Modern Data Architecture on Azure

Design Data-centric Solutions on Microsoft Azure

Sagar Lad

Apress®

Modern Data Architecture on Azure

Design Data-centric Solutions on Microsoft Azure

Sagar Lad

Apress®

Modern Data Architecture on Azure: Design Data-centric Solutions on Microsoft Azure

Sagar Lad
Navsari, Gujarat, India

ISBN-13 (pbk): 978-1-4842-9759-9 ISBN-13 (electronic): 978-1-4842-9760-5
https://doi.org/10.1007/978-1-4842-9760-5

Copyright © 2023 by Sagar Lad

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Development Editor: Laura Berendson
Editorial Assistant: Shaul Elson
Copy Editor: Kezia Endsley

Cover designed by eStudioCalamar

Cover image designed by Freepick

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub (github.com/apress). For more detailed information, please visit https://www.apress.com/gp/services/source-code.

Paper in this product is recyclable

अनुगृहिता अस्म्यहम्

This book is dedicated first, to my wonderful wife Vini for being the perfect life partner and the one who I can always count on. Secondly, to all my friends cum big brothers in Netherlands: Priteshbhai, Nirmalbhai, Siddeshbhai, Abhishekbhai and Pradipbhai, who have become family now.

Table of Contents

About the Author xi

About the Technical Reviewer xiii

Acknowledgments xv

Introduction xvii

Chapter 1: Introduction: Fundamentals of Data Management 1

Introduction to DAMA and DMBOK 2

Essential Data Concepts 3

Types of Data 3

Data Management Principles 6

The Data Lifecycle 7

Consistency Models 8

Data Ingestion Patterns 8

Data Platform Paradigm 9

Data Management Principles and Challenges 12

Preparing a Data Strategy 12

Data Quality Measurements 15

Dealing with Substantial Volumes of Data 17

Data Management Frameworks 20

The Strategic Alignment Model 20

The Amsterdam Information Model 22

The DAMA DMBOK Framework 23

The DAMA Wheel....29
Data Governance....30
Data Architecture....31
Data Modeling and Design....32
Data Storage and Operations....33
Data Security....33
Data Integration and Interoperability....34
Document and Content Management....34
Reference and Master Data....34
Data Warehousing and Business Intelligence....35
Metadata....35
Data Quality....36
Understanding the Environmental Factors Hexagon....37
Understanding the Knowledge Area Context Diagram....38
Conclusion....39

Chapter 2: Build Relational and Non-Relational Data Solutions on Azure....41

Data Integration Using ETL....42
Data Extraction....43
Data Transformation....44
Data Loading....44
Designing ELT Pipelines Using the Azure Synapse Server....45
Online Analytical Processing for Complex Analyses....47
Semantic Data Modeling....51
Challenges of Using OLAP Solutions....53
Managing Transaction Data Using OLTP....54

Managing Non-Relational Data 60
Key-Value Pair Databases 61
Column Family Databases 62
Document Databases 62
Graph Databases 63
Handling Time-Series and Free-Form Search Data 65
Working with CSV and JSON Files for Data Solutions 72
Conclusion 75

Chapter 3: Building a Big Data Architecture 77

Core Components of a Big Data Architecture 78
Data Ingestion and Processing 79
Data Analysis 81
Data Visualization 82
Data Governance 83
Using Batch Processing 83
Azure Synapse Analytics 86
Azure Data Lake Analytics 86
Azure Databricks 87
Azure Data Explorer 88
Real-Time Processing 89
Real-Time Data Ingestion 92
The Lambda Architecture 95
The Kappa Architecture 99
Internet of Things (IoT) 105
Data Mesh Principles and the Logical Architecture 107
Conclusion 112

