O'REILLY®

The Enterprise Big Data Lake

DELIVERING ON THE PROMISE OF HADOOP
AND DATA SCIENCE IN THE ENTERPRISE

Early Release
RAW & UNEDITED

Alex Gorelik

The Enterprise Big Data Lake

Alex Gorelik

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

The Enterprise Big Data Lake
by Alex Gorelik

Printed in the United States of America.

Published by O'Reilly Media, Inc. , 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com .

Editor: Tim McGovern
Production Editor: FILL IN PRODUCTION EDITOR
Copyeditor: FILL IN COPYEDITOR
Proofreader: FILL IN PROOFREADER
Indexer: FILL IN INDEXER
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

January -4712: First Edition

Revision History for the First Edition
2016-06-24: First Early Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781491931486* for release details.

978-1-491-93148-6

[FILL IN]

Table of Contents

CHAPTER 1

Starting a Data Lake

As discussed in the previous chapter, the promise of Data Lake is to provide a place to store the data in the enterprise, so it will be available for analytics and data science, but what's the best way to get started? This chapter discusses various paths enterprises take to build the data lake. Apache Hadoop is an open source project that's quickly becoming synonymous with a Data Lake. We are going to have a dedicated chapter discussing Hadoop, but it is worth while to review what it is and some of its key advantages for supporting a Data Lake.

The What and Why of Hadoop

Hadoop is a massively parallel storage and execution platform. It automates many difficult aspects of building a highly scalable and available cluster. As the diagram below illustrates, Hadoop starts with a distributed file system HDFS, although some Hadoop distributions like MapR and IBM provide their own file system to replace HDFS. HDFS automatically replicates data on the cluster to achieve high parallelism and availability. For example, if Hadoop is configured to use its default replication factor of three, it stores each block on three different nodes. This way, when a job needs a block of data, the scheduler has a choice of three different nodes to decide which one is the best one to use based on what other jobs are running on it, what other data is located there and so forth. Furthermore, if one of the three nodes fails, the system dynamically reconfigures itself to create another replica of each block that used to be on that node while running current jobs on the remaining two nodes.

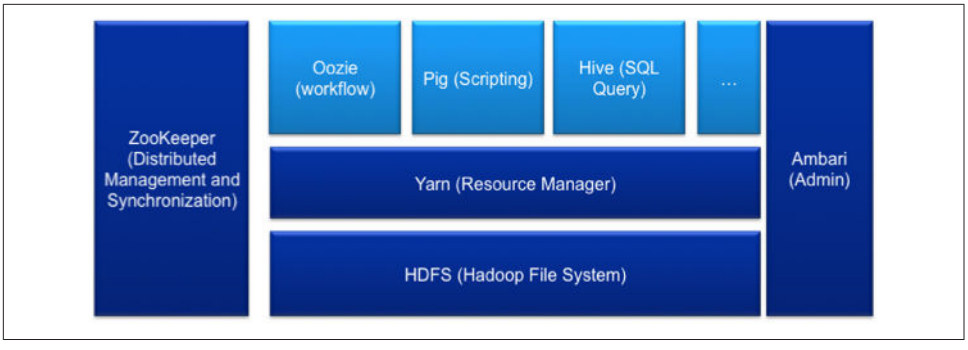


Figure 1-1.
A Hadoop-based data storage system.

MapReduce is programming model that has been implemented to run on top of Hadoop and to take advantage of Hadoop file system to create massively parallel applications. It allows developers to create two types of functions – mappers and reducers. Mappers work in parallel to process the data and stream the results to reducers that assemble the data for the final output. For example, a program that counts words in a file can have a mapper function that reads a block in a file, counts the number of works and outputs file name and the number of words it counted in that block. The reducers will then get a stream of word counts from the mappers and add the blocks for each file before outputting the final counts. An intermediate service called sort and shuffle makes sure that the word counts for the same file are routed to the same reducer. The beautiful thing about Hadoop is that the MapReduce job does not have to know or worry about the location of data, optimizing which functions run on which nodes or which nodes failed and are being recovered.

However, the main attraction of Hadoop is that it is a platform and an ecosystem of open source projects and proprietary products that solve a wide variety of use cases. The most prominent projects include: Hive – a SQL like interface to Hadoop files; Spark – an in-memory execution system; Yarn – a distributed resource manager; Oozie – a workflow system; and many, many others.

There are several key properties of Hadoop that make it so attractive as a long-term data storage and management platform:

1. Extreme scalability. At most enterprises, data only grows and often exponentially. This growth requires more and more compute power to process. Hadoop is designed to keep scaling by simply adding more nodes. It is used at some of the largest clusters in the world at places like Yahoo and Facebook. This ability to keep scaling is often referred to as "scaling out".

2. Cost-effectiveness. Hadoop is designed to work with off-the shelf hardware and run on top of Linux and use many free open source projects. This makes it very cost-effective.
3. Modularity. Traditional data management systems are monolithic. For example, a traditional relational database does not expose its data except through a relational query, so if there is a better or faster query engine, it cannot leverage existing data files. RDBMS also requires tight schema control (referred to as schema on write) – before you can add any data, you have to pre-define the structure of that data (called schema) and have to carefully change that structure if data changes. Hadoop, on the other hand, is designs from the ground up to be modular, so the same file can be accessed any application. For example, it can be accessed by Hive to perform a relational query or by a custom MapReduce job to do some heavy-duty analytics. This modularity makes Hadoop extremely attractive as a long-term platform for managing data, since new data management technologies will be able to use data stored on Hadoop through open interfaces.
4. Loose schema coupling or schema-on-read. Unlike a traditional relational database, Hadoop does not enforce any sort of schema when the data is written. This allows what's called "frictionless ingest" – data can be ingested without any checking or processing. Since we do not necessarily know how the data is going to be used, by using frictionless ingest, we can avoid paying the high price of processing for data we may not need and, potentially, not processing it correctly for the future applications. It is much better to leave it in the original or raw state and do the work as needed when the requirements and use case become solidified.

Now, if we are building a long-term storage and analytics system for our data, we would want it to be – cost effective, highly scalable and available, require minimal work to add data and be extensible to support future technology, applications and projects. As you can see from the brief discussion above, Hadoop fits the bill beautifully.

Since most data lakes today are built on a Hadoop cluster, one aspect that affects the adoption of a data lake Starting a data lake is closely related to the state of Hadoop adoption at the enterprise. At some enterprises, Big Data proponents are struggling to bring Hadoop in and are looking for projects to show quick ROI and turn that success into wide adoption. At other enterprises, Hadoop adoption is a fait accompli and a battle is raging between business and IT for control of Hadoop. Business sees Hadoop as a way to bypass IT and get what they need quickly and cost-effectively, while IT recognizes the nightmare scenario of Hadoop proliferation quickly turning into a unsupportable morass that eventually ends up on their lap to deal with and sees an enterprise data lake as a way to get ahead of the problem. For all parties involved, agreeing on a plan for building and adoption is essential to get to their goals as quickly—and painlessly—as possible.

In this chapter, we will cover some of the most popular scenarios for Data Lake adoption. For companies where Hadoop is being widely adopted by business, a data lake is often built by IT to try to prevent proliferation of "data puddles" – small independent clusters built with different technologies often by systems integrators who are no longer engaged in the project.

For companies trying to introduce Hadoop, there are several popular approaches:

- Some companies start with a data science project, show great ROI and then expand it to a full data lake
- Other companies start by off-loading some existing functions to Hadoop and then add more data and expand into a data lake
- Yet a third set of companies start with a new operational project on Hadoop and then decide to add analytics
- Finally, some companies build data lake as a central point of governance

Which one is right for you? That depends on what your company is, what your role is, and a number of other considerations that we will examine in this chapter.

Preventing Proliferation of Data Puddles

With all the excitement around Hadoop, there are many vendors and system integrators out there selling immediate value to business. These folks promise quick ROI often, with cloud-based solutions. For many business teams whose projects languish in IT work queues and who are tired of fighting for priority and attention, it seems like a dream come true. In weeks or months, they get the projects they have been demanding from IT for years.

Many of these projects get started and produce quick wins, causing other teams to undertake similar projects. Pretty soon, many business groups have their own "shadow IT" and their own little Hadoop clusters (sometimes called data puddles) on premises and in the cloud. These single-purpose clusters are usually small and purpose-built using whatever technology System Integrators (SIs) or local resources are familiar with and loaded with data that may or may not be rigorously sourced.

The unfortunate reality of open source technology is that it is still not stable enough, nor standard enough for this proliferation. Once the SIs move on and the first major technical challenge hits – jobs don't run, libraries need to be upgraded, technologies are no longer compatible, these data puddles end up thrown back to IT.

To prevent this scenario, many CIOs want to get ahead of the train and build a data lake, so when business teams decide that they need Hadoop, they have both the compute resources and the data for their projects already available in the data lake. By providing managed compute resources with pre-loaded data, yet giving users autonomy through self-service, an enterprise data lake gives business the best of both

worlds: support for the components that are difficult for them to maintain (Hadoop platform and data provisioning), and freedom from IT when it comes to the projects.

Leading with Data Science

Finding a high visibility data science project that affects the top line is another very attractive strategy. Data Warehouses that started as a strategic imperative promising to impact the business ended up supporting reporting and operational analytics and, while essential to running the business, are mostly perceived as necessary cost, rather than a strategic investment. Many data warehousing and analytics teams see data science as a way to visibly impact the business and to become strategically important again.

The most practical way to bring data science into an organization is to find a highly visible problem that

- Can be solved through machine learning or advanced analytics
- Is well defined and well understood
- Can show quick, measurable benefit
- Requires data that the team can easily procure

Once such a problem is identified, most organizations invest into a small Hadoop cluster – either on premises or in an Amazon cloud (depending on data sensitivity). They bring in Data Science consultants, run through the process and quickly produce results and show the value.

Most organizations perform two or three of these projects and then use the success to justify a Data Lake. This is sometimes referred to as "Xerox PARC" model. Xerox established PARC – or Palo Alto Research Center to research the office of the future in 1970. In 1971, Xerox PARC invented laser printer which became the main staple of Xerox business for years to come and while many other industry changing technologies were invested at PARC, none of these were successfully monetized by Xerox on the scale of laser printing. This analogy works because the results of data science are inherently unpredictable. For example, a long, complex project may produce a stable predictive model or the model may produce only marginal improvement (for example, if the model is right 60% of the time, it is only a 10% improvement over randomly choosing the outcome, which will be right 50% of the time).

This investing for the future approach sounds good and it is very tempting to build a large Hadoop cluster, load it up with data and declare victory. Unfortunately, I have spoken to dozens of companies where exactly such pattern played out: they had a few data science pilots that quickly produced amazing results. They used these pilots to secure multi-million dollar data lake budgets, built large clusters, loaded petabytes of data and are now struggling to get usage or show additional value.

