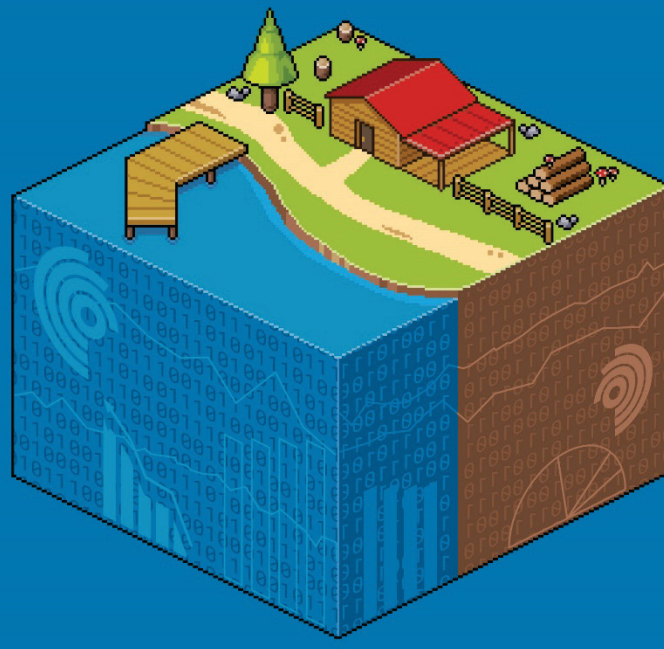
BILL INMON,
DAVE RAPIEN, AND VALERIE BARTELT

THE DATA LAKEHOUSE

THE BEDROCK FOR ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, AND DATA MESH



The Data Lakehouse

The Bedrock for Artificial Intelligence, Machine Learning, and Data Mesh

Bill Inmon

Dave Rapien

Valerie Bartelt

Technics Publications

115 Linda Vista, Sedona, Arizona, USA
https://www.TechnicsPub.com

Edited by Jamie Hoberman
Cover design by Lorena Molinari

First Printing First Edition 2023

ISBN, print ed. 9781634621571
ISBN, Kindle ed. 9781634621618
ISBN, ePub ed. 9781634621779
ISBN, PDF ed. 9781634621793

Library of Congress Control Number: 2023938916

Preface

Wherever you look, there are people everywhere going nuts over artificial intelligence. Or machine learning. Or data mesh. Everyone is out there championing and touting great new technological advances.

And indeed, there is great promise in new technology and new advances.

But every new technology starts with the assumption that there is data that will feed these new technologies. Having a source of data that artificial intelligence (AI), machine learning (ML), and data mesh will operate on is just a basic assumption everyone has.

Everyone wants their organization to be data-driven.

But in every case, these underlying assumptions are wrong. Unfortunately, AI, ML, and data mesh are as susceptible to the age-old paradigm of GIGO (garbage in/garbage out) as all of their predecessors. GIGO applies to AI, ML, and data mesh as it applies to all of the technologies that have ever been developed.

The truth is that there is not a solid foundation of data on which to operate.

But now there is the data lakehouse. When properly built, the data lakehouse provides the foundation of data that supports new and sophisticated technologies. The data lakehouse provides a bedrock foundation to build further analytical capabilities.

To make these technologies actually work, it is necessary to have a solid foundation of data. And it is not enough just to have data. You have got to have data that is:

- Believable
- Malleable
- Sharable

Once you have the foundation of data that has these qualities, then, and only then, can you advance to senior-level applications such as AI, ML, and data mesh. And when properly built, the data lakehouse provides this foundation.

So what are the qualities that make up the bedrock foundation of data that supports applications of the future?

The first observation is that we must account for different kinds of data. In particular, there is structured, textual, and analog data. These three different types of data have very different properties. The skills learned in mastering one

kind of data do not apply to other types of data. These different types of data are as different as Antarctica, the Amazon River, and the Sahara Desert. All three places reside on Earth. But all three places are fundamentally different from each other.

Stated differently, the rules and means of navigation, manipulation, and even survival in each of the different types of data are dramatically different. Yet to support applications and processing, we must become conversant in the different traits and idiosyncrasies of the different types of data.

This book is about the bedrock foundation of data needed to thrive and survive in modern information systems. This book is about the data lakehouse.

This book is for the architect, the business person, and the system developer.

We hope you find it useful.

We want you to succeed with artificial intelligence, machine learning, and data mesh.

Bill Inmon, Valerie Bartelt, and Dave Rapien
June 2023

Contents

Introduction

Once the world had simple applications. But today's world has all sorts of data, technology, hardware, and fancy gadgets. Data comes to us from many places and in many forms. And the volume of data is just crushing.

There are three different types of data that an organization uses for analytical purposes.

First, there is classical structured data that principally comes from executing transactions. This structured data has been around the longest. Second, there is textual data from emails, call center conversations, contracts, medical records, and elsewhere. Textual data was a "black box" that could only be stored but not analyzed by the computer. Third, there is the world of analog/Internet of Things (IoT). Textual extract, transform, and load (ETL) technology has opened the door of text to standard analytical techniques. Machines, such as drones, electric eyes, temperature gauges, and wristwatches, all generate data. Analog/IoT data is in a much rougher form than structured or textual data. And a tremendous amount of this data is generated in an automated manner. Analog/IoT data is the domain of the data scientist.

At first, we threw all of this data into a pit called the "data lake." But we soon discovered that merely throwing data

into a pit was pointless. To be useful, data needed to (1) be related to each other and (2) have its analytical infrastructure carefully arranged and made available to the end user.

Unless we meet these two conditions, the data lake turns into a swamp, and swamps start to smell after a while.

A data lake that does not meet the criteria for analysis is a waste of time and money.

Enter the data lakehouse. The data lakehouse adds elements to the data lake to become useful and productive. Stated differently, if all you build is a data lake without turning it into a data lakehouse, you have just created an expensive eyesore. Over time that eyesore is going to turn into an expensive liability.

With the data lakehouse, achieving a level of analytics and machine learning is possible that is not feasible or possible any other way. But like all architectural structures, the data lakehouse requires an understanding of architecture and an ability to plan and create a blueprint.

CHAPTER 1

Believable Data

Every end user will tell you, "I need to access my data." And indeed, there are many ways to access data. For example, the end user can use transactions to look at data. Reports. Spreadsheets. The Internet. Knowledge graphs. And so forth. The are literally hundreds of ways to access data.

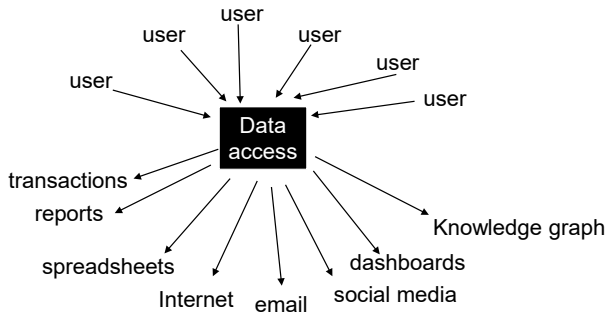


Figure 1.1. The first instinct the end user has for approaching the computerization of data is the need for and the ability to access data.

However, almost immediately, a second need arises in the form of the question, "Do I actually believe the data that I am accessing?" The end user discovers quickly that it is

one thing to access data. And it is another thing altogether to believe the data the end user is accessing.

Accessing data and believing data are not the same thing.

As a simple example of not believing the data you can access, consider the case where someone created a spreadsheet showing that Bill Inmon makes a million dollars a month. So there it is, on the computer in a computerized database, a million dollars a month.

The problem is that it is not the truth. Bill Inmon does not make a million dollars a month. Yet, it certainly is on the computer. And it certainly is what the database says. And the end user can certainly access it. And it certainly is not the truth.

If you were to use this information as a basis for making decisions, you would be making some very bad, very incorrect choices. This data is accessible but not believable.

In a way, this data is worse than just being incorrect. It is misleading and it looks like it is credible. It is a pitfall waiting to snare an unsuspecting person into making bad decisions.

So you see, it is one thing to access data. But if the data is not believable, the data can lead to great errors in decision-making and judgment.

Becoming a mature end user

There is then an implicit step that the end user must make when approaching a computerized system. The end user must graduate from wanting to access data to wanting to access believable data.

Fortunately, this is an intuitive step to make.

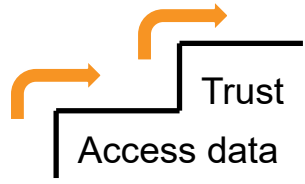


Figure 1.2: Steps the end user makes.

The simple steps shown are merely the first steps the end user takes in becoming a mature analyst and end user. There are a lot of other steps that require answering questions, such as these:

- Can I analyze the data?
- Can I see how the data changes over time?
- Can I combine the data with other data for analysis?

There indeed are many steps awaiting the end user in their journey to becoming competent with computer technology. But the two most basic are:

- Can I access my data?
- Can I believe the data that I am accessing?

As a simple exercise in the believability of data, consider a military commander who asks the simple question, "How many tanks do I have under my command?" This is a very straightforward question. The commander asks two different battalions to answer this question.

But when the commander asks one group of people, the commander gets one answer. And when the commander asks another group of people, the commander gets an entirely different answer. And trying to make a decision based on the answers that have been given is a dicey proposition.

When the commander asks Battalion ABC how many tanks are available, Battalion ABC says there are 1,500 tanks available. Then when the commander asks Battalion XYZ how many tanks are available, the commander gets the answer that 10,000 tanks are available. The commander could be making a disastrous decision if the commander doesn't know which answer is right.

Accessing the data for the commander is not the issue. Understanding what has been said is the issue.

How many tanks do we have?

1,500 tanks	10,000 tanks
Battalion ABC	Battalion XYZ
Maintained Field tested Fully armed In place	In reserve Untested Not armed In storage

Figure 1.3: Both battalions are correct. The question needs to be further qualified to get the desired answer.

Upon further investigation, the commander finds out that Battalion ABC has reported tanks that are ready for operation today. These tanks have been maintained, field-tested, fully armed, and are in position to be used. Now, battalion XYZ also has tanks. But the tanks that battalion XYZ has are stored in reserve, have not been field tested, are not armed, and are in storage.

In truth, the question needs to be refined. Instead of asking how many tanks do we have, the commander should have asked how many battle-ready tanks do we have? Or, how many tanks in reserve do we have? Making decisions on information that is unreliable or not fully qualified is dangerous.

Making a good decision requires not just data, but believable data.

The moving goalpost

There are different kinds of goals.

One kind of goal is a finite goal. American football has a goal line. In order for a team to score, the ball must be carried over the goal line. The goal line is fixed. It doesn't change throughout the game. It is an example of a finite goal.

The other kind of goal is an ever-moving goal. For example, suppose a person desires to become an accomplished cook. The person starts by scrambling eggs and boiling water. But quickly the person discovers that, around the world, there is a wealth of dishes to be cooked. There are Mexican dishes. Chinese dishes. French dishes. Japanese dishes. There is no end to the dishes that can be learned.

The person whose goal it is to become a good cook will never learn and cook all the dishes in the world. There is always something on the horizon that the person has not tried yet. Always.

But that does not mean that the person won't become a good cook. It just means that the goalpost keeps moving.

Cooking – unlike football – has a moving goalpost.

The quest for data believability is a moving target. There can be a constant improvement in the believability of data. But there is always some way to improve the believability of data. Like cooking, there is no end to the ways to improve data believability.

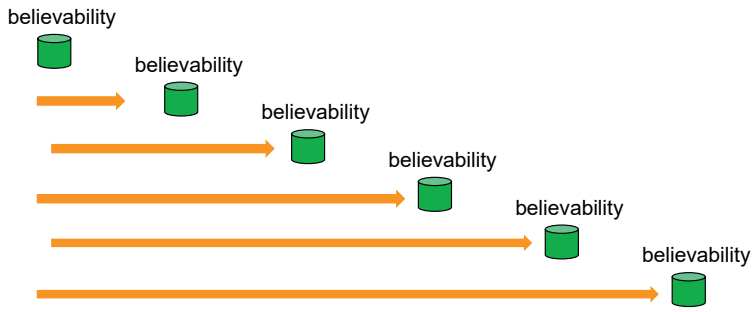


Figure 1.4: Over time, the believability of data improves.

Elements of believability

So what are the elements of data believability? In addition to the simple accuracy of data, we need to know:

- The source of the data

- When the organization first captured the data
- All of the data transformations
- Whether data audits/edits have been done
- Whether the data is complete
- Other data that corroborates the existing data
- Data usage and context
- Responsible parties for data capture and lineage
- Where the data was captured
- Metadata and context there is for the data
- Alterations made to the data
- Additions and appendages made to the data.

And this is just the shortlist. There are many other aspects that pertain to data that are required for the end user to fully understand the data.

Summary

The believability of data is the foundation on which the remainder of the world of technology relies. Without the believability of data, the world is subject to the famous saying, GIGO (garbage in, garbage out).

CHAPTER 2

The Bedrock Foundation

When you build an office building, you set your foundation on bedrock. Your building will collapse one day if you don't find bedrock for your foundation. Having the proper foundation for a structure sets the stage for the long-term happiness and safety of the structure's inhabitants.

The last place you would want to place your structure on is a beach. The first hurricane that passes by will knock your building down, and you'll be left with rubble.

Applications

Complex and sophisticated applications such as artificial intelligence, machine learning, and data mesh all depend on a foundation of data on which to operate. If we build these complex and elaborate technologies on sand, they will not operate properly. To function properly and fulfill their purpose, these technologies must rest on bedrock.

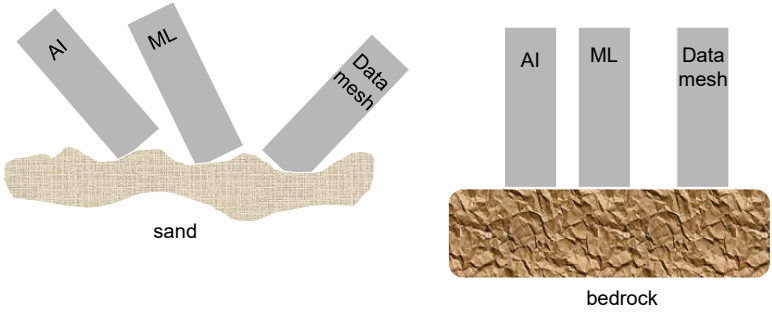


Figure 2.1: For applications to have stability and believability, they must be built on bedrock and not sand.

The bedrock foundation of these technologies is data. It is not enough for these technologies to access data. The data that they access has to be believable. If the data they operate on is not believable, the technology will deliver incorrect and misleading results despite the elegance of the technology itself.

The data lakehouse is the bedrock for these technologies.

AI and medicine

As a simple example of the need for a proper data foundation, suppose there is an AI application for medicine. The nurse that enters patient data has entered the patient's incorrect blood type. The nurse entered blood type O when the blood type should have been A.

If AI is trusted to the point that AI controls a blood transfusion, the results would be disastrous for the patient. If the data entered and used by AI is not correct, there is little AI can do to amend the validity of the data.

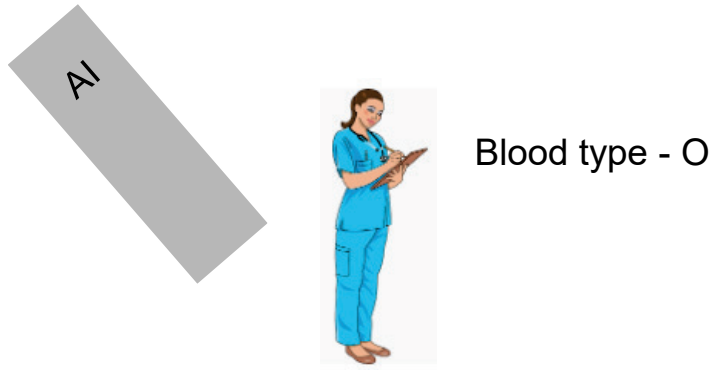


Figure 2.2: Actual blood type is A.

As another example of the need for believable data, suppose there is a bank customer. The bank records show this customer has a small amount of money in her account. The bank treats her as if she were a pauper. But in another bank account, the bank shows that she has inherited a large amount of money. The lady is a millionaire. But there is no way to connect the different accounts, and therefore, the bank does not see the total amount this customer has with this bank.

Data mesh shows that the lady should be treated as a pauper. If the totality of the data could be uncovered, the true status of the lady would also be uncovered.

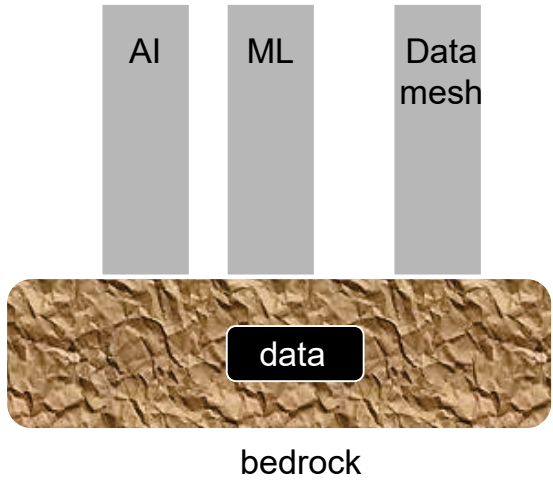


Figure 2.3: All applications must operate from a firm foundation of bedrock believable data. When the application operates on a bedrock of stable, accessible, and believable data, the application will most likely be a success. Conversely, when the application sits on unbelievable data, the application will definitely fail.

The elements of believable data

So what are the elements of a bedrock foundation? There are, in fact, lots of elements that make up the bedrock foundation of believable data. These elements can be found in the data lakehouse. These elements include:

- **Accurate**. This is the most basic element. If data is not accurate, then it is not useful
- **Complete**. The data supporting an application must be as complete as is warranted.

- **Timely**. When the analyst goes to look at data, the analyst assumes that the data being used is the latest version of the data. Looking at out-of-date data can be very misleading.

- **Accessible**. To be useful, data must be accessible. Some data must be accurate up to the second. Other data has a more relaxed parameter of usability. But the timeliness of data is an important factor in its usefulness.

- **Integrateable**. Data must exist in a state where the data fits with and can be meaningfully combined with other data. There are differing degrees of integration. Some data simply cannot be integrated with other data. But most data can indeed be integrated with other data. The ability to access integrated data is essential to the usefulness and believability of the data.

- **Malleable**. Data is like putty. To be useful, it needs to be able to be shaped. The data lakehouse serves these needs through:

    - **Granularity**. Granular data is able to be examined in many ways. The less granular the data, the less valuable the data becomes to the person doing the analysis

- **Metadata-enhanced**. Raw data is almost useless data. Metadata is needed to allow the end user to see what to analyze.
- **Documentation**. In addition to metadata, well-documented data is clear and concise. The end user knows what he/she is dealing with to analyze and use data.
- **Diversity**. The bedrock foundation serves a wide variety of data types and data structures.

Summary

The foundation of believable data, when properly created, sets the stage for success with applications of data. When built properly, the data lakehouse suits these needs.

CHAPTER 3

How to Avoid Bad Data

There are many ways that data can get to be bad. Most of the time, this takes place when we first enter data into a system. Ensuring this step goes correctly is vital for safeguarding acceptable data quality. Other times data goes bad due to incompatibility issues. Having no documentation can also be very detrimental to the quality of the data. Thus, we need to document the data when we first get the data.

Data quality issues arise in both structured and unstructured data. Structured data are considered standard formats, such as text, numbers, or dates. Unstructured data does not reside in a relational database. Some examples would be images, audio, or geospatial data. Some ways to ensure data quality for unstructured data would be to use software that will adequately store the data. There is a potential of collecting unnecessary data when it comes to unstructured data (i.e., real-time data), so incorporating a system to ensure only relevant data remains is important. When it comes to lots of data flowing into the database, which is relevant to both

unstructured and structured data, ensuring data quality in real time is essential.

Entry errors

Entry errors are often due to the person inputting data incorrectly into the system or due to an error already present in the document. Mistakes that are in the document could be a result of issues such as transcription or handwriting errors. We must devote time to ensure the transcription is as true to the source as possible. Many transcription tools are available, each with unique strengths and weaknesses. To minimize errors during this step, having a person read the final transcription before entering it into the system may be worthwhile. A person can also check their work to minimize handwriting errors.

