IN THIS CHAPTER

- Defining and scoping the data lake
- Diving underwater in the data lake
- Dividing up the data lake
- Making sense of conflicting terminology

Chapter 1
Jumping into the Data Lake

The lake is the place to be this season — the data lake, that is!

Just like the newest and hottest vacation destination, everyone is booking reservations for a trip to the data lake. Unlike a vacation, though, you won't just be spending a long weekend or a week or even the entire summer at the data lake. If you and your work colleagues do a good job, your data lake will be your go-to place for a whole *decade* or even longer.

What Is a Data Lake?

Ask a friend this question: "What's a lake?" Your friend thinks for a moment, and then gives you this answer: "Well, it's a big hole in the ground that's filled with water."

Technically, your friend is correct, but that answer also is far from detailed enough to really tell you what a lake actually is. You need more specifics, such as:

- How big, dimension-wise (how long and how wide)
- How deep that "big hole in the ground" goes
- How much variability there is from one lake to another in terms of those length, width, and depth dimensions (the Great Lakes, anyone?)
- How much water you'll find in the lake and how much that amount of water may vary among different lakes
- Whether a lake contains freshwater or saltwater

Some follow-up questions may pop into your mind as well:

- A pond is also a big hole in the ground that's filled with water, so is a lake the same as a pond?
- What distinguishes a lake from an ocean or a sea?
- Can a lake be physically connected to another lake?
- Can the dividing line between two states or two countries be in the middle of a lake?
- If a lake is empty, is it still considered a lake?
- If one lake leaves Chicago, heading east and travels at 100 miles per hour, and another lake heads west from New York . . . oh wait, wrong kind of word problem, never mind. . . .

So many missing pieces of the puzzle, all arising from one simple question!

You'll find the exact same situation if you ask someone this question: "What's a data lake?" In fact, go ahead and ask your favorite search engine that question. You'll find dozens of high-level definitions that will almost certainly spur plenty of follow-up questions as you try to get your arms around the idea of a data lake.

TIP

Here's a better idea: Instead of filtering through all that varying — and even conflicting — terminology and then trying to consolidate all of it into a single comprehensive definition, just think of a data lake as the following:

> A solidly architected, logically centralized, highly scalable environment filled with different types of analytic data that are sourced from both inside and outside your enterprise with varying latency, and which will be the primary go-to destination for your organization's data-driven insights

Wow, that's a mouthful! No worries: Just as if you were eating a gourmet fireside meal while camping at your favorite lake, you can break up that definition into bite-size pieces.

Rock-solid water

A data lake should remain viable and useful for a long time after it becomes operational. Also, you'll be continually expanding and enhancing your data lake with new types and forms of data, new underlying technologies, and support for new analytical uses.

REMEMBER

Building a data lake is more than just loading massive amounts of data into some storage location.

To support this near-constant expansion and growth, you need to ensure that your data lake is well architected and solidly engineered, which means that the data lake

- Enforces standards and best practices for data ingestion, data storage, data transmission, and interchange among its components and data delivery to end users
- Minimizes workarounds and temporary interfaces that have a tendency to stick around longer than planned and weaken your overall environment
- Continues to meet your predetermined metrics and thresholds for overall technical performance, such as data loading and interchange, as well as user response time

Think about a resort that builds docks, a couple of lakeside restaurants, and other structures at various locations alongside a large lake. You wouldn't just hand out lumber, hammers, and nails to a bunch of visitors and tell them to start building without detailed blueprints and engineering diagrams. The same is true with a data lake. From the first piece of data that arrives, you need as solid a foundation as possible to help keep your data lake viable for a long time.

A really great lake

You'll come across definitions and descriptions that tell you a data lake is a centralized store of data, but that definition is only partially correct.

A data lake is *logically* centralized. You can certainly think of a data lake as a single place for your data, instead of having your data scattered among different

databases. But in reality, even though your data lake is logically centralized, its data is *physically* decentralized and distributed among many different underlying servers.