The biggest mistake companies make is to go big and hope for the best. They build large data lakes, hire sizable data science teams and expect amazing insights to keep pouring in. If you choose to go the analytical route, consider the following recommendations that a number of IT and Data Science leaders have discussed with me:

- Have a pipeline of very promising data science projects that you will be able to execute as you are building up the data lake to keep showing value. Ideally, make sure that you can demonstrate one valuable insight a quarter for the duration of the data lake construction
- Broaden the data lake beyond the original data science use cases as soon as possible by moving other workloads onto the lake – from operational jobs like ETL to governance to simple BI and reporting
- Don't try to boil the ocean right away – keep building up the cluster and adding data sources as you keep showing more values
- Focus on getting departments, teams and projects onto the data lake

In summary, data science is a very attractive way to get to the data lake. It often affects the top line, creating the ROI through the value of the business insight, raising awareness of the value of data and the visibility of the data team. The key to building a successful data lake is to make sure that the team can continue producing such valuable insight until the data lake diversifies to uses cases and creates sustainable value for the wide range of teams and projects.

Off-loading existing functionality

One of the most compelling things about Hadoop is its cost, which can be up to 100 times lower than the cost of a relational data warehouse of similar performance and capacity. Since the size of a data warehouse only increases and IT budgets often include the cost of expansion, it is very attractive to off-load some processing from a data warehouse instead of growing the data warehouse. The advantage of this approach is that it does not require a business sponsor because the cost is coming out entirely out of IT budget and because the success is entirely up to IT – the off-loading should be transparent to the business users.

The most common processing to offload to Hadoop is the T or transformation part of ETL, which stands for Extract, Transform, and Load. ETL is the common way most enterprises move and transform bulk data around by Extracting it from source systems, Transforming it on ETL server and then Loading it into the target system. This work is done entirely in batch and is completely transparent to the business users. Teradata is the leading provider of large massively parallel data warehouses. For years, Teradata has been advocating ELT approach to loading data warehouse (Extract and Load into Teradata and then Transform using Teradata's powerful multi-node engines) since general ETL tools did not scale well to handle the volume of data that needed to be transformed. Hadoop, on the other hand, can handle the volume

with ease and very cost-effectively. Therefore, Teradata is now advocating doing the transformations in Hadoop and then loading data into Teradata database, to perform queries and analytics.

Another common practice is to move processing of non-tabular data to Hadoop. Many modern data sources from web logs to twitter feeds are not tabular. Instead of the fixed columns and rows, they have complex data structure and variable records. These types of data can be processed very efficiently on Hadoop in their native format, instead of needing to be converted to a relational format and uploaded into a data warehouse for processing using relational queries.

A third class of processing that's commonly moved to Hadoop is real-time or streaming processing. New technologies like Spark, which allows multi-node massively parallel processing of data in memory and Kafka – a memory queuing system as well as many other similar technologies, are making it very attractive to perform large scale in-memory processing of data on Hadoop for real-time analytics, CEP (complex event processing) and dashboards.

Finally, Hadoop can be used to scale up existing projects at a fraction of the cost. One company that I have spoken with has moved some complex fraud detection processing to Hadoop and for the same compute resource cost was able to run 10 times more data 10 times faster on Hadoop than on a relational database, creating orders of magnitude more accurate models and detection.

Once IT teams move such automated processing to Hadoop and accumulate large data sets, they come under pressure to make this data available to data scientists and analysts. To go from automated processing to a data lake, they usually have to go through the following steps:

- Add data that's not being processed by automated processing jobs to create a comprehensive data lake
- Provide data access for non-programmers including data visualization, reports, dashboards and SQL access to data. To facilitate adoption by analysts, provide a comprehensive, searchable catalog and create automated data governance policies around data access, sensitive data handling, data quality and data lifecycle management
- Insure that SLAs for automated jobs are not affected by the work that analysts are doing by setting up prioritized execution and resource governance schemes

New Operational Project

Instead of off-loading existing functionality to Hadoop, some companies use it to support a new operational project such as IOT (Internet Of Things – processing of machine data and logs from various devices) or social media customer analytics. For

example, a large IT manufacturer builds devices that send their logs to the factory on daily basis (this is called "call home logs"). The manufacturer used to process the logs and store only 2% of the data in a relational database used for predictive modeling. The models predicted when a device would fail, when it would need maintenance, and so forth. Every time the log format or content changed or the analysts needed another piece of data for their predictive models, developers would have to change the processing logic and analysts would have to wait months to gather enough data before they could run new analytics. With Hadoop, this company is able to store the entire log files at a fraction of the cost it used to spend to store only 2%. Since analysts can now access all the data they as far back as they like, they were able to deploy a number of new analytics for internal data quality initiatives as well as customer facing ones.

In many ways, the path of starting with a new operational project is similar to the offloading process for an existing project. The advantage of a new project is that it creates new visible value for the company. The drawback is that it requires additional budget. Moreover, a project failure even if it has nothing to do with Hadoop can taint enterprise's view of big data technology and negatively affect its adoption.

Across verticals, Hadoop is a great choice for marketing analytics, social media analytics and weblog analytics. In addition, here are some popular new projects that Hadoop is used for in different verticals:

- Financial Services: GRC (Governance, Risk and Compliance) – portfolio risk analysis, and a myriad of regulations including Basel 3, Know Your Customer, Anti-Money Laundering and many others; fraud detection; branch location optimization; and automated trading
- Healthcare: governance and compliance; medical research; patient care analytics; IOT for medical devices, wearable devices and remote healthcare
- Pharmaceuticals: research, process manufacturing optimization
- Manufacturing: collecting device information for IOT, quality and preventive maintenance
- Education: admissions; student success
- Retail: pricing
- Adtech: automated bidding, exchanges

There are many more examples and more projects are being deployed and discussed every day!

Central Point of Governance

With more and more government and industry regulations and stricter enforcement, governance is becoming a major focus for many enterprises. Governance aims at providing users with secure, managed access to data that complies with governmental

and corporate regulations. It generally includes management of sensitive and personal data, data quality, data lifecycle management, metadata management, and data lineage. A subsequent chapter on data governance will go into a lot more details on this topic. Since governance provides compliance to governmental and corporate regulations and these regulations apply to all systems in the enterprise, governance requires enterprise to implement and maintain consistent governance policies. Unfortunately, implementing and maintaining consistent governance policies across heterogeneous systems that use different technologies and are managed by different teams with different priorities presents a very formidable problem for most enterprises.

Imagine the challenges of trying to create a consistent data quality policy, trying to track lineage of data across various systems and integration technologies, or trying to find all sensitive data in all systems and treat it appropriately. Instead, enterprises are bringing all data into a Hadoop based data lake, so it can be treated using consistent governance policies and made available to the analysts and project teams in a controlled manner. All the compliance reporting, eDiscovery and other regulatory activities are then performed against the data lake.

Furthermore, Hadoop, as a file system, lends itself well to the idea of bimodal IT, an approach that recommends creating different zones with different degrees of governance. By creating and keeping separate zones for raw and for clean data, Hadoop supports various degrees of governance in one cluster.

Which Way is Right for Me?

Any one of these approaches can lead to a successful data lake. Which way should you go? It usually depends on your role, your budget and the allies you can recruit. Generally, it is easiest to start a data lake by using the budget that you control. However, regardless of where you start, for a data lake to take off and become sustainable, you will need to convince analysts throughout the enterprise to start using it for their projects. Therefore, you will first need a plan of how to convince the analysts to use the data lake.

IT

If you are an IT executive or Big Data champion, the following decision tree should help you formulate a data lake strategy:

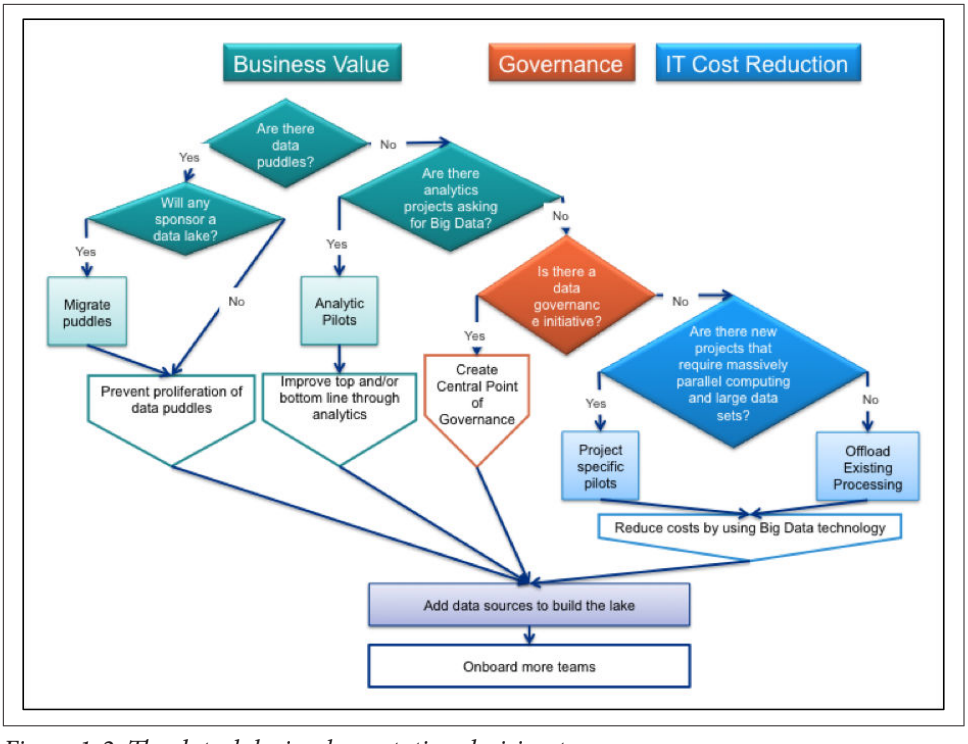


Figure 1-2. The data-lake implementation decision tree

1. Are there any data puddles (i.e., are business teams using Hadoop clusters on their own)?
   a. If yes, are there any projects that would agree to move to a centralized cluster?

i.If yes, use the cost of the project to justify a centralized cluster

ii.If not, justify building a data lake to avoid proliferation of data puddles. Use previous proliferations (e.g., data marts, reporting dbs) as examples.