Chapter 4: Data Management Patterns and Technology Choices with Azure 113

Data Patterns and Trends in Depth 114

CQRS Pattern 114

Event Sourcing 117

Materialized Views 117

Index Table Pattern 118

Analytical Store for Big Data Analytics 120

Azure Synapse Analytics 120

Azure Databricks 122

Azure Data Explorer 125

Building Enterprise Data Lakes and Data Lakehouses 126

Enterprise Data Lakes 127

Enterprise Data Lakehouses 131

Data Pipeline Orchestration 133

Real-Time Stream Processing in Azure 138

Conclusion 141

Chapter 5: Data Architecture Process 143

Guide to Data Modeling 143

Conceptual Data Model 145

Logical Data Model 146

Physical Data Model 147

Focus on Business Objectives and its Requirements 147

Data Lake for Ad Hoc Queries 150

Enterprise Data Governance: Data Scrambling, Obfuscation, and DataOps 155

Data Masking Techniques 158

Master Data Management and Storage Optimization 163

Master Data Management 164
Data Encryption Patterns 170
Conclusion 174

Chapter 6: Data Architecture Framework Explained 175

Fundamentals of Data Modeling 175
The Network Data Model 177
The Hierarchical Data Model 178
The Relational Data Model 179
The Object-Oriented Data Model 180
The Dimensional Data Model 181
The Graph Data Model 182
The Entity Relationship Data Model 183
The Open Group Architecture Framework 184
Preliminary Phase 187
Defining the Architecture Vision 187
Business Architecture 188
Information System Architecture 188
Technology Architecture 188
Opportunities and Solutions 189
Migration Planning 189
Governance Implementation 190
Architecture Change Management 190
DAMA DMBOK 190
The Zachman Framework 195
Conclusion 198

Index 199

About the Author

Sagar Lad is an Azure Data Solution Architect working with a leading multinational software company. He has deep expertise in implementing data management governance and analytics solutions for large enterprises using cloud and artificial intelligence. He has more than ten years of IT experience and is an experienced Azure cloud evangelist with a strong focus on driving cloud adoption for enterprise organizations using Microsoft Cloud Solutions. He loves blogging and is an active blogger on Medium, LinkedIn, and the C# Corner developer community. He was awarded the C# Corner MVP in September 2021 for his contributions to the developer community.

About the Technical Reviewer

Kapil Bansal is a PhD scholar and lead DevOps engineer at S&P Global Market Intelligence, India. He has more than 15 years of experience in the IT industry, having worked in the areas of Azure cloud computing (PaaS, IaaS, and SaaS), Azure Stack, DevSecOps, Kubernetes, Terraform, Office 365, SharePoint, release management, application lifecycle management (ALM), Information Technology Infrastructure Library (ITIL), and Six Sigma. Kapil completed the advanced certification program in strategy for leaders from IIM Lucknow and cybersecurity and cyber-defense from IIT Kanpur.

Kapil has worked with IBM India Pvt Ltd, HCL Technologies, NIIT Technologies, the Encore Capital Group, and Xavient Software Solutions, in Noida, and has served multiple clients based in the United States, the UK, and Africa. This includes T-Mobile, World Bank Group, H&M, WBMI, Encore Capital, and Bharti Airtel (in India and Africa).

Kapil also reviewed the Apress titles *Hands on Kubernetes on Azure,* Practical Microsoft Azure IaaS: Migrating and Building Scalable and Secure Cloud Solutions, Beginning SharePoint Communication Sites, and many more.

Acknowledgments

I wish to express my gratitude to my colleagues cum friends Thimo ten Veen, Ilse Epskamp, and Michael Hoogkamer, for their moral support and for helping me climb the ladder.

Introduction

This book is intended for data solution architects, data engineers, and IT consultants/architects who want practical insights on designing modern data architecture implementations in Azure.

In this book, you:

- Learn about the fundamentals of data architecture, including data management, data handling ethics, data governance, and metadata management.
- Analyze and understand the business needs to choose the best Azure services and to make informed business decisions.
- Learn about Azure cloud data design patterns for relational and non-relational data, batch and real-time processing, ETL/ELT pipelines, and more.
- Modernize your data architecture using Azure as a foundation to leverage data and using AI to enable digital transformation. You learn to secure and optimize overall data lifecycle management.
- Understand the various data architecture frameworks and their best practices.