TECHNICAL STUFF

The data services that you use for your data lake, such as the Amazon Simple Storage Service (S3), the Microsoft Azure Data Lake Storage (ADLS), or the Hadoop Distributed File System (HDFS) manage the distribution of data among potentially numerous servers where your data is actually stored. These services hide the physical distribution from almost everyone other than those who need to manage the data at the server storage level. Instead, they present the data as being logically part of a single data lake. Figure 1-1 illustrates how logical centralization accompanies physical decentralization.

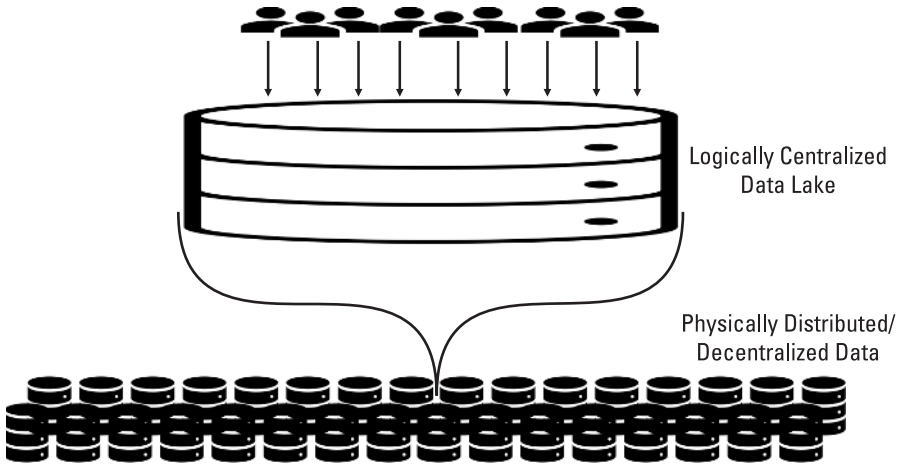


FIGURE 1-1: A logically centralized data lake with underlying physical decentralization.

Expanding the data lake

How big can your data lake get? To quote the old saying (and to answer a question with a question), how many angels can dance on the head of a pin?

Scalability is best thought of as "the ability to expand capacity, workload, and missions without having to go back to the drawing board and start all over." Your data lake will almost always be a cloud-based solution (see Figure 1-2). Cloud-based platforms give you, in theory, infinite scalability for your data lake. New servers and storage devices (discs, solid state devices, and so on) can be incorporated into your data lake on demand, and the software services manage and control these new resources along with those that you're already using. Your data lake contents can then expand from hundreds of terabytes to petabytes, and then to exabytes, and then zettabytes, and even into the ginormousbyte range. (Just kidding about that last one.)

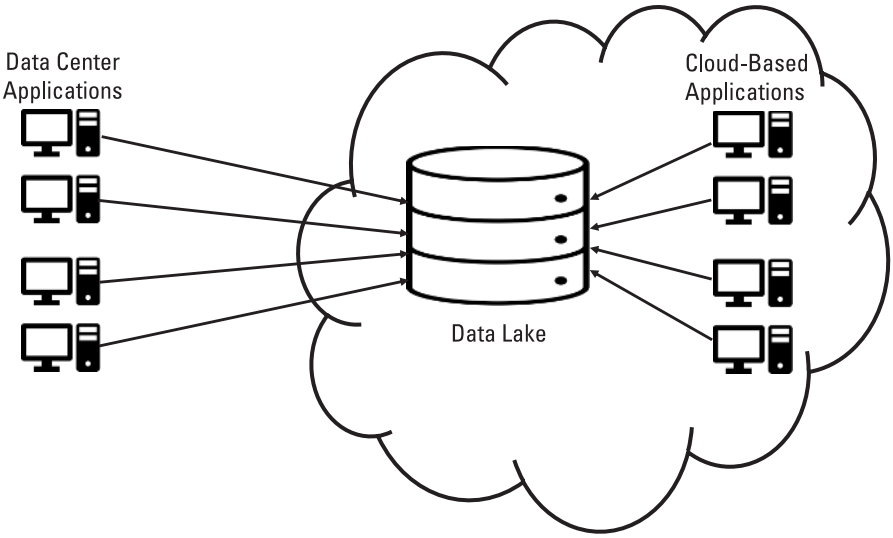


FIGURE 1-2: Cloud-based data lake solutions.