Having formats available for applicable fields may help avoid these entry errors. This is called an *input mask*, which offers different formats for values. This would notify the person inserting the data that a particular format is required. For instance, an input mask can indicate a series of characters needed when adding values to the field, including placeholders and additional characters comprising parentheses and hyphens. These input masks help to prompt the data entry person of the particular format each field needs. Such as the following:

Field	Input Mask (indicates different formats that are needed)
International Phone Number	_ _ - (_ _ _) _ _ _ -_ _ _ _
US Zip Code	_ _ _ _ _ - _ _ _ _
Date	_ _ - _ _ - _ _ _ _
Social Security Number	_ _ _ - _ _ - _ _ _ _

Table 3.1: Input mask examples.

Also, indicating a particular data type needed in a field could avoid entry errors. Many different data types and precision lengths can be indicated for each field. Many data types will exist, or you may create your own data types. Some data types include the following:

Data Type	Definition
INT	Integer
VARCHAR	Variable characters with texts and numbers
NUMBER	Numeric values
Yes/No	A binary value is needed
DATE	Indicates a day, month, and year, such as 3/19/2023 for the US
MEMO	A large amount of text

Table 3.1: Various data types.

Key incompatibility

Additional checks are needed when inserting data to avoid entering data that conflicts with what is already in the

system, which can be due to data integration errors. A poor data connection could be the culprit. This connection or link between the form for entering data and the data source could lead to key incompatibility or attribute inconsistency. See Figure 3.1 below for examples of key incompatibility or attribute inconsistency issues.

For key incompatibility issues, it is possible that the primary key, or unique identifier field, may contain duplicate primary keys when additional data is entered into the system. Duplicate primary key fields are not allowed in a database, which would lead to an error. Or there may be a different primary key in the system than the record or row we are trying to insert. This could cause duplication issues that would also create data errors in the system. Concerning attribute inconsistency issues, integration errors could occur if the data for a field is required but is missing or in a different format when entering new data into the system.

Duplicate records

Duplicate records, or adding the same information more than once, often occur when data is transferred to another system. Duplication is not just limited to the primary key. Not knowing the most reliable data can cause a loss of confidence in the data that you have. For instance, making

critical decisions for your business would be challenging if you have difficulty determining even how many customers you have due to duplication. In this scenario, knowing who you are marketing to and how productive your company is would be challenging.

Misspellings

Misspellings could also be an issue when integrating data. In this situation, it is difficult to determine which data is correct. For example, is the person's name "Mary" or "Marie"? The stakes grow higher based on the particular data in question, especially if the data is starkly different. The integration process, which involves entering records into a system, can slow down if someone needs to manually confirm that the data is correct. Taking advantage of formatting the data and restricting the data to specific data types helps mitigate these errors.

Compatibility

Various incompatibility issues could lead to bad data (language, distillation, and context incompatibility). We will get to these, but first, what is compatibility? This ensures that your system will appropriately interface with an upgrade or another system's data.

Context incompatibility could also be an issue when integrating multiple data sources. Contextual data is any relevant facts surrounding the situation at hand. For example, marketing data could involve other information such as customer data, social media interactions, economic changes like stock market prices, and environmental changes like seasons and the weather. Different contextual data must be synthesized together, especially today, where there are many opportunities to obtain relevant data, such as through GPS tracking or Internet of Things (IoT) devices, like wearable devices that track a person's data in real time. If it is not possible to connect the disparate data sets, then contextual incompatibility results, leaving gaps in some areas that can be considered for achieving a holistic view of the marketing strategy's success or failure.

Distillation incompatibility is also a concern. First, what is distillation? Knowledge distillation compresses a larger model into something smaller that can simulate the real world. Typically, the larger model trains the smaller model offline, online, or through self-distillation.

In the most common offline distillation, many neural network models often train the smaller model. Neural networks mimic the neurons in the brain and are trained by processing examples. In online distillation, also called *parallel computing*, the larger and smaller model process data simultaneously. During self-distillation, the larger and smaller models are the same and train each other.

Deep learning is a part of knowledge distillation and includes speech and image recognition. Deep learning processes data similarly to how the human brain works and offers insights based on patterns in speech, images, and so on. If there are any compatibility issues during processing, then knowledge distillation will not be successful.

If a current feature works differently after integrating data, this would be considered language incompatible. Often, you can check to see the compatibility level of a database, which may help avoid language incompatibility.

Documentation

Not keeping documentation can be another data quality issue. If we do not accurately document data, then much time will result in searching for the data you need. Often, people refer to lots of unorganized data, where things are hard to find, such as a *data swamp*. A data swamp usually consists of a collection of random data with no structure or quality rules. To avoid a data swamp, collecting and documenting data that is only relevant to your business is important. If you have a lot of data, you may want to work towards creating a data lake that can store and process large quantities of structured and unstructured data. An advantage of using a data lake over a data warehouse is its

lower costs due to its capability of storing most of the data in raw form.

No matter the size of the data, keeping detailed documentation is an essential step for maintaining data accuracy. Not having data documentation is a recipe for disaster due to not knowing the meaning or purpose behind the data you are storing. Properly documented data can be easily used and understood by all team members.

For instance, a data dictionary could help to minimize many data quality issues. A data dictionary is a collection of information about the data you are using. It often offers metadata or data about your data. There are many metadata standards available to use as guidelines to follow when logging the data. We can also include meanings and interpretations in the data documentation.

Most importantly, the documentation often contains rules for using the data. The best practice is to collect information about the data when you first receive it. This will provide a good reference point for future team members who will interact with the data.

Summary

Data quality is often deemed an uncontrollable phenomenon. Yet through reviewing entry errors, key incompatibility, duplicate records, misspellings, and ensuring good documentation, we can minimize and prevent bad data. Data quality is a core metric for success in data teams, so we must take the causes and solutions seriously.

Some evaluation criteria for data quality include accuracy, completeness, reliability, relevance, and timeliness. For accuracy, some questions would be whether the data is valid and valuable. Completeness involves whether there are potential gaps in the data. Reliability is whether we can trust the data. Relevance is its applicability to the business' needs. Timeliness involves the data being able to make up-to-date decisions.[1]

---

[1] https://www.techrepublic.com/article/tips-to-improve-data-quality-for-unstructured-data/.

CHAPTER 4

Different Types of Data

The data in the organization can be divided into three basic types: structured data, textual data, and machine-generated analog/IoT data.

Data volume across the types of data

There are many different aspects to the different kinds of data found in the organization. One of the most important is data volume. The volume of data to handle greatly impacts the technology that stores and manages the data.

There is a large difference in the volume of data found in each of the different types of data.

There is only a relatively small amount of data that is structured data. Structured data is transaction-based data that is a by-product of doing day-to-day business in the organization. Textual data is found in many places, including in contracts, emails, telephone conversations,

medical records, and many other places. Analog/IoT data is data generated by machines, including electronic eyes, drones, wristwatches, alarm clocks, vehicles, and many other devices. Machine-generated exists in many different places.

When you look at the volumes of data associated with each of the different types of data, it is seen that compared to textual data, there is a relatively small amount of structured data. Stated differently, there is much more textual data than there is structured data. The same ratio is found comparing text to analog data. There is far more machine-generated data than there is textual data.

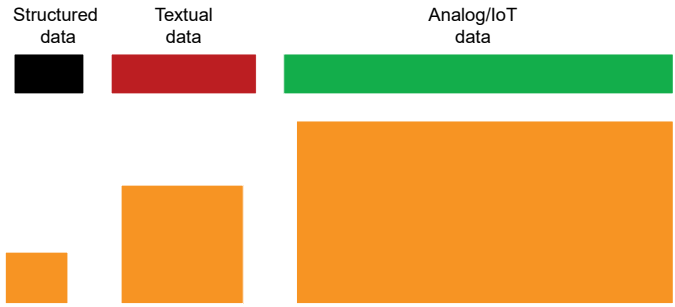


Figure 4-1: The volume of data found in the corporation.

The business value of data

Certainly, the volume of data differential is important in the data lakehouse. But it is only one of the factors we

need to consider. Another important factor is the business value of the data being captured and stored in the data lakehouse. Just because there is a lot of data does not mean that there is business value throughout the data. Stated differently, some data contains a great deal of business value, and other data does not contain much business value at all.

In the following figure, the business value of the different types of data appear in red. Nearly all structured data contains business value. Some structured data contains no business value, but not much.

When it comes to text, some text contains business value. But there is also a significant portion of text that does not contain any business value. When a person writes an email saying they want to purchase a product, there is great business value. But when a person writes an email confirming a dinner time with a friend, there is no business value.

When it comes to machine-generated data, there is a tremendous amount of business value in only a small fraction of the data. Most of the data generated by a machine is rote data that reflects little or no business value. For example, a lathe generates data about its process. For three months, the lathe operates properly. The data generated during those three months is unremarkable. But one day, the lathe develops a problem. Once the problem

arises, the lathe does not operate properly. For this one day, the lathe generates very remarkable data. The percentage of useful data generated by the lathe is very low.

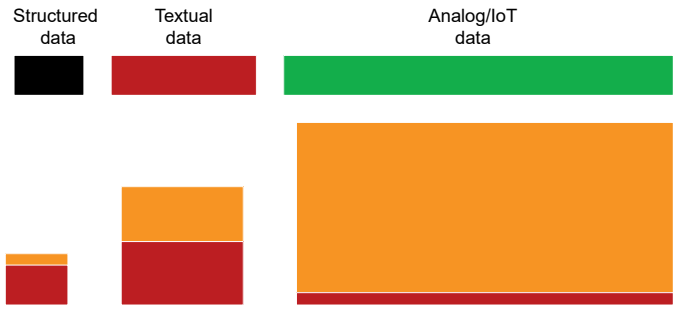


Figure 4-2: The business value found in each segment of data type.

Probability of access of data

In conjunction with the business value of data spread across the different types of data is the probability of access of the data. The probability of access of data in the data lakehouse corresponds exactly to the business value of the data.

The probability of data access occurs where there is business value.

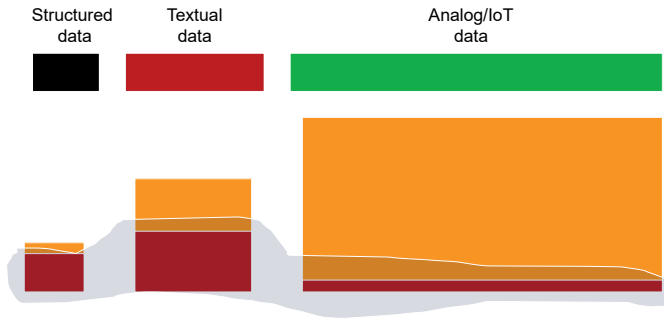


Figure 4-3: The probability of access of the data corresponds directly to the business value of the data found in each data type.

From many different perspectives, it makes no sense to keep data that will not be accessed in the same storage location as data with a high probability of access. It is expensive to do so. It is wasteful of machine cycles to store the different types of data together. And storing the different types of data together disrupts the ability of the analyst to do cogent analytical processing.

In a word, it simply is not a good practice to store data with a high probability of access with data that does not have a high probability of access in the data lakehouse.

Whenever data is separated by probability of access, we need to consider whether there would be a need to search the data placed into bulk storage. In other words, when I put data into bulk storage, does that mean that, in the future, when there are unknown requirements that will arise, can I go back and find and analyze the data that has been placed in bulk storage?

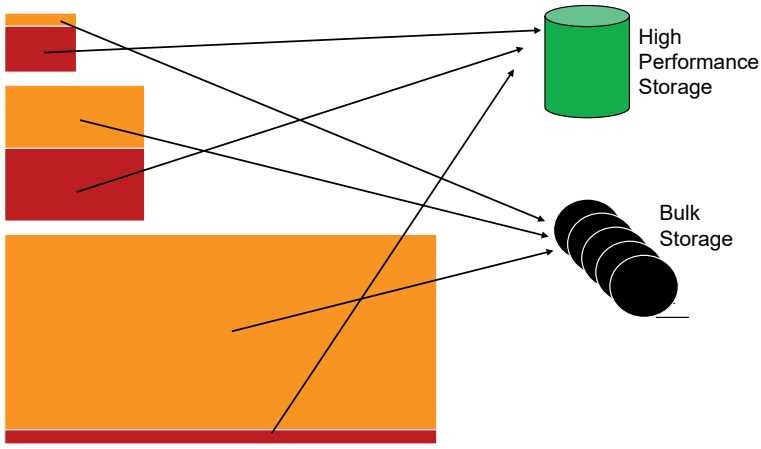


Figure 4-4: We place data with high business value in high-performance storage and data with low business value in bulk storage.

Even though data has been placed in bulk storage, it does not mean we can examine it in the future. Of course, it is not as easy or efficient to retrieve and analyze data in bulk storage as it is in high-performance storage. But it can still be done. And as long as going back and analyzing data in bulk storage is not done regularly, there is no problem.

Once the desired data is found in bulk storage, it is easy to place that data in high-performance storage.

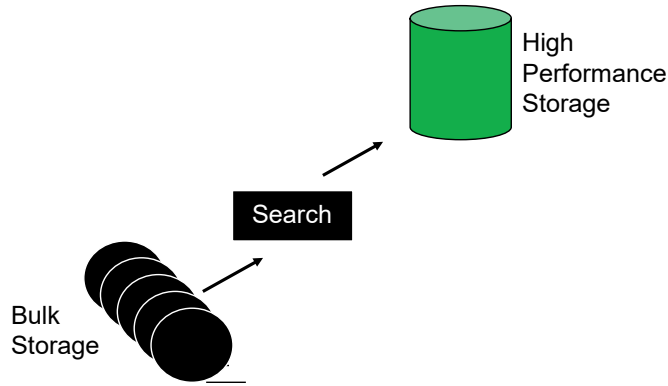


Figure 4-5: If unknown future requirements arise, we can examine bulk storage.

Data degradation

One factor to consider when placing any type of data in high-performance storage, is that the probability of data access degrades over time. Stated differently, the older that data gets, the less likely that data is relevant to solving today's problems.

The degradation of data over time is true for all types of data.

Bulk storage as an archival mechanism

Because the probability of access and the usefulness of data degrades over time, it is necessary to augment high-performance storage with bulk storage, where bulk storage is used as a means to archive data.

Once put into archival storage, the archive can always be retrieved if necessary. But if archiving is processed properly, there should be very few occasions where a search of archival data is necessary.

Summary

The different types of storage carry with them their own unique traits that greatly affect the way the data is stored and used in the data lakehouse.

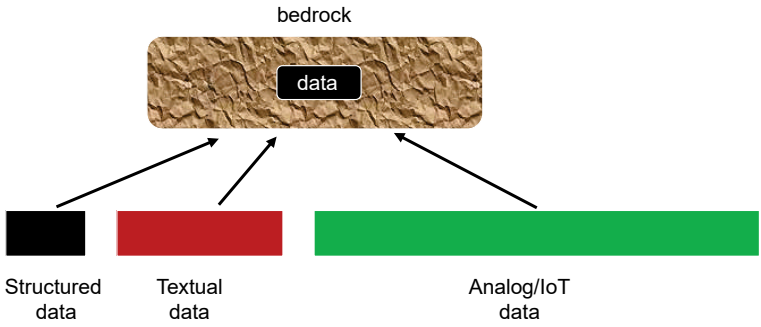


Figure 4-6: The three different forms of data found in the bedrock.

CHAPTER 5

Abstraction of Data

Abstraction of data is a very useful and needed method of dealing with large amounts of complex data. Humans use abstraction to deal with volume and complexity in everyday life of many things.

When a doctor refers to medication, he/she is abstractly referring to the many medications a patient may be required to take. When an executive refers to assets, he/she refers to the money the organization has available, the intellectual property the organization holds, the buildings and offices the organization owns, and so forth. When a construction manager refers to building equipment, the construction manager refers to the hammers, saws, and drills the construction company has at its disposal.

It is much simpler to refer to objects abstractly than it is to refer to calling out each object by itself. Data architects need to use abstraction as a tool to deal with the number of objects found in data and the complexity of those objects.

Each of the different types of data has their own modes and means of abstraction. Structured data is abstracted by means of data models. Textual data is abstracted by means of ontologies and taxonomies. Analog/IoT data is abstracted by means of distillation algorithms.

Data models

The data model used for structured data contains an entity relationship diagram (ERD), a data item set (DIS), and a database schema.

The ERD identifies the major subject areas of the organization and the relationship between those entities. For example, a typical ERD might include customer, product, order, and shipment as entities.

The data item set expands the entity into its component parts. The key, attributes, and dependent data for a given entity exist in the dis.

The database schema mirrors the DIS and describes such features as physical attributes, indexes, and unique key values for the data. In many ways, the database schema merely takes the DIS and adds details to the data depicted in the DIS.

The different components of the data model interconnect with each other. There is a DIS for every entity found in the ERD. And there is a database schema for every DIS.

To understand this interconnection, consider some different maps. A globe shows all the oceans and countries in the world. Then there is a map of the state of Texas. Then there is the map of the streets in Dallas. Each of these maps has its own unique levels of detail. And each of these maps relates to the other. So you can find the state of Texas on a globe. And you can find the city of Dallas on a map of Texas.

The data represented in the data model normally does not contain derived or summarized data. To that end, the data model contains only granular data. The data model normally represents the internal data of the organization. Normally the data model does not represent the data external to the organization. The data model is based on data that can be altered if needed.

The elements of the data model provide the context for the data residing inside the data model. For example, when the data model specifies the attribute NAME, the only data residing in the space allocated for NAME is, in fact, the name of an individual or a company. You would never put a telephone number or Social Security Number in the space the data model has allocated for NAME.

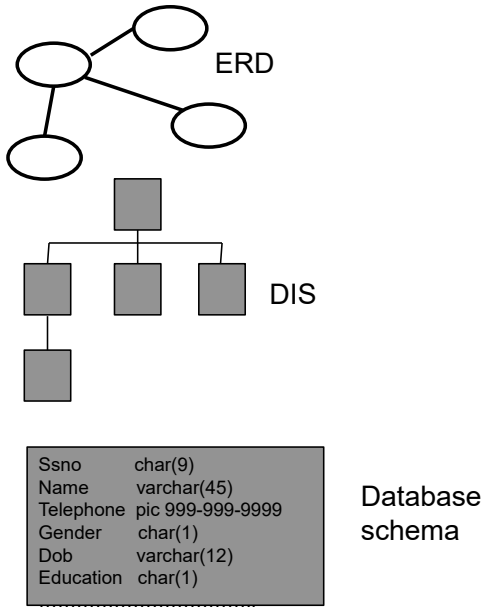


Figure 5.1: The abstraction of structured data.

Ontologies and taxonomies

The abstraction of text has some similarities to the data model and some dissimilarities to the data model. The two major components of the abstraction of text are ontology and taxonomy.

An ontology is a group of related taxonomies, and a taxonomy is a classification of like objects.

For example, consider an ontology of countries, states, and cities. Each taxonomy that resides inside the ontology is unique unto itself. But there is a relationship between the elements of each taxonomy to the elements residing in another taxonomy. For example, the city of Paris resides in France. The state of Texas resides in the United States.

A taxonomy is a classification of like objects. For example, a tree may be an elm tree, a pecan tree, a pine tree, and so forth. Each of the elements residing in the taxonomy has the same relationship to the general value of the taxonomy. For example, if you had a tree taxonomy, you would not put a Porsche or a Sherman tank in the classification of being a tree.

Taxonomies are a depiction of the external world. In this regard, taxonomies are quite different from the data model. And taxonomies are an abstraction of text that does not change. For example, when a person writes and then delivers a speech, there is no opportunity to go to the text of the speech and modify what was said.

There are different types of ontologies. Some ontologies are generic to any subject. The words "I like…" or "I love…" can be used in any type of text. A second type of ontology is specific to a discipline. Doctors have medical terms. Lawyers have legal terms. Construction workers have construction terms. The ontology for a doctor would not suit the purposes for a lawyer, for example.

A third type of ontology is a specific termed ontology that is specific to an organization. For example, an oil company has terms used only in the oil company and nowhere else.

Ontologies are infinite for all practical purposes. When an ontology is built, it is built to suit the needs of a select community. It is recognized that the elements of the ontology can be expanded indefinitely.