CHAPTER 1

Introduction: Fundamentals of Data Management

The 21st century is the age of data. Companies need better data-management solutions because there are enormous volumes of it produced every day. Today's successful businesses and organizations must comprehend the what, why, and how of data management.

The process of gathering, storing, and using data in a way that is economical, secure, and effective is known as *data management*. Data management enables individuals, groups, and networked devices to optimize data utilization in order to make good decisions.

This chapter covers the following topics:

- Introduction to DAMA and DMBOK
- Essential data concepts
- Data management principles and challenges
- Data management frameworks
- The DAMA wheel

© Sagar Lad 2023
S. Lad, *Modern Data Architecture on Azure*, https://doi.org/10.1007/978-1-4842-9760-5_1

Introduction to DAMA and DMBOK

DAMA (the Data Management Association) is a non-profit and vendor-independent organization consisting of business and technical professionals that research data and information management. This organization is responsible for the development and execution of best practices, policies, and design architectures for managing the data lifecycle. The DAMA guide, called *DMBOK* (the Data Management Body of Knowledge), is a standard reference guide containing processes, principles, and best practices for data management.

The main goals of the DMBOK are as follows:

- Determine best practices, roles and responsibilities, and maturity models for data management
- Standardize management practices
- Establish a formal vocabulary
- Define the scope of the practices
- Provide a vendor-neutral overview of management practices and alternatives for various scenarios

DAMA is committed to furthering the ideas and methods of data and information management and to help DAMA members and their organizations meet their goals regarding data. Through its community of experts and the establishment of certification and training programs, DAMA sponsors and facilitates the development of the DMBOK in order to achieve this purpose.

The development of DMBOK2 has spanned close to 30 years. All of the contributors are seasoned professionals and many of them have names you are familiar with. The top practitioners in the field today describe what works regarding data management practices, experience, and expression.

DAMA (originally known as the Data Administration Management Association) is a global not-for-profit organization that works to improve ideas and methods related to information management and data management. It describes itself as a vendor-independent, entirely voluntary organization with a membership consisting of business and technical experts. DAMA also has numerous international, continental, and national chapters throughout the world. Its international branch is known as DAMA International (DAMA-I).

Essential Data Concepts

Data is the general term for information in several formats, including text, numbers, photos, and videos. It is essential to today's digital environment because it influences decision-making, provides companies with access to insights, and promotes innovation. Analytics, machine learning, and artificial intelligence are all built on data, which has led to breakthroughs across many industries.

Let's look at the fundamentals of data in more detail.

Types of Data

It is important to understand the type of data you are working with. As Figure 1-1 shows, there are two types of data: qualitative and quantitative. These types can be further classified into four categories.

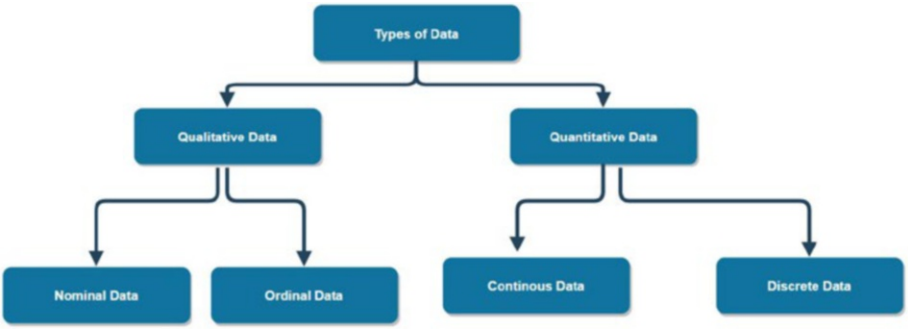

Figure 1-1. *Types of data*

Qualitative Data

Data described in the form of expressions or feelings that can't be expressed in the form of numbers is known as *qualitative data*. It is basically in the form of words or labels that can be collected from documents, audio, video, images, and so on. Qualitative data describes how people see things.

Examples:

- Favorite animal
- Cars

Qualitative data can be further categorized into nominal and ordinal data.