TIP

Cloud providers give you pricing for data storage and access that increases as your needs grow or decreases if you cut back on your functionality. Basically, your data lake will be priced on a pay-as-you-go basis.

Some of the very first data lakes that were built in the Hadoop environment may reside in your corporate data center and be categorized as *on-prem* (short for *on-premises*, meaning "on your premises") solutions. But most of today's data lakes are built in the Amazon Web Services (AWS) or Microsoft Azure cloud environments. Given the ever-increasing popularity of cloud computing, it's highly unlikely that this trend of cloud-based data lakes will reverse for a long time, if ever.

As long as Amazon, Microsoft, and other cloud platform providers can keep expanding their existing data centers and building new ones, as well as enhancing the capabilities of their data management services, then your data lake should be able to avoid scalability issues.

TECHNICAL STUFF

A multiple-component data lake architecture (see Chapter 4) further helps overcome performance and capacity constraints as your data lake grows in size and complexity, providing even greater scalability.

More than just the water

Think of a data lake as being closer to a lake resort rather than just the lake — the body of water — in its natural state. If you were a real estate developer, you might buy the property that includes the lake itself, along with plenty of acreage

surrounding the lake. You'd then develop the overall property by building cabins, restaurants, boat docks, and other facilities. The lake might be the centerpiece of the overall resort, but its value is dramatically enhanced by all the additional assets that you've built surrounding the lake.

REMEMBER

A data lake is an entire environment, not just a gigantic collection of data that is stored within a data service such as Amazon S3 or Microsoft ADLS.

In addition to data storage, a data lake also includes the following:

- One or (usually) more mechanisms to move data from one part of the data lake to another.
- A catalog or directory that helps keep track of what data is where, as well as the associated rules that apply to different groups of data; this is known as *metadata.*
- Capabilities that help unify meanings and business rules for key data subjects that may come into the data lake from different applications and systems; this is known as *master data management.*
- Monitoring services to track data quality and accuracy, response time when users access data, billing services to charge different organizations for their usage of the data lake, and plenty more.

Different types of data

If your data lake had a motto, it might be "All data are created equal."

In a data lake, data is data is data. In other words, you don't need to make special accommodations for more complex types of data than you would for simpler forms of data.

Your data lake will contain structured data, unstructured data, and semi-structured data (see Figure 1-3). The following sections cover these types of data in more detail.

Structured data: Staying in your own lane

You're probably most familiar with *structured data,* which is made up of numbers, shorter-length character strings, and dates. Traditionally, most of the applications you've worked with have been based on structured data. Structured data is commonly stored in a relational database such as Microsoft SQL Server, MySQL, or Oracle Database.

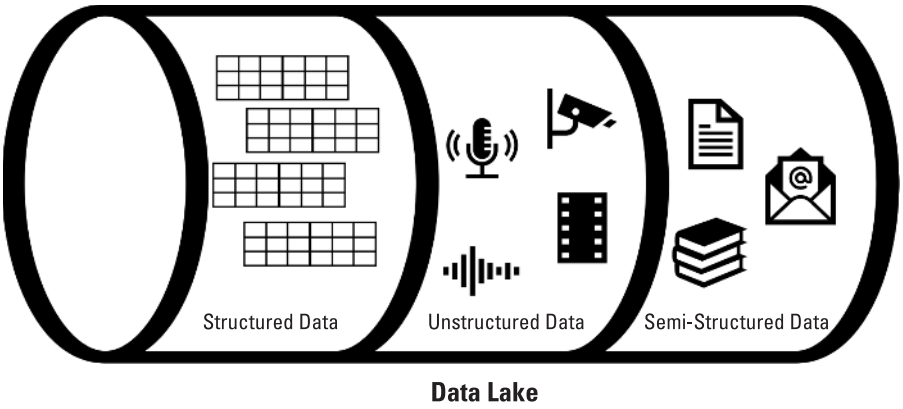


FIGURE 1-3: Different types of data in your data lake.