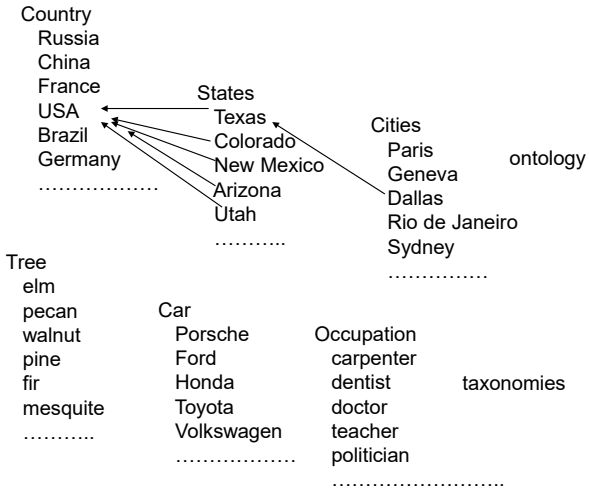


Figure 5.2: The abstraction of text.

Distillation algorithms

The abstraction of analog/IoT data differs greatly from the abstraction of structured or text data. Because of the disparity between business value and non-business value

found in analog data, it is necessary to have an algorithm (or algorithms) to distill the useful analog data from the unuseful analog data.

The distillation algorithm abstracts Analog/IoT data.

The distillation algorithm can take many forms, depending on the analog data and the ultimate business value. And, of course, the algorithm can be altered over time as conditions change.

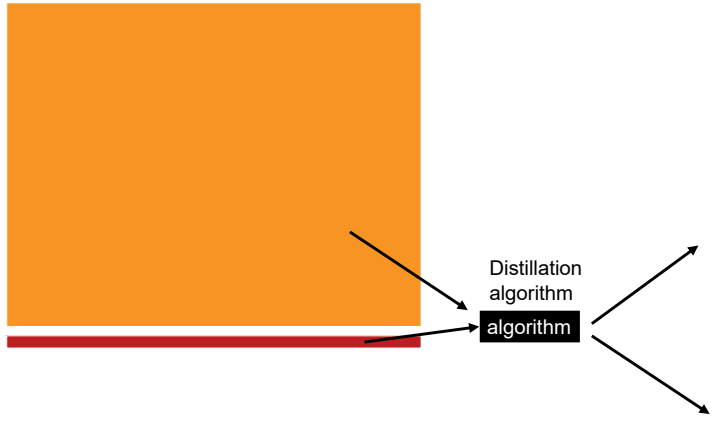


Figure 5.3: The abstraction of analog/IoT data.

Summary

At first glance, it may seem that the data model and the ontology are the same thing. However, there are some

important stark differences between the two types of abstraction. The data model is inward facing, looking at the internal workings of the organization. The ontology is external facing, depicting the external world. The data model contains explicit metadata. The ontology contains implicit metadata. The data that the data model depicts is able to be altered, if necessary. The text that the ontology abstracts is not able to be changed. (It may even be illegal to go back and change text.)

The data used for the data model is finite. However, the text and the portrayal of the external world that the text operates on are not finite. The external world can go on forever. So, there are indeed differences between the data model and the ontology, even though there are similarities as well.

There are also many differences between the data model, ontology, and distillation algorithm. The data model and ontology are abstractions of data, while the distillation algorithm is a description of processing.

There is another type of important abstraction that occurs in the foundation of data. That abstraction is the abstraction of the lineage of data as data moves through the processes of the organization.

Typically data is captured as part of a transaction. Once captured, the data is gathered with other like data. Then the data begins a journey going from one process and one

data to another. At each point of departure, the data is combined, calculated, and merged with other data.

Finally, the data reaches the point where it is used for analytical processing. The analyst needs to know about the entire journey the data has traveled to be able to do analytic processing successfully. The calculations, summarizations, and mergers are all necessary for an analyst to do his/her analysis properly.

The abstractions of data belong in the bedrock of data, the data lakehouse, that support the many applications that rely on corporate data.

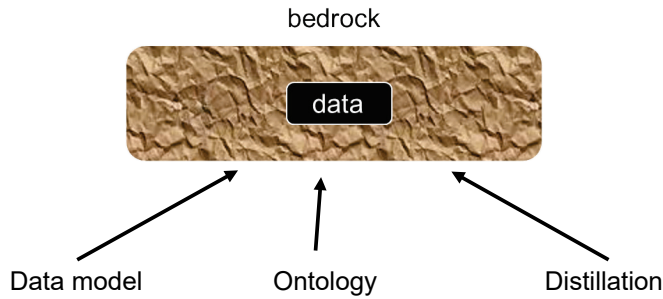


Figure 5.4: The abstractions of the different types of data belong in the bedrock as well.

The proper abstractions of data make the data lakehouse usable and comprehensible for all parties needing to access and use the data.

CHAPTER 6

The Structured Environment

The structured environment is the portion of the bedrock data foundation (the data lakehouse) that is most familiar. The structured environment was the first environment to make its appearance in technology.

We call the structured environment "structured" because the structure of each record is identical. There will be different contents in each record, but the basic layout of the data is identical.

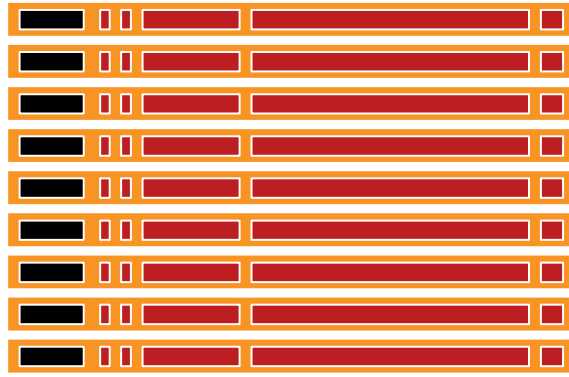


Figure 6.1: The structure of each record is the same.

For a variety of reasons, the structured data environment is optimized for computer processing. The structure of the computer works in an optimal manner on structured data. As shall be seen, the computer is not optimized for working in other environments.

Transaction-based data

The result of a business activity, usually a business transaction, creates a structured record. Typically, someone makes a purchase. Or someone makes a phone call. Or someone creates an invoice. Some business occurrence occurs that triggers the creation of a structured record.

Many business activities generate a structured record. The structured record acts as a log of the business activities.

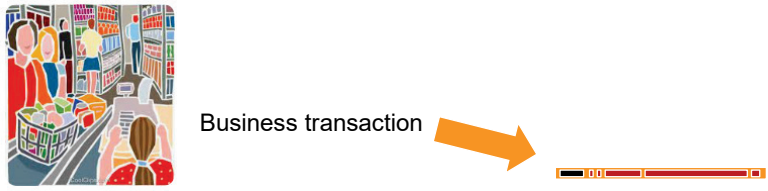


Figure 6.2: Usually, a business transaction generates a structured record.

Structured records

There are some basic elements found in every structured environment. The most basic structural element is the record. The structured record holds a collection of data, including one or more keys of data, several attributes of data, and other data. Typically, the system uses one or more indexes of data to point directly to the record. As an example of a structured record, the record might contain:

- Part number – key
- Part description – attribute
- Unit of measure – attribute
- Packaging – attribute
- Weight – attribute

And so forth.

A key might be a Social Security Number, a part number, a passport number, and so forth. An attribute might be a person's name, gender, and date of birth. In general, an attribute's values depend upon a key's values.

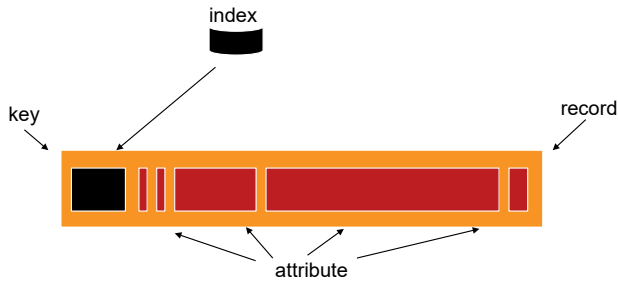


Figure 6.3: The basic elements of structured data.

The structured records have a rigid design. For example, you would never find a telephone number in the space allocated to an attribute named Gender. As such, the context of the structured record of data is embedded in a very solid way. In fact, the context of data gives the structure to structured data.

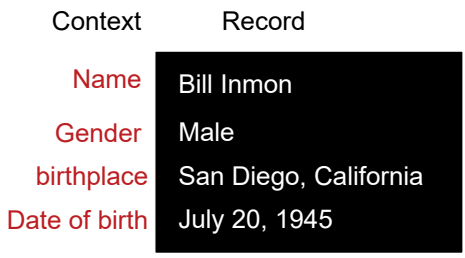


Figure 6.4: The context that describes the attributes of the data found in structured data is carried with each record.

Keys

The keys found in a structured record can either be unique or non-unique. For example, a unique key for a record might be Social Security Number. And the same record may have a key for gender, which of course, would be non-unique.

The key takes the form of an index in a database, which allows data to be accessed directly and efficiently. Without an index, the computer would have to embark on a slow and clumsy sequential search of all the data in the

database when searching for a particular record. The index on the structured record allows the record to be accessed directly in the database.

Once the records accumulate inside the computer, the structured records are placed into a table or a database. The table or database resides on disk storage outside of the computer. But the disk storage is attached directly to the computer and allows the computer to have direct access to the data.

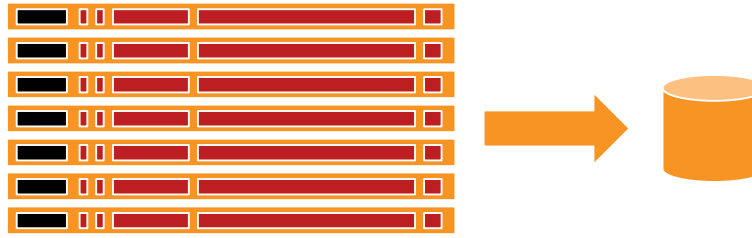


Figure 6.5: The structured data is placed inside a table or database.

Once the rows of data are placed in a database, or more accurately, a table within a database, the rows can be accessed directly.

An essential part of the bedrock data that encompasses the structured environment is the abstraction of the structured data in the form of a data model. The data model needs to be available, as well as the actual structured data.

Online transaction processing

One of the major values of accessing structured data directly in a database is the opportunity to do online transaction processing (OLTP). OLTP processing allows the organization to perform interactive processing. Interactive processing allows the business to incorporate the computer directly into their day-to-day activities. Standard business facilities, such as ATM processing, bank teller processing, and reservation systems, all depend on the ability to do online processing.

Central to online processing is the ability to run transactions quickly and consistently. In addition, system availability becomes a factor. The organization cannot conduct business if the system goes down. The mantra that online systems run by is that the system runs at the speed of the slowest transaction. The mantra dictates that each transaction uses a minimal amount of resources.

The net result is that the computer alleviates the business from doing tedious work. And OLTP provides an important facility for businesses.

The OLTP environment brings some other system requirements not found elsewhere. For example, with OLTP there is a need for expedient backup and recovery processing. Should a transaction fail in the middle of the

OLTP job stream, it becomes very important that the failed transaction does not corrupt the data the system uses.

For example, if a bank deposit fails in the middle of the day, the failure of the transaction must not cost the bank or the customer any money. If the banking transaction fails in the middle of a transaction, the truncation may have erroneously moved money from one account to another. For this reason, the ability to backup and recover online transaction processing becomes an absolute necessity.

In addition, when transactions can update data in an online mode, it becomes necessary to protect one transaction from trying to update the same data at the same time as another transaction is trying to update that data.

Collisions of transactions against the same data at the same time can destroy the integrity of the data.

Enterprise data

Although applications are useful and critical to how we do business, the same data element can appear in different places with different values. For example, data element ABC has a value of 13 in one place and a value of 1000

elsewhere. Trying to make decisions on this confusion of data became impossible.

The lack of data integrity does not allow organizations to look across all of their data, and therefore, there is no enterprise view of data. The organization can not answer such questions as:

- Across the enterprise, how many customers do I have?
- Across the enterprise, how many different sales did I make?
- Across the enterprise, how many products and services do I have?

The data warehouse requires that data from applications be integrated into an enterprise understanding of data, thereby improving data integrity.

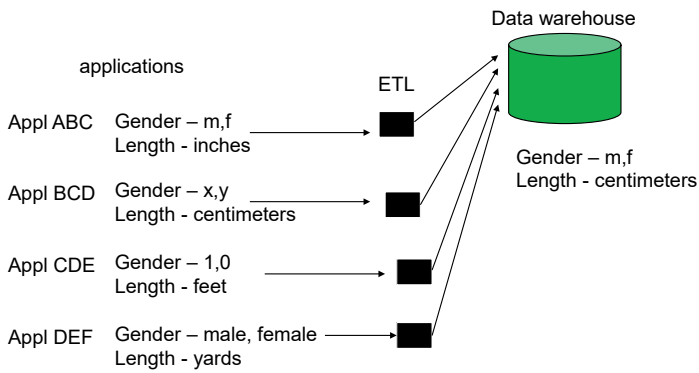


Figure 6.6: The transformation of application data to enterprise data.

In this example, it is seen that there are four applications. Each of the applications shares common data like gender and the length of some product. However, each of the applications has its own interpretation of the data. In one application, gender is specified as m,f. In another application, gender is 1,0. And so forth. And the measurement of the product in one application is in terms of inches, while in another, it is in terms of feet.

To achieve an enterprise understanding of the data, it is necessary to transform the data into a single format using Extract, Transform, and Load (ETL). In fact, the transforms that need to be done to achieve enterprise data are much more complex and numerous than those simple ones shown in the diagram.

Summary

The following figure shows that structured data becomes one of the foundations of the bedrock of data.

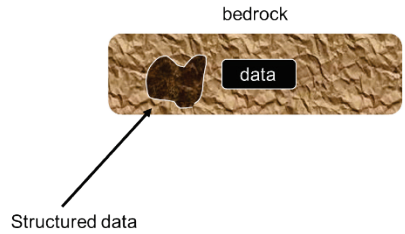


Figure 6.7: Structured data becomes the first of the types of data that belong in the data lakehouse.

CHAPTER 7

Textual Data

Textual data is one of the most useful types of data we do not sufficiently use today. It is not used for many reasons we will explain in detail, but the main reason is that using textual data is difficult.

Types of textual data

Textual data comes in many forms, including text documents, Word documents, PDF documents, HTML pages, JSON documents, XML documents, manuals, emails, texts, comments, blogs, and blocks of text in Excel cells.

Figure 7.1: Text can come to you in many formats natively.

You can also transcribe textual data from vocal sources such as audio files, phone conversations, radio shows,

recorded meetings, and doctor visits. These vocal sources are converted to text using conversion software like Dragon Speak, Google Speech-To-Text, Cortana, or transcription services. You can even convert pictures into text using optical character recognition (OCR) software. While it is true that many transcription software products and OCR products are not perfect, they do a decent job of turning audio files and picture files into usable textual data.

Figure 7.2: Text data can be converted from other formats.

Textual data can be ambiguous. Often, it does not have any metadata to tell us what that data means. Textual data rarely has integrity or a pre-defined form. To confuse people even more, most textual data companies store it intermixed with numerical data, pictures, and minutia.

What makes textual data valuable is when you extract specific useful pieces of data and add context to that data so that it can be interpreted by categorization, graphs, or correlations.

Consider a PowerPoint presentation, a PDF, or a business Word document. You have textual data that we want to be able to find and utilize. On the same page, we have a graph and a partial spreadsheet. How do you characterize and represent the graph and the cells of the table within that page, along with the textual data? Are the picture and cell data textual data, or are the table and graph names all we need when characterizing it? If we do not include the graph or table, have we lost any data of importance, considering that that data is already in our structured database? Unfortunately, most projects never begin because managers do not know how to answer these questions.

Language barriers in utilizing textual data

Textual data is also exceedingly difficult to work with because its format is not straightforward. To begin with, there are no less than seven major sets of characters that make up words and textual data in the world. These sets include the American Alphabet, multiple Arabic character sets, Russian letters, and Hebrew characters.

On the earth today, there are more than 7,100 languages spoken. Of these, there are 23 main languages, and about 80% of the world speaks one of ten languages. Since each language has its own idioms, idiosyncrasies, and formats,

attempting to work with all these languages becomes a monumental task.

Another reason that textual data is not currently used is that most people do not know how to speak their language well. They use shortcuts, slang, imperfect verbs, nouns, and adjectives. Most people also do not use proper punctuation, especially in text. How can you extract data from imperfect sources?

Words have different meanings

Another rationale managers create for not using their valuable text data is that each word can have multiple meanings based on the words around it. For example, if someone uses the word "trust" in a sentence, that word takes on completely different meanings based on the context of the use of the word. We all know that "trust" may mean believing in another person, but "trust" in a financial sentence might mean a vehicle to store money and assets to be divided with another person or organization according to rules. Using "trust" in a computer-based sentence means a relationship in which two computers are allowed to share specific data. The same word means completely different things based on the other words in the sentence.

Extracting business meaning

Most organizations do not see the benefits of working with the massive amount of textual data they currently have because they do not see a pathway to get started. The problem with this myopic mindset is that most of your questions can be answered by properly dissecting, understanding, and using the textual data stored throughout the organization. You need to take the first step in your thousand-mile journey!

Adding context and metadata to specific text will allow you to better understand your customers, products, documents, and valuable data.

There are two main ways to extract meaning from textual data: 1) manually; and 2) Textual ETL. Manual extraction requires training and understanding by each of the employees doing the work. It is tedious and will vary by the human doing the work. Other organizations will not be able to get the desired context from your text because they rarely know your industry and its organizations as well as your employees.

Using an automated solution, such as a Textual Extract, Transform, and Load (Textual ETL) procedure, will require some upfront decisions and subject matter expertise. It will

require some training data and evaluation, but it will consistently return the same results time after time. An automated Textual ETL process will also return the results in the overall least expensive, quickest, most repeatable, and least biased way. This automated procedure can be done in the background as documents and comments are collected by your organization.

A few caveats should be understood about textual data extraction.

- **It is not perfect**. No people or process will return perfect results because language is not perfect. Different dialects, slang, times, cultures, and people can produce different meanings from the exact same sentence. Sarcasm, emphasis, word choice, and metaphor can wreak havoc on language and interpretations.

- **Sentiment returns feelings about a product or situation**. Sentiment is not an exact science. If a person can return 80% correct sentiment results from a written sentence, you have a competent person in that language. With true sentiment analysis, it is not about just one sentence. Rather, it is about the feelings from the thousands of reviews of one product.

- **Natural language processing (NLP) attempts to tag words relative to other words within**

sentences. NLP serves many purposes in theory. It evaluates grammar. It attempts to discover sentiment. It predicts words that are missing from sentences. It completes sentences based on similar sentences that it has evaluated. NLP does all of this if it has been trained on those words. In a small batch, NLP is quite effective and amazing. However, as you move to a general environment, NLP requires so much training that it becomes extremely resource intensive and expensive. NLP as a science, is not so much about text extraction as it is about textual prediction based on tagging, algorithms, human assumptions, and an exceptionally large testing dataset.

- **Textual data extraction requires context**. This requires documents to be separated into similar contextual types. This also requires a subject matter expert with some initial training data to help build a set of taxonomies, ontologies, and rules, which we will call a "nexus." The nexus is used to extract important data from textual data, which can be exported to a structured database and evaluated using traditional visualizing tools. This nexus-based approach, called Textual ETL, requires training on a minor dataset and produces more usable data per record than NLP because its purpose is different. The time and resources to

complete this type of textual data extraction are also greatly reduced without reducing the quality of results.

Summary

You must go through the three major data cleansing steps when looking at your textual data as a possible data source. You need to **E**xtract the textual data from its source and retrieve only the text. You need to **T**ransform the data into a format that can be utilized with other data. Finally, you need to **L**oad that transformed data into a structure that can be utilized to return answers to business problems.

CHAPTER 8

Analog/IoT Data

The third type of data found in the bedrock of data is machine-generated analog/IoT data. Machine-generated data is data generated by the operation of a machine. As the machine operates, telemetry data is created to measure the machine's work.

Like structured data and text, machine-generated belongs in the data lakehouse as well.

