WHITEPAPER

ARCHITECTING AZURE DATA LAKES

TALAVANT

FEBRUARY 2018

TALAVANT

ABOUT TALAVANT

Moving Information Forward.

We exist because there is a better way to make data work for business - better resources, strategy, sustainability, inclusion of the company as a whole, understanding of client needs, tools, outcomes, better ROI.

We're different. Not only are our methods inclusive of all facets of a business – no matter the level of service you need, but we also commit to investing in our own knowledge and growth, and apply it to your people, processes and technologies.

Microsoft Gold Partner

ARCHITECTING AZURE DATA LAKES

CONSULTANTS:

Jordan Anderson
Senior Manager of Consulting Services | Talavant
Madison, WI

Sean Forgatch
Senior Consultant | Talavant
Milwaukee, WI

TABLE OF CONTENTS

TALAVANT

EXECUTIVE SUMMARY

ABSTRACT

As the volume, velocity, and variety of data grow, organizations increasingly depend on data lakes for data storage, governance, blending, and analysis.

A data lake enables organizations to collect a larger volume and variety of data without the rigidity and overhead of traditional data warehouse architectures. Additionally, data lakes provide a place for data-focused users to experiment with datasets and find value without involving IT or spinning up a large project. This Whitepaper outlines the benefits of a data lake solution and defines an architecture for data ingestion, storage, curation, governance, and organizational distribution.

THE CASE FOR A DATA LAKE

Traditional enterprise data warehouses (EDW) and data marts require planning, design, modeling, and development before data is made visible to end-users. During this period, usually days to weeks, key elements in the business may have changed, requiring re-design and protracting time-to-value. EDW rigidity and rigor often entice end-users to build their own solutions using spreadsheets, local databases, and other proprietary tools. This inevitably creates data silos, shadow IT, and a fragmented data landscape. Furthermore, the scarcity of cataloged business data resources limit the data that the organization uses to answer business questions, resulting in decision makers acting on incomplete information.

A well-designed data lake balances the structure provided by a data warehouse with the flexibility of a file system. It's important to understand that a data lake is different than a file system in that raw data cannot simply be added to the lake and made available to the organization. Process and design are still required to enable end-users, and security and governance still apply. The application of our recommended architecture will enable the data organization to be nimble while retaining control.

AUDIENCE

While the beneficiaries of a data lake solution span the entirety of the organization, the users that access the lake directly are limited. Not all business users will be interested in accessing the data lake directly, nor should they spend their time on data blending and analysis. Instead, the whole of the lake is made available to data scientists and analysts, while a vetted and curated dataset is made available to the organization at large.

ARCHITECTURE

OVERVIEW

The overall architecture and flow of a data lake can be categorized into three primary pillars: data lake operations, discovery, and organization.

1

ORGANIZATION

DESIGN patterns of our data lake are going to be the foundation of our future development and operations. Essentially, at the highest of levels, you need a place to land, stage, and surface data. The biggest difference between traditional systems is that when we land the data, it will live there indefinitely. Though Azure Data Lake is specifically designed for large scale analytics – and usually housed on an HDFS platform – it can also be built on Azure Blob Store or even SQL Server, depending on the specific use-case.

SECURITY is necessary to enforce not only compliance but also to keep the lake from becoming a "data swamp" or a collection of unorganized data within the data lake. Azure Data Lake allows users to utilize Azure Active Directory to control enterprise security as well as security features specific to Azure Data Lake that controls access.

2

OPERATIONS

DATA MOVEMENT involves the ingestion and extraction of data in the data lake. Data might be pushed or pulled depending on the technology chosen and purpose for the movement of data. Ingestion is the most critical component as our source systems can produce a variety of data streams and formats.

DATA PROCESSING, as it relates to data lakes, involve both real-time and batch processing. This could occur both internally and externally to the data lake. The reason this is included is that a streaming pipeline, in most cases, would route both to a real-time dashboard as well as to the data lake for later batch processing. Data volume, velocity, and business requirements help determine the ultimate pattern for processing data.