In a database, you define columns (basically, fields) for each of your pieces of structured data, and each column is rigidly and precisely defined with the following:

- **A data type,** such as INTEGER, DECIMAL, CHARACTER, DATE, DATETIME, or something similar
- **The size of the field,** either explicitly declared (for example, how many characters a CHARACTER column will contain) or implicitly declared (the system-defined maximum number for an INTEGER or how a DATE column is structured)
- **Any specific rules that apply to a data column or field,** such as the permissible range of values (for example, a customer's age must be between 18 and 130) or a list of allowable values (for example, an employee's current status can only be FULL-TIME, PART-TIME, TERMINATED, or RETIRED)
- **Any additional constraints,** such as primary and foreign key designations, or *referential integrity* (rules that specify consistency for certain columns across multiple database tables)

Unstructured data: A picture may be worth ten million words

Unstructured data is, by definition, data that lacks a formally defined structure. Images (such as JPEGs), audio (such as MP3s), and videos (such as MP4s or MOVs) are common forms of unstructured data.

Semi-structured data: Stuck in the middle of the lake

Semi-structured data sort of falls in between structured and unstructured data. Examples include a blog post, a social media post, text messages, an email message, or a message from Slack or Microsoft Teams. Leaving aside any embedded or attached images or videos for a moment, all these examples consist of a long string of letters, numbers, and special characters. However, there's no particular structure assigned to most of these text strings other than perhaps a couple of lines of heading information. The body of an email may be very short — only a line or two — while another email can go on for many long paragraphs.

In your data lake, you need to have all these types of data sitting side by side. Why? Because you'll be running analytics against the data lake that may need more than one form of data. For example, you receive and then analyze a detailed report of sales by department in a large department store during the past month.

Then, after noticing a few anomalies in the sales numbers, you pull up in-store surveillance video to analyze traffic versus sales to better understand how many customers may be looking at merchandise but deciding not to make a purchase. You can even combine structured data from scanners with your unstructured video data as part of your analysis.

If you had to go to different data storage environments for your sales results (structured data) and then the video surveillance (unstructured data), your overall analysis is dramatically slowed down, especially if you need to integrate and cross-reference different types of data. With a data lake, all this data is sitting side by side, ready to be delivered for analysis and decision-making.


TECHNICAL STUFF

In their earliest days, relational databases only stored structured data. Later, they were extended with capabilities to store structured and unstructured data. Binary large objects (BLOBs) were a common way to store images and even video in a relational database. However, even an *object-extended* relational database doesn't make a good platform for a data lake when compared with modern data services such as Amazon S3 or Microsoft ADLS.

Different water, different data

A common misconception is that you store "all your data" in your data lake. Actually, you store all or most of your *analytic* data in a data lake. Analytic data is, as you may suspect from the name, data that you're using for analytics. In contrast, you use *operational* data to run your business.

What's the difference? From one perspective, operational and analytic data are one and the same. Suppose you work for a large retailer. A customer comes into one of your stores and makes some purchases. Another customer goes onto your company's website and buys some items there. The records of those sales — which customers made the purchases, which products they bought, how many of each product, the dates of the sales, whether the sales were online or in a store, and so on — are all stored away as official records of those transactions, which are necessary for running your company's operations.

But you also want to analyze that data, right? You want to understand which products are selling the best and where. You want to understand which customers are spending the most. You have dozens or even hundreds of questions you want to ask about your customers and their purchasing activity.

REMEMBER

Here's the catch: You need to make copies of your operational data for the deep analysis that you need to undertake; and the copies of that operational data are what goes into the data lake (see Figure 1-4).