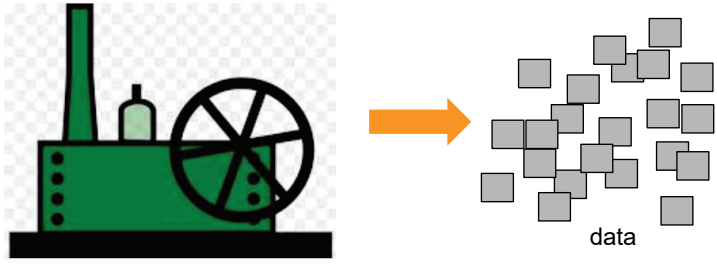


Figure 8.1: Typically, machine-generated data is cranked out by the machine while the machine is in operation.

Once the machine generates the data, it is offloaded to some storage mechanism. The machine has many ways to generate the data, such as by an electric eye, heat sensor, pressure sensor, etc. Many variables are recorded when generating machine-generated data.

Many devices and machines can generate data as a by-product of their operation. Typically those devices might be:

- A manufacturing machine, such as a pump or a lathe
- An electronic eye that is doing surveillance
- A wrist watch
- A drone in the sky
- An automobile

And so forth.

The truth is that there are many, many sources of machine-generated data.

Figure 8.2: Machine-generated data comes from many sources.

A disparity in the usefulness of data

Nearly all machine-generated data has the property that once the machine generates the data, the majority of the

data is not of interest or value to the organization's business. The machine normally does not know how to distinguish between data that is useful and data that is not useful. The machine just sits there and generates data as it operates.

Most of the data that is generated can be called *dross* data. Dross data is data that is accurate and reflects the operation of the machine but is not useful to the business of the organization. The data that is not dross, however, is useful data.

An electronic eye

To understand the creation of dross data, consider the electronic eye camera that watches a parking lot. The camera scans the lot for 24 hours a day, taking pictures. No cars are in the parking lot at night, but the camera still takes pictures. During the day, the camera takes pictures of people parking their cars and driving in the lot. These daily pictures are taken and are not useful. Most daily activity is of no great consequence.

But one day, a car is broken into while parked in the lot. For the short amount of time that the car is being assaulted, the pictures taken by the electronic eye are very useful. From the standpoint of a percentage of the time that pictures are taken, the break-in pictures are not very

much. They represent less than 1% of 1% of the pictures that have been taken. But those pictures of the break-in are very important.

It does not make sense for the company to hold on to pictures that don't show any interesting activity in the parking lot. Normal day-to-day activity is not of any value. In fact, it is expensive and a waste of resources for the company to store those uninteresting pictures. But the pictures that are of interest have great value in being stored.

This pattern of machines taking pictures of a parking lot that is generally unuseful is repeated again and again in the world of machine-generated data. Nearly all machines experience this disparity in the usefulness of the data that is recorded.

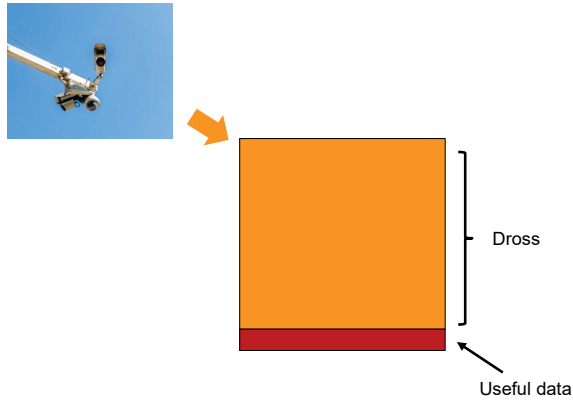


Figure 8.3: The vast majority of the data produced by machines is not useful or interesting data.

What is required to manage the volume of data that machines generate is a means of distilling the useful data from the unuseful data. There is always some means of making this recognition.

Manual scan

The crudest means of distilling useful data from unuseful data is to manually scan the data. This means of separating data always works. But it is crude. It means that someone has to watch hours and hours of video (or look at data gathered in some other way), looking for just a few seconds or a few records. Furthermore, if the person happens to miss the interesting part of the video, the interesting part may never be found.

The only good news is that this manual option, although painful and crude, is always available. But it is not recommended unless there is no other way to access and distill the machine-generated data.

Separation by date

Another crude way to separate the data is by date. For example, all the machine-generated data kept for the past

month go to archival storage. Only the most current week's worth of data is kept.

This approach is simple to implement. But it has many pitfalls. The first pitfall is that we may want to look at data older than a week. The second pitfall is that we will store enormous amounts of useless data.

However, the manual option to distill analog/IoT data is always available and is better than nothing.

Data filtering

A much more practical option is to filter out the types of data that are likely to be needed. The data that is not likely to be needed is sent to bulk storage. The data that is likely to be needed is sent to high-performance storage.

Although this option for distillation is a very good option, it depends on the fact that certain types of data will be necessary in the future. Unfortunately, it is sometimes difficult to determine what data we might need in the future. However, an experienced engineer or analyst can tell us what data they will likely need in the future.

One problem with this approach is that unneeded data will still be stored in high-performance storage. However,

that is the price to pay for dealing with machine-generated data.

All in all, this approach is better than the approaches we have already discussed.

Thresholding data

In the thresholding approach, the analyst determines the thresholds that are of interest to the analyst. A record is written whenever a measurement exceeds or falls below the threshold. In doing so, the analyst only looks at data that is of interest. The record that has been written is sent to high-performance storage. Once in high-performance storage, the record of data that has exceeded the threshold is immediately obvious. The remaining records are sent to bulk storage.

Once the threshold has been crossed, as much data as possible related to breaking the threshold that is even remotely of interest is captured and placed onto high-performance storage.

The issue with the thresholding approach is that the thresholds have to be set. If the thresholds are set too low, the analyst will miss important data. If the thresholds are set too high, the analyst will have unnecessary data placed in high-performance storage. Therefore, it is important to

set the correct thresholds. Of course, we can adjust the thresholds over time.

The thresholds approach saves lots of space and time and is the preferred approach if possible.

Time sequencing

Another approach to distilling machine-generated data is that of time sequencing data. In the time sequencing approach, we decide that certain time periods are more likely to hold interesting data than other periods.

In the case of the parking lot electronic eye, we may decide that there is no need to look at surveillance data from 8:00 pm to 5:00 am as there is normally no one in the lot then.

This approach works well if the analyst is correct in their assumptions.

It may well be that under normal conditions, there are no cars in the lot during certain hours. But on one rare occasion, someone parked their car at 3:00 am and committed a felony during that timeframe. In that case, the video of the felony would have been missed.

Summary

Once we separate the useful from the unuseful data, we place it in bulk storage. Once in bulk storage, the unuseful is still available if needed, and we get these benefits:

- We save money on the cost of storage, as bulk storage is cheaper
- It takes less time to process data
- The ease with which we analyze data is greatly enhanced.

Once the machine-generated environment is built properly, the bedrock of data is further solidified.

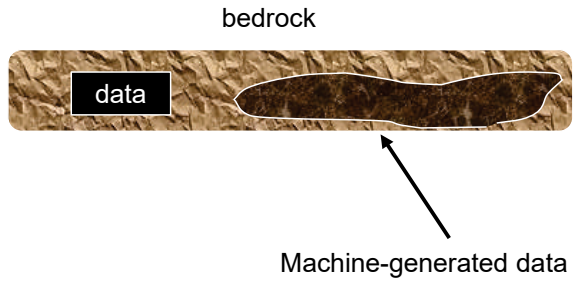


Figure 8.4: Analog/IoT data that has been distilled is the third type of data that belongs in the data warehouse.

CHAPTER 9

Bulk Storage and the Data Lakehouse

Almost as an afterthought, bulk storage appears in the bedrock of data. Bulk storage is not as prominent as high-performance storage because analytics is not normally done on bulk storage. The designer focuses most of the attention of architecture and design on high-performance storage because that is where the end-user spends their time.

However, bulk storage plays an important role in the bedrock of data because, in many ways, bulk storage allows the analyst to be free to do their work productively. Bulk storage sets the stage for the data lakehouse to be used efficiently and effectively.

Bulk storage is storage where large amounts of storage can be used to inexpensively store data. Bulk storage is not fast or efficient to access compared to high-performance storage. But bulk storage can still be accessed electronically and is a place where data can be stored over long periods of time.

In many ways, bulk storage plays the same role to high-performance storage as a relief pitcher plays in baseball. Bulk storage is absolutely necessary but does not play a role of prominence in the architecture. But when bulk storage is needed, it is very important.

Pros and cons

There are good points and bad points that all relate to bulk storage.

As a con, bulk storage cannot be accessed directly under normal circumstances. To find data in bulk storage, we access the data sequentially. The problem with sequentially storing data is that searches of the data require large amounts of time. Therefore, we should never use bulk storage to support online transaction processing. And because we can only access data sequentially, we cannot use bulk storage for standard analytical processing. Furthermore, when a search of bulk storage is required, the search normally requires a lot of custom coding. This limitation constrains the ways we can use bulk storage.

However, there are also some very real advantages to bulk storage. Even though bulk storage is awkward to access, bulk storage still stores the data in an electronic form and therefore is always available to the organization. We can

also store data for a long period of time. Bulk storage does not deteriorate over time to any great extent.

However, the real advantage of bulk storage is that it is inexpensive compared to other forms of storage. As a result, the organization can afford to store an almost endless amount of data in bulk storage. And given the volumes of data that converge on the organization, the storage cost is not a trivial subject. In addition, when cloud considerations arise, the storage cost becomes even more of an issue. Stated differently, bulk storage alleviates the storage costs for the entire organization.

With these pros and cons, bulk storage is a good place to store data with a low access probability. Many kinds of data have a low probability of access. In some cases, laws require organizations to store data for long periods, even though there is very little chance of data access. In other cases, data just grows old and stale over time.

Bulk storage is also a good place to store most machine-generated data that is unlikely to be accessed or otherwise used for analysis. When a machine is operating normally and producing normal results, the measurements taken at that moment are not of great interest.

Probability of access

When there are huge amounts of storage, the computer has to do a lot of work to find data. By placing data with a low probability of access in bulk storage, the system does not have to consider that data when searching for data that has a higher probability of access.

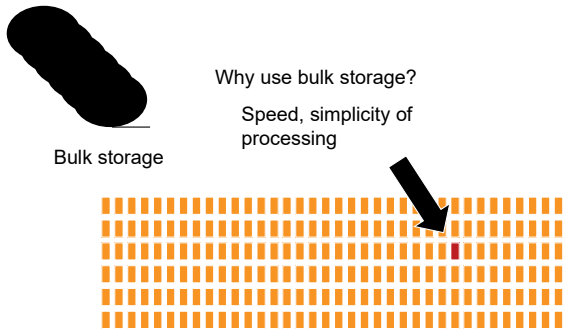


Figure 9.1: Why use bulk storage?

For all practical purposes, data with a high probability of access "hides" behind other data when there is a lot of data to be processed. By making readily available data with a high probability of access, the analyst streamlines the processing for analysis. Queries run more quickly. The cost of query processing goes down. There is great benefit in not hiding data with a high probability of access in a jungle of data with a low probability of access.

There is great benefit in separating data according to its probability of access or low probability of access.

The question then becomes, "Can the probability of access be determined before the data is used?" The answer is that the probability of access can sometimes be determined before the data is used. As an example of the ability to separate data that is useful from data that is not useful, consider text.

Storing large amounts of raw text in a database is certainly possible. But when you examine raw text, you find that only a fraction of the text is ever useful for analytical processing. To illustrate this point, consider the sentence, "She placed the aileron on the tail and the wings."

When you look at this sentence, you see that there are some words, like "aileron", "tail", and "wings" that are important. But there are a lot of other words that will never be used for any kind of analysis, like "She", "the", "on", and "and". We should only place words we will use in storage. And these words can be anticipated before considering the usage of the text.

It is possible to ascertain data that does and does not have a high probability of usage.

But the usage of words is hardly the only criteria for determining probability of access. A much more common means of determining the probability of access is the age of data. The probability of access for all data diminishes over

time. The only difference in the rate of diminishment is that some data degrades faster than other data. But over time, the probability of access of all data diminishes. As such, bulk storage is a good place to archive data when its probability of access lessens.

Incidental indexes

One feature that is sometimes useful for managing bulk data is to produce what can be termed an "incidental" index for bulk data. In normal circumstances, indexes are produced for the regular access of data in an efficient fashion. The driving factor behind the production of an index is the anticipation of high access of data.

Even though bulk data has a low probability of access, it still does not have no probability of access. So there may still be a need to create indexes for bulk data. We use an incidental index when there is a random need for accessing the bulk data. It is a "just in case" circumstance.

This type of indexing typically can be done with machines that may have idle cycles, such as off-hours, when nothing else is running. And building an incidental index may save huge amounts of time if there is a need to search bulk data.

Typically, when searching bulk data is needed, the search must be done quickly. But this is a paradox because

searches of bulk data cannot be done quickly. In such a case, an incidental index may save much gnashing of teeth.

Metadata and bulk storage

Another component of bulk data storage is the need for metadata. Just because bulk data does not have a high probability of access does not mean that bulk storage does not need metadata. Indeed, if we dump bulk data into storage without metadata, it will be very difficult to ever use and find that data again. Consequently, metadata descriptions of data are as necessary for bulk storage as they are for high-performance storage.

Summary

Even though bulk storage is not the center of attention for the bedrock of data, bulk storage is nevertheless an important and necessary component of bedrock data. Bulk storage sets the stage for and complements high-performance storage.

It is debatable whether bulk storage belongs in the data lakehouse. Certainly, the addition of bulk storage enhances the capabilities of the data lakehouse. But whether bulk

storage can be said to be part of the data lakehouse is an entirely different issue.

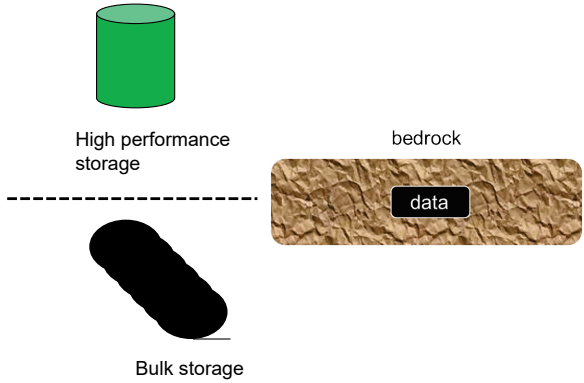


Figure 9.2: Bedrock data is housed in both high-performance storage and bulk storage.

CHAPTER 10

Data Architecture and Data Engineering

Data architecture and data engineering are the yen and yang of technology. Both disciplines work together in a complementary manner and are different sides of the same coin.

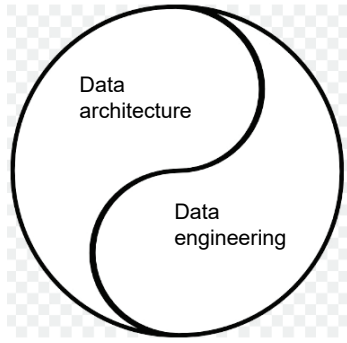


Figure 10.1: Data architecture and data engineering are the yen and yang to each other.

To understand this symbiotic relationship between the data architect and the data engineer, consider data architecture without data engineering. Data architecture without data engineering is like a boat in the ocean without a sail. The boat may not be sinking, but it isn't

going anywhere either. Data architecture by itself is a pretty pointless exercise. To realize the architecture, the data architect needs the data engineer.

On the other hand, consider data engineering without data architecture. Data engineering without data architecture is like a boat without a rudder. To keep afloat in dangerous water, a boat has to have a rudder. Otherwise, a wave can come along and sink the boat. And without a rudder, the boat will never reach its intended destination. Data engineering without data architecture is a pointless exercise.

How the roles work together

To further clarify the roles of the data architect and the data engineer, consider the construction of a house. The architect makes plans for the house: where the kitchen will be, where the bedrooms will be, where the living room will be, and so forth.

And the data engineer is the person who takes the floor plan created by the architect and turns the floor plan into a real house—with a foundation, plumbing, electricity, air conditioning and heating, and so forth.

The data architect and the data engineer work together to build complex information systems. The architect looks at

long-term considerations, such as where the house is to be placed on the lot, what rain storms and snow storms will be expected, what wear and tear will occur on each part of the house like the garage door opener, the number of people the house will hold, and so forth.

The data engineer looks at very tactical issues such as how to lay cement, where to put pipes, how to wire a house according to building codes, what the ceiling supports will be, and so forth.

Together the data architect and the data engineer complement each other and blend their skills and perspectives to create a modern information systems environment.

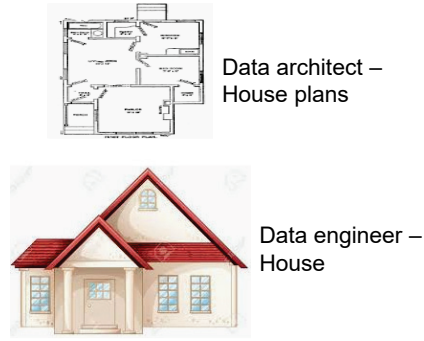


Figure 10.2: The roles of the different people.

Not surprisingly, the data architect and the data engineer have much in common. They indeed have some areas of interest that do not overlap. But there still is a lot of common territory between the two disciplines.

Roles and data types

When it comes to structured data, some of the interests that the data architect and the data engineer have in common include entities, relationships, keys, attributes, indexes, volumes of data, and so forth.

While the data architect and the data engineer have common interests, they nevertheless look for different things. For example, the data architect looks at data in the structured environment and determines if the data is:

- Defined at the highest level of modeling
- Transformed when a transformation is necessary
- Mapped with a complete lineage
- Archived properly
- Designed to accommodate large volumes of data.

The data architect is looking at the larger picture and the long-term view of the project at hand.

The data engineer looks at such things as:

- Data normalization
- Summarized and derived data
- Correct sources chosen
- Transformations well-defined.

The data engineer looks at the mundane details of how the project will be realized in terms of code, database, operating systems, and so forth.

But structured data is only one of the common grounds between the data architect and the data engineer. A second important ground of commonality is that of text.

The data architect and the data engineer share a common interest in ontologies, taxonomies, sentiment analysis, corelative analysis, language, homographs, acronyms, and so forth.

The data architect has an interest in such things as:

- The source of ontologies
- The interrelationships of taxonomies
- The overlap of taxonomies
- The hierarchical levels of the taxonomies
- The maintenance of the taxonomies.

The data architect is interested in the completeness of the ontology, the usage of bulk storage, the transformation of data into the bedrock foundation, and so forth.

The data engineer has an interest in such things as:

- The freshness of the taxonomies
- The relation of the ontology to the entities of the organization

- The completeness of the taxonomies
- The level of specificity of the taxonomies.

And so forth.

The data engineer is interested in the ETL transformation of text into a database, the database that will be used, the flow of data to and from bulk storage to high-performance storage, and so forth.

There is yet another common ground of interest between the data architect and the data engineer. That level of interest centers around the analog/IoT data found in the organization. The data architect and the data engineers are both interested in algorithms used for data distillation, the data structure and the components of the different types of data found in the analog environment, the management of bulk storage, and so forth.

Both the data architect and the data engineer have a stake in the success of the analog/IoT environment.

The data architect concerns himself/herself with such things as:

- The rate at which analog data is created
- The level of granularity of the analog data
- The business needs satisfied by the analog data
- The efficiency of the distillation algorithm.

The data architect is concerned with such things as the volume of data that will be encountered, the algorithm that will be used for distillation, the data content and structure of the data to be placed in high-performance storage, and so forth.

The data engineer looks at such things as:

- The ability to do maintenance against the distilled data
- The accuracy of the distillation algorithm
- The analytical processing passed by the distilled data
- The need to occasionally redefine the parameters of distillation.

The data engineer is concerned with such things as the actual coding of the distillation algorithm, the loading of data into both bulk storage and high-performance storage, making high-performance storage available to the end user, and so forth. Another concern of the data architect and the data engineer is the ability to track and move data across the different data types. Not all data can be used for this cross-pollination. But when data can be used across different data types, there exist great possibilities.

However, a number of issues must be addressed when data crosses from one type of data to the next. These issues are the concern of both the data architect and the data engineer.

Yet another concern of the data architect and the data engineer is data lineage. Data typically has a flow with the organization. As a rule, transformation of data occurs when we move the data. And some data is moved repeatedly. In the data flow across the enterprise, both the algorithm doing the transformation and the data selection for transformation are issues.