1. If cannot get approval, wait for puddles to run into trouble – it won't take long

1. If there are no data puddles, are there groups that are asking for Big Data and/or Data Science? If not, can you sell them on sponsoring it?
   a. Make sure they can identify low risk, high visibility projects
   b. Try to line up more than one project per team and more than one team to maximize the chances of success
   c. Go down the data science/analytics route
2. If there are no groups ready to sponsor a big data project, is there a data governance initiative?
   a. If yes, try to propose and get approval for the single point of governance route

3. Otherwise, review the top projects and identify any that require massively parallel computing and large data sets and would be more cost-effective using Hadoop.
4. Finally, find existing workloads to off-load

Data Science or Business Analytics Team

If you are in charge of Data Science or Business Analytics, you will most likely go down the data science route to justify a data lake. While it is difficult to get the enterprise to commit significant resources to build a large data lake, it is usually feasible to create a data puddle – a small purpose-built cluster to showcase the value of data science or advanced analytics. Best practices include

- Finding a high visibility problem with well understood data sets and solving it using advanced analytics as a pilot
- Building a pipeline of other high visibility problems that can be solved with a high degree of certainty
- Using the success of the initial pilots to justify a larger data lake investment

Often, business teams find it easier to build a pilot cluster in the cloud and hire consultants to configure and deploy it. However, it is ideal to get IT involved early and let them co-sponsor the projects. By pooling the business value of data science projects and IT cost reductions from off-loading processing to Hadoop, it is often possible to justify a larger, more powerful cluster and have it adopted quicker.

Data Governance Professionals

Data Governance professionals sometimes regard big data and Hadoop as a far removed future problem. They feel that they first have to implement data governance policies for legacy systems before tackling new technologies. This approach, while not without merit, misses the opportunity of using Hadoop as a cost-effective platform to provide centralized governance and compliance for the enterprise. Instead of retrofitting legacy systems and convincing the teams responsible for those systems to commit their limited personnel resources to implement the governance policies and to use expensive compute resources to execute those policies, it is frequently much more straight forward and cost effective to have them simply ingest their data into Hadoop so a standard set of tools can implement consistent governance policies:

- data can be profiled and processed by a standard set of data quality technologies with uniform data quality rules
- sensitive data can be detected and treated by a standard set of data security tools
- retention and eDiscovery functionality can be implemented in a uniform way across the systems

- compliance reports can be developed against a single unified system

CHAPTER 2
Governing the Data Lake

During the first generation of data warehousing, analysts were usually non-technical domain experts. They were not comfortable with data manipulation and often not comfortable with computers in general. Therefore, traditional data warehousing teams consisted of multiple technologists with various functions

- Database and system administrators to manage and optimize the physical and software systems
- Data architects and data modelers to model the business systems and translate those logical models into physical database schemas (a set of tables and fields and their relationships).
- ETL developers to load data into the data warehouse
- Data analysts to create and manage data quality rules and maintain metadata catalogs and repositories
- Data Stewards to govern data and create and manage compliance rules
- BI architects to design reporting metadata layer
- Report developers to create OLAP cubes, reports and dashboards

With all these people, no wonder any change took months to incorporate.

Current generations of analysts grew up with computers and are much more comfortable with data and data manipulation. These analysts often perceive IT data warehousing team as roadblocks instead of essential assistants. They want to be able to find data and do their own analysis. In fact, oftentimes the one size fits all of the data warehouse and strict data quality rules conflicts with what they are trying to achieve.

For example, a data quality rule may specify that salary should be a number between 0 and 1b or NULL if unknown. A data warehouse ETL job that implements this rule would then "fix" any incoming data to convert a string in the salary field into a NULL to indicate that the salary is not a number and it is not known. A data scientist, on the

other hand, may find strings immensely useful and would want to treat it differently depending on what the string is. For example, if salary field is set to "full time student", data scientist may choose to make the value 0; if the field says "unemployment", she may choose to set it to an average unemployment benefit for the state; if it says retired, she may choose it to an average retirement benefit for the state; and if it says self-employed, to an average income.

The Era of Self-Service

To accommodate this new breed of analysts, enterprises are turning to self-service approaches. The first step in establishing self-service data culture is to create a data lake, so data can be collected and managed in one place. Moving data into a data lake has several important advantages:

- It avoids the need to provide and manage access to many different systems. This may not be as big of a problem for the organizations that have implemented a Single Sign On solution, but still requires quite a bit of work to decide what data in which systems is appropriate for which analysts to see
- The data lake also isolates analysts from professionally maintained environments like the enterprise data warehouse, so their queries and programs don't affect production jobs
- Data in data lakes can be de-identified or encrypted to make it safe for the analysts to see and use
- Finally, data lake can provide a single point of audit and tracking for all analyst access

Once the data lake is established, the analysts need a way to find and understand data. This is a formidable task when you consider the wide variety of data in most enterprises. A large retailer discussed having over 30,000 data sources feeding their data lake and how each source may provide hundreds and even thousands of tables.

Organizing a Data Lake

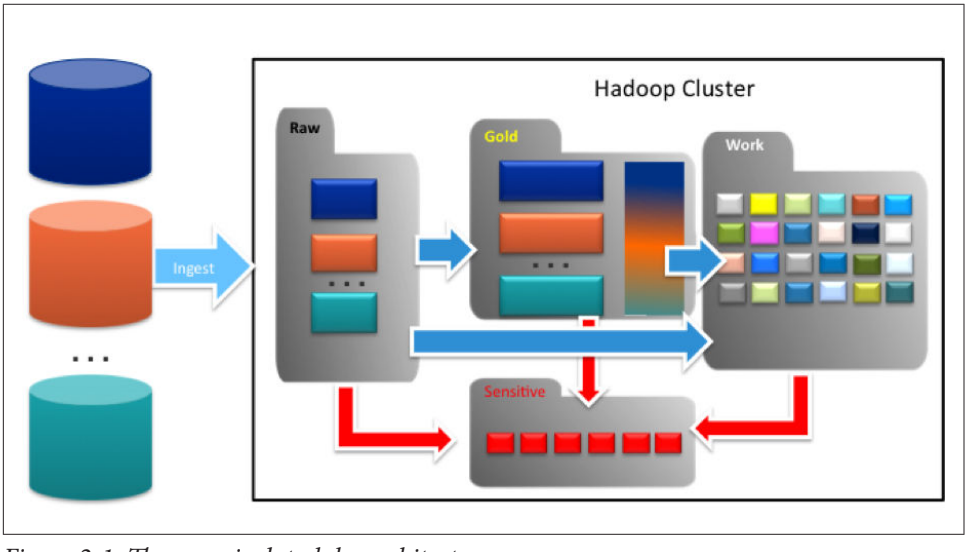


Figure 2-1. The generic data lake architecture.

Some assistance for finding data may be provided by creating and carefully maintaining directory structure and naming conventions. The illustration above reflects a pretty common Hadoop cluster architecture with Raw Landing area where the files are loaded from external systems. Gold area where cleansed, curated and aggregated files are kept, Work area where users work on projects and Sensitive data zone where sensitive data is kept in encrypted volumes.

Raw or Landing Zone

The Raw or Landing Zone is used to house raw ingested data. A naming convention is usually created to identify where data comes from:

All ingested raw data is usually kept in a single folder (e.g., /Landing), in that folder, there is usually a folder per source system (e.g., /Landing/EDW or /Landing/Twitter), and a folder per table or some other grouping (e.g., /Landing/EDW/Customer_dimension or /Landing/Twitter/Mybrand1)

If the table is reflected periodically, there may be a partition for each time new data is loaded (e.g., /Landing/EDW/Customer_dimension/20140101.csv or /Landing/Twitter/Mybrand1/20140101.json). Usually, to avoid very large folders, a more elaborate folder tree may be created with a folder for each year and month for example and just that month's partitions as files (e.g., /Landing/EDW/Customer_dimension/2014/01/20140101.csv or /Landing/Twitter/Mybrand1/2014/01/20140101.json).

Landing zone data is usually raw and loaded directly from source systems. Data Scientists, who need raw data, can go there directly. Analysts who need cleaner data may use data from Gold zone that frequently mirrors Landing area, but contains cleansed and summarized versions of the raw data.

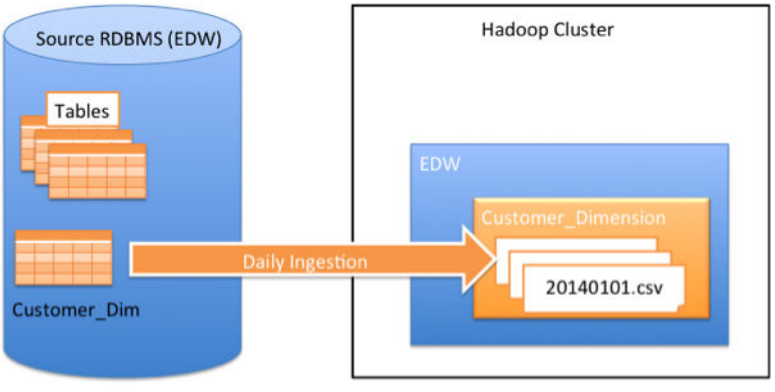


Figure 2-2. governing_a_data_lake_-_landing_zone.png

Gold Zone

This zone is sometimes also called Prod to indicate that data is production ready or Cleansed to indicate that data has been run though data quality tools and/or curation process to clean up (or cleanse) data quality problems. Like the Landing Zone, it often has a folder per source system (e.g., /Gold/EDW or / Gold /Twitter) and a folder per table or some other grouping (e.g., / Gold /EDW/Customer_dimension or / Gold /Twitter/Mybrand1)

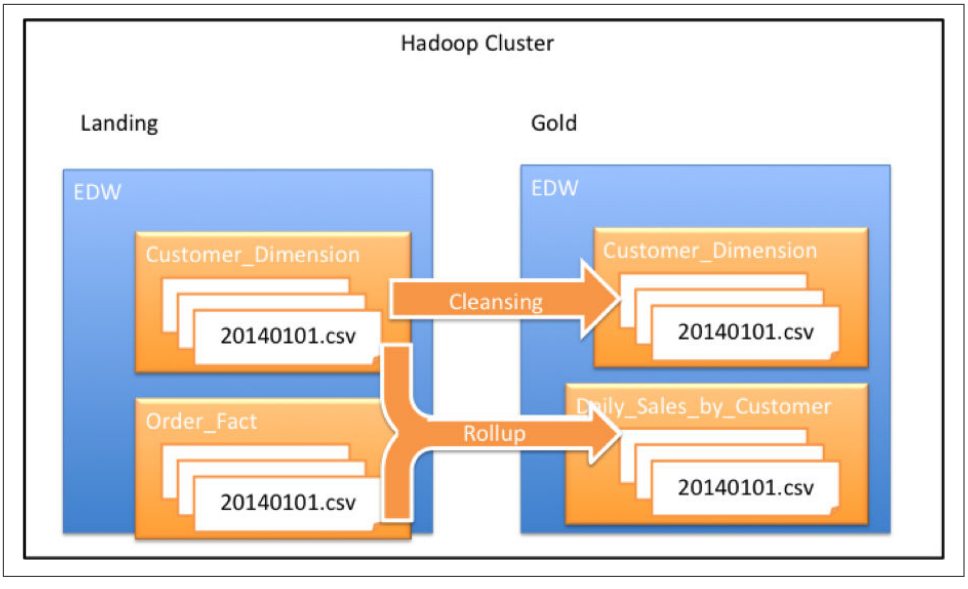


Figure 2-3.