Nominal Data

Variables without any numerical rank are known as nominal data. Since one color cannot be ranked above another, the color of hair might be regarded as nominal data. You can't execute mathematical operations on nominal data in any particular way. This data is split into different categories but lacks any meaningful order.

Examples:

- Color of a car
- Skin color
- Nationality
- Language

Ordinal Data

Data that can be classified in the form of order or rank can be defined as ordinal data. It is a type of qualitative data that is simpler than nominal data. Examples of ordinal data include grades given to the students, such as A, B, and so on. Ordinal data is always ordered but the categorical values are unequally distributed among the classified categories.

Examples:

- Rating or feedback for training
- Rating for an interview
- Economic status

Quantitative Data

Numerical values can be used to express quantitative data, which can be counted. We can also perform statistical data analysis on the top of this quantitative data. It provides a response to the questions "how much," "how many," and "how often." It captures more information about the respective data. Quantitative data includes numbers like price of a car, the height of a building, and so on.

We can statistically manipulate quantitative data. Numerous graphs and charts, including bar graphs, histograms, pie charts, and line graphs, and so on can be used to display quantitative data.

Discrete Data

Discrete data refers to the data that is unique or separate. Integers and whole numbers are examples of discrete data. Discrete data include things like the overall number of employees in the company, for example. We can't convert this type of data into decimals or fractional values.

Examples:

- Price of an iPhone
- Total employees in a company
- Number of months in a year

Continuous Data

Fractional numbers are the representation of continuous data. This could be an Android phone's version, someone's weight, and so on. Information that can be broken down into smaller pieces is represented by continuous data. Any value in a range can be assigned to the continuous variable.

Examples:

- Temperature of a room
- Speed of a car
- Wakeup time

Data Management Principles

Like any other management processes, data management also has a set of principles used to maintain a balance between strategic and operational needs:

- Treat data as an asset
- Data values should be measured in economic terms

- The quality of the data should be managed
- Metadata should be used to manage the data
- Data management is cross functional and requires various skills and expertise
- Data management requirements should be handled from the enterprise point of view
- The data lifecycle should be managed
- Associated risks should also be managed with the data
- Data management requirements and outcome must drive technology decisions
- Successful data-driven initiatives require leadership support and commitment

The Data Lifecycle

The data lifecycle has various phases, whereby data moves from one state to another. The data lifecycle consists of the following major activities:

- *Data ingestion*: Data is collected from various sources via the data pipeline. The data pipeline can be in the form of ETL (Extract, Transformation, and Load) or ELT (Extract, Load, and Transform). Data ingestion gets the data from various sources and makes the data ready for consumption by the end users.
- *Data storage*: Data should be stored in a secure and encrypted format at an agreed-upon location. Data should be easily accessible from this storage location.

- *Data processing and analysis*: This is a set of procedures used on data to extract data in the right output form after being verified, transformed, and integrated. To guarantee the usefulness and integrity of the data, processing procedures must be meticulously documented. Data quality control, statistical data analysis, modeling, and result interpretation are all included in this stage.
- *Data visualization*: The outcome of the data analysis should be in the form of visualization reports that provide business insights.

Consistency Models

There are various types of consistency models for any project implementation:

- *ACID*: Atomic, Consistent, Isolated, and Durable
- *BASE*: Basically Available, Soft State, Eventual Consistency
- *CAP*: Consistent, Available, Fault Tolerant

Data Ingestion Patterns

Data can be ingested in multiple ways. As Figure 1-2 shows, based on the latency requirements, data can be ingested using the batch pattern and the near real-time streaming pattern.