ORCHESTRATION in essentially cloud automation. Giving us the ability to execute multiple processes on multiple systems within a data pipeline, and scheduling those pipelines.

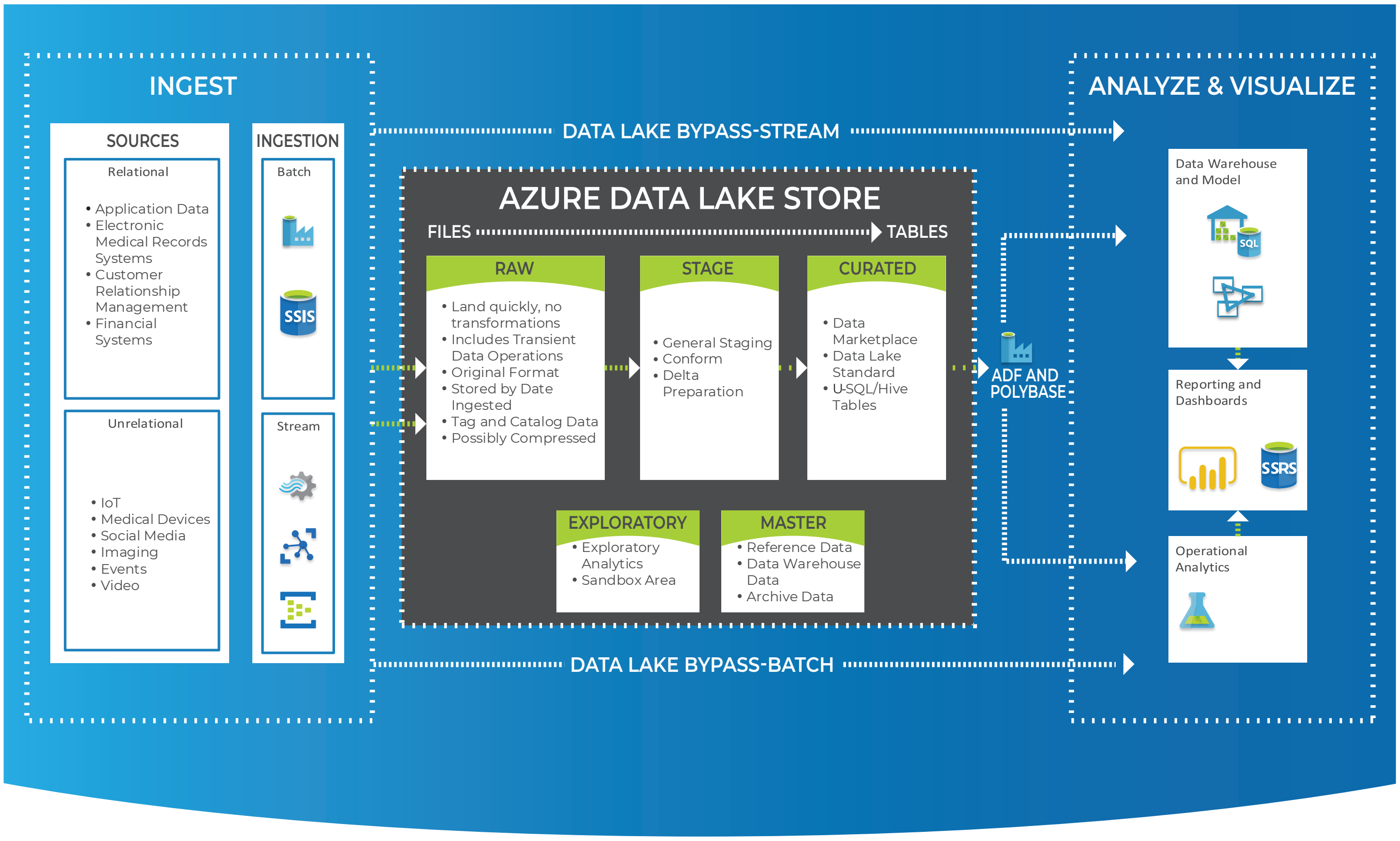
3

DISCOVERY

METADATA is the data about data. This ranges from data lineage such as the format and schema, to capturing information about which source it came from, what day and time it was captured on, as well as the data location.