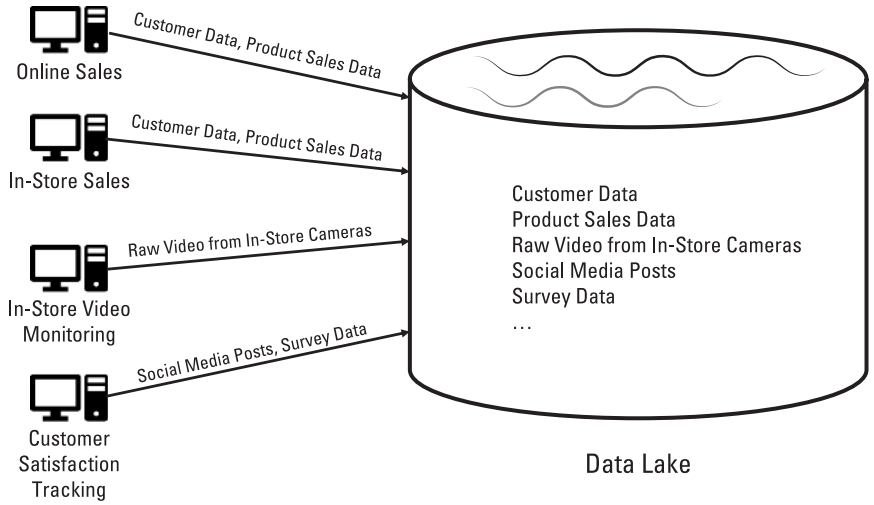


FIGURE 1-4: Source applications feeding data into your data lake.

Wait a minute! Why in the world do you need to copy data into your data lake? Why can't you just analyze the data right where it is, in the source applications and their databases?

Data lakes, at least as you need to build them today and for the foreseeable future, are a continuation of the same model that has been used for data warehousing since the early 1990s. For many technical reasons related to performance, deep analysis involving large data volumes and significant cross-referencing directly

in your source applications isn't a workable solution for the bulk of your analytics.

Consequently, you need to make copies of the operational data that you want for analytical purposes and store that data in your data lake. Think of the data inside your data lake as (in used-car terminology) previously owned data that has been refurbished and is now ready for a brand-new owner.

But if you can't adequately do complex analytics directly from source applications and their databases, what about this idea: Run your applications off your data lake instead! This way, you can avoid having to copy your data, right? Unfortunately, that idea won't work, at least with today's technology.


TECHNICAL STUFF

Operational applications almost always use a relational database, which manages *concurrency control* among their users and applications. In simple terms, hundreds or even thousands of users can add new data and make changes to a relational database without interfering with each other's work and corrupting the database. A data lake, however, is built on storage technology that is optimized for retrieving data for analysis and doesn't support concurrency control for update operations.

Many vendors are working on new technology that will allow you to build a data lake for operational, as well as analytical purposes. This technology is still a bit down the road from full operational viability. For the time being, you'll build a data lake by copying data from many different source applications.

Refilling the data lake

What exactly does "copying data" look like, and how frequently do you need to copy data into the data lake?


REMEMBER

Data lakes mostly use a technique called ELT, which stands for either *extract, transform, and load* or *extraction, transformation, and loading.* With ELT, you "blast" your data into a data lake without having to spend a great deal of time profiling and understanding the particulars of your data. You extract data (the *E* part of *ELT*) from its original home in a source application, and then, after that data has been transmitted to the data lake, you load the data (the *L*) into its initial storage location. Eventually, when it's time for you to use the data for analytical purposes, you'll need to transform the data (the *T*) into whatever format is needed for a specific type of analysis.


TECHNICAL STUFF

For data warehousing — the predecessor to data lakes that you're almost certainly still also using — data is copied from source applications to the data warehouse using a technique called ETL, rather than ELT. With ETL, you need to thoroughly understand the particulars of your data on its way into the data warehouse, which

requires the transformation *(T)* to occur before the data is loaded *(L)* into its usable form.

With ELT, you can control the *latency,* or "freshness," of data that is brought into the data lake. Some data needed for critical, real-time analysis can be *streamed* into the data lake, which means that a copy is sent to the data lake immediately after data is created or updated within a source application. (This is referred to as a *low-latency data feed.*) You essentially push data into your data lake piece by piece immediately upon the creation of that data.

Other data may be less time-critical and can be "batched up" in a source application and then periodically transmitted in bulk to the data lake.

You can specify the latency requirements for every single data feed from every single source application.

REMEMBER

The ELT model also allows you to identify a new source of data for your data lake and then very quickly bring in the data that you need. You don't need to spend days or weeks dissecting the ins and outs of the new data source to understand its structure and business rules. You "blast" the data into your data lake in the natural form of the data: database tables, MP4 files, or however the data is stored. Then, when it's time to use that data for analysis, you can proceed to dig into the particulars and get the data ready for reports, machine learning, or however you're going to be using and analyzing the data.