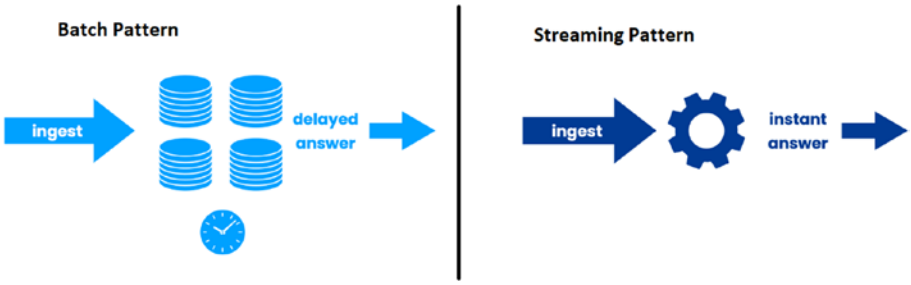

Figure 1-2. *Batch vs streaming pattern*

In the batch pattern, data points that have been grouped together are collected in a specific time interval. In the stream pattern, data is collected continuously from various data sources during an interval window, without any additional latency. Data will arrive as soon as it is available at the source end.

Data Platform Paradigm

The amount of data that businesses collect today is enormous, and businesses rely on that data to make important business decisions, enhance product offerings, and provide better customer service. But if businesses can't use that data right away? What good is it then? How data is stored affects how easily it is to access.

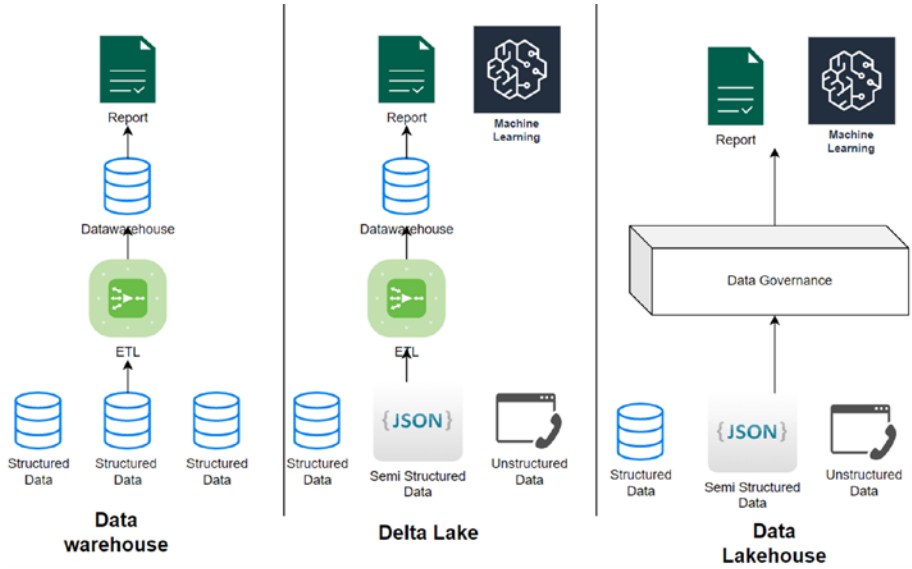

Figure 1-3. *Data warehouse vs data lake vs data lakehouse*

As shown in Figure 1-3, there are three paradigms of data storage as mentioned here:

- *Data warehouse:* A data warehouse is a centralized data repository used by an organization to house enormous amounts of data from various sources. A data warehouse acts as an organization's single source of "data truth" and is essential to reporting and business analytics. Typically, data warehouses combine relational datasets from several sources, such as application, business, and transactional data, to store historical data.

- *Data lake*: A data lake is a centralized, extremely adaptable storage space where massive amounts of organized and unstructured data are kept in their unprocessed, unaltered, and unformatted forms. A data lake uses a flat architecture and object storage in its unprocessed state to store data, as opposed to data warehouses, which save relational data that has previously been "cleaned." Data lakes are adaptable, reliable, and affordable and allow enterprises to obtain advanced insight from unstructured data. This is opposed to data warehouses, which have difficulty handling data in this format.
- *Data lakehouse*: This novel big-data storage architecture combines all the benefits of data lakes and data warehouses. All data, whether organized, semi-structured, or unstructured, may be stored in a single location with the best machine learning, business intelligence, and streaming capabilities possible.