TAGGING is what drives data discovery in an organization. In a data lake, we could potentially have petabytes of data. Much of which might not be funneling to our production systems yet, because value has not been identified. Tagging is both an automatic and manual business process, and is unique to an organization.

ARCHITECTURE



1

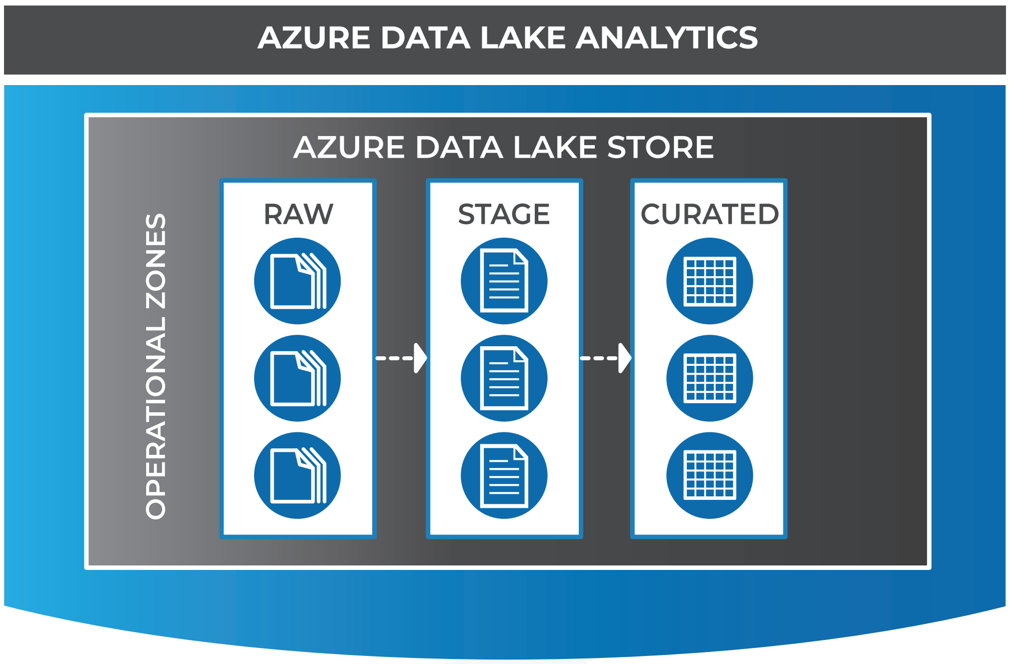
ORGANIZATION

DESIGN

The overall design of a data lake is vital to its success. Coupled with security and the right business processes, the data lake will provide valuable new insights; ultimately providing more tools to executives for making strategic decisions.

Nonetheless, if a data lake is designed poorly without a method of organization, it can quickly become a collected mess of raw data with no efficient way for discovery or smooth acquisition, and later increasing ETL development time, efforts, and limiting success.

The first and foremost design effort should be focused on the storage hierarchy design and how data will flow from one directory to the next. Though this is a simple concept, as laid out below, it will set up the entire foundation of how a data lake is leveraged from the data ingestion layer to the data access layer.