Everyone visits the data lake

Take a look around your organization today. Chances are, you have dozens or even hundreds of different places to go for reports and analytics. At one time, your company probably had the idea of building an *enterprise data warehouse* that would provide data for almost all the analytical needs across the entire company. Alas, for many reasons, you instead wound up with numerous *data marts* and other environments, very few of which work together. Even enterprise data warehouses are often accompanied by an entire portfolio of data marts in the typical organization.

Great news! The data lake will finally be that one-stop shopping place for the data to meet almost all the analytical needs across your entire enterprise.

Enterprise-scale data warehousing fell short for many different reasons, including the underlying technology platforms. Data lakes overcome those shortfalls and provide the foundation for an entirely new generation of integrated, enterprise-wide analytics.

WARNING

Even with a data lake, you'll almost certainly still have other data environments outside the data lake that support analytics. Your data lake objective should be to satisfy *almost* all your organization's analytical needs and be the go-to place for data. If a few other environments pop up here and there, that's okay. Just be careful about the overall proliferation of systems outside your data lake; otherwise, you'll wind up right back in the same highly fragmented data mess that you have today before beginning work on your data lake.

The Data Lake Olympics

Suppose you head off for a weeklong vacation to your favorite lake resort. The people who run the resort have divided the lake into different zones, each for a different recreational purpose. One zone is set aside for water-skiing; a second zone is for speedboats, but no water-skiing is permitted in that zone; a third zone is only for boats without motors; and a fourth zone allows only swimming but no water vessels at all.

The operators of the resort could've said, "What the heck, let's just have a free-for-all out on the lake and hope for the best." Instead, they wisely established different zones for different purposes, resulting in orderly, peaceful vacations (hopefully!) rather than chaos.

A data lake is also divided into different zones. The exact number of zones may vary from one organization's data lake to another's, but you'll always find at least three zones in use — bronze, silver, and gold — and sometimes a fourth zone, the sandbox.

Bronze, silver, and gold aren't "official" standardized names, but they are catchy and easy to remember. Other names that you may find are shown in Table 1-1.

TABLE 1-1 Data Lake Zones

Recommended Zone Name	Other Names
Bronze zone	Raw zone, landing zone
Silver zone	Cleansed zone, refined zone
Gold zone	Performance zone, curated zone, data model zone
Sandbox	Experimental zone, short-term analytics zone

All the data lake zones, including the sandbox, are discussed in more detail in Part 2, but the following sections provide a brief overview.

WARNING

The boundaries and borders between your data lake zones can be fluid (Fluid? Get it?), especially with streaming data, as I explain in Part 2.

The bronze zone

You load your data into the bronze zone when the data first enters the data lake. First, you extract the data from a source application (the *E* part of ELT), and then the data is transmitted into the bronze zone in raw form (thus, one of the alternative names for this zone). You don't correct any errors or otherwise transform or modify the data at all. The original operational data should look identical to the copy of that data now in the bronze zone.

TIP

Your catchphrase for loading data into the bronze zone is "the need for speed." You may be trickling one piece of data at a time or bulk-loading hundreds of gigabytes or even terabytes of data. Your objective is to transmit the data into the data lake environment as quickly as possible. You'll worry about checking out and refining that data later.

The silver zone

The silver zone consists of data that has been error-checked and cleansed but still remains in its original format. Data may be copied from a source application in JavaScript Object Notation (JSON) format and land in the bronze zone in raw form, looking exactly as the data was in the source system itself — errors and all.

You'll patch up any known errors, handle missing data, and otherwise cleanse the data. Then you'll store the cleansed data in the silver zone, still in JSON format.

