Netezza), Microsoft SQL Server, Teradata, SAP HANA, and HP Vertica. ETL tools that are commonly deployed include Informatica, Oracle Data Integrator, IBM DataStage, and Ab Initio.

■ **Note** When the EDW and data marts first became central and mandatory to running the business, information architects began to understand the need for these platforms to also be highly available, recoverable, and secure. As Hadoop clusters and NoSQL databases are assuming similar levels of importance to lines of business today, the demand for similar capabilities in these platforms is driving the creation of new features and capabilities in these distributions. This is illustrated by the growing focus on improved availability, recoverability, and security in the more recent software releases in the open source community and being offered by the various vendors creating distributions.

An Incremental Approach

Early data warehousing design efforts sometimes suffered from "paralysis by over analysis" with a focus on elegant IT designs but not mapped to requirements from lines of business. Designs of early EDWs often took 12 months or more, well outside the bounds of business needs or the attention spans of business sponsors. Some early practitioners relied on a classic waterfall approach where the scope of the effort for the entire EDW was first determined, and then time and resources were allocated.

Figure 1-7 illustrates the waterfall approach. Lengthy project plans, delays, and lack of attention to the business often led to the lines of business taking matters into their own hands, developing and deploying independent data marts, or creating pseudo data marts in spreadsheets to solve their most immediate problems.

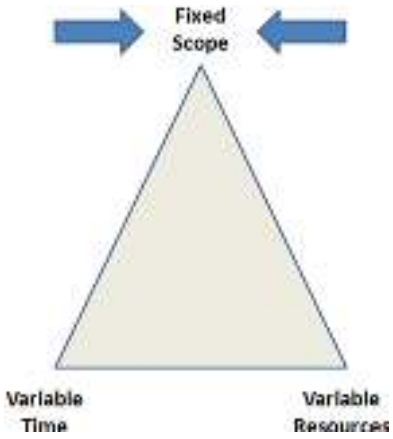

Figure 1-7. *Waterfall approach*

In light of these problems, many turned away from the waterfall approach and switched to an agile incremental approach to design and development. Partnerships were formed between IT and the lines of business. Time frames of 120 days or less for implementation and evaluation of the progress toward a business solution became commonplace in many organizations. Figure 1-8 represents the incremental approach and illustrates a fixed time and fixed resources being assigned.

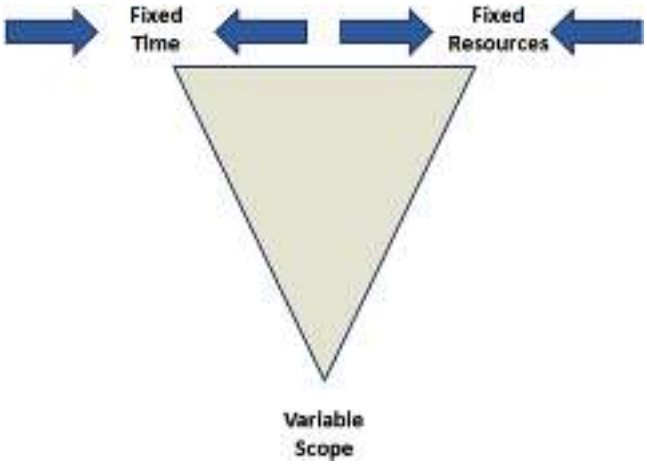

Figure 1-8. *Incremental approach*

Though Figure 1-8 shows a variable scope, there should be some real business value demonstrated that is aligned to the business goals at each increment in the process. So, in practice, the methodology applied is often a blended balancing of the incremental and waterfall approaches.

Using this approach, the usefulness of the solution is re-evaluated at each step along the way. Defining and evaluating the EDW and dependent data marts in shorter time increments means that IT designs and development can be adjusted before they became too misaligned with business expectations. Return on investment can be calculated at regular intervals and reflect any changes in scope.

Some companies choose to embed business analysts within their IT organizations to drive ongoing identification of incremental requirements. Others create business intelligence centers of excellence as virtual organizations, periodically bringing together analysts in the lines of business with IT architects and developers. Ongoing communications and flexibility among the teams is critical to success regardless of the approach used.

■ **Note** Successful Big Data projects that provide new business solutions are also usually developed using an incremental approach. Ongoing dialog in the organization among the teams regarding what is being discovered and the potential business impact is essential to success.

Faster Implementation Strategies

Early data warehouse implementations were based on entirely customized designs. Since data models had not matured and were not widely available, a significant amount of time was spent defining and designing the data models from scratch. Workload characteristics and workload changes over time were very unpredictable, making the

specification of servers and storage difficult. As lessons were learned and data warehouse designs matured, best practices became understood and solutions emerged that built upon these experiences.