If there are summarized or derived files, these go here as well (e.g., / Gold /EDW/ Daily_Sales_By_Customer or / Gold /Twitter/BrandTwitterSummary)

The folder is then broken into dates similar to the Landing zone. When the files are landed in the Landing zone, automated programs cleanse and summarize these files and place results in Gold zone.

For most analysts using the Data Lake, Gold Zone is the main source of data. To make it more accessible, it is very common to create a SQL view of each file in the Gold Zone using Hive, Impala, Drill or one of a dozen other systems.

Work Zone

Most of the analyst work happens in the Work zone, also known as Dev or Project zone. These zones are usually structured to reflect the organizational structure of the analyst organizations.

Projects folder usually has a folder per project (e.g., Projects/Customer_Churn) and inside this folder there are folder to reflect the details of the project

In addition to Projects, clusters have a /Users folder with a folder for each user (e.g., / Users/fjones112) where each user does their private work.

Encrypted Zone

An Encrypted Zone is sometimes created to keep files containing sensitive data. The Encrypted Zone can contain explicitly encrypted data or it can utilize transparent encryption provided by Gazzang (recently acquired by Cloudera and included as part of Cloudera Navigator) and similar tools. There are several best practice approaches to structuring Encrypted Zone:

- Have an encrypted copy of a file in encrypted zone and a clear copy that's missing sensitive fields in Gold Zone. Access to encrypted files is only provided explicitly and temporarily on as needed basis.
- Have an encrypted copy of a file and a de-identified copy. De-identification is the process of replacing real sensitive data with similar made up data in order to retain properties of the original data. For example, a feminine Hispanic first name may be replaced with a different feminine Hispanic first name, because Data Scientists sometimes derive missing gender and ethnicity information from names; or an address may be replaced with a random valid address within a certain distance of the real address in order to facilitate geographical analysis. The difficult part of de-identification is maintaining consistency. The same value usually has to be replaced with the same identical value in all files, so it can be joined as part of the processing (e.g., so the same customer can be identified in multiple files). This makes it complex – the system has to maintain the mappings of real values to randomly generated values; fragile – a single letter misspelling which may be easily handled by identity resolution systems can cause a de-identification system to generate two completely different values; and vulnerable – if hackers get access to de-identification software, they can run a list of common names through it and get a table of value mappings using for all the files.
- Sometimes, for certain types of files, it is very difficult to identify sensitive data. For example, a medical health record EMR is an XML file that may contain up to 60,000 elements. It is very difficult to review all the elements to find sensitive ones and to regenerate the file taking out or de-identifying sensitive values. In such cases, companies find it easier to just keep encrypted values around.

Data Lake users and zones

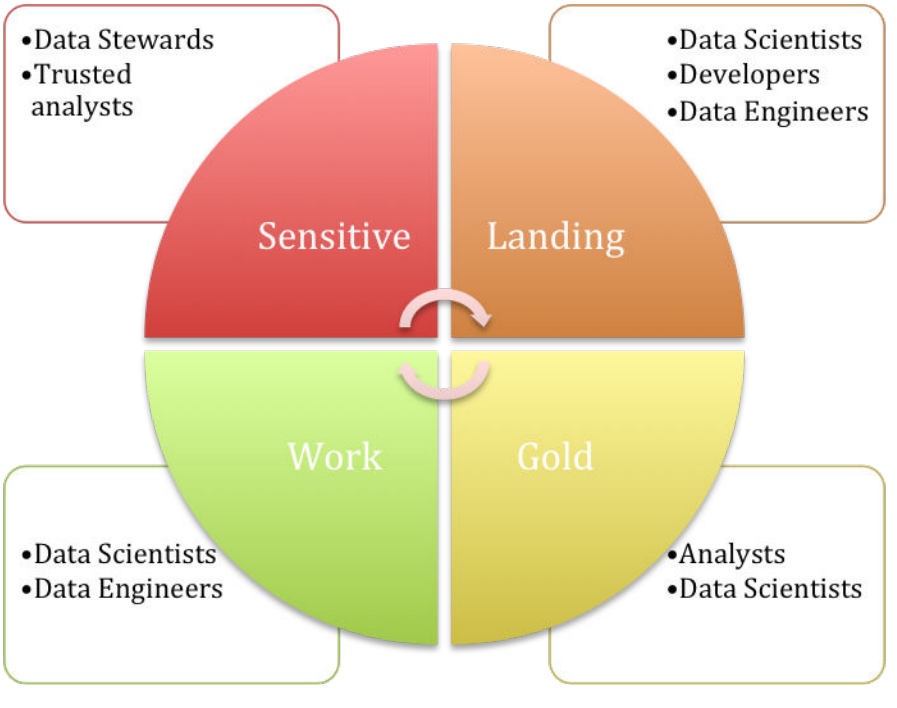
Different Data Lake Users usually use dedicated zones.

- Landing Zone – usually only highly technical developers, Data Engineers and Data Scientists get access to the Landing Zone. In general, users of the Landing Zone must have a very compelling reason to perform their own data treatment and summarization; and have the ability to write code.
- Gold Zone – this is usually the most popular zone. Most non-developers are confined to this zone. Because it usually provides a SQL interface to data and data is

usually cleansed, it is a starting point for most reporting and BI analysis. Even analysts who are not comfortable with SQL can usually use standard BI tools against Hive or other SQL on Hadoop tables. Even developers and data scientists usually prefer to use cleansed data to avoid extra work, unless there is a compelling reason not to. Gold Zone is often maintained by IT and ideally, well documented, either through directory structures and naming conventions or through HCatalog – the data dictionary developed around Hive Metastore that is being used by more and more other projects.

- Work Zone – this area is usually the domain of developers, data scientists and data engineers. Either individual folders or project folders contain both intermediate and final results of data. This is usually the least-documented part of the Data Lake. Unfortunately, it is frequently the largest since Data Science projects are explorative in nature and frequently require massive experiments. A typical data science project may create hundreds of experimental files before coming up with a good model or abandoning the approach altogether.

- Encrypted Zone – this zone contains sensitive data. Usually, only authorized people can get access to data in Encrypted Zone.



One Cluster or Many?

One question that many companies struggle with is whether to have one Hadoop cluster or break data into several. There are important trade offs to both approaches. There are several arguments for having multiple clusters:

- Security and access – original Hadoop distributions did not have sophisticated authorization mechanisms. One way to prevent unauthorized access in the past was to physically separate data sets into different clusters. With the advances in Hadoop authorization and encryption schemes, these concerns can now be addressed on a single cluster using open source projects like Apache Ranger (promoted by HortonWorks) or Apache Sentry (promoted by Cloudera).
- SLAs for production jobs – another concern has to do with making sure the production jobs can complete according to their SLAs. If a lot of operational processing or production ETL is performed on Hadoop, the administrators will have to set up elaborate job management schemes to make sure that production jobs are executed in priority order and are not impeded by non-production jobs. Often times, it is simpler to just isolate production job environment into a separate cluster and let the users do what they want in sandbox or dev environment. The main decision point seems to be around how much production work really is there. Imagine that we have a 100 node cluster. If production jobs scale linearly and take 2 hours on this cluster, we have two choices. Either build another 100 node cluster for them that will be idle 22 hours a day, or build a smaller, say 10 node cluster for them and have them run 20 hours instead of 2. On the other hand, if production jobs take 80% of compute resources on the 100 node cluster, then creating a separate 90 node cluster and allocating 10 nodes to experimentation will create a fully utilized isolated production environment. Although the experiments will take much longer to run on the smaller 10 node clusters, the analysts and developers will not have to wait for production jobs to finish to run their experiments, so their productivity will probably not be affected all that much.
- Organizational Control – often organizations want to control their compute resources and data. However, unless there are lots of automated production jobs running, the compute usage of Hadoop for exploratory analytics is so uneven, that pooling resources to build a larger shared cluster usually makes more sense to take advantage of scale out processing on a single large cluster with much less investment in both Hadoop infrastructure and Hadoop management and administration.

Building a Catalog

While the directory structure and naming convention help analysts navigate the cluster, they are not sufficient:

- There is no search - analysts have to browse to the right directory, which works when the analyst knows what she wants, but is not practical when she is just exploring thousands of sources/folders
- There is a very useful Hadoop utility called Hue that allows the user to look at a file. Unfortunately, it only shows a small initial segment of the file and for most large files understanding what's inside from a small segment may be difficult. Below is a screenshot of a twitter feed in Hue
- In order to decide whether a file is good for their project, the analyst needs to understand what's inside the file – e.g., is there any NY data; how many tweets are there? What's the order amounts? If the analyst is looking for customer demographics such as age, just because she can preview a few lines of a file and see age there, it doesn't tell her how many customers in the file actually have age.
- The analyst also needs to be able to tell where the file came from. Not all data can be trusted. Some of it comes from failed data science experiments, some from systems that are notoriously inconsistent and some from well curated and trusted systems. It is critical to know where data originated. It is just as critical to understand what was done to data. Is the file in the Landing Zone, Cleansed Zone or work zone? Depending on the analyst's needs, either raw or cleansed data may suffice or, if it is in someone's work folder, the analyst may have to carefully study the file description or talk to the file or project owner. Frequently, a lot of attributes may have been curated and treated in the way the analyst needs, while other attributes will need different treatment and need to be obtained from the raw files.

To address these shortcomings, companies need a detailed catalog or inventory. Given the shear number of files in the lake (and 30 years of history of people ignoring or circumventing processes and procedures designed to document and track data), the catalog has to be automated as much as possible and extremely easy and useful to analysts to take notes and jot down annotations.

Several vendors are providing data catalog capabilities including Teradata Loom, Waterline Data Inventory, Informatica Governed Data Lake, Tamr Data Catalog and Oracle Big Data Catalog as well as Cloudera Navigator and Apache Atlas developed by Hortonworks and its partners. When choosing a vendor, several important capabilities should be considered:

- Native Hadoop processing.
- Automated data discovery and classification.
- Integration with enterprise repositories.
- User friendliness.

Native Hadoop Processing

Because Hadoop Data Lake is going to be the largest data system in the enterprise, it requires the power of Hadoop to process and catalog data in the lake. The whole point of Hadoop is not just a cost-effective place to store data – we have had that for years, but a very cost-effective place to process data. Not leveraging Hadoop's processing capabilities to catalog Hadoop data will simply not scale. While Teradata Loom and Waterline Data Inventory were designed natively on Hadoop, other tools in the list were designed to work with Relational Databases and either require data to be loaded into a relational DBMS like Tamr Data Catalog or have proprietary engines that run outside Hadoop.