Summary

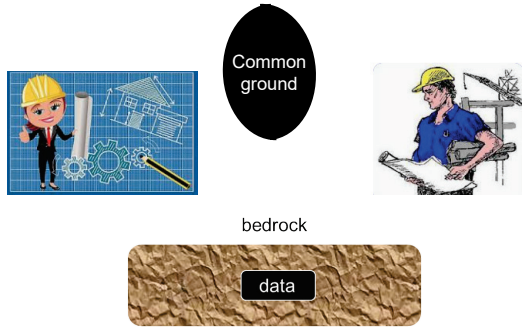


Figure 10.2: Both the architect and engineer build the bedrock.

Both disciplines work together to create the bedrock of information: the data lakehouse. The data architect and data engineer complement each other greatly. Both disciplines have a vested interest in:

- Creating a successful information environment
- Building on the work created by the other discipline.

CHAPTER 11

Business Value

The worlds of technology and business are intermixed, although, at times, it may not seem that way. Technology exists to further the goals and advances of business. And business foots the bill for technology. When technology strays from this basic equation, it withers and dies. And when technology advances the cause of business, the business prospers and so does technology. It then becomes beneficial to examine the relationship between business and technology, especially in light of the bedrock of data needed to run both business and technology.

Business value is the driver

In every case, business drives the ultimate satisfaction and value of technology. Business is the horse that pulls the cart of technology. And the best way that technology can support business is by building and maintaining a bedrock of data that the business can rely upon to make sound business decisions.

The goals of business value are simple when viewed at a high level:

- Make money
- Increase the number of customers
- Keep existing customers as customers.

Of course, there are many facets and smaller objectives to achieving these simple business goals, including:

- Introducing new products
- Advertising
- Packaging products
- Going into new marketplaces
- Pricing

It's all about money

Making money at first sounds like a simplistic and cold objective. But making money opens the door to many other aspects of business. When a business is making money, the business can:

- Expand into new market places
- Expand into new products
- Expand into new packaging

And so forth.

Making money is the key to long-term continuity and success. Money is the blood of business.

While making money is the key objective, actually making money is a complex process. The following figure shows that from a business perspective, there are many facets to achieving the simple goals of achieving business value.

Figure 11.1: Many factors must be in place to achieve business value.

In truth, all technology that is successful in the long run is focused, in one way or the other, on achieving these business goals. And creating bedrock data is the best way to lay a foundation for achieving those goals.

The bedrock of data

But achieving a solid foundation of bedrock data is a complex thing to do. We must align many moving parts to achieve the solid bedrock foundation.

The problem is that all of those components of technology must work together. Building and supporting the bedrock of corporate data is like a symphony orchestra. There are many different parts to the symphony orchestra, just as there are many different parts to the building and support of the bedrock of data. To make beautiful music, the different parts of the orchestra must be coordinated and orchestrated.

Once the bedrock foundation is in place, then the organization is in a position to support the business needs of the organization.

When you are thrashing through the complexity of the technology jungle, it is easy to forget that the ultimate goal is business value.

The difficulty of coordination

But coordinating the different technical components is difficult. This is because:

- There are a lot of pieces to the technological foundation
- Each of the pieces is very different from the other pieces

- The different pieces of technology require sequencing to work together
- The time frame required for sequencing the different parts of the technology is very different
- Different parts of the technology work at different speeds
- Iterative development of the coordination is required.

In short, what is required is a grand master—a ringmaster of a three-ringed circus.

Fiefdoms

Because of all of this complexity, confusion, and iterative processing, it is normal for the technology components to start to develop small fiefdoms. Soon one part of the technological landscape starts to disengage from other parts of the landscape. Then another part of the technology landscape disengages, and so forth. Each fiefdom sees itself as its own separate environment, not as a part of a whole.

And as the technological component disengages from the community, it starts to think of building its own technology for technology's sake. And what is lost is the focus on business value. At this point, the technological component can't even relate to business processes. Instead,

the entire focus is on the intricacies of the technology for the sake of the technology.

When technology components start to build their own fiefdom, the vision of supporting the organization's business becomes lost.

Summary

What is needed is for every technology component to refocus on building and supporting the bedrock of data that ultimately supports the business. In doing so, the technology of the organization becomes focused on supporting the business of the organization.

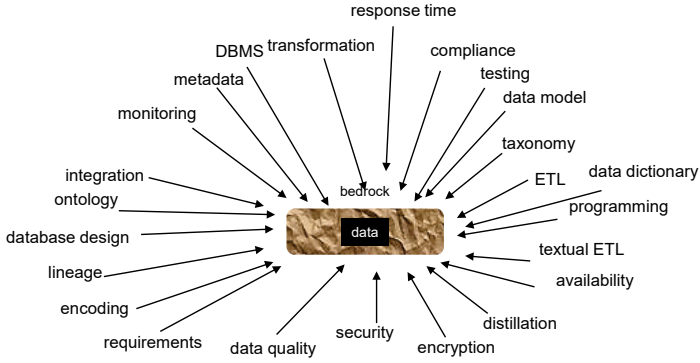


Figure 11.2: Since the bedrock leads to business value, technology needs to focus on building and supporting the bedrock of data.

CHAPTER 12

The Hierarchy of Data Needs

Maslow's Hierarchy of (Human) Needs has been drilled into our society since 1943. As humans, we have Physiological needs to survive, such as air, water, food, and shelter. We have Safety Needs, such as personal security, employment, health, and property. We have needs for Love and Belonging as people want friendship, intimacy, and family. We have Esteem Needs because we want to be respected and have recognition and freedom. We have the desire to be the best we can be, which Maslow puts at the top of the pyramid as Self-actualization Needs.

The bottom two tiers are needed for survival. Once we are secure in our survival, we need others and respect for success. We look to the top of self-esteem and self-actualization for growth.

Interestingly, humans are content to remain somewhere in the middle of the needs structure. For many of us, success is enough. Most of us fall short of fully making our mark, being fully creative, or completing our inner potential. We live, compete, and are happy with lives filled with Safety,

Love, and Respect. Even those who reach the top will tend to drop back to the comfortable and just tell stories of the "Glory Days."

Figure 12-1: Maslow's Hierarchy of (Human) Needs.

This is the same for most businesses with regard to data. There is a "Hierarchy of Data Needs" for your company to compete and succeed as well. Your company will work through stages of this Hierarchy as well.

The five levels of the Hierarchy of Data Needs from the bottom up include: Collect, Move and Store, Transform, Label, Consolidate, and Aggregate, and, at the top, Optimize and Learn.

As you move up this Hierarchy, you are looking to believe your data, integrate your data, access your data, and finally automate your data.

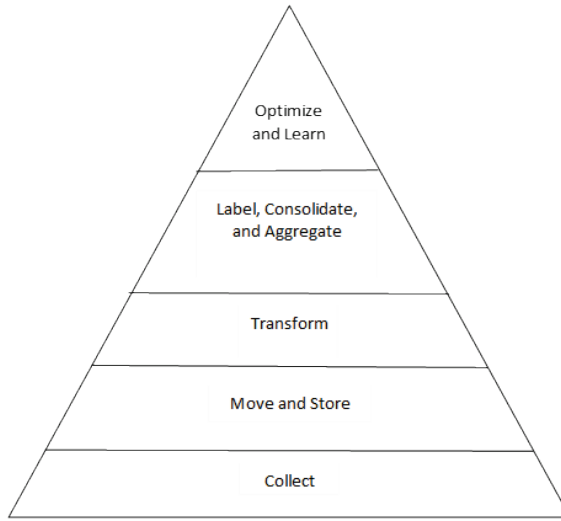


Figure 12-2: The five levels of the Hierarchy Data Needs.

Collect data

At the bottom of the tier, we collect data. We collect more data than we will ever need or could ever consider using. It needs to be pointed out that as a world, we are in the Data Age, not the Information Age, because we are great at collecting as much data as we can. We need that data to have context and meaning to be considered "Information." 90% of all today's digital data globally was generated in the past two years. This has been true for decades. That means that stored digital data is growing almost 100-fold every four years!

Data is collected from data entry, sensors, computers logging events, and machine or Internet of Things (IoT) systems. In every case, the key is for the data to be complete, accurate, unbiased, and singular. Even textual data needs to come from root sources, but in those cases, much of this is in an irregular format. You also want to store appropriate metadata with textual data so that it can be referenced back to its source.

The key to this bottom tier is that the correct data is collected and classified properly. This data is the basis for all your systems and most of your decisions and so it must be right and it must be believable.

Move and store data

The second tier is to move and store your data. This means that the source systems have reliable transfer systems. Your storage for structured and unstructured data needs to be redundant and accessible. Your Batch or Online Transaction Processing (OLTP) data transfer systems require verification and rollback procedures. Your extract, transform, and load procedures must match your business needs and data governance guidelines.

Relational databases and data warehouses have been the de facto storage and retrieval systems for structured data for decades. With the introduction of the data lake, data lakehouse, and parquet- and JSON-based file stores, businesses have thrown traditional storage and movement requirements to the wind for the chance to collect more data for possible future usage. These decisions have made the next steps of the hierarchy exceedingly difficult, if not impossible. These decisions have been made by persuasive techies claiming business acumen, convincing business decision-makers who want to be considered tech savvy. In most cases, these decisions prove themselves wasteful, if not destructive.

Transform data

The middle tier is to transform your data into something useful for your business decisions. This is one of the most difficult tiers because it requires both data knowledge and business understanding. The central tier is where data builds a competitive advantage for your company. This tier, along with the move and store tier, is where you integrate data from multiple systems into usable fuel for your decision support systems, expert systems, business intelligence systems, and business analytics systems.

This tier is difficult because it requires meticulously cleaning your "very dirty" stored data. Then you need to transform that data into a specific format that your data governance committee has dictated. The problems occur when your root systems do not properly collect, store, or label the data required to make correct business decisions. Better stated, if your base tier data is not believable and therefore not stored correctly, you cannot integrate it with your other data.

In this tier, we clean the data, transform it, prepare it for reporting systems, and detect anomalies in the data. In this deviation detection, through great insights, we save money in solving problems before they become real problems.

If a process is trending in the wrong direction, and you detect and fix it, you can save your business.

Label, consolidate, and aggregate data

The fourth tier consists of labeling, consolidating, and aggregating data. This is the root of business analytics and reporting systems. It is here that we bring data together to give information and insights to decision-makers when requested. We develop metrics to measure how we are doing. We segment our customers for better marketing

and customer service. We create OLAP cubes by aggregating data into dimensions so that we can discover trends, associations, outliers, deviations, and sequential patterns.

This tier adds value to the data because it is here that we really begin to use the data.

This can be done with your textual data by mining and contextualizing key phrases. Then with the mined data, you can add metadata to those specific contextualized terms and store them as columnar data.

From a general point of view, this is the tier where we access the data. Everything else builds up to this tier. This tier is where most businesses find comfort, success, and competitive advantage. When considering Maslow, this is our Self-Esteem Tier, and most companies are content to exist here.

Optimize and learn from data

The top tier in the Hierarchy of Data Needs is the Optimizing and Learning Tier. This tier uses computer algorithms that use existing data to explain itself. Once it understands existing data, it can predict trends as new

data arrives. This is the basis of machine learning. These algorithms can experiment with expected outcomes versus actual outcomes. They then can predict what will happen with better accuracy as even more data comes to the company.

If we can predict what will happen, we can then prescribe appropriate actions to deal with expected outcomes. These prescriptions are the basis for expert systems or systems that advise users based on what an industry expert might suggest. Your Navigation App on your phone is a great example of an expert system. If the app knows where you want to go and knows standard or actual traffic, the system can advise on what directions to take to best get to your destination.

It is at this tier that we optimize business decisions based on data.

These give your company a true competitive advantage over non-data competitors. This is the tier where we automate data.

As you advance in this tier, the algorithms advance too, and you start to have systems that emulate human intelligence. These are also systems containing three or more layers, considered neural networks, which have the ability to "learn" from enormous amounts of data. These

are called deep learning systems. These systems are the engines for artificial intelligence systems.

Artificial intelligence systems can make decisions similar to humans. They can drive cars (often with much better ability than humans), control fleets of robots, guard your buildings, and produce natural language answers that outperform humans.

Summary

If you want to use data, you need a plan.

1. You need to only collect data that you need.
2. The data needs to be believed and trusted by your employees. If your employees distrust the data, you need analysts who can work with those employees to fully understand what the data is saying despite what personal beliefs the employees hold.
3. You need to store it in a place that is accessible yet secure.
4. You need to transform it into a format that is usable by your employees and the applications they use.

5. You can integrate your stored and transformed data with each other to have a fuller view of your data from disparate systems.

6. You then aggregate your data with proper metadata for your reporting and business analytics systems.

7. Finally, you can create systems that learn with your data to optimize business decisions or even generate new innovative technologies.

CHAPTER 13

Data Integration in the Data Lakehouse

The whole intention of the data lakehouse is to support good decision-making—from the CEO to the clerk and everyone in between. The bedrock foundation allows the organization to become truly data-driven. Without the bedrock of believable data, the organization can never become data-driven.

The most important key to providing the data that the organization needs is providing an integrated data foundation. The data lakehouse must contain integrated data. That is non-negotiable. It simply is not adequate to just throw data into the data lakehouse and hope it meets someone's needs. Throwing data into the data lakehouse without integrating it wastes time, money, and opportunity.

Data integration is mandatory to produce the foundation for the organization to make decisions. Integration is not negotiable.

Stated differently, when an organization elects not to integrate the data on which they make decisions, it will never achieve reliability using AI, ML, and data mesh. This is because these technologies assume that data is available and reliable. And that is a false assumption unless a bedrock of data supports those technologies.

The problem is that technology vendors and consultants detest having to do integration. And that presents a problem to the organization because organizations have traditionally depended on those vendors and consultants for advice on conducting the information processing department.

Different types of integrated data

When the subject of integration arises, knowing what data to integrate is important. Three basic types of data appear in the bedrock foundation of data within the data lakehouse:

- Application-based structured data
- Text-based data
- Mechanically generated analog data.

Each of these three types of data belongs in the data lakehouse's bedrock foundation.

Automating integration

There is technology to aid in the integration of data.

For application-based structured data, there is ETL technology. For text, there is textual ETL. For analog/IoT data, there is distillation software. Each of these types of software supports the need for integration in a sophisticated and automated manner.

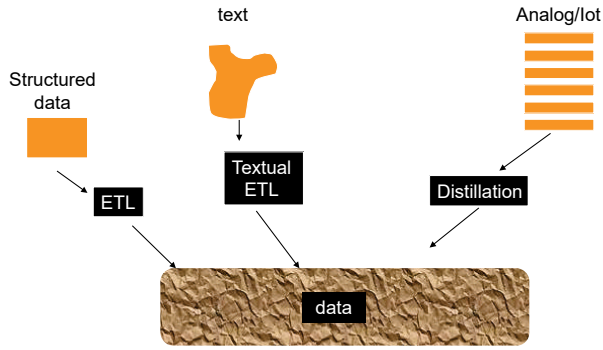


Figure 13-1: Bedrock data is integrated.

The net result of the process of integrating data is the transformation of data itself. Another way of thinking about the bedrock of data is that the bedrock contains transformed data.

One of the interesting aspects of the transformed data is that the process of transformation is very different for each of the different types of data. Stated differently, there is little or no commonality of processing between ETL, textual ETL, and distillation.

ETL

ETL is the process of transforming structured, application-based data. As a rule, transactions generate structured data that the business runs to do its daily activities.

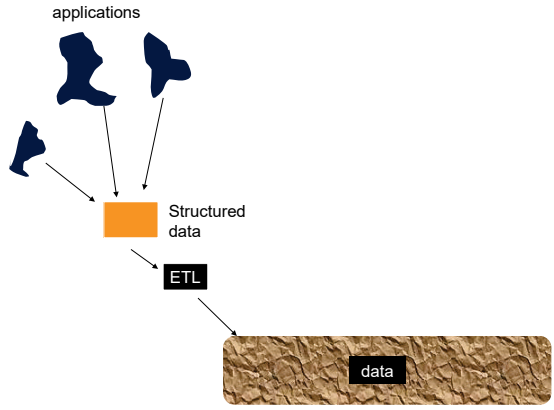


Figure 13-2: ETL takes application data as input and turns it into integrated data.

There are many facets to transforming application-based data into enterprise-based data using ETL, including resolving:

- Naming conventions
- Encoding practices
- Physical characteristic differences
- Attribute measurements
- Key value differences
- Attribute existence criteria
- Granularity discrepancies
- Definitional discrepancies

- Data selection criteria
- Summarization and derivation discrepancies.

The net result of integrating application-based, transaction-based data is the ability to have a true enterprise understanding of the transactions done by the organization.

Textual ETL

The integration of text into bedrock data is very different from the integration of structured data. The first major difference is in the source of the data found in textual analytics. Transactions generate structured data, yet voice conversations and presentations generate textual data. Textual data comes from printed sources, such as newspapers, documents, and ad memos. Textual data comes from the Internet, email, and other electronic forms of data.

Furthermore, textual data is free form. Unlike transactional data where every occurrence is well understood and rigid, textual data is freeform. The metadata that describes structured data is not present when dealing with text. People just do not talk in the same fashion that a bank follows to cash a check.

So there are significant differences in how textual data is processed as opposed to how structured data is processed.

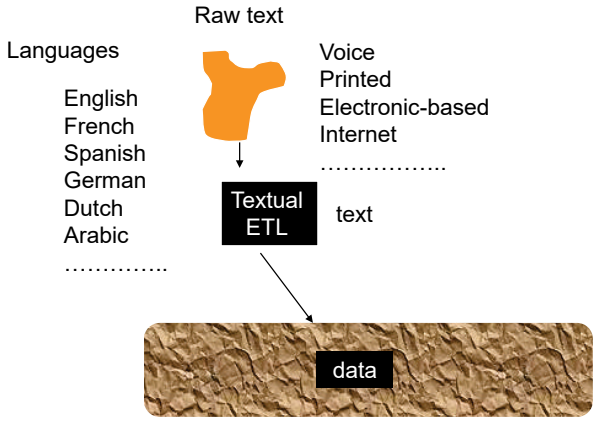


Figure 13.3: Raw text is integrated when placed in bedrock data.

Some of the elements of the integration of textual integration include:

- An ontology that describes the general venue that is encompassed by the text
- Taxonomies that are found within the ontology
- Nexus that contains both taxonomies and business rules
- Recognition of words based on the proximity to each other
- Homographic resolution of terms
- Ability to deidentify selected data
- Ability to recognize commonly used terms
- Ability to operate in multiple languages
- Ability to recognize sentiment in text

And so forth.

Distillation algorithms

The integration of analog/IoT data differs from ETL or textual ETL. The essence of analog data integration is removing data with a low probability of access in the bedrock data. We simply cannot store all of the analog data generated. This is especially true of data with a very low probability of access.

The raw analog data needs to go through a distillation process to separate the data that is of interest with a high probability of access from the data that is not of interest with a low probability of access. We send the data with a high probability of access to the collection of bedrock data.

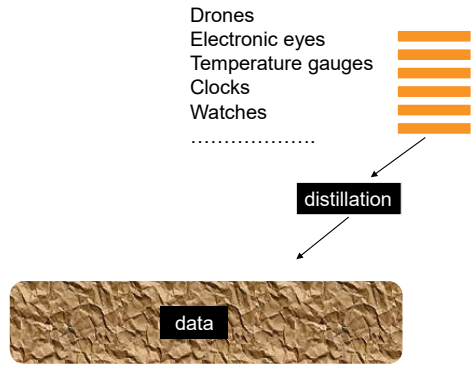


Figure 13-3: Mechanically-generated data is distilled and integrated into bedrock data.

The sources of data for analog data include:

- Drones
- Electronic eyes

- Automobiles
- Temperature gauges

And so forth.

The elements of analog distillation include such things as:

- The algorithms used for distillation
- The changes to the algorithms that have occurred over time
- The thresholds of selection
- The changes to the thresholds of selection that have occurred over time
- The timings that measurements have been recorded
- The changes in the timings that have occurred over time.

Summary

When built as suggested, the bedrock of data is built to support the applications that need the data to support the many analytical needs of the organization.

Figure 13-4: Bedrock data serves as the basis for analytical processing across the enterprise.