One set of solutions to emerge were predefined data models based on common data analysis needs. The models became available for industries (such as retail, communications, and banking) and provided horizontal analytic solutions (for example, financials, sales, marketing, and supply chain). Such models are available today from software vendors and consulting companies and feature definitions of logical designs and sometimes also include physical schema. They cover key business areas and contain the tables and other data elements needed to start the deployment of the data warehouse. Some are also packaged with ETL scripts useful in extracting data from popular ERP and CRM transaction processing sources and loading the data into the data models. Of course, most organizations customize the models based on their own unique business requirements. However, the models do provide the starting point for many data warehousing projects and are most successfully deployed when using the incremental approach.

As we noted earlier, configuring servers and storage for data warehousing workloads also presented challenges for information architects and server and storage architects. Given that data volumes grow at a much more rapid rate than the evolution of faster access times to physical disk drives, most found their platforms became throughput-bound if not enough attention is paid to the overall system design. In recent years, the notion of deploying appliance-like platforms configured for data warehousing and data marts has become quite common. There are several such offerings available from relational database vendors who also provide servers and storage. The availability of flash in storage further helped speed performance where the database software was optimized to take advantage of the flash. More recently, the dramatic reduction in the cost of memory, introduction of new processors capable of addressing huge memory footprints, and further refinement in the database's ability to store and retrieve frequently accessed data in-memory led to huge query response and analysis performance improvements. All of these have served to mitigate many of the complex database tuning and design tasks previously necessary.

That said, as certain server and storage bottlenecks such as throughput are overcome, others will naturally arise since there is always a physical limitation somewhere in a system. Business analysts will continue to demand new analytic applications that take advantage of new platform capabilities and push the bounds of the technology.

■ **Note** At the time of publication, the number of applications for Hadoop clusters and NoSQL databases was quite small but growing. There were also a growing number of appliance-like server and storage platforms available for these data management engines. As the business value of the solutions that require such engines is understood, time to implementation and the ability to meet service levels will grow in importance. So, it is expected that the desire for such optimally configured appliances will grow and their popularity will follow a trajectory similar to what was observed in the adoption of data warehousing appliances.

Matching Business Intelligence Tools to Analysts

How data is accessed and utilized is driven by the needs and skills of the individuals in the lines of business. For those who need to see the data to make decisions, the tools they might use can range from simple reporting tools to extremely sophisticated data mining tools. Modern infrastructures sometimes also include engines for automated recommendations and actions, as well as information discovery tools.

Figure 1-9 illustrates the range of tools and techniques and their relative user community sizes and relative complexity.

Figure 1-9. *Potential business analyst and user community size for various analyst tools*

The simplest way to convey information to business analysts is through pre-defined reports that display KPIs selected by developers of the reports. Reports have limited flexibility in the information that can be viewed, but they also assure that a wide variety of business users can become consumers of the information because of the simplicity in accessing them. The reporting tools the developers use generate SQL for accessing needed data. Report developers often judge the quality of reporting tools by the clarity with which they present the KPIs and the ease and flexibility in which reports can be generated, shared, and distributed. For example, a variety of template types are often supported for printing such as PDF, RTF, and XSL.

Ad hoc query and analysis tools provide a greater degree of flexibility since business analysts can pose their own *what-if* questions by navigating database tables themselves. Developers create business metadata to translate cryptic table names into meaningful business-oriented data descriptions. The ease with which business users can navigate the data is also dependent on the underlying schema design in the database. As we described earlier, star schema with dimensional models and hierarchies are particularly easy to navigate. Figure 1-10 illustrates an interface showing a fact table consisting of sales surrounded by dimensions that include time, products, offices, and others. Behind the interface, these tools also generate SQL to access the data. In addition to flexibility, modern ad hoc query and analysis tools are judged by the data visualization capabilities these tools provide.

Figure 1-10. *Typical ad hoc query tool interface to facts and dimensions*

Typical ad hoc query, analysis, and reporting tools you are likely to see being used today include Oracle Business Intelligence Foundation Suite, SAP Business Objects, IBM Cognos, MicroStrategy, Tableau, QlikView, Pentaho, and Tibco Jaspersoft. Of course, many would say that Microsoft Excel is the most popular tool for this type of work in their organization. In most organizations, a variety of vendors' tools are used.

A smaller but growing subset of business analysts deal with massive amounts of data and seek to uncover hidden patterns and / or predict future outcomes using their data. The kinds of analyses range from the simple statistics that you probably learned in college (for example, mean, standard deviation, and so on) to models based on more sophisticated data mining algorithms.