Automated Data Discovery and Classification

The sheer scope and complexity of data in the data lake makes it impossible for humans to manually classify or tag all data with business metadata. Therefore, an automated approach is required to complete this classification. While all the tools allow the analysts to tag data in some fashion, some, like Waterline Data Inventory provide an automated discovery engine that learns from analyst tagging and automatically classifies other data sets.

Integration with enterprise repositories

Hadoop Data Lake is only part of enterprise data eco-system and as such has to integrate with the rest of the governance infrastructure. Many tools provide enterprise data repository. Many enterprises have extensive metadata investments already that need to be leveraged. While Hadoop may be the most scalable and cost effective platform to process data both inside and outside Hadoop, Hadoop-only solutions are limiting and will not address enterprise requirements.

User friendliness

A lot of metadata solutions are designed for IT and governance specialists. To be widely adapted, the inventory solution must be intuitive and usable by non-technical analysts

Tools Comparison

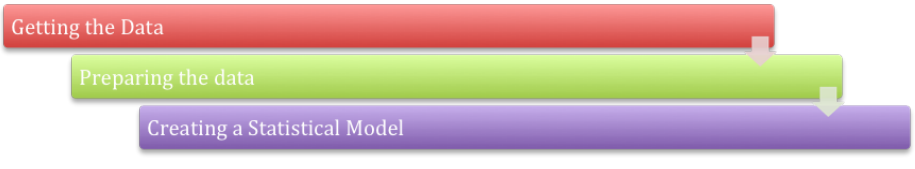
The following table lists all announced data-cataloguing products and evaluates their capabilities. Since the tools from Oracle and Informatica and Apache Atlas have not been released at the time of writing, not all information about these tools is known.

	Hadoop Support	Tagging	Enterprise Metadata Repository	Business Analyst Focused UI
Waterline Data Inventory	Native	Automated	Y	Y
Teradata Loom	Native	Manual	Y	
Cloudera Navigator	Native	Manual		
Informatica Governed Data Lake*	HDFS and Hive	Manual	Y	Y
Tamr Catalog*	Hive only	Manual	Y	TBD
Apache Atlas*	Native	Manual		TBD
Oracle Data Catalog*	TBD	TBD	Y	TBD

Products marked with * are not yet commercially available at the time of writing.

Finding Data and Getting It into Shape for Analytics

Data comes in all shapes and sizes and very different degrees of quality. Depending on the project, data needs to be transformed into the right shape. A typical advanced analytics projects will consist of the following steps:



If the model is unstable and does not produce expected results, the process may have to be repeated: additional or different data may need to be found and/or data may need to be treated differently. In practice, most of the time this process is iterative and sometimes involves quite a bit of experimentation and trial and error and hundreds of iterations.

We will go through each step in the process:

- Provisioning Data
  - — Finding data
  - — Understanding data

  - Accessing data
- Preparing Data
  - Combining data sets
  - Deciding what data to keep
  - Cleaning or treating data appropriately
  - Transforming data into the right shape
- Creating a Statistical Model
  - Training a Model
  - Testing a Model

Even when analysts are not doing advanced analytics, but creating reports or visualizations, data still needs to be provisioned and prepared.

Finding data

To find data, the analyst has to be able to search for data in the data lake and beyond and to determine whether a given data set is the best data set. Data Inventory discussed in the previous section is a necessary technology to facilitate that. Ideally, the search should be possible using all of the above: business terms, technical metadata and data contents.

Business metadata is usually how an analyst thinks of data. She may not even know all the possible technical names for it. For example, a field containing customer id may be called cust_id, id, cid, clientid, account number, kunnr, and any number of other synonyms and abbreviations. Sometimes, there is a temptation to use content search to find data – after all, it works for Google. Content search works very well for distinct values such as names or long identifiers such as credit card number. Unfortunately, majority of values in a data lake are ids, counters, quantifies and codes that are often represented by small numbers and short codes. These small numbers are pretty anonymous - in other words, do not really contain enough information about what data means. For example, 59 may be a customer id, a customer age, a year of birth, quantity of items in an order, number of employees in a department, and hundreds of other unrelated items. And because we are looking for data sets not individual values, the search for 59 may return hundreds, thousands or even millions of files without any real way to telling what "59" means. Furthermore, because most work in the data lake involves statistical analysis – either data science and advanced analytics or visualization and BI, the analysts are usually not looking for a specific customer using their customer id, but rather for data sets containing customer ids. Therefore, it is paramount to allow them to search the metadata in business terms that they are used to.

Understanding data

Once a potentially relevant data set is found, the analysts need to understand it. This usually involves the gathering information about the data set:

- What is the format of the data set – is it in JSON, Avro, Delimited File, etc.
- What is the schema of the data sets – what are the fields. Some data set formats like JSON and Avro are self-describing and have field names, while others like logs or delimited files may not even have a header row. Without understanding the schema, it is very difficult for the analyst to understand what's in the data set

For each field, the analysts usually need to look at the profile – statistical analysis of the values in the field. Generally profile includes one or more of the following

- Cardinality or number of unique values (e.g., how many unique customer ids are in this data set)
- Selectivity – s the measure of uniqueness of the field. Selectivity of 1 or 100% means the field is unique, while low selectivity denotes that the field contains many duplicate values. Selectivity is calculated as Cardinality divided by number of rows
- Number of NULLs or missing values (e.g., how many customers are missing age)
- Density – how full is the field. It is usually calculated as number of non-null values/number of rows
- Min and Max values
- For numeric fields, mean and standard deviation are frequently useful

If addition to the field profile, analysts usually need to examine the most frequent values and their counts. Sometimes, examining data and its distribution is the only way to tell what's inside a field. In addition to looking at the most frequent values, data scientists frequently need to understand the distribution of data and the easiest way to do it is to plot it and see the curve.

While profiling information is very intuitive for tabular data – the stats are for each column aggregated across all the rows, it gets trickier for hierarchical data such as JSON or XML files.

While the format may be different, conceptually JSON files represent the same data as tabular files. For example, an order can be represented as a set of tables in a relational database or a set of tabular files related to each other by what is called primary key-foreign key relationships. In the illustration below, four tables are used to store information about Orders, Customers and Products. Primary key-foreign key relationships are captured with lines between primary key fields and foreign key fields with one to many relationships illustrated with 1:N, meaning that for each CustomerID in Customers table, there may be many orders with the same CustomerID in Orders table.

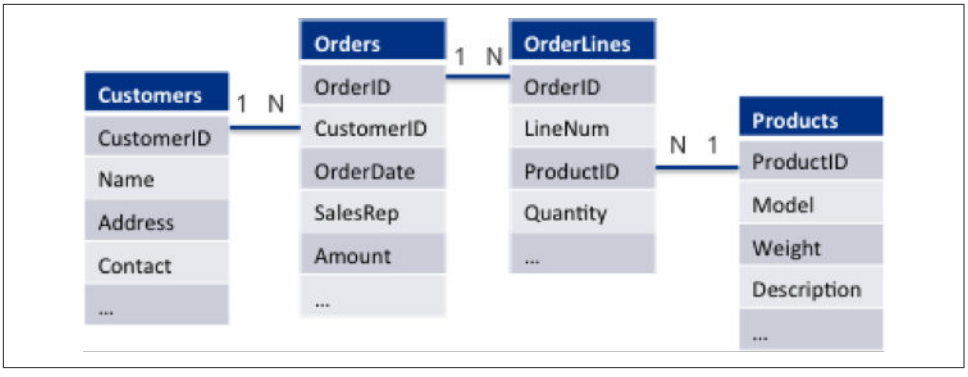


Figure 2-4.

The same information may be presented as a single JSON file where the hierarchy represents the relationships expressed by primary and foreign keys in relational systems. Instead of four different tables, a single JSON file will capture all the attributes and relationships. The following sample JSON file snippet captures all the information for an order. Please note that there are no CustomerIDs and ProductIDs required to related Orders, Customers and Products. Customer information is embedded in Orders record and Product information is embedded in OrderLine and so forth.

{“Order”

{

“OrderID” : “123R1”,
“Customer” {

“Name” : “Acme Foods”, “Address” : “20 Main St, Booville, MD”, Contact : “Zeke Gan”, …

}

“OrderLine” {

“LineNumber” : “1”,

“Product” {“Model” : “XR1900E”, “Weight” : “20”, “Description” : “XR1900E is the latest …”, …

}

“Quantity” : 1,

…

}

}

Since the information in both cases is really the same, a simple process called shredding is often used to extract fields from hierarchical files. For example, customer name may be extracted using its full hierarchical name Order.Customer.Name. This name is called XPATH and shredding basically creates a unique field for each unique XPATH in a hierarchical file. Most profiling tools such as the ones from Informatica and IBM, require hierarchical files to be explicitly shredded or converted to a tabular format before they can be profiled, while more modern profiling tools from Trifacta or Waterline Data shred hierarchical data automatically.

In addition to understanding the contents of a file, it is important to understand where data came from and what was done to it. This information is called Lineage or Provenance. Lineage information is automatically captured by most ETL (Extract, Transform and Load) tools like Informatica, IBM Infosphere, and Talend, as these tools move and transform data. However, a lot of transformation and movement is done using ftp, scripts written in Pig or Python and open source Hadoop tools such as SQOOP and Flume. These tools do not capture or expose the lineage of data. Since lineage is only useful if you can trace it all the way back to the source, these lineage gaps are critical to fill. Some tools try to fill this missing lineage either by scraping system logs (Cloudera Navigator), instrumenting these open source systems to report its lineage (Apache Atlas), requiring users to manually provide lineage for every job they write (Apache Falcon) or by examining the content of the files and trying to deduce the lineage (IBM Infosphere Discovery and Waterline Data).

However, even if lineage is available, it is often expressed in the language of the tool that generated data and not all analysts are versant in all tools. Frequently, multiple tools are used to generate a data set and it is extremely difficult to trace through Pig scripts, Python and Java programs, ETL (Extract, Transform and Load) tools and file systems commands to figure out what happened to the file. It is much easier for both the analysts creating the files as well as analysts trying to understand the files, if the files are annotated with some notes about what was done. Today, most analysts take notes in their notebooks, making it impossible for anyone else and, frequently, themselves to figure out what happened. Several approaches are often tried to address this: SharePoint or Wikis are set up to capture the notes in spreadsheets. In reality, what we find is that the landing zone that's maintained by IT is well documented in such Wikis, while the work zone, where you need the notes most, is not. Another approach is use notes and tags in a Data Inventory system described in the previous section. This allows the analysts to not only read the notes once they found the file, but to search for the files using the information contained in the notes and tags.