CHAPTER 14

Analytics

The primary purpose of the bedrock structure of data within the lakehouse is to support analytical processing. With analytical processing supported by a firm foundation of data, the organization can make well-founded, informed decisions. Conversely, the organization can only guess what decisions to make without a firm foundation of bedrock data.

Three basic types of data within the bedrock foundation of the data lakehouse are structured data, textual data, and analog/IoT data. These three types of data form the bedrock foundation. Although the foundation supports mostly analytical processing, sometimes, although rare, there is also operational use of the bedrock data.

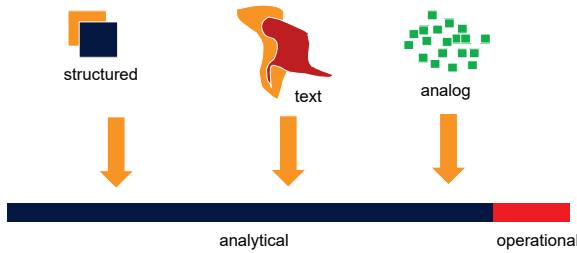


Figure 14-1: The primary purpose of bedrock data is to support analytical processing.

Structured analytics

One type of analytics we can do from the bedrock environment is based on structured data only. We need to ensure though that we are analyzing *integrated* structured data. This way, the organization can do analytical processing across the enterprise. Stated differently, it is a mistake for organizations to put unintegrated application data into the bedrock foundation.

By using the data found in the bedrock foundation, the organization can look across the enterprise and answer important questions such as:

- How many customers do we have across the enterprise?
- What kind of sales are we making across the enterprise?
- What location is more productive and profitable than other locations across the enterprise?
- How are sales activities across the enterprise varying by product line? By location? Over time?

The bedrock foundation of structured data lends itself to all sorts of useful analytical analyses, such as:

- **Departmental analysis**. How is one department performing in relation to another department?

- **Trends**. What changes are occurring over time?
- **KPIs**. What are the key performance indicators and how are they performing?
- **Outliers**. Where and who are the outliers and what conditions are causing them to be an outlier?

Text analytics

The second kind of analytical processing using bedrock data is text analysis. We assume that text has been passed through textual ETL and is in a form suitable for analysis. We assume that both text and context are supported for the purposes of textual analytics. Stated differently, placing raw text in the bedrock foundation is not a good policy for many reasons.

Once the foundation of textual data is in place in the bedrock data, we can do all sorts of analysis. A typical analysis of text might be the analysis for understanding customer sentiment. Hospitality organizations need to understand their customers and gather customer sentiment in a variety of fashions. Customer sentiment is gathered directly from the customer, the Internet, email, call centers, and so forth.

The customer's sentiment is gathered and realized through the analysis of what most customers are saying.

Another form of analytics that comes from text is correlative analytics. In corelative analytics, the analysis is made of variables that occur together. A good example of corelative analytics is the analysis of doctors' records. Doctors' records contain a wealth of corelative information, such as, for 1,000,000 COVID patients:

- How many had a serious case of COVID?
- How many were smokers?
- How many had cancer?
- How many were obese?
- How many were over 65 years old?
- How many were men? Women?
- How many were taking furosemide?

And we can do many other forms of corelative analysis.

Analog/IoT analytics

Yet another type of analytics we can do using data from the bedrock foundation is the analytics that flows from analog/IoT data. Analog/IoT data analytics can either look at the totality of data or the individual records found in the analog/IoT data.

As an example of analytics searching for a single record, consider an analysis that looks at the output of machine manufacturing. The analyst looks for machines that are producing defective output, examing each product made by each machine. When the analyst finds a faulty machine, adjustments are made to the machine. In such a fashion, the quality of manufacturing is improved.

Combining text and structured data

One of the more intriguing and potentially powerful forms of analytic processing within the bedrock data environment is the combined analysis of structured and textual data simultaneously. On occasion, it is possible to combine data from both environments. In doing so, different and powerful perspectives of data can be created.

For example, suppose the purchases made by a customer exist in the structured environment. In this case, the purchases that have been made, when the purchases were made, where the purchases were made, and the price paid for the purchases are gathered in the structured environment. Then, in the textual environment, it is found that the customer has voiced opinions as to the products purchased.

When the structured data is merged with the textual data, a very complete and accurate picture of the customer starts to appear.

Understanding the customer allows the manufacturer to improve their products and services. In addition, by understanding the customer, the manufacturer increases the chances of gaining new customers.

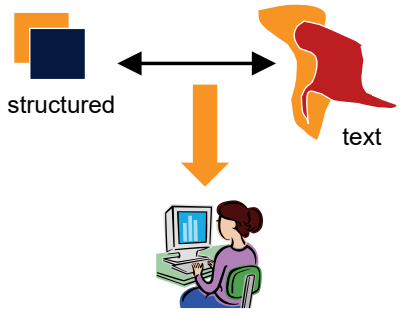


Figure 14-2: Combining text and structured data analytics can lead to even more powerful analytics, such as analyzing the 360-degree analysis of the customer, customer trends, and store satisfaction.

While the merger of structured data and textual data is a great advance over current methods of hearing the voice of the customer, it is not without complications. To achieve a cross-environment analysis, it is necessary to have some method of connecting the data from both environments.

The problem is that structured data operates on keys, attributes, and indexes. And people normally don't talk or write using these components. Therefore, there needs to be

some way to connect language and structured data to perform analysis on both environments.

There are occasions where there are such linkages. For example, in comments on websites such as Yelp, we can identify the organization, the date of the comment, and other important information. This structured information is carried with each comment made by a user. When there is identifying information, we can perform cross-environment analysis. But when there is no way to identify a connection between structured data and textual data, it is very difficult (if not impossible) to analyze both kinds of data.

Of course, there is a strong connection between the two environments in some cases. And in other cases, there is not. As an example of a very weak connection, it is possible to connect the two environments using a person's name. But such a connection is fraught with pitfalls and is not to be trusted.

As a simple example, the name, *William Inmon*, exists in the structured environment. In the textual environment, there is *Bill Inmon*. Is *William Inmon* the same thing as *Bill Inmon*? Maybe it is and maybe it isn't. Assuming that these names refer to the same person can lead to erroneous conclusions.

Connecting all three environments

Of course, it is possible to make a connection across the three environments. And when we can make such a connection, very interesting analytics can be done.

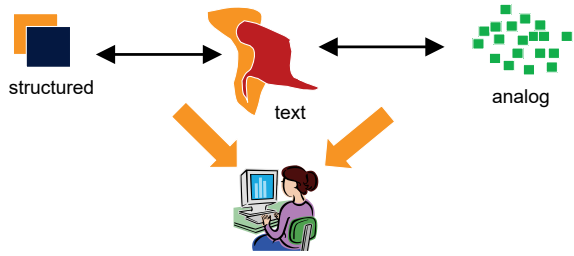


Figure 14-3: Analytic processing across all three types of data.

The problem is that the connecting data between the different environments is almost always very weak data. And the weakness of the connection between the different environments prevents any major analytical processing from occurring.

Performing analytics

We can analyze bedrock data in three ways:

- By dashboards
- By knowledge graphs
- By spreadsheets

Each of these three methods of analytical processing has its advantages and disadvantages.

Dashboards are useful for static data. That is, data that is well-defined and whose structure and relationships to other data do not frequently change. Dashboards are ideal for a company following its KPIs. Dashboards are not suitable for dynamically changing data and data whose relationship with other data constantly changes. Dashboards are good for summary data and less useful for analyzing individual data elements. One of the most appealing aspects of dashboards is the visualization possible using a dashboard. Top management is particularly fond of powerful visualizations.

Knowledge graphs are good for dynamic data where relationships between data elements constantly change. Knowledge graphs are useful for relating very different kinds of data together. Knowledge graphs are useful for detailed data and less useful for summarized data.

Spreadsheets are ubiquitous. The great value of spreadsheets is their immediacy and their great flexibility. Any user can find and use a spreadsheet, apply a spreadsheet to any kind of data, and directly enter data into a spreadsheet. But with the great flexibility of spreadsheets comes a problem. The problem is that spreadsheets do not carry with them data that has a high degree of integrity. Just because a number appears on a

spreadsheet does not mean that the number is either accurate or believable. Because anyone can enter any values they want in a spreadsheet, the data found on a spreadsheet is always open to speculation.

Summary

There are then different ways to do different types of analytical processing. The analytics can be trusted as long as the analytical processing is based on the bedrock of data.

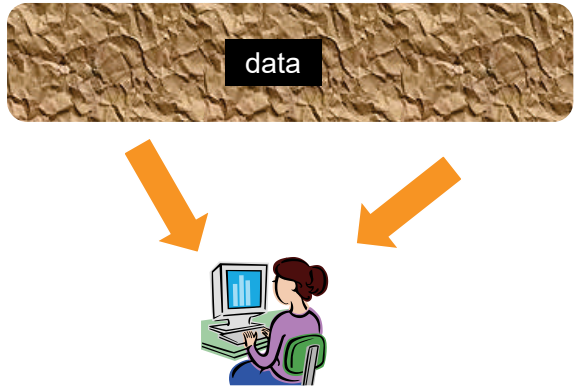


Figure 14-4: Bedrock data supports analytical processing.

CHAPTER 15

Soft Data

The essence of the bedrock foundation data in the data lakehouse is believability. Therefore, if the data in the bedrock foundation is not believable, we should not place that data in the data lakehouse.

When a person accesses data from the bedrock foundation, the person must have confidence that the data retrieved is accurate and complete.

When it comes to structured, textual data, and analog/IoT data, there is usually little question as to the authenticity of the data. This kind of data can be called "hard" data.

But there is another kind of data that can be placed in the bedrock foundation, and that data is called "soft" data. Soft data is data that comes from a spreadsheet, the Internet, or from the government. The issue with soft data is its accuracy and authenticity, which is at odds with bedrock data.

Should soft data be placed in bedrock data at all? Is soft data believable enough to be housed with other data whose values are well established and vetted?

We can place some soft data into the bedrock foundation. However, we must take care to ensure the validity of the soft data. Other soft data should never be placed in the bedrock foundation.

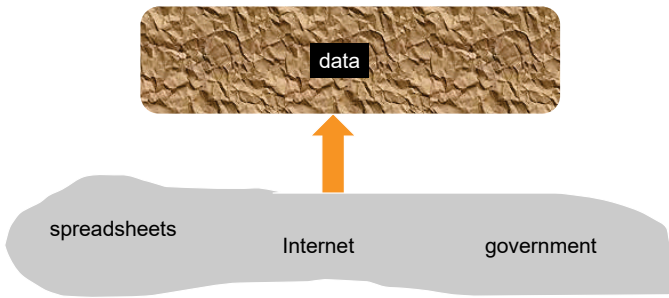


Figure 15-1: "Soft data" can be included in the bedrock data.

Spreadsheet data

The first kind of data we can place in the bedrock foundation is spreadsheet data. However, there are some stringent restrictions on placing spreadsheet data in the bedrock foundation.

There are many problems with spreadsheet data. The first and biggest problem is that spreadsheet data can come from anywhere. In fact, data found on a spreadsheet may

be entirely fictitious. Anyone can put anything they want on a spreadsheet. So the first problem with spreadsheet data is that it may not be believable. If data on the spreadsheet is not believable, we should not place it in the bedrock foundation.

A second issue with spreadsheet data is that the data found on a spreadsheet has no usable or reliable metadata. Although a spreadsheet contains rows and columns, this specification is only within the context of the spreadsheet. Therefore, the row and column definitions may or may not have any relevance to the business of the organization whatsoever.

One consequence of the fact that spreadsheets have no reliable metadata is that numeric values have no context on a spreadsheet. For example, the number 1977 appears on a spreadsheet. Now what is 1977? Is it a year? The number of sheep in the pasture? The amount your account is overdrawn? The size of the freshman class at Yale?

The truth is that when the number 1977 stands alone on a spreadsheet, it has no meaning. It could be anything.

The net result is that only text can be taken from a spreadsheet. And even then, text has to have context. So we must embed the context of text inside the text itself to include that text within the bedrock layer. Sometimes we can derive the context from the text, and sometimes we

can't. And, of course, text must pass through textual ETL before it can be placed inside the bedrock foundation.

Internet data

Another abundant source of data for the bedrock foundation is the Internet. And as long as the data can be certified and verified, there is no reason why we cannot place Internet data in the bedrock foundation.

There are, however, some obstacles for taking data from the Internet and placing it into the bedrock foundation. The first obstacle is accessing the data on the Internet. Some sites do not want people taking data from their site. And in any case, the website the Internet data is hosted on requires its own custom code. Furthermore, the required code is constantly changing because the Internet site itself is constantly changing.

The good news is that there are very limited privacy issues with taking data from the Internet. The reason why data is even placed on the Internet in the first place is that the data is public domain data.

Also, in almost every case, the data found on the Internet is a one-time-only access. It is very unusual, although possible, for data to be constantly updated on the Internet.

Government data

The third likely source of soft data comes from the government. The government sends down a wealth of data that may be useful. For example, the government declares interest rates, population numbers, inflation rates, employment rates, etc. There is no reason why we should not place these proclamations from the government in the bedrock foundation.

Summary

However, from all soft sources of data, a certain amount of certification of the data is required before placing it in bedrock data. This certification produces the believability of the data. Some of the elements of the certification of soft data include:

- How was the soft data calculated? Exactly what calculations have been made?
- What data was selected for the calculations and gatherings of data? What data was excluded?
- When was the data gathered and calculated? When was the data taken off of the Internet?

- What is the source of the data found on the Internet?
- Who made the calculations and where?

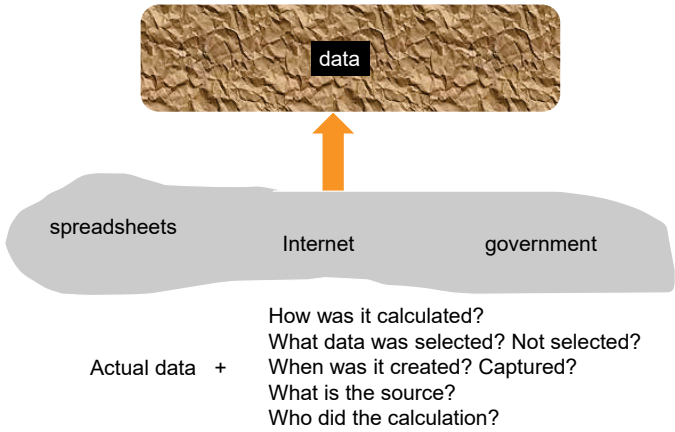


Figure 15-2: If the certification of soft data cannot be made, then the soft data should not be placed in the bedrock foundation.

CHAPTER 16

Descriptive Data

Due to the differences in the types of data in the bedrock foundation, the descriptive data is very different for each of the different types of data as well. And all of the descriptive data, despite fundamental differences, is needed to have a complete picture of what is in the bedrock foundation.

The analyst depends on all of the different kinds of descriptive data.

The descriptive data found in the analytical infrastructure exists to point to the detailed data beneath it. Stated differently, the analytical infrastructure is a roadmap to the detailed data. The descriptive data tells the analyst where to find data, what the data means, and how to combine the data.

As such, the analytical infrastructure is where the analyst starts. The descriptive data serves all sorts of analysts, including data scientists, business analysts, clerks, and

even management. In a word, the descriptive data serves all communities that wish to use the bedrock data.

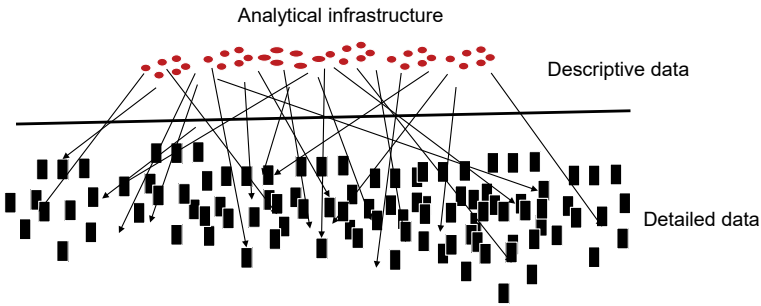


Figure 16-1: There are two levels of data found in bedrock data.

We will describe each of the different components of the analytical infrastructure that appear in Figure 16-2.

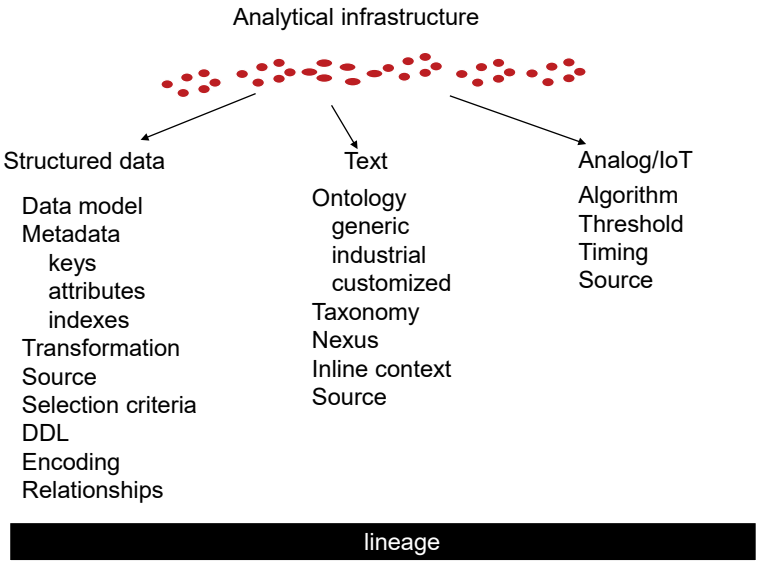


Figure 16-2: Components of the analytic infrastructure.

Data model

The descriptive data for the structured environment portion of the bedrock data foundation begins with the data model. The data model is an abstraction of the structured data found in the bedrock foundation.

There are different levels of modeling within the data model. The ERD (entity relationship diagram) level is where we define the major entities of the organization. The relationships between the different entities also exist in the ERD. Typical entities in the ERD might be customer, product, shipment, sale, and so forth.

Beneath the ERD is the dis (data item set). The data item set takes each of the entities and does a further description of the entity. A dis exists for each entity found in the ERD. In the dis are found keys, attributes, and the relationship between different organizations of data that exist in the entity. And for each grouping of data in the dis is found a physical definition of the data, including actual definitions, key identification, attribute name and structure, and indexes. The dis sets the stage for more detailed database design.

In many ways, the data model is an abstraction of the structured data within the organization.

It is very useful to have an abstraction. Data quickly becomes complex and abstracting allows the designer and analyst to access and analyze the data in the bedrock.

Metadata

A second related component of the analytical infrastructure of the structured component of the bedrock foundation is the metadata definition of data, which is similar to the physical layer of the data model.

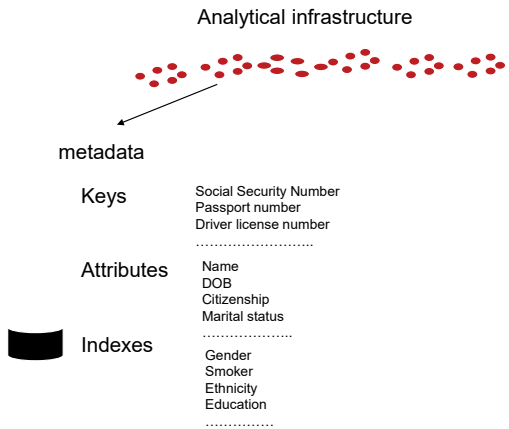


Figure 16-3: The metadata layer does contain some physical characteristics that are peculiar to the DBMS that are not found in the lower level of the data model.

The physical layout of metadata includes the actual definitions of data as described by the database management system (DBMS). We define keys, attributes, and indexes.

Transformation

The analytical infrastructure contains a definition of the transformations performed upon structured data. Often the largest transformation done is the transformation of application data to enterprise data. But transformations of data are dome elsewhere as well. The transformation specification for structured data includes such aspects as:

- Naming transformations
- Encoding transformations
- Measurement transformations
- Currency transformations
- Calculation transformations
- Data selection transformations

And so forth.