Data Lake Zones

OPERATIONAL ZONES

Raw Zone – This is the first area in the data lake where data is landed and stored indefinitely in its native raw format. The data will live here until it is operationalized. Operationalism occurs once value has been identified by the organization, which does not occur until a measurement has been defined. The purpose of this area is to keep source data in its raw and unaggregated format. This may be that we want to start capturing social media data, but don't know just how we will use it. Access to this area should be restricted from most of the business. Transformations should not occur when ingesting into the RAW zone but rather the data should be moved from source systems to the raw zone as quickly and efficiently as possible.

Data tagging is also included in this zone. This is where both automated and manual tagging can be accomplished. Tagging of datasets can be stored within Azure Data Catalog. This will allow business analysts and subject-matter experts to understand what data lives not only in Azure Data Lake, but Azure at large.

The folder structure for organizing data is separated by source, dataset, and date ingested. Big data tools such as U-SQL allow for utilization of data across multiple folders using virtual columns in a non-iterative manner.

Stage Zone – The Stage zone is where we land our data for processing and ultimately prep the data to be loaded into a Hive or U-SQL table in the Curated Zone. Normal operations in this zone include decompression, cleansing, and aggregation. The results of these activities will often be stored in an output file. From here, the data can go two places, either to a curated table, or directly to an analytical store such as an Azure Data Warehouse, utilizing Polybase for data acquisition.

Curated Zone – The Curated zone is your trusted layer of data within the data lake. It is now decompressed, cleansed, and ready for storage. This is your database layer in either U-SQL or Hive. Data here are exclusively separated by their own databases, and are conformed and summarized. This is the primary level of access to data where security allows for self-service BI and exploratory analytics. Depending on tool selection, data may need to be extracted back to a file for consumption outside of the data lake.

The zones listed above account for the primary zones within a data lake. Apart from these zones are other supporting zones that may be required to enable deeper analytics inside the data lake.

SUPPORTING ZONES

MASTER Zone – Master data will always be trusted data and will often source from the data warehouse. This can also include archived data from a data warehouse if such practices are implemented. This data is going to be used to support analytics within the data lake.

EXPLORATORY Zone – This is an open area where data scientists and analysts can discover new value from the data lake. They can leverage multiple zones to find the best possible value from new data. This zone should be organized by user and project.

TRANSIENT Zone – Acts as a temporary zone and supports ingestion of the data. A use case for this zone is that you may want to decompress data in this zone if you are moving large amounts of compressed data across networks. The data should be short lived, hence the name Transient.

SECURITY

In a big data solution, data must be secured in transit and at rest. There must be a way to limit visibility for circumstances such as conforming to compliance standards. Azure Data Lake provides enterprise grade security in the areas of authentication, auditing, and encryption.

Data Access Control – There are two Access Control Lists within Azure Data Lake, Access ACLs and Default ACLs. The Access ACL controls the security of objects within the data lake, whereas the Default ACLs are predefined settings that a child object can inherit from upon creation.

At a high level, a folder has three categories of how you can assign permissions: “Owners”, “Permissions”, and “Everyone Else”. Each of which can be assigned Read, Write, and Execute permissions. You have the option to recursively apply parent permissions to all child objects within the parent.

It’s important to have the security plan laid out at inception, otherwise, as stated above, applying permissions to items is a recursive activity. Access settings such as Read, Write, and Execute can all be granted and denied through the Azure Portal for easy administration as well as automated with other tools such as Powershell. Azure Data Lake is fully supported by Azure Active Directory for access administration.

Data Security – Role Based Access Control (RBAC) can be managed through Azure Active Directory (AAD). AAD Groups should be created based on department, function, and organizational structure. It is best practice to restrict access to data on a need-to-know basis. How that plan is laid out depends on the specific security policy of the company. When designing a data lake security plan, the following attributes should be taken into consideration.

1. Industry specific Security – If you are working under HIPAA for patient records, it is required that users only have access to a patient's record if it's in conjunction with their specific role. So protected health information (PHI) may need to be scrubbed before data can be surfaced for mining and exploration. If this is the case, it could change the security plan to be more restrictive.