Providing access to data

Once analysts find the data sets that they need, they need to obtain access to those data sets. The easiest access model is to provide all analysts access to all data. Unfortunately, this is only possible if data is not subject to government regulations (e.g., not Personally Identifiable Information or Credit Card Information); is not copyrighted with restricted access (e.g., purchased or obtained from external sources for very specific or limited use), and is not considered critical to the company. Every company usually has data that it considers critical – everything from IP information to customer lists to engineering designs and financial information. Therefore, except for limited set of projects mostly dealing with public and research data and not sensitive internal data, enterprise cannot give full access to all data in data lakes to everyone.

Authorization

Authorization is the common way of managing data access. It involves explicitly assigning permissions to perform specific actions such as read or update on specific data assets such as files and tables to specific analysts. To streamline this process, security analysts usually create roles (collections of permissions) and assign those roles to groups of analysts.

Most legacy systems provide their own internal authorization mechanism. Since more and more companies are opting for more and more applications – often in the cloud, instead of using a single integrated application from a single vendor, single sign on systems became very popular. With single sign on, users log in once and their credentials are supported by all the applications and systems.

Unfortunately, there are several notable challenges with this approach:

- It is very difficult to predict what data the analyst will need for which project
- Unless the analyst has access to data, she cannot tell whether that's data that she needs
- There is high cost of maintaining authorizations
  - Whenever a new employee is hired, the security admins need to provide appropriate authorizations
  - As an employee changes roles or projects, the security admins need to provide new privileges and revoke old privileges
  - When a new data set shows up, the security admins need to figure out all the users who may need access to this data set
  - When an analyst needs a data set that contains sensitive data, a version of that data set needs to be created that either removes or de-identifies sensitive data

There are several approaches enterprises are taking to address these challenges:

- De-identifying sensitive data by removing, encrypting, or replacing it with generated random data; and granting access to these de-identified data sets to everyone
- Implementing self-service access management by creating a metadata only catalog that allows the analysts to find all available data sets and then request access to the relevant ones from data set owners or security admins

De-identifying Sensitive Data

Let's examine the problem of de-identifying sensitive data. The first challenge is that most enterprises don't know where all sensitive data is. This is especially true of Data Lakes where data is supposed to be loaded with "frictionless ingest". In other words – raw data is dumped into the data lake without any processing. This makes it very fast to load putting minimum stress on source systems, but makes it very difficult to figure out what sort of data it is and whether it is sensitive both in traditional sense and company-specific sense. Most profiling tools have built in rules that let enterprises scan the data lake and look for sensitive data. Some are fully automated like DataGuise and Waterline Data, while others like Informatica, Talend and IBM InfoSphere, require programmers to create ETL jobs for each data set.

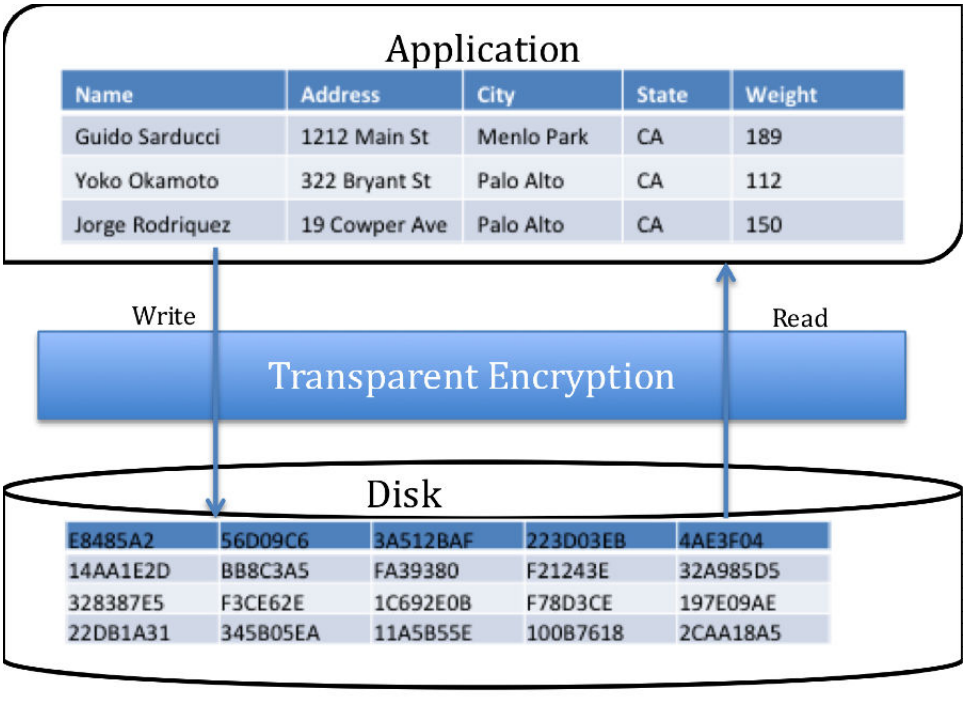
Once all sensitive data has been identified, encrypting sensitive data seems like a simple and straight forward way of making sure analysts can use whatever they want. There are several levels of encryption:

- Transparent
- Explicit
- De-identification

Let's say we have a data set (for simplicity, let's make it tabular) that contains some patient information at a healthcare provider

Name	Address	City	State	Weight
Guido Sarducci	1212 Main St	Menlo Park	CA	189
Yoko Okamoto	322 Bryant St	Palo Alto	CA	112
Jorge Rodriquez	19 Cowper Ave	Palo Alto	CA	150

Transparent encryption (like the one provided by Cloudera Navigator) automatically encrypts data on disk when it is written and automatically decrypts it when it is read. This is done to prevent someone from accessing or copying raw disk volumes and reading them one byte at a time to recreate the data file thereby avoiding all access controls.



However, transparent encryption does not prevent analysts with read privileges on the file from seeing sensitive data. For that enterprises usually deploy explicit encryption and encrypt each value separately. While it may seem straight forward with many open source encryption functions available and many tools from DataGuise, Informatica, Blue Talon, Vormetric and many others providing encryption, it makes the sensitive data completely unusable as illustrated by the diagram below.

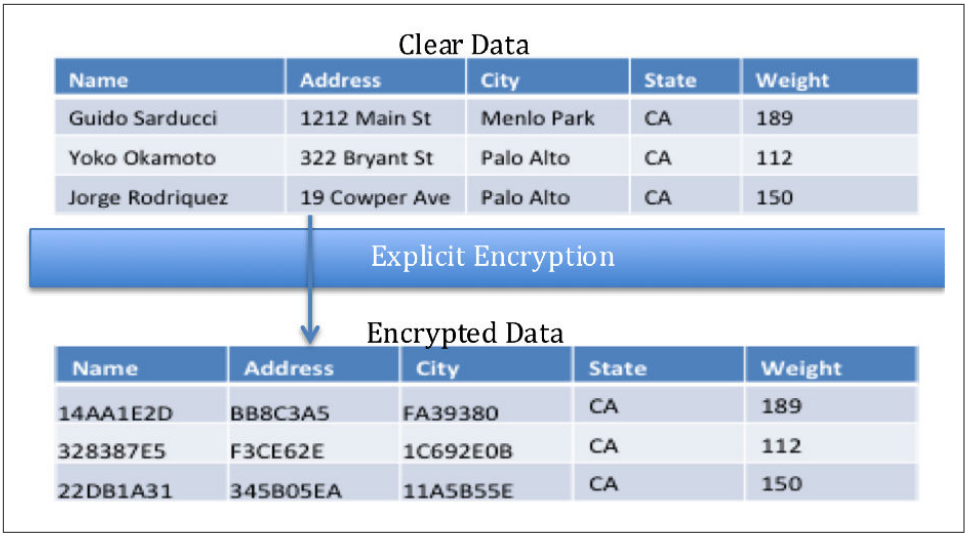


Figure 2-5.

Unfortunately, that creates real problems for data scientists trying to use the data sets. One Data Scientist that I interviewed for this book told me how at his company all data in the data lake is encrypted unless he can prove that the attributes are not sensitive. The data scientist did not care for that approach. As he rhetorically put it - *"How am I expected to prove that an attribute is not sensitive if I can't find or view it?"*

Even if only truly sensitive attributes are encrypted, often these attributes encode very important information that data scientists use to derive variables for their models. For example, in a data set where there is person's name, but the gender is missing, it is often possible to tell gender from the First Name. Similarly, it is sometimes possible to figure out ethnicity from the first and last names. If the first name is encrypted, it will not tell us anything. While encrypting address is necessary, it prevents geographical analysis. To preserve such important information, a class of techniques called "de-identification" or "anonymization" techniques have been developed. These techniques replace sensitive information with randomly generated sensitive information that preserves the important characteristics of the original data values. For example, a Hispanic feminine first name would be replaced with a different Hispanic feminine first name; a Japanese last name would be replaced with a different Japanese last name; and an address may be replaced with some other valid address within a 10 mile radius as described in the following illustration.

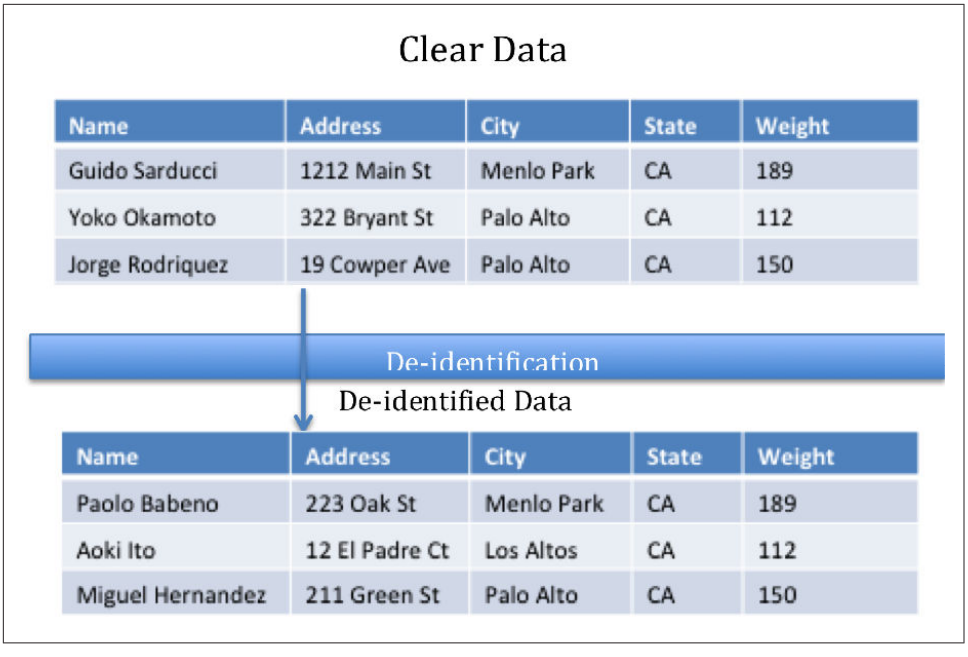


Figure 2-6.