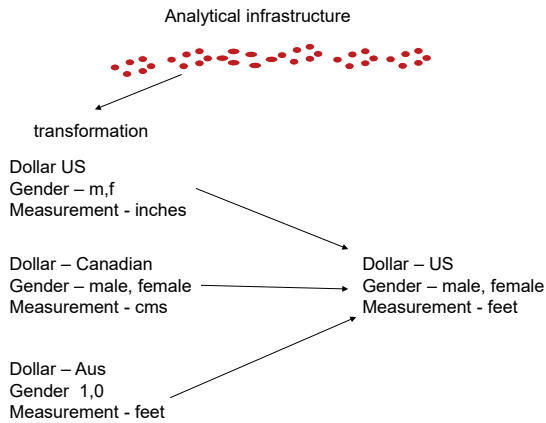


Figure 16-4: Examples of transformations.

Source

An important part of the analytical infrastructure for structured systems is identifying the source of structured data. Usually, the beginning source of structured data is transactions. We can collect transactions from a wide variety of sources. Sometimes there are other data sources other than transactions that find their way into the structured environment.

Consider an airline reservation transaction as a simple example of data collection from a transaction. During the making of the reservation, we collect the following types of information:

- Passenger name
- Flight
- Date of the flight
- Destination and origination ports
- Cost of the flight
- Seating

And so forth.

The transactions can come from various sources, such as bank teller activities, ATM activities, demand deposit transactions, airlines reservation, and so forth.

Selection criteria

One of the most important parts of the analytical infrastructure for structured data is identifying the selection criteria for calculations. It is not sufficient to identify the algorithms used in a calculation. It is also necessary to identify what data has been included and excluded from the calculation. The analyst needs the selection criteria to do an accurate and incisive analysis.

The inclusion and exclusion of data can make a big difference in the accuracy and believability of the calculation. The analyst using the bedrock data for analytics needs to know what data participated in a calculation.

DDL

Another component of the analytical infrastructure is that of the DDL (data definition language). DDL defines the database definitions to the DBMS. While the dis and the physical definition of data help define data, the DDL always contains other information useful in defining the database to the DBMS. In addition to keeping the DDL in the descriptive layer of processing for the bedrock foundation, the analyst needs to keep track of the changes to the DDL over time.

Encoding

Another important component of the descriptive level of the analytical infrastructure is the documentation of the encoding. Encoding refers to the values saved inside the database that have meaning. As a simple example, the values m and f might be encoded in the gender attribute. In this case, m refers to male and f refers to female. But there are many other encoding structures contained in the database.

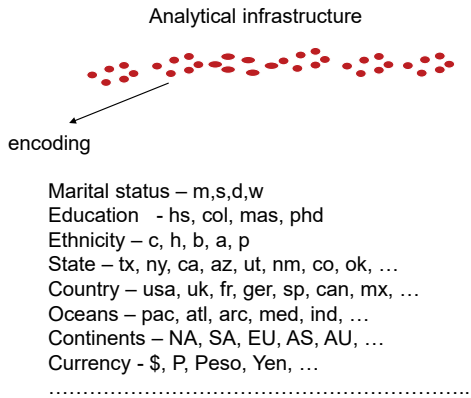


Figure 16-5: For the analyst to understand the data stored in the bedrock, the analyst must understand the encoding of data values.

Relationships

One of the more important components of the analytical infrastructure of the bedrock data is the identification of

the relationships among data elements. The relationship of one data element to another forms the basis for much of analytical processing. Stated differently, much analysis relies on the existence of relationships of data. Therefore, it is important to identify those relationships.

Data relationships can take many forms. Some relationships can be built and supported entirely with an application. The DMBS supports some relationships, whereas others are casual and exist only infrequently under certain conditions. Still, other relationships are mandated to exist whenever the data exists to support the relationship. Relationships can take many forms, such as:

- Application-supported relationships
- DBMS-supported relationships
- Implicit relationships
- Explicit relationships
- Inferred relationships.

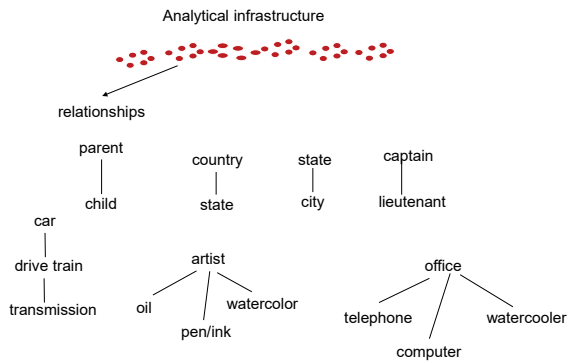


Figure 16-6: Relationships take many forms.

Text

Another class of descriptive data is the descriptive data that supports text and text analytics. While there are similarities to the descriptive data that supports structured data and text, there are indeed some distinctive differences. Perhaps the main difference is in the embodiment of context.

In the structured world, the context of data is very explicit and therefore embodied in the very fabric of the system. For example, in the structured world, an attribute may be named "gender". The implication is that all data found in the attribute will relate to gender. Or a table may be named CUSTOMERFILE. The implication is that information found in the file relates to customers. So, in the structured world, the context of data is explicitly and obviously found in the metadata that applies to the structured data. In the structured environment, context is supplied by the very existence of metadata that describes the data itself.

But there is no such explicit embodiment of context in the world of text. People do not talk in terms of context. People do not write in terms of explicit context. Instead, context is implicitly embedded in language. Context certainly exists in text. But context is not explicitly defined in the same way that it is defined in the structured environment. To understand context in language, it is

necessary to disambiguate the text. Yet context in textual analytic processing is as important as context in structured systems. And context plays a very important role in the analytical infrastructure needed for making sense of the bedrock data foundation in the data lakehouse.

Ontology

The primary component of descriptive data for text is the ontology. An ontology is the collection of two or more related taxonomies. As a rule, the ontology provides a complete description of a business or a discipline.

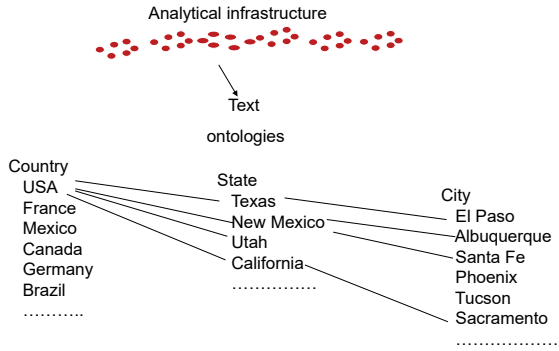


Figure 16-7: Together, the taxonomies for country, state, and city form this ontology.

There might be an ontology for car manufacturing. There might be an ontology for piloting an airplane. There might be an ontology for teaching school, and so forth. As a

simple example of an ontology, consider an ontology that contains country, state, and city. Every state has a country it belongs to and every city has a state and a country it belongs to. A geographer might use this ontology.

Ontologies have three levels: generic, industrial, and customized. The generic level of the ontology is for words and terms used generically. The subject matter of the generic level of the ontology does not matter when looking at terms generically. The industrial level of the ontology contains industry-specific terms peculiar to the industry. Medicine has its own terms. Legal has its own terms. Accounting has its own terms.

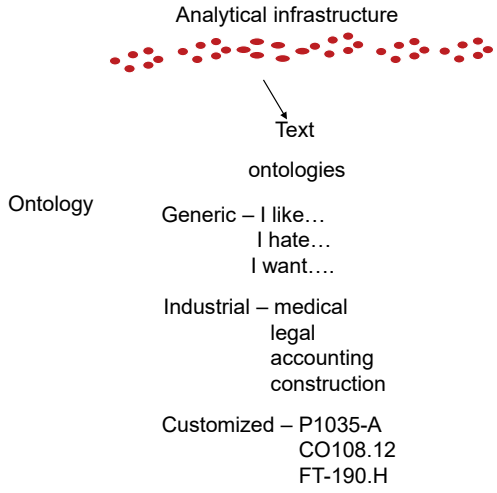


Figure 16-8: Ontologies have three levels: generic, industrial, and customized.

The customized level of the ontology contains company-specific designations. For example, we may specify an oil well as W10.25A. Or a compressor may be specified as C1098-VC. Every industry has terminology specific to the individual company and no one else. These customized terms belong in the ontology as well.

Taxonomy

The next type of construct necessary in understanding text is the taxonomy. A taxonomy is merely a classification of words. Each of the words in a taxonomy has the same relationship to the class as all of the other elements found in the taxonomy.

Taxonomies are placed into an ontology.

At first glance, an ontology and a taxonomy appear to be the same thing or at least very closely related. Yet there are some significant differences between the taxonomy and the ontology. The contents of the ontology are heterogenous and the contents of the taxonomy are homogenous. An ontology will contain many different elements that are fundamentally different from each other. For example, an ontology for a car manufacturer might contain taxonomies for raw materials, car sales, and

quality assurance. Each of these taxonomies is very different from each other.

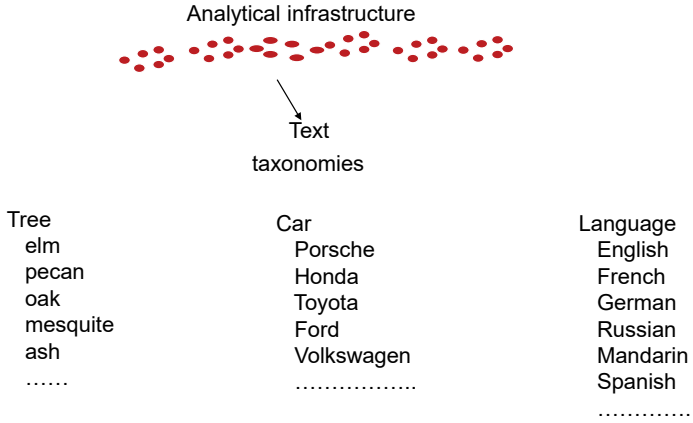


Figure 16-9: A taxonomy contains only classifications of data whose relationship to the classification is the same. Unlike an ontology, the contents of a taxonomy are homogenous. As a simple example, the taxonomy for a tree might contain elm, oak, pine and pecan. But the taxonomy for a tree would not contain a football, a golf club, or a steak dinner. A football is not a kind of tree.

It is not sufficient to use only ontologies and taxonomies to do text analytics. Text has other requirements as well.

Nexus

One of the other elements required by textual disambiguation is looking at business rules implied in the text itself.

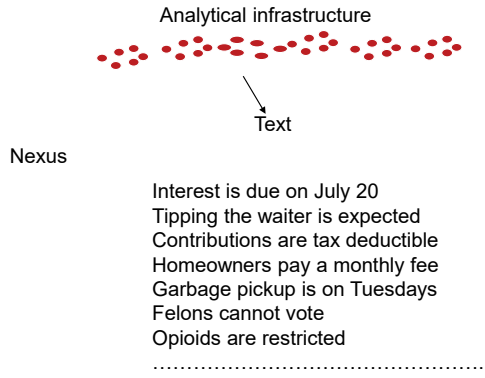


Figure 16-10: These business rules form what can be called a nexus.

Inline context

Related to business rules is the process of inline contextualization. Inline contextualization is a necessary component of ontological/taxonomical resolution for the disambiguation of text.

Much text can be disambiguated by looking at ontologies and taxonomies as a guideline. But other forms of text do not make use of ontologies and taxonomies.

Most text is free form. The writer/speaker can say anything they want any way they want. But some text is not free form. For example, legal contracts and laboratory reports are typically not free form. When text is not free form, the meaning of words can be derived by the appearance of text in the text itself.

For example, in a contract, we can infer the name of the person signing the contract by finding the beginning and ending delimiters in the contract's text. Once we find the beginning and ending delimiter, the meaning of the text between the delimiters can be inferred.

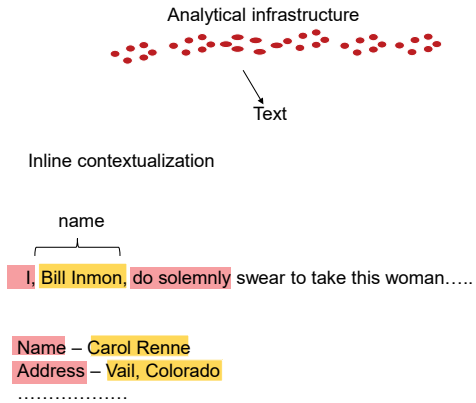


Figure 16-11: Suppose there is an employment form where a person's name is required. The beginning delimiter is "NAME". The person's name exists between the beginning delimiter and the end of the line.

Source

The source of textual data is as important in the world of text as the source is important in the structured environment. Typically the sources of text are:

- Voice recordings
- Email

- Printed text
- Electronic text
- Internet

Each of these sources of text has its own idiosyncrasies. Voice text requires transcription. And some degree of accuracy is always lost in doing the transcription. Printed text requires OCR (optical character recognition) transcription. The accuracy of the OCR transcription depends on many factors, such as ink strike, font, and stability of the paper. Internet access depends on the site that the internet data resides on. Each Internet site is different and each site changes frequently. Email access depends on the ability to filter spam and blather and remove system overhead data. If spam and blather are not filtered, the size of the email stream grows to an unworkable size.

Analog/IoT

The third type of data found in the bedrock foundation of the lakehouse is analog/IoT data. Analog/IoT data is data that a machine has generated. Most of the data generated by the mechanical monitoring of a process is not of interest. But occasionally, analog/IoT data is of serious interest. As such, analog/IoT data requires a distillation

process that separates the interesting data from the uninteresting data.

Algorithm

Of special interest is the distillation algorithm. The distillation algorithm has the intelligence to tell when an analog record will or will not be useful.

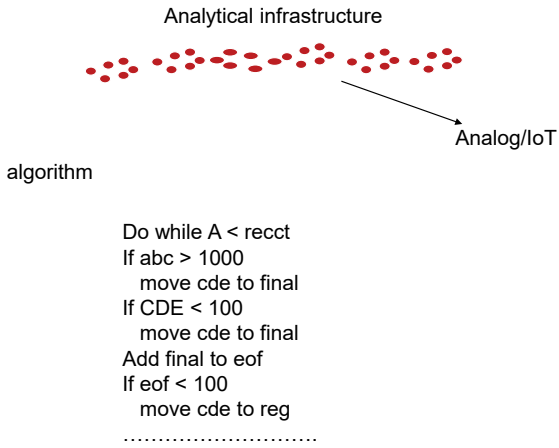


Figure 16-12: The algorithm is of special interest to the analyst trying to use analog/IoT data.

Threshold

In addition to the algorithm used to separate analog/IoT data, the parameters that define the data thresholds for

further analysis are of interest as well. The algorithmic thresholds determine the boundaries where a record will be written to an access file. Throughout the day, records are written due to the monitoring process. Most records are normal and not of interest. However, occasionally a measurement is made that is out of the bounds of normality. The measurement may be either too high or too low. In this case, a record is written to the high probability of an access file. The determination of whether a record is written depends on the values applied to the algorithm.

Timing

On occasion, time boxing captures analog/IoT data with a high probability of interest to the analyst.

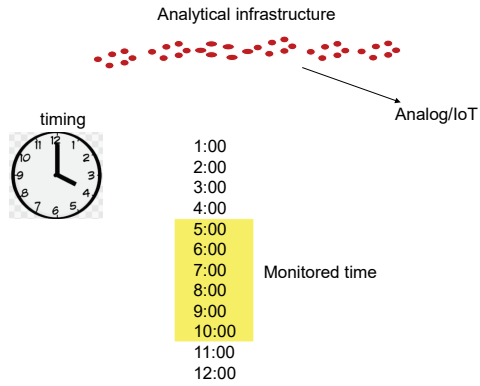


Figure 16-13: In a time boxing approach, the analyst chooses a time slot when an interesting activity is anticipated. All activity that occurs within that time box is captured.

Rather than depend on defined thresholds to determine when an activity is of interest, the analyst "time boxes" the monitoring activity.

Source

The source of analog/IoT data is important. Certain machines collect data at one rate. Other machines collect data at another rate. Certain machines have a high degree of accuracy in the collection of data. Other machines have a low accuracy in the collection of data. Some machines collect data based on one type of measurement. Other machines collect data in a different form of measurement.

The analyst needs to know all these nuances in the machines that generate the analog/IoT data.

Lineage

Across all of the different types of data is the data that reflects lineage. It is normal for data to flow from one database to another in the organization.

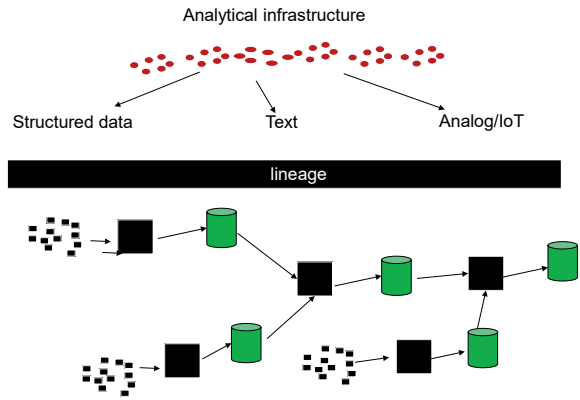


Figure 16-14: The data lineage is of great use to the analyst operating on the bedrock foundation of the data lakehouse.

Summary

For all practical purposes, the data lakehouse's bedrock data is a cornucopia of valuable data. The most basic type of data found in the bedrock data of the data lakehouse is the detailed data integrated into the bedrock. This bedrock data allows the organization to become truly data-driven. But having bedrock detailed data is insufficient to make the data lakehouse useful. Another type of data is needed. That data can be called the *analytic infrastructure* or simply *descriptive data*. The descriptive data describes the detailed data that resides in the bedrock of data, so the analysts can find the data they need.

CHAPTER 17

The Data Catalog

We place the analytical infrastructure in a structure called the data catalog. The data catalog is analogous to the card catalog found in a library. When you enter a large public library, you don't just go to the stacks and start looking for a book. Instead, you go to the library's card catalog. Here you can quickly and easily search for your book. Once you have found the references to your book, you can easily find it in the stacks.

The data catalog operates in a similar manner. The data catalog has references to all the documents and databases that are in the organization. With the data catalog, you can save enormous amounts of time by using the data catalog to find what you are looking for in bedrock data.

The data catalog points to the detailed data found in the data lakehouse. The data catalog sits above this bedrock data. The data catalog contains a wide diversity of information about bedrock data:

- Metadata
- Data models

- Ontologies
- Taxonomies

And so forth.

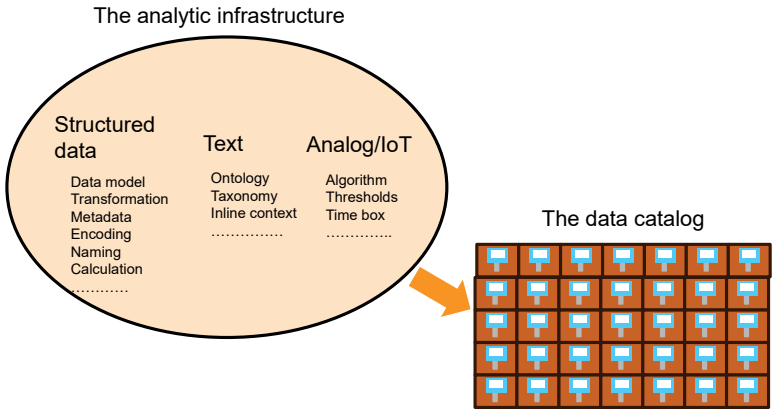


Figure 17-1: The data catalog contains information found in the analytic infrastructure.

Eternal maintenance

One aspect of the data catalog often overlooked is that the data catalog is always undergoing change. The data catalog is constantly being updated. There are many reasons for the constant barrage of updates, including:

- Business conditions are constantly changing
- Systems are constantly changing
- New systems are constantly being added

Usage

The data catalog should be open and available to anyone in the organization who analyzes data. The only exclusions are people trying to do malice to the organization.

This means the data catalog is open to:

- Management
- Clerks
- Day-to-day operations
- Auditors
- Analysts

And so forth.

Structures inside the different data types

The data catalog is structured so that one type of data in the catalog relates to another type of data. This is especially true for the data residing in the card catalog relating to a single data type.

It is possible to have relationships of descriptive data across the different types of data as well. Although there are not many of these relationships across the different types of data, when there are relationships can be very important.