REMEMBER

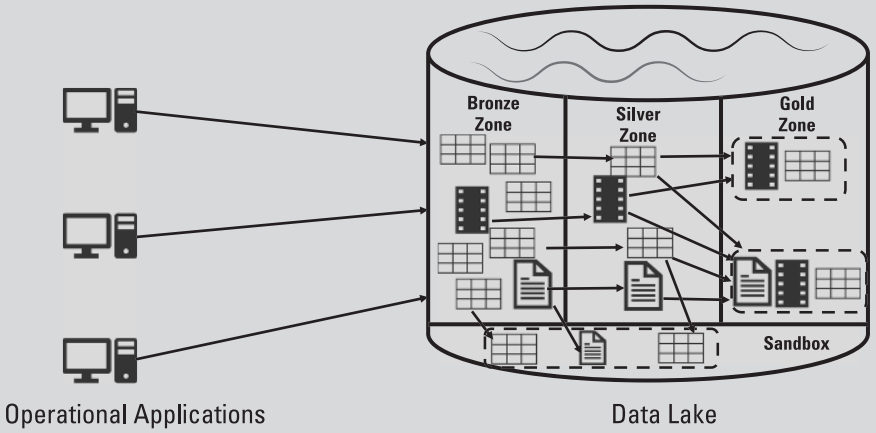
Not all data from your bronze zone will be cleansed and copied into your silver zone. The data lake model calls for loading massive amounts of data into the bronze zone without having to do upfront analysis to determine which data is definitely or likely needed for analysis. When you decide what data you need, you do the necessary data cleansing and move only the cleansed data into the silver zone.

The gold zone

The gold zone is the final home for your most valuable analytical data. You'll *curate* data coming from the silver zone, meaning that you'll group and restructure data into "packages" dedicated to your organization's high-value analytical needs.

LINKING THE DATA LAKE ZONES TOGETHER

The following figure shows the progressive pipelines of data among the various zones, including the sandbox. Notice how not every piece or group of data is cleansed and then sent from the bronze zone to the silver zone. You'll spend time refurbishing, refining, and transmitting data to the silver zone that you definitely or likely need for analytics.



Likewise, select data sets are sent from the silver zone to the gold zone. Remember that another name for the gold zone is the curated zone, meaning that you've especially selected certain data to be consolidated and then placed in "packages" within the gold zone.

You might transmit raw, uncleansed data from the bronze zone into the sandbox along with data from the silver zone, depending on the specifics of your experimental or short-term analytical needs.


TECHNICAL STUFF

You will almost certainly replicate data across the various gold zone packages, but that's not a problem at all. As long as you carefully control the data flows and the replicated data, you're unlikely to run into problems with uncontrolled data proliferation.

The sandbox

Your bronze, silver, and gold zones combine to form a *data pipeline.* In your gold zone, you create data packages that are closely aligned with high-value, pervasive

analytical needs and that will provide data-driven insights to your organization for a long time.

But what about shorter-term analytical needs or experiments that you want to run with your data? You may be building new machine learning models to predict customer behavior, optimize your supply chain, or determine new treatment plans for a hospital system's patients. You need to experiment with different machine learning techniques, and you need actual data for your work.

Head over to the sandbox and start playing. You'll load whatever data you need for your short-term or experimental work and do your thing. The data lake isolates the sandbox from the data pipeline, so you can do whatever you need without interfering with your organization's primary analytical work.

Data Lakes and Big Data

Turn the clock back to the early 2010s when big data burst onto the scene. Almost every organization was exploring how this new generation of data management technology can overcome many of the barriers and constraints of relational databases, particularly for analytical storage.

Big data promised — and delivered — significantly greater capacity than was possible with relational databases. With big data, you can store unstructured and semi-structured data alongside your structured data. You can also bring new data into a big data environment with lower latency than with relational databases.

Wait a minute! That sounds just like the description of a data lake! So, is a data lake just another name for big data?

Well, sort of . . . possibly . . . or maybe not. . . .

The best way to think of the two disciplines in relation to one another is as follows:

- Big data is the underlying core technology used to build a data lake.
- A data lake is an environment that includes big data but also potentially other data management technologies along with services for data transmission and data governance.

THE THREE (OR FOUR OR FIVE OR MORE) VS OF BIG DATA AND DATA LAKES

Quick quiz: Name all the Vs of big data and data lakes. You can start with the original three: volume, variety, and velocity. But you'll also find blog posts and online articles that mention value, *veracity* (a formal term for accuracy), visualization, and many others. In fact, don't be surprised if one day you read an article or blog post that also includes Valentine's Day!

The original three Vs of big data came from a Gartner Group analyst named Doug Laney, way back in 2001. Volume, variety, and velocity were primarily aspirational characteristics of data environments, describing next-generational characteristics beyond what the relational databases of the time were capable of supporting.