2. Data Security Groups – These should align with the flow of the data through the data lake. Depending on how the data lake will be surfaced to end users and services, data will be more restricted at ingestion into the data lake and become more available as it's curated. The data should be as explorable as possible without increasing risk.

3. Apart from data security at the data lake store level, keep in mind the applications that will be layered over the store itself.

For instance, Azure Data Lake Analytics, where you pay by the query. You will want production access restricted to the service accounts running those jobs. Environments can be managed by creating additional Azure subscriptions.

Data Encryption – Data is secured both in motion and at rest in Azure Data Lake Store (ADLS). ADLS manages data encryption, decryption, and placement of the data automatically. ADLS also offers functionality to allow a data lake administrator to manage encryption.

Azure Data Lake uses a Master Encryption Key, which is stored in Azure Key Vault, to encrypt and decrypt data. Managing keys yourself provides some additional flexibility, but unless there is a strong reason to do so, leave the encryption to the Data Lake service to manage. If you choose to manage your own keys, and accidentally delete or lose them, the data in ADLS cannot be decrypted unless you have a backup of the keys.

KEY TYPES

1. **Master Encryption Key (MEK) –** The ADLS Master Encryption Key stored in Azure Key Vault
2. **Data Encryption Key (DEK) –** Encrypted by the MEK. Generates the BEKs for the data.
3. **Block Encryption Key (BEK) –** Unique encryption keys generated for each block of data associated with an ADLS data file. The BEK is what's used to decrypt the actual data.

2

OPERATIONS

DATA MOVEMENT

Data Movement includes primarily the tools, practices, and patterns of data ingestion into the data lake. Processing within the data lake and extraction out of the data lake can also be considered data movement, but much attention should be directed at the ingestion level due to the various sources and tool types required. A data lake is often designed to support both batch ingestion as well as streaming ingestion from IoT Hubs, Event Hubs, and streaming components.

Considerations

1. Metadata – Metadata can be captured either during the ingestion process or in some sort of batch post process within the data lake. A metadata strategy should already be in place before data ingestion planning begins. The strategy should include knowing what data is going to be captured at ingestion, which can be captured in Azure Data Catalog, HCatalog, or a custom metadata catalog that will support data integration automation as well as data exploration and discovery. The format of the data is also important; what format will be used later in the data lake for processing? It may be more beneficial to use various HDFS specific files such as AVRO depending on the scenario.

2. Batch and Real Time – Azure Data Lake and associated tools for HDFS allow you to work across thousands of files at petabyte size. A streaming dataset may place small files throughout the day, or a batch process may place one or many terabyte size files per day. Partitioning data at ingestion should be considered, and is useful for gaining maximum query and processing performance.

Also consider that not all organizational data flows will include a data lake. For example, SSIS packages integrate relational databases with your data ware-house.

3. Cloud and On-Premise – Many organizations will want to ingest data from both on-premise and cloud sources. It can sometimes be time-consum-ing to transfer large amounts of data from on-premise systems to Azure. If this become a bottleneck, Microsoft provides a private connection to Azure via Azure Express Route.

PROCESSING

Once ingestion of data has been completed, the next step is to begin any processing of the data while it resides in the data lake. This can involve data cleansing, standardization, and structuring. Proper data processing tool selection is critical to the success of a data lake.

Within Azure, there are two groups of tools available for working with Azure Data Lake. Azure Data Services and Azure HDInsight. Though similar in nature, one is not a direct replacement of the other, in some cases they directly support each other just as a data lake can support an enterprise data warehouse. For most cases, Azure Data Services will be able to solve the lower 80 percent of data lake requirements, reserving HDInsight for the upper 20 percent and more complex situations. Azure Data Services is all about bringing big data to the masses, allowing organizations to acquire and process troves of data as well as gain new insights from existing data assets through machine learning, artificial intelligence, and advanced analytics.