The statistical functions that business analysts work with to bring clarity to the data usually fit in the following categories:

- Basic statistical functions such as summary, sort, rank, and frequency
- Density, probability, and quantile functions
- Special functions such as gamma functions
- Test functions such as chi square, simple and weighted kappas, and correlation

Advanced data mining algorithms are used when there is a need to understand what variables are critical in accurately predicting outcomes and in defining the predictive models that will subsequently be used to predict the outcomes. The models are often applied where there are hundreds of variables present but only a dozen or fewer that impact the outcome. The data mining algorithms can be categorized as follows:

- Clustering algorithms: Used to explore where certain business outcomes fall into to certain groups with common characteristics such as teenagers, males, and so on.
- Logic models: Used where if certain events occur, others will follow (and often referenced as decision trees).

- Neural networks: Somewhat black box mathematical models trained against sample sets with known outcomes.
- Anomaly detection algorithms: Used to detect outliers and rare events.

The vendors you are likely to find installed in your current architecture providing statistical and data mining capabilities include the SAS Institute, IBM SPSS, R (an open source statistical engine), and Oracle Advanced Analytics.

Historically, statisticians and data miners were also domain experts and were sometimes referred to as "quants." With the growing popularity of Hadoop, the new role of data scientist has emerged. Early data scientists were especially adept at using advanced programming techniques that took advantage of Hadoop's features.

■ **Note** There is much debate today about the skills and definition of the data scientist role. Some still believe the data scientist is a combination of a statistician and Hadoop programming guru. However, many hired with those skills have shown that they lack the domain expertise needed to understand what to look for in the data and the potential impact on the business. In many organizations today, data scientists are paired with business domain experts, and they work as a team to assure success.

Early in this century, it was recognized that there was a growing need to explore massive data sets that might include structured, semi-structured, and streaming data. The information discovery tools that were introduced enable exploration of data where a schema is not pre-defined. The tools generally either have their own proprietary data store engines, such as Oracle Endeca Information Discovery, or rely on Hadoop to enable exploration of data sets and combinations of data. The data analyzed is typically gathered from OLTP sources, EDWs, NoSQL databases, and Hadoop. Tibco Spotfire, Oracle Big Data Discovery, and some of the business intelligence tools we previously mentioned in this chapter can directly access Hadoop and are used for information discovery.

Finally, for certain problems, action must be taken in real time. Examples might include recommending products that could be of interest during a web site shopping visit or equipment that should be checked out for maintenance because its failure is predicted in the near future.

Web site activity data is typically analyzed using predictive analytics models. The models' results are periodically provided as updates (using batch feeds) to a real-time recommendation engine. The engine then recommends that the web site serve up specific web pages or notifications as guided by the models. As more analyses are made, the recommendations are fine-tuned and become more accurate. Often, reporting tools are used to monitor the results of these automated actions.

Other business problems, such as pending equipment failure, might require immediate action prior to any detailed data analysis since there is latency in the previously described learning process. Business rules engines or event processing engines can be pre-programmed to take specific action as a result of detected events.

These are often deployed in Internet of Things solutions in order to trigger an immediate action based on what sensors are detecting.

Later in this book, we will describe how to uncover the need for these various tools and solutions and then subsequently describe technical considerations as they become part of the information architecture design.

Evolving Data Management Strategies

As business workload demands changed and drove new technical requirements, relational databases evolved and introduced new capabilities intended to address those requirements. However, some found that a technology based on a concept of data's fitting neatly in rows and columns introduced too much overhead or was misaligned with the problems that needed to be solved. It is largely for those reasons that NoSQL databases and Hadoop engines began to appear around the turn of this century.

Coincidentally, they appeared at a time when the "open source" movement was gaining momentum and, in turn, helped to fuel that momentum. In the open source model, vendors and individuals have access to source code and these "committers" submit updates and utilities they are willing to share. Source code for NoSQL databases can be licensed from the Apache Software Foundation and GNU. Hadoop licenses can be obtained from Apache. As new features are incorporated into new releases of the open source code, the software vendors then determine what to include in their own distributions. Though the distributions can be downloaded for free, the vendors believe they can ultimately become profitable and successful companies by generating revenue through subscriptions (including support) and by offering services for a fee.

NoSQL Databases

The NoSQL database terminology dates to the late 1990s and was intended to describe a broad class of non-relational database engines designed to handle rapid updates and ingest the largest quantities of data while providing horizontal scalability. Such update and ingestion workloads had become a challenge for certain online applications (such as shopping carts on web sites) where fast update performance was critical despite a huge number of users of the application.