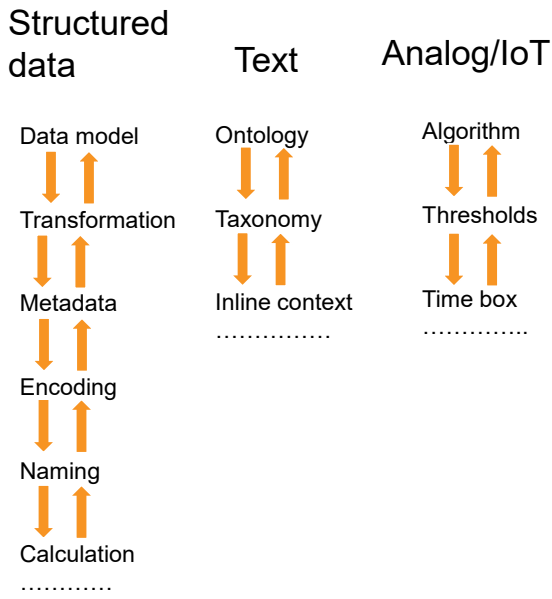


Figure 17-2: Some of the most interesting analyses occur between the different descriptive data types.

Summary

Analytical software can be used against the data found in the data catalog, just as analytical software can be used against detailed data in the bedrock. And of course, the data catalog can be analyzed independently of the bedrock data, if necessary.

Unlike bulk storage which may or may not be called part of the data lakehouse, the data catalog is definitely part of the data lakehouse.

CHAPTER 18

The Evolution of Data Architecture

Data architecture is at the heart of doing many kinds of processing. Without data architecture, there is no firm footing of data on which to operate. AI, ML, and data mesh all depend on data for success in their environment.

The bedrock foundation of data, which is found in the data lakehouse, is the result of a well-thought-out and well-crafted data architecture.

It may not be obvious that the bedrock foundation of data, the data lakehouse, is the result of a long evolution, but that is exactly the case.

In the beginning…

In the very beginning, there were only simple programs. The simple programs could read some input, process it

and produce some output. The first simple programs could do repetitive jobs in the organization accurately and quickly. The first early programs saved a lot of clerical time.

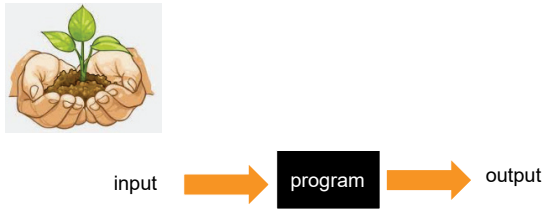


Figure 18.1: In the beginning, there were simple input/output programs.

Applications

Soon people discovered that they could write sophisticated applications. The computer and programming could do a lot more than just repetitive activities. Soon people began to create sophisticated applications. One program could prepare data for another program, and so forth. In doing so, the value of technology grew in the eyes of the organization.

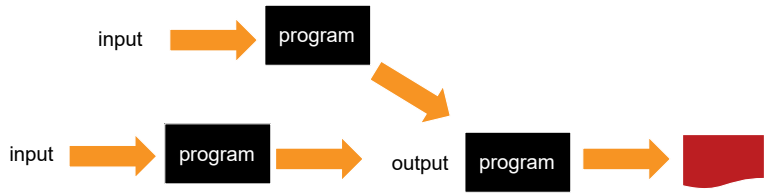


Figure 18.2: Then there were applications.

But the volume of data began to be an issue. The new applications began to churn through so much data and produce even more data that the storage media of the day, punched cards and paper tape, no longer sufficed as a medium to store data. There needed to be another way to hold and manage data.

Magnetic tape files

Entering this picture was the magnetic tape file. The magnetic tape file could store a much larger amount of data than could ever be stored on earlier media. There were lots of advantages of the magnetic tape file over punched cards. Storage on magnetic files was less expensive. Magnetic tape files did not require a fixed-length record. Magnetic files could be reused. All of these characteristics of magnetic tape files were significant advantages over punched cards.

And with magnetic tape came the notion of the master file.

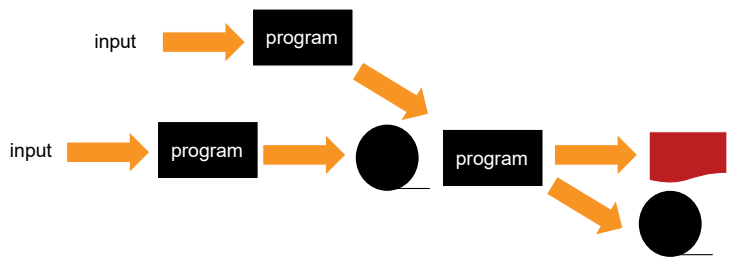


Figure 18.3: Then came master files.

The master file was useful for gathering and storing data about the major entities of the organization, such as customer, product, and shipment. The idea behind a master file was to concentrate related information in a single place.

But very quickly, magnetic tape files had problems. Although magnetic tape files could store data more efficiently than punched cards, you had to read the entire file to access a single record. This introduced long and wasteful processing. While the magnetic tape file had solved many of the problems of punched cards, the magnetic tape file introduced a new set of challenges.

In addition to requiring a full file search to find a single data record, magnetic tape files did not hold data for long periods. The oxide wore off the tape and the file became worthless when the magnetic tape files were stored for any length of time.

Disk storage

Into this world came disk storage. With disk storage, we could store and access data electronically. A database management system (DBMS) manages the data.

At first, disk storage was expensive and the volume of disk storage was limited. But over time, the cost of producing

magnetic disk storage dropped and became very affordable. And the internal access speeds improved with each new release of disk storage.

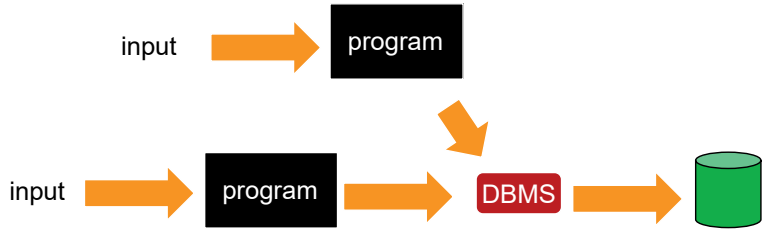


Figure 18.4: Next came DBMS and disk storage.

One of the innovations that disk storage brought was the ability to access data directly. It was no longer necessary to read an entire file to access a single record.

OLTP

Once data could be accessed quickly, the world saw the advent of something called online transaction processing (OLTP). With OLTP, it was possible to do sub-second transaction processing. OLTP gave birth to processing such as bank teller (ATM) and airline reservation processing. Because of OLTP, the computer was positioned in the organization front and center in front of the customer. OLTP elevated the computer into a position where the computer became an essential part of the processing of the organization's day-to-day business.

OLTP elevated the role of the computer from doing back-office tasks to directly interfacing with the customer. When computer availability failed or when response time slowed, the business suffered. The computer became the workhorse of the business. Soon there were OLTP applications everywhere.

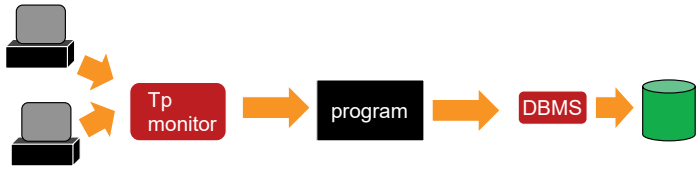


Figure 18.5: Next came the tp monitor and online processing.

Personal computer

Into this arena came the personal computer. The personal computer became very popular for a variety of reasons. The personal computer was inexpensive. The personal computer was portable so that it could be transported anywhere. The personal computer came with software such as spreadsheets, which opened up the world of technology and computer processing to anyone.

The personal computer opened the door of computation to an audience never exposed to the capabilities of the computer.

But most of all, the personal computer gave the gift of autonomy to the end user. No longer was the end user dependent on the IT department to do their processing. The end user could do their own independent processing with the personal computer.

For years the IT department had been the sole arbiter of what applications were built and what access to the computer was allowed. But with the personal computer, the IT department lost control over computation.

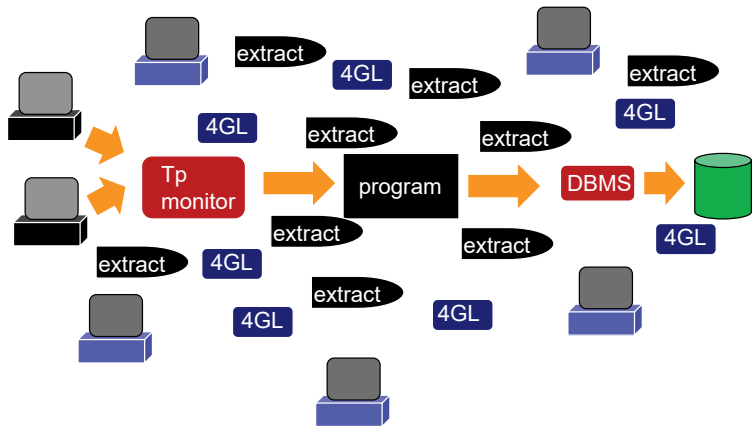


Figure 18.6: Then there was 4GL and the extract programs.

4GL processing and the extract program

With the personal computer came an explosion of technology. Soon both the IT department and the end user were busy developing their applications. Abetting the end-

user was technology known as 4GL (or 4th generation language) processing. 4GL processing freed the end user from needing the IT department to do its own processing and programming.

In addition to 4GL processing, soon there was a proliferation of extract programs. The extract program was simple in concept. The extract program would read a database, find some data, and deliver that data to another database. Extract programs were needed to move data from one application to another.

It was not obvious at the time that extract programs carried with them some very poisonous seeds. The problem with extract programs is that the same data element is stored in different places. In one place, data element ABC had a value of 39. In another place, data element ABC had a value of 60. And in yet another place, data element ABC had a value of 1000. So trying to make a decision based on the value of ABC was like trying to hit a target blindfolded.

The problem was that once the same data element was stored in multiple places, the data element was not kept up to date. So the organization woke up one day and found that it had the same data element with different values scattered everywhere, and no one knew the right data value.

The extract programs coupled with the many applications led to data disintegrity. The problem became not finding data, but finding believable data.

The problem of data disintegrity was an architectural problem, not a technical problem. Adding more technology to the organization only made the problem worse, not better.

What was needed was a basic transformation from application data to enterprise data. But transforming data from application to enterprise was not the only problem. In addition, storing data for lengthy periods of time became necessary. Before the data warehouse application, transaction processing stored data for the minimum possible time, typically a few weeks to a month. If application data were stored for longer periods, transaction response time began to suffer. So online transaction processing applications jettisoned data as soon as possible to preserve response time.

However, it was discovered that there was value in storing data for more than a few weeks. Historical data became important data, and there was no place for historical data in OLTP. Historical data was useful in finding and analyzing the habits of consumers.

Because of the need for an enterprise view of data and a need to store data for a lengthy period, an architectural construct grew called the data warehouse.

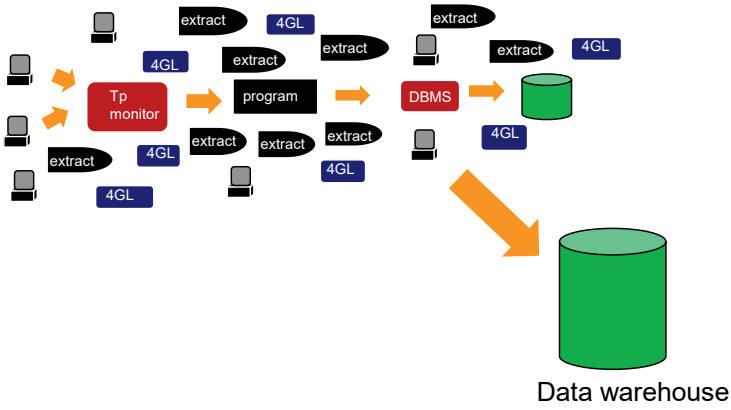


Figure 18.7: Out of the miasma came the data warehouse.

The data warehouse

The data warehouse became an ideal place for analytical processing. With the data warehouse, there was:

- An enterprise view of data
- Immediately available data fit for analysis
- Granular data that could be reshaped in many ways
- Historical data that could be used for lengthy analysis.

Before the data warehouse, analytical processing was almost unknown. The data warehouse provided the enterprise view of data. But we learned quickly that different organizations needed that enterprise data to be reshaped into a form and structure familiar to the end

user. Fortunately, the data found in the data warehouse was granular and could support many different views of the data. Soon marketing wanted their own data. And sales wanted their own data. Finance wanted their own data, and so forth. The granular data found in the data warehouse could support all the different needs of the data and, at the same time, provide a foundation for reconcilability, if reconcilability ever became an issue.

Data marts

To satisfy the need for the parochial uses of the data, there grew an architectural structure called the data mart.

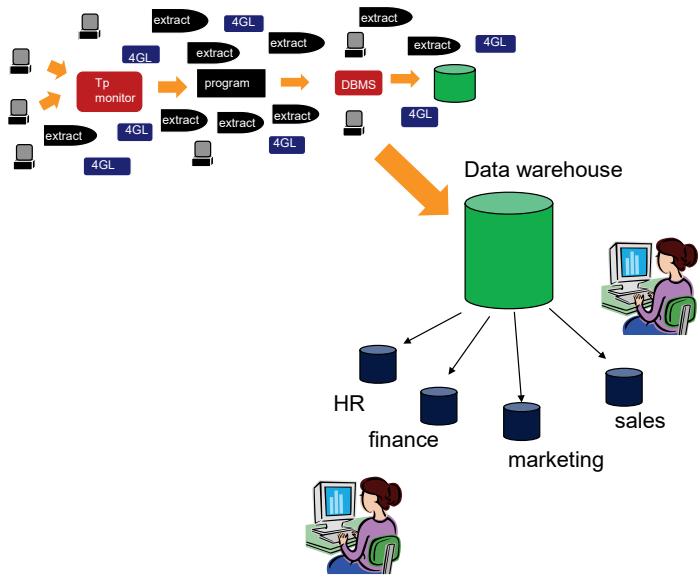


Figure 18.8: Then came data marts.

The data mart used the granular data found in the data warehouse and reshaped that data into the form and structure suitable to the end user's needs. It became possible to reconcile differences between different departments because all of the departments started with the same source of granular and integrated data: the data warehouse.

The architecture of the data warehouse continued for quite a while (and it continues even today).

The Internet and IoT data

Another form of data to make an appearance was from the Internet. The Internet opened up a wide world of diverse data. It was possible to find many diverse forms of data about every imaginable subject on the Internet. Furthermore, a terrific volume of data could be found on the Internet. And the data on the Internet came from a worldwide foundation of sources.

Another type of data that made its appearance was machine-based data. This machine-based data was called analog or IoT (internet of things) data. Machines were capable of producing a huge volume of data as they operated. Some of the analog data was very valuable. Much of the machine-generated data was of little or no value.

The organization faced the problem of trying to make a rational organization of these very different forms of data.

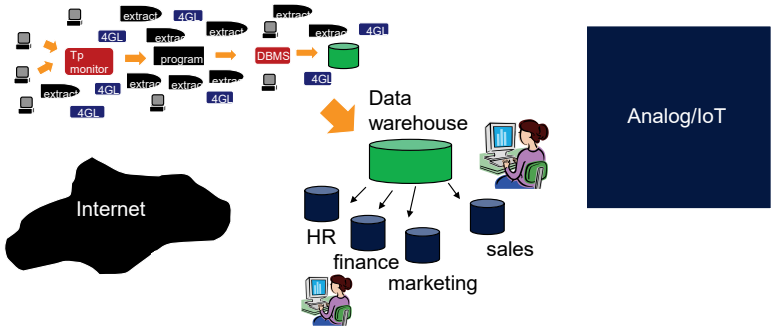


Figure 18.9: Next came the Internet and IoT data.

The data lake

Into the fray of technology and data came the architectural structure called the data lake. Certain vendors came along and declared that all data could be thrown into what was a data lake. Once the data was in the data lake, it could be analyzed. At least, that was the theory. What transpired was that people put their data into the data lake only to discover that it was never used for analysis.

Quickly the data lake turned into a data swamp. Or data sewer.

There were lots of reasons why no one used the data found in the data lake. The data in the data lake was unintegrated and, therefore, no one knew what anything was. The data

lake was large and no one could find anything in it. The data lake did not attempt to integrate its data. There was no way to rationally relate one type of data in the data lake to any other type of data in the data lake. Because the data was in such terrible shape, no one could relate one data element to another.

And there were other reasons why the data lake was an architectural failure.

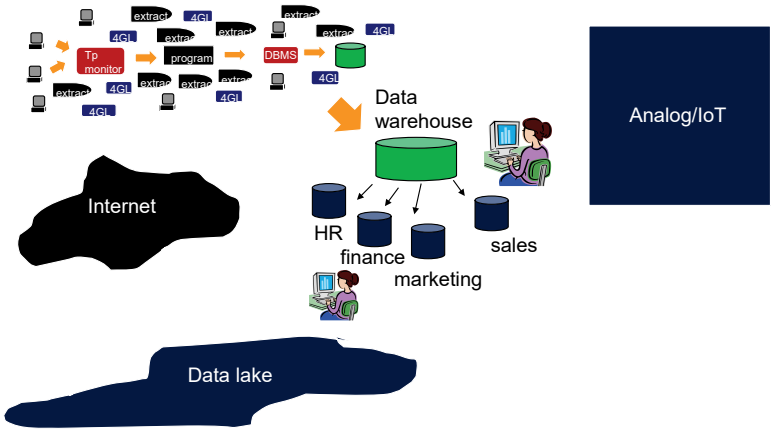


Figure 18.10: Then the data lake appeared.

The data lakehouse

From the mess that was the data lake came the data lakehouse. The data lakehouse added features to the data lake. An analytical infrastructure was added and data was integrated before placing it into the data lakehouse.

Once these two features were added, the data lakehouse became a viable architectural component to serve the architectural and analytical needs of the organization.

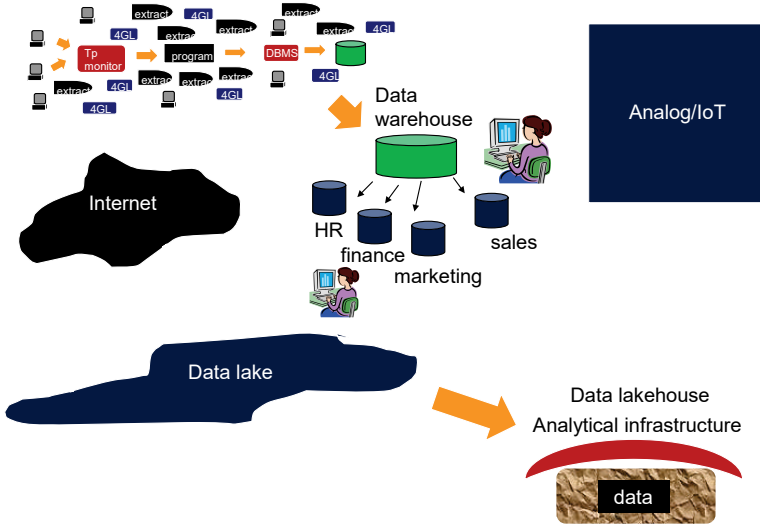


Figure 18.11: Finally, the data lakehouse and bedrock data emerged.

Summary

Is the data lakehouse the final architectural form of data architecture? The answer is absolutely not. While the data lakehouse is a sophisticated architectural answer to the needs of today, there will likely be future architectural enhancements and additions to the data lakehouse to support tomorrow's needs.

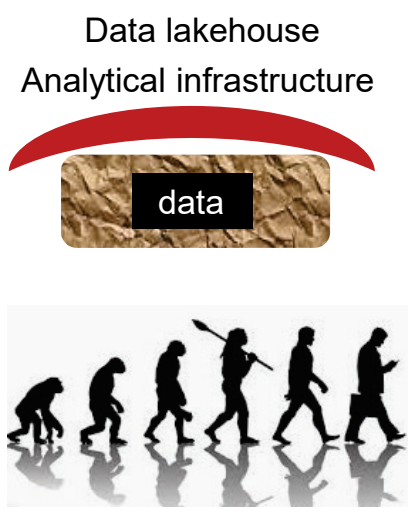


Figure 18.12: The data lakehouse will continue to evolve.

Index