Platforms and Tools

The following diagram outlines the Azure Data Services and Azure HDInsight tools available for working alongside Azure Data Lake. Though many modern tools have multiple functionalities, the primary use is marked.

TOOL	STAGE						PLATFORM
	INGEST	PROCESS	STORE	ANALYZE	VISUALIZE	EXPLORE	
ADLCopy	○						Azure Data Services
Analysis Services				○			Azure Data Services
Blob Store			○				Azure Data Services
Data Catalog	○					○	Azure Data Services
Data Factory	●						Azure Data Services
Data Lake Analytics		○		○			Azure Data Services
Data Lake Store	○		○				Azure Data Services
DistCp	○						Azure HDInsight
Event Hubs							Azure Data Services
Hadoop		○	○				Azure HDInsight
Hbase		○	○				Azure HDInsight
Hive/Hcatalog		○					Azure HDInsight
Hive LLAP	○	○					Azure HDInsight
IoT Hubs	○						Azure Data Services
Kafka		○					Azure Data Services
Pig	○	○					Azure HDInsight
Polybase		○					Azure Data Services
Power BI					○	○	Azure Data Services
R Server				○	○		Azure Data Services
Reporting Services					○		Azure Data Services
Spark		○		○			Azure HDInsight
SQL Data Warehouse	○	○	○				Azure Data Services
Sqoop	●						Azure HDInsight
SSIS-IR		○					Azure Data Services
Storm	○	○					Azure HDInsight
Stream Analytics				○			Azure Data Services
Zeppelin					○		Azure HDInsight

● = *Also provides orchestration*

Part of designing a processing pattern across a data lake is choosing the correct file type to use. Listed in the below two diagrams are a list of modern file and compression types. It is often a trade-off between speed and compression performance, as well as supported metadata features.

Files

FILE TYPE	CAPABILITY
Avro	Compressible; splittable; stores schema within the file; good for unstructured and schema differentiating data
Parquet RC File	Columnar format; compressible
Sequence	Columnar format and similar to compressible; splittable; packs smaller files into a Sequence file; improves processing in HDFS
CSV/Text	Commonly used in nearly every organization; easily parsed; often a good use case for bulk processing; not always best choice for HDInsight depending on use case

File Compression

COMPRESSION TYPE	CAPABILITY	USE
Bzip2	High compression; low speed; works well for archival purposes, not HDFS queries	Archiving
Gzip	Mid compression; mid speed	Avro, Sequence
LZO	High speed; lower compression; works well for text files	Text
Snappy	High speed; lower compression	Avro, Sequence

Tables

When processing data within Azure Data Lake Store (ADLS), it is a common practice to leverage fewer larger files rather than a higher quantity of smaller files. Ideally, you want a partition pattern that will result in the least amount of small files as possible. A general rule of thumb is to keep files around 1GB each, and files per table no more than around 10,000. This will vary based on the solution being developed as well as the processing type of either batch or real-time. Some options for working with small files would be combining them at ingestion and expiring the old (Azure Data Lake has simple functionality for automatically expiring files) or inserting them into a U-SQL or Hive table, and then partitioning and distributing those tables.

U-SQL and Hive processes data in Extents and Vertices. Vertices are packages of work that are split across the various compute resources working within ADLS. Partitioning distributes how much work a Vertex must do. An Extent is essentially a piece of a file being processed, and the Vertex is the unit of work.

Table Partitioning in Azure Data Lake – Table partitioning within Azure Data lake is a very common practice that brings significant benefits. For example, if we have a directory that is distributed by month, day, and the specific files for that day, without partitioning, the queries would have to essentially do a full table scan if we wanted to process the previous month's data. If we distribute this data by month, we would only have to read 1 out of 12 partitions. Conversely, let's say we were tracking orders of customers, we would not necessarily want to partition by a customer ID as that could result in many small files which would most likely be the contributor to increased overhead and high utilization of compute and memory resources.