Early NoSQL databases did not support SQL, hence the name for this class of data management engines. Over time, SQL support of varying degrees has been added to many of the available NoSQL databases. Early NoSQL databases also did not provide traditional atomicity, consistency, isolation, and durability (ACID) properties provided by a relational database. This support was deemed as undesirable since it required too much overhead that got in the way of the performance needed. Today, many of the NoSQL databases are claiming to support at least some of the ACID properties. However, it is generally recognized that they are not intended to be used as a substitute for OLTP relational database engines or where joining many types of data across dimensions is required.

A variety of NoSQL database types have emerged. These include the following:

- Key Value Pairs: Databases that consist of keys and a value or set of values and that are often used for very lightweight transactions and where the number of values tied to a key grows over time.
- Column-based: Databases that are collections of one or more key value pairs, sometimes described as two-dimensional arrays, and are used to represent records so that queries of the data can return entire records.
- Document-based: Similar to column-based NoSQL databases, these databases are designed for document storage and feature deep nesting capabilities, enabling complex structures to be built such that documents can be stored within documents.
- Graph-based: Databases that use treelike structures with nodes and edges connected via relations.

Horizontal scalability of NoSQL databases is enabled using a technique called *sharding*. Sharding is simply the spreading of data across multiple independent servers or nodes in a cluster. Performance is dependent upon the power of the nodes but also upon how well the spreading of the data provides a distribution that also matches the performance capabilities of the individual servers. For example, if all of the most recent data is put on a single node and most of the activity is related to recent data, the application will not scale well. Many NoSQL database vendors have focused on automating the sharding process to provide better load balancing and make it easier to add or remove capacity in the cluster.

Though not as robust as relational databases in supporting high availability failover scenarios, NoSQL databases do enable replication of data to provide database availability in case of server or node failure. Copies of data are typically replicated across nodes that are different from the nodes where the primary data resides.

There are dozens of NoSQL database engines of the various types we have described. Some that you are more likely to encounter include Apache Cassandra, MongoDB, Amazon DynamoDB, Oracle NoSQL Database, IBM Cloudant, Couchbase, and MarkLogic. As the feature list for these databases can rapidly change, understanding the capabilities that are provided by the version being considered for deployment is very important. As an example, some added in-memory capabilities in only their more recent versions.

Hadoop's Evolution

Streaming data feeds from web sites increasingly caused headaches for companies and organizations seeking to analyze the effectiveness of their search engines at the beginning of this century. Such data streams include embedded identifiers and data of value, but also other miscellaneous characters. Figure 1-11 provides an illustration of typical data found in web logs. Clearly, this type of data does not fit well in a relational database. Doug Cutting was working on an approach to solve this problem by developing a new engine that he called *Nutch* as early as 2002. In 2003 and 2004, Google published two

important papers describing the Google File System (GFS) and MapReduce. The notion of a distributed file system was not new at the time, but Google's papers laid out a vision of how to solve the search problem.

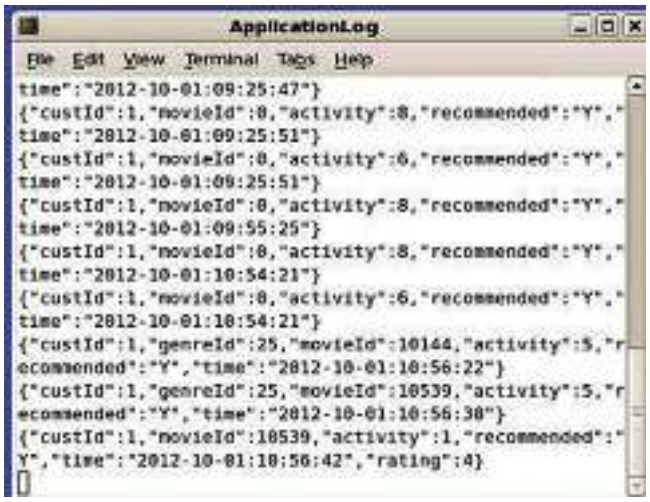

Figure 1-11. *Typical web log data stream*

Cutting understood the importance of the Google papers and made modifications to his own effort. MapReduce was able to map the data streams and reduce the data in the streams to data of value. GFS provided clues on how to scale the engine and such scalability was seen as particularly critical given the number of deployed web sites was exploding. In 2006, Cutting joined Yahoo! and renamed his storage and processing effort after the name of his son's toy elephant. Hadoop was born. That same year, Hadoop became an Apache Software Foundation project.