Several tools including DataGuise, IBM Infosphere Optim and Informatica provide these capabilities.

Even if we de-identify sensitive data, some analysts will need access to the real data. Furthermore, even when there is no sensitive data, most enterprises compartmentalize data access and provided it on an as-needed basis only. Since Data Science is by its nature exploratory, it is difficult to predict what data the analyst is going to need. Even for simple analytics, there is a lot of power that comes from understanding what data is available and getting access to it. As a compromise between very high maintenance tightly managed privileges and a free for all approach that requires no management, companies are turning to self-service access management.

Self-Service Access Management

Self-service access management has several distinct characteristics

- Analysts can explore (search and browse) metadata for all data sets that may be made available to them
- Analysts can request access to a data set from the data set owner
- The owner of data set decides who can access it, in what way and for how long
- All the requests, justifications and permissions are tracked for security audit purposes

The following set of steps illustrates a self-service access management and data provisioning system

Step 1: Data Owner publishes data assets to the Catalog

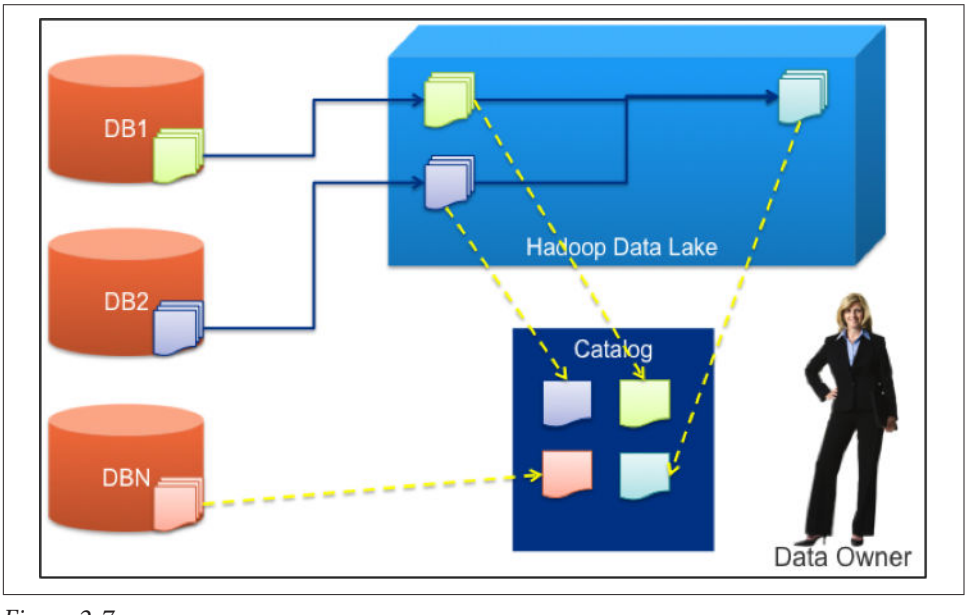


Figure 2-7.

Step 2: Data Analyst finds data sets in Catalog

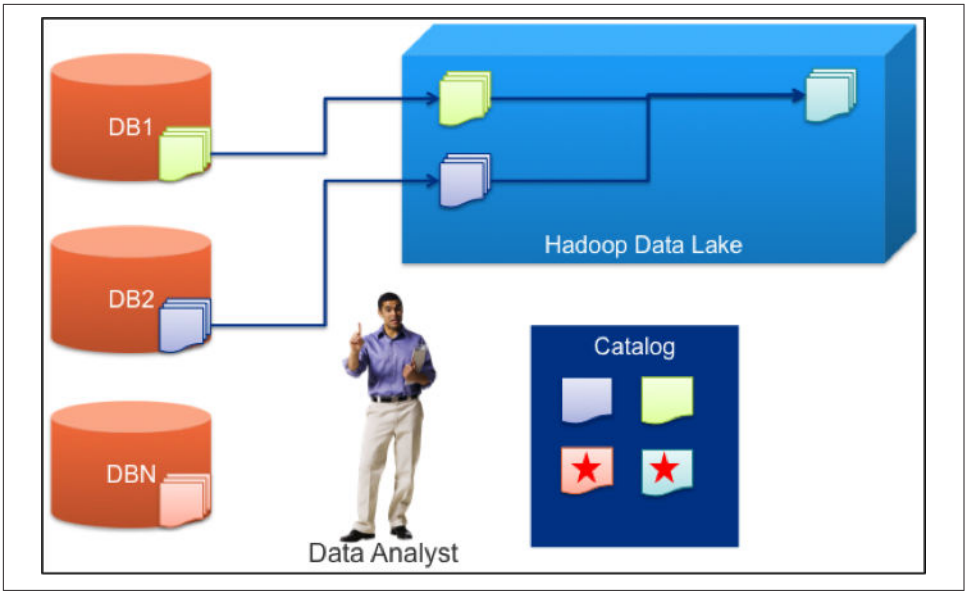


Figure 2-8.

Step 3: Analyst requests access from Data Owner

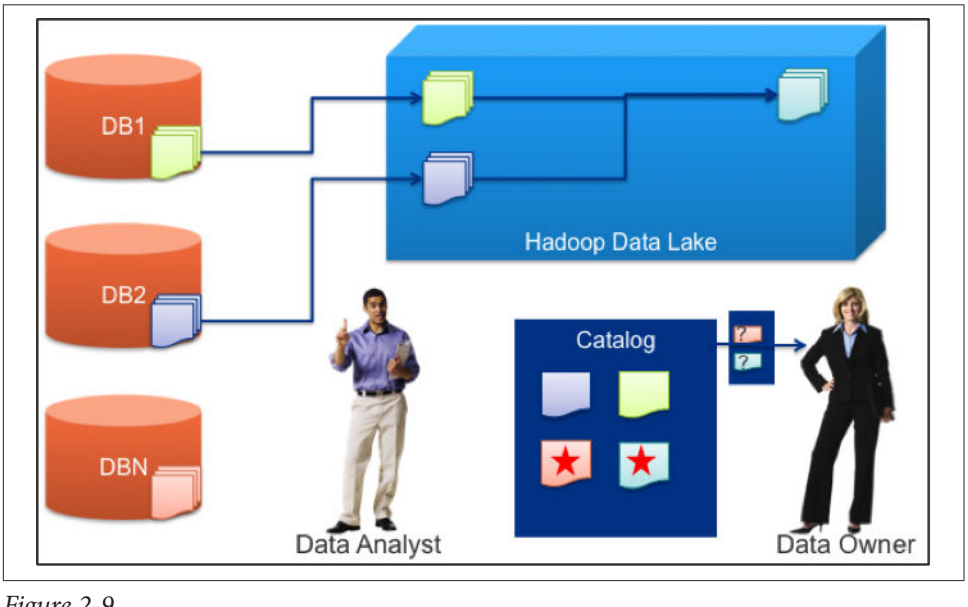


Figure 2-9.

Step 4: Data Owner approves the request

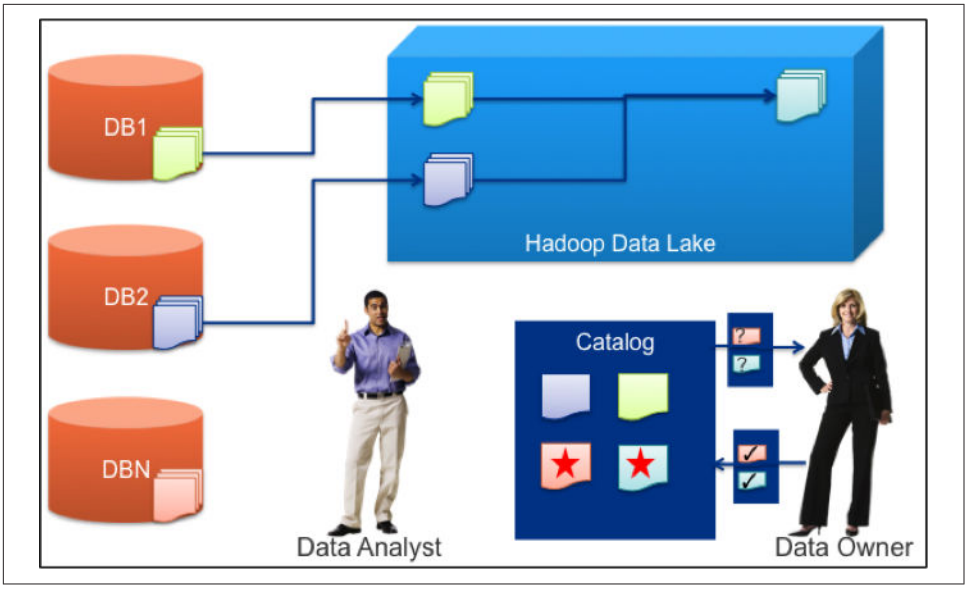


Figure 2-10.

Step 5: Data Sets are provided (provisioned) to the Analyst

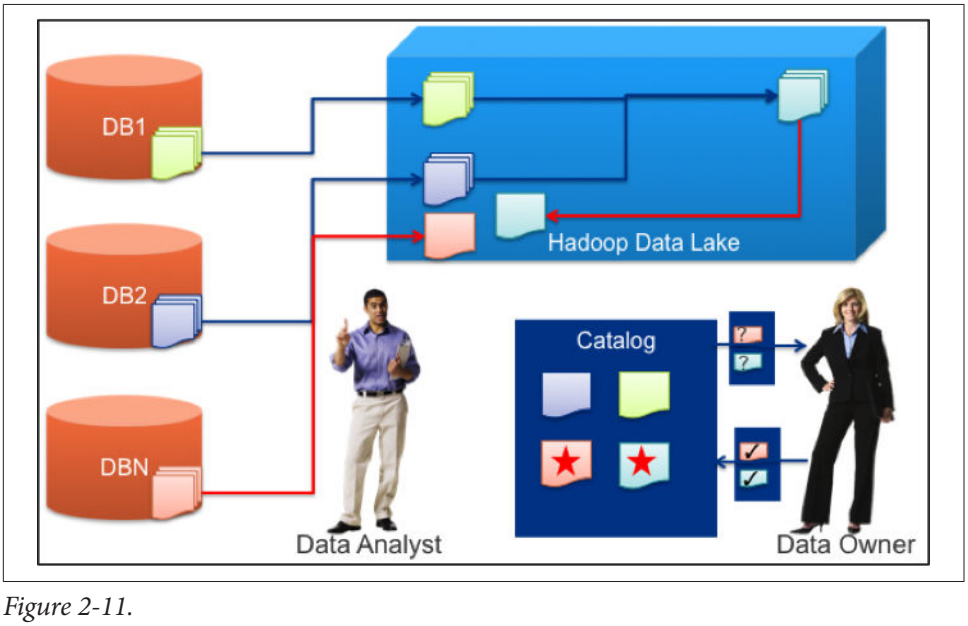


Figure 2-11.