Partitioning considerations

1. **Partition Key should have a low number of distinct values**
2. **Avoid partitioning until partitions would be at least 1GB in size**
3. **Avoid too many small files**

Table Distribution in Azure Data Lake – Table Distribution is for finer-level partitioning, better known as Vertical Partitioning. For our last example in discussing Table Partitioning, we reviewed when we wouldn't use Table Partitioning for customer orders. We could however utilize Table Distribution for something more granular such as customers. Table Distribution allows us to define buckets to direct where specific data will live. In a WHERE predicate, the query will know exactly which bucket to go to instead of having to scan each available bucket. Distribution should be configured such that each bucket is similar in size. Remember we still want to avoid many small files here as well.

Incremental Updates in Azure Data Lake – How will updates be handled? In Azure Data Lake, we cannot update an existing file after it has been created like we would in a traditional relational database system such as SQL Server. Instead, we need to load appended files, and construct our program to view/retrieve the master set plus the appended file or just the appended file. This is what occurs when doing an INSERT. Utilizing a Hive or U-SQL table, we can lay schema across our data files as well as table partitions to drive query performance and functionality. Sometimes it is a better approach to drop and recreate tables instead of trying to manage many small append files.

Orchestration

Orchestration – better understood as simply cloud automation or job automation – is the process of scheduling and managing the execution of processes across multiple systems and the data that flows through those processes.

An example of a big data pipeline across Azure would be ingestion from an Event Hub where we are capturing factory sensor data, processing that data through Azure Machine Learning and]surfacing the results in a PowerBI dashboard in real-time. This entire process is orchestrated through Azure Data Factory (ADF). We will refer to workflow automation as Orchestration. In the HDInsight environment, some tools are better suited to solving certain problems than others. Therefore, an orchestration solution is what enables us to work across an ever-increasing variety of technology and tools.

It is a common misconception that ADF has replaced mature ETL tools such as SQL Server Integration Services (SSIS). ADF can be understood as a higher-level orchestration layer that can call and pass data to other tools such as SSIS, U-SQL, Azure Machine Learning, and more.

3

DISCOVERY

METADATA AND TAGGING

Metadata-capture and tagging are closely related, and support data exploration and governance within the business.

Metadata is what we talk about when capturing data about data, such as the source that the data originated from, how the data are structured and formatted, and when it was captured.

Tagging is used to understand the inventory of a Data Lake through the notion of tags, similar to those attached to social media and blog posts. Consider a data lake in a healthcare organization is housing data from multiple electronic medical record systems. You might have a tagging strategy of *EMR > System Name > Patient Encounters* (the dataset). This allows individuals throughout the organization to search for EMR system data by Tag.

When devising a tagging strategy, the following should be considered:

- When does the data get tagged?
- Who will own and be responsible for tagging the data sets?
- What will the naming convention be so that data sets can be easily discoverable by other business units?

METADATA MATURITY:

Traditional Metadata – This includes information about flat files, tables, connection strings, file store location, file name, data types and length, etc. You can leverage this metadata in patterned ETL development and management.

Modern Metadata – Utilizing file formats that allow metadata to be stored within a data file. Storing metadata within the file itself assists in solving problems such as evolving schema changes.

Advanced Metadata – Automated processing and pre-defined aggregations upon loading into the data lake. If you have worked with Microsoft's PowerBI application, an example would be the Quick Insights feature that shows simple relationships in the data and quick aggregations of the data.

CONCLUSION

Data lakes are an important tool for organizations seeking data operations flexibility and agility while retaining governance.

Azure Data Lake empowers organizations to harness data without traditional database rigidity and overhead, drastically reducing time-to-value.

Utilizing the foundational concepts and tools outlined in this Whitepaper, the implementation of a successful data lake will become an integral part of your organization's data landscape and drive innovation and performance through enhanced decision-making capabilities.

THANK YOU

Moving Information Forward.

talavant.com