A distributed file system enables highly parallel workloads to occur across massive amounts of storage. MapReduce is co-located with the data providing the scalability needed. When this combination was discussed by early proponents, it was often described as solving a Big Data problem where data had huge volume, variety, and velocity. Over time, the Big Data terminology has taken on much broader meaning as vendors have positioned many different solutions to address many different though similar problems.

Today, Hadoop clusters are seen as the ideal solution for processing many types of workloads. Some of these clusters are now used to speed ETL processing by providing highly parallelized transformations between source systems and data warehouses. Other Hadoop clusters are used for predictive analytics workloads as analysts use tools such as R or SAS or leverage Hadoop's own machine learning and data mining programming library named Mahout. Data scientists sometimes write code (using Java, Python, Ruby on Rails, or other languages) and embed MapReduce or Mahout in that code to uncover patterns in the data. Increasingly, many also access data in the cluster through various SQL interfaces such as Hive, Impala, or other similar vendor offerings.

■ **Note** Newer data management solutions described in this book were invented to provide optimized solutions by addressing specific emerging workload needs. However, the perception about many of these data stores, promoted as open source, is that they are cheaper. This can lead to the application of these software distributions outside of their technical sweet spots in order to reduce cost of acquisition and support. As developers complain about the limited functionality compared to other engines that were seen as more costly, vendors that create the software distributions often add new features and capabilities in response. The unanswered question is whether many of the resource utilization and performance benefits of these distributions will disappear as they overlap more with other legacy data management solutions and with each other.

Hadoop Features and Tools

The Apache Software Foundation provides incubators for Hadoop features and tools and classifies these as development projects. As new releases occur, the results of these projects make their way, in varying degrees, into Hadoop distributions from vendors that include Cloudera, Hortonworks, IBM, MapR, and Pivotal. Apache Hadoop project status updates are posted on the apache.org web site.

If you are new to Hadoop, some definitions of key projects that the distributors and developers often talk about as Hadoop technologies and features could be useful. Some of the core data management features include the following:

- HDFS: The Hadoop Distributed File System.
- Parquet: A compressed columnar storage format for Hadoop.
- Sentry: A system that enables fine-grained, role-based authorization to data and metadata stored in Hadoop.
- Spark: An engine that enables Hadoop in-memory data processing.
- YARN: A framework used in scheduling and managing jobs and cluster resources.
- Zookeeper: A coordination service for distributed applications.

Important features for data transfer and collection in Hadoop include the following:

- Flume: A service for collecting and aggregating streaming data including log and event data in HDFS.
- Kafka: A publish-and-subscribe message broker platform designed to handle real-time data feeds.
- Sqoop: A tool used to transfer data between Hadoop and databases.

Programming tools, application programming interfaces (APIs), and utilities in Hadoop include the following:

- Hive: A SQL-like construct (HiveQL) for querying data in Hadoop.
- MapReduce: An early Hadoop programming paradigm that performs a "map" (filtering and sorting) and then a "reduce" (summary operation) for data that is distributed across nodes.
- Oozie: A workflow job scheduler used in managing Hadoop jobs.
- Pig: A data-flow language and parallel execution framework for data processing.
- Spark GraphX: An API that enables viewing of data as graphs and collections, transformations, and joins of graphs to resilient distributed data sets (RDDs), and creation of custom graph algorithms in Spark.
- Spark MLib: A machine learning library implemented in Spark.
- Spark SQL: An API that enables querying of RDDs in Spark in a structured (Hive) context.
- Spark Streaming: An API that enables creation of streaming processes in Spark.
- Solr: A full text indexing and search platform.

Creators of Hadoop-based applications sometimes seek the capabilities provided by a NoSQL database as part of their designs. HBase provides a NoSQL columnar database that is deployed on HDFS and enables random reads and writes. It is especially useful in handling sparsity of data problems. In addition to supporting ad hoc queries, HBase often is used for providing data summaries.

Layout of a Hadoop cluster on the underlying servers and storage requires the designation of name nodes, data nodes, and nodes that will provide the services enabling the features that we previously mentioned. Proper deployment of services across the cluster eliminates critical single points of failure that could bring the entire Hadoop cluster down. Data is normally triple replicated to assure that it is available in the event of node failures.

Some debate remains about when it is appropriate to include a Hadoop cluster as a component in the information architecture as opposed to suggesting a data warehouse deployed using a relational database. Table 1-1 attempts to highlight the strengths of each. As the capabilities in the data management engines are rapidly changing, you should revalidate these characteristics based on the most current information available when you consider deployment options for projects of your own.