Over the years, other industry analysts, bloggers, consultants, and product vendors added to the list with their own Vs. The difference between the original three Vs and those that followed, though, is that value, veracity, visualization, and others all apply to tried-and-true relational technology just as much as to big data.

Don't get confused trying to decide how many Vs apply to big data and to data lakes. Just focus on the original three — volume, variety, and velocity — as the must-have characteristics of your data lake.

WARNING

You'll find varying perspectives on the relationship between big data and data lakes, which certainly confuses the issue. Some technologists reverse the relationship between big data and data lakes; they consider a data lake to be the core technology and big data to be the overall environment. So, if you run across a blog post or another description that differs from the one I use, don't worry. As with almost everything about data lakes and much of the technology world, you'll find all sorts of opinions and perspectives, especially when you don't have any official standards to govern a discipline.

The Hadoop open source environment, particularly the HDFS, is one of the first and most popular examples of big data. Some of the earliest data lakes were built, or at least begun, using HDFS as the foundation.

TECHNICAL STUFF

For purposes of establishing a data lake foundation, Amazon's S3 and Microsoft's ADLS both qualify as big data. Why? Both S3 and ADLS support the three Vs of big data, which are as follows:

- Storing extremely large *volumes* of data
- Supporting a *variety* of data, including structured, unstructured, and semi-structured data
- Allowing very high *velocity* for incoming data into the data lake rather than requiring or at least encouraging periodic batches of data

TIP

Think of big data as a core technology foundation that supports the three Vs of next-generation data management. Big data by itself, however, is just a platform. It's the natural body of water — the lake itself — at a popular lakeside resort. When you divide your big data into multiple zones, add capabilities to transmit data across those zones, and then govern the whole environment, you've built a data lake surrounding that big data foundation. You've done the analytical data equivalent of building the docks, the restaurants, and the boat slips surrounding the lake itself.

The Data Lake Water Gets Murky

In addition to data lakes, you may come across references to data ponds, data puddles, data rivers, data oceans, and data hot tubs. (Just kidding about the last one.) What's going on here?

WARNING

Your job when planning, architecting, building, and using a data lake is complicated by the fact that you don't have an official definition published by some sort of standards body, such as the American National Standards Institute (ANSI) or the International Organization for Standardization (ISO). That means that you or anyone else can define, use, and even publish your own terminology. You can call a smaller portion of a data lake a "data pond" if you want, or refer to a collection of data lakes as a "data ocean."

Don't panic! Of all the "data plus a body of water" terms you'll run across, *data lake* is by far the most commonly used. All the characteristics of a data lake — solid architecture, support for multiple forms of data, a support ecosystem surrounding the data — apply to what you can call a data pond or any other term.

If William Shakespeare were still around and plied his trade as an enterprise data architect rather than as a writer, he would put it this way: "A data lake by any other name would still be worth the time and effort to build."

BACK TO THE FUTURE WITH NAME CHANGES

In the early 1990s, data warehousing was the newest and most popular game in town for analytical data management. By the mid-'90s, the concept of a data warehouse was adapted to a *data mart* — essentially, a smaller-scale data warehouse. The original idea behind a data mart called for the data warehouse feeding a subset of its data into one or more data marts — sort of a "wholesaler-retailer" model.

The first generation of data warehouse projects, especially very large ones, was hallmarked by a high failure rate. By the late '90s, data warehouses were viewed as large, complex, and expensive efforts that were also very risky. A data mart, on the other hand, was smaller, less complex, and less expensive, and, thus, considered to be less risky.

The need for integrated analytical data was stronger than ever by the end of the '90s. But just try to get funding for a data warehousing project! Good luck!

Time for plan B.

Data warehouses went out of style for a while. Instead, data marts became the go-to solution for analytic data. No matter how big and complex an environment was, chances are, you'd refer to it as a data mart rather than a data warehouse. In fact, the idea of an *independent* data mart sprung up, and the original architecture for a data mart — receiving data from a data warehouse rather than directly from source systems — became known as a *dependent* data mart.

Fast-forward a couple of decades, and it's back to the future. First, big data sort of evolved into data lakes. Now you have analysts, consultants, and vendors complicating the picture with their own terminology. This won't be the last time you'll see shifting names and terminology in the world of analytic data, so stay tuned!