There are many advantages to this approach. As an IT executive at a large enterprise that I interviewed for this book explained,

> "People are afraid to share data unless they can make sure it is used appropriately. By giving them the power to decide who can use it and how, we created an environment where they feel safe sharing data. Before we implemented this self-service approach, obtaining data required months of negotiations up and down the management chain. Everyone always asked for everything they could think of to make sure they did not have to go through the pain and delays of negotiations again. This made data owners distrustful of the true needs of the requesters and forced them to institute strict review processes where the requesters had to provide very detailed requirements and justifications, causing ever more work and more delays. It was virtually impossible to explore data in such an environment.
>
> With the self-service access management, requesters can study the data sets in the catalog and figure out what they need before they even place access requests, so there are many fewer requests, and a lot less data is being requests even when the requests are made, since the analysts have done pretty extensive exploration with the catalog and found what data is fit for purpose. Finally, because access requests are pretty automated, making additional requests is quick and straightforward."

In addition to giving data owners control over who uses data and giving analysts the ability to explore the data sets and obtain access quickly, by granting access to data for a specific period of time, this approach eliminates both the maintenance associated with managing permissions to all possible data sets; as well as the inevitable legacy access that the analysts retain after the project is over, just in case they may need it. With self-service approach, they just request for the access to be quickly reinstated.

Once the authorization has been obtained, physical access to data can be provided to analysts in a variety of ways depending on the nature of the data sets and the needs of the projects. A very popular way to granting access is creating an external Hive table for the data set. External Hive tables do not copy or change the data sets and can be created or deleted with negligible compute costs (since they are just metadata definitions). The analysts are then granted access to the Hive tables.

For some projects, analysts may want to make copies of the files or create their own Hive tables (for example, with different input formats that tell Hive to parse and interpret data). In such cases, they can be given a copy of a data set or read access to the data set itself.

Combining data

Since most data science tools and visualization tools work with a single data set and since data frequently comes from different sources and data sets, these data sets need to be combined into a single data set. For example, analyst may get social media information about company products and then need to relate it to the product list to figure out which tweets are about which products. Analysts may get customer information from a CRM system and customer issues from the Bug tracking system and order shipments from the ERP system. Sometimes combining data is straightforward and requires joining data sets on common keys such as CustomerID. Often times however, especially when combining company data with external data, there are no common ids and data needs to be combined using what's called *Identity Resolution*. This can be done using specialized systems such as MDM or Master Data Management systems with probabilistic matching engines that have built in rules for understanding and aligning different types of data such as customer data or product data. Some modern tools like Tamr are using machine learning to broaden identity resolution beyond few well-understood domains and apply it to the broad spectrum of data sets.

Data Relevance – Deciding what to keep

Not every attribute in a data set is going to be relevant to the project, therefore, the first step of treating a data set is to identify which attribute may be relevant. Sometimes, when doing a report or visualization, the analyst already knows what attributes they need. In other cases, especially when doing advanced analytics, a data scientist may not know which attributes will influence the model. For example, if the project is to predict sales prices for houses, the data scientist may have access to hundreds of attributes including the color of the house, number of bedrooms, the brand of a water heater, and so forth. While only few of these attributes will have influence on the price of the house, the trick is knowing which few will. A data scientist will frequently plot attributes against the outcome and against other attributes to identify which attributes make a difference and which do not.

Data Quality - Cleaning or treating data appropriately

Data is usually not clean. Cleaning data usually involves several steps:

- Normalizing or harmonizing data
- Addressing data quality issues

Normalizing data

Same information may be stored differently in different data sets and even in the same data set. The most common normalization challenges include:

- Normalizing to common units of measure. Different records and data sets may use different units of measure (e.g., imperial measurements vs. decimal, different packaging levels – crates vs boxes vs units) and normalization will convert all measurements to the same unit of measure.
- Different representations - data sets may represent the same data differently. For example, one data set may have date of birth, while the other has age and one would need to be converted to the other representation. Frequently, geographical information can widely range in representations from addresses to areas to precise latitude and longitude.
- Different formats – some data sets may have separate fields for first, middle and last name, while others may combine it into one. Similarly for addresses.
- Inconsistent accuracy – some measurements may be more detailed than others. For example, one data set may capture timestamps to a second, which another to a minute. One may use dollars and cents and the other just dollars. If the two are to be combined, data usually needs to be rounded to the same level of accuracy.

There are many other instances specific to different industries and companies and all need to be resolved before data can be combined and used.

Once data is normalized, its quality needs to be assessed and addressed. Data Quality is a well-developed discipline of Data Management and involves defining quality rules; applying those rules to data to detect violations, often called exceptions; and fixing those exceptions. Data Quality is a big topic and entire books have been written about it, so we will just provide a quick summary designed to get the general approach across.

Data Quality rules come in many shapes and sizes, but can generally be broken into several broad categories:

- Scalar tests – these rules can be applied to a specific value. For example, Customer Name is a required field and should have a value; Salary should be a number; Age should be between 0 and 150
- Field level – applied to all the values in a field. The most common examples have to do with field uniqueness – for example, customer id should be unique; and field density – for example, Customer Name cannot be empty; but may have other rules such as average income should be between X and Y. While some of these rules like density may seem redundant – for example, Customer Name not empty can be expressed as a scalar test, the advantage of doing it at field level is that we can provide tolerances – for example, we can tolerate up to 10% of customer names being empty.
- Record level – applied to all the fields in a single record. For example if US_CITIZEN field is True then Social Security Number should not be empty; Each root element in Orders record should have 3 children
- Data set (table/file level) – these are not common

- Cross-data set level – referential integrity rules are very common in relational systems – they basically state that a primary key should be unique and a foreign key field should not have any value that do not exist in the primary key field; count (orders) from orders where fulfilled = 1 should be the same as count(distinct order_id) from shipments; or number of rows in file1 should be smaller or equal than number of rows in file2

Some of the data quality rules can be fixed programmatically, while others require manual intervention also called curation. For example, a missing Customer Name may be looked up in master customer list or missing gender can be deduced from salutation or the first name. On the other hand, sometimes, there is no way of programmatically fixing data quality problems. In such cases, data needs to either fixed manually or fixed differently depending on the project it is being used for. For example, if there is a transaction for an account number that's missing a digit, the analyst curating the data may need to manually search through accounts to see which account it can match and then look at the account history to see if there is a transaction for that amount on that date; or if the analyst has customer income information missing and no place of getting it, she may decide to take out the records with missing income, or treat it as 0 or replace missing income with average income depending on the type of the project she is doing.

Data Quality is a particularly difficult problem for Data Science for two reasons:

1. It is very difficult to tell from the results of data science that data had quality problems
2. Data curation may be very different for different projects and even for the same project, different treatments (fixes) may need to be attempted until the right one is found

Because most advanced analytics are probabilistic and/or require machine learning, it is virtually impossible to tell from the outcome whether data was bad. If I run a simple BI report, I can often spot if the numbers don't make sense – for example, if Finland sales exceed US sales when usually, US sales dwarf sales for Finland. On the other hand, if customer segmentation project breaks my 3 million customers into 12 buckets, it is very difficult and may be impossible to spot whether it is right or wrong. Even with simpler problems like linear regression, all a data scientist can usually tell is that the model is not stable. It is very difficult to tell whether it is because of bad data or because there is really no predictive model for this set of attributes. That's why data scientists often try different treatments. For example, they may try to substitute a missing income with average income and, if that doesn't produce a stable model, try to substitute it with 0, and if that doesn't work, try to take out all the records with missing income.

Transforming data into the right shape

Once the analyst finds the right data set, she needs to shape it to fit the project.

Depending on the level of technical sophistication and complexity of the problem, this process can be as simple as pointing your visualization tool like Tableau to a Hive table or as complex as writing custom R and Python code.

For Data Science projects, data frequently needs to be transformed from attributes to a variable matrix – a tabular data set with relevant attributes; to a feature matrix – a matrix of numeric features. For example, if we are trying to predict how likely forecasted sales deals are to close, we may locate attributes such as purchase history, customer size, sales person record and so forth. From these, we may derive variables such as: how many times has the customer purchased from us; number of employees at the customer company; total budget for our project; smallest and largest purchase the customer ever made with us; how long does a deal take on average to close, etc. From these variables, we may then derive the features: how many times has the customer purchased from us; customer size (0 – small; 1 medium; 2 large; 3 very large); total budget; will this purchase exceed the largest purchase that the customer ever made (0 yes, 1 no), has the sales cycle within 10% of how long it takes for a deal to close on average, etc. These transformations require both domain understanding to figure out what features to create and significant coding effort to generate the features.

Data Prep Tooling

The most popular tool for getting data into shape is Excel, although analysts frequently use databases and SQL; R; Python, Pig and other scripting languages; as well as integrated analytical tool sets like SAS and SPSS. Recently a number of new Data Prep or Data Wrangling tools such as Informatica Rev, Paxata, DataMeer, Trifacta and dozens of others, have hit the market, offering a machine learning enhanced approach to wrangling data that combines visualization, machine learning and ability to sample and work with massive data sets.

Creating an Analytical Model

There are many machine learning techniques. This book is about setting up the data infrastructure rather than creating the models. However, it is often useful to work through an example to understand what Data Scientists are trying to do and, therefore, what data infrastructure support they need. For the purposes of illustration, we will look at a simple example of creating a linear regression model to predict how likely are the forecasted deals in our sales pipeline to close. Assuming we have already created a data set in the previous steps that contains a feature matrix and historical outcomes, at this point Data Scientist is going to divide a data set into two: training set and test set. She is then going to run a linear regression model against the training

set. The result of the model is a mathematical formula that contains a set of coefficients – one for each feature and one independent that, if applied to a record in a feature matrix, should predict the outcome as either 1 or 0. This is why we needed to convert all the data into a feature matrix – so we can create a mathematical model.

Once the model is created, data scientist is going to run it against the test set. In other words, she will apply the formula to each record in the test set and see if it correctly predicted the outcome. There are various ways to measure the effectiveness of the model. If the formula correctly predicts the outcomes in the test data set, the model is called "stable". If not, it is called "unstable" and data scientist will need to either add or remove features or apply different treatments to the existing features and try again. Often it takes hundreds of tries to produce a stable model.

As you can see, the path to success is long and difficult. It requires forethought, planning and a wide array to technology and skills. Understanding the type of work that the analysts are going to be doing and the type of data that they will be working with is key to creating a successful roll-out plan. By staging Data Lake roll out to show incremental success, and by systematically implementing governance as new Zone come on-line and new data is ingested and new teams are on-boarded, the Big Data team will be able to successfully generate demand from business teams wanting to adopt the Data Lake and earn the confidence of governance, risk and compliance organizations.