Data lakes of all varieties are typically the starting point of a data lakehouse; the data is then transformed to the *delta lake* format (an open-source storage layer that gives data lakes reliability). Data lakes with delta lakes enable ACID transactional procedures from conventional data warehouses.

So, these are major basic concepts of data that you should understand before you start any data-driven project or an initiative.

Data Management Principles and Challenges

Data is the new oil. Without making data-driven decisions, organizations cannot thrive in the current economic environment. In this digital era, data becomes the key to competitive advantage, meaning a company's ability to compete will increasingly be driven by how well it can leverage data, apply analytics, and implement new technologies.

Managing an unexpected data surplus is a challenge faced by many firms when implementing a data-first strategy. It can, of course, be crippling to have no data. But it can be just as crippling to have more data than you know what to do with. Organizations must prepare a roadmap for gathering, evaluating, and managing their data if they want to take advantage of the potential that data offers.

In order to use the data efficiently and prepare a better data strategy, you need to understand the data management principles in depth.

Preparing a Data Strategy

Creating a data management strategy is one of the most essential data management principles. Organizational efforts need a strategic roadmap for successful data management. Building a strong foundation through a data strategy is crucial to an organization's success. Recommended elements for any data strategy include understanding the importance of gathering relevant data, setting up a platform that promotes data analysis, gaining meaningful insights from the data you collected, and using those insights to enhance decision-making. These are all necessary for harnessing the power of data in an organization.

As shown in Figure 1-4, the data strategy roadmap emphasizes data governance programs, mastering data management of internal and external structured data, and other crucial elements.

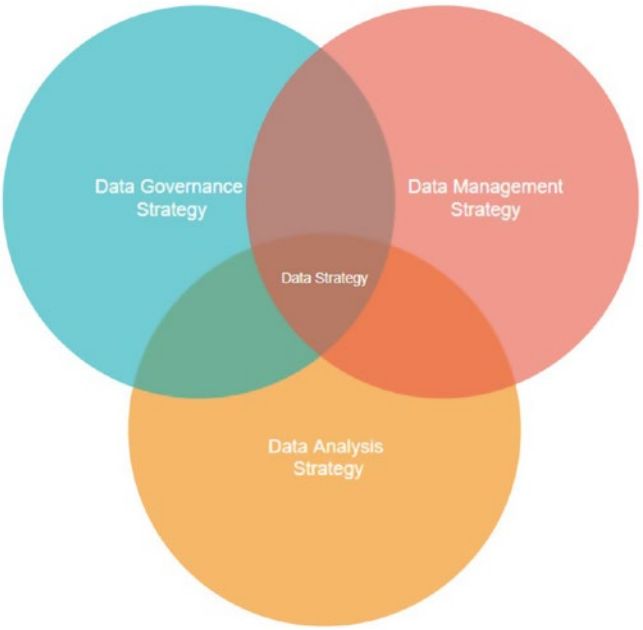

Figure 1-4. *Data strategy pillars*

Defining Roles and Responsibilities

You have to clearly define roles and responsibilities inside the data management system in order to practice good data management. Everyone's job in data management is different but interdependent. There are three main roles in data management organizations:

- *Data stewards*: The administration and supervision of an organization's data assets is known as data stewardship. Stewards enable the delivery of high-quality, consistently accessible data to business users. This is the link between the IT department and the business side of an organization, and they carry out data usage and security standards that have been established through enterprise data governance projects.

- *Data owners*: One or more data sets inside an organization are subjected to classification, protection, use, and quality of a data owner. Data owners make sure that the data glossary is complete and approved. They are also responsible for maintaining the quality of the data.
- *Data custodians*: In order to fulfil the requirements outlined by the data owner in the Data Governance Framework, a data custodian is in charge of putting security controls in place and keeping them up to date for a specific data collection.

These components work together to form a framework for handling data during the course of a project or program.

Data Lifecycle Management

Data lifecycle management is the method used to manage data from the point of data entry to the point when data is destroyed. As shown in Figure 1-5, a successful data lifecycle management process gives a business's data structure and organization, thus enabling important process goals, including data security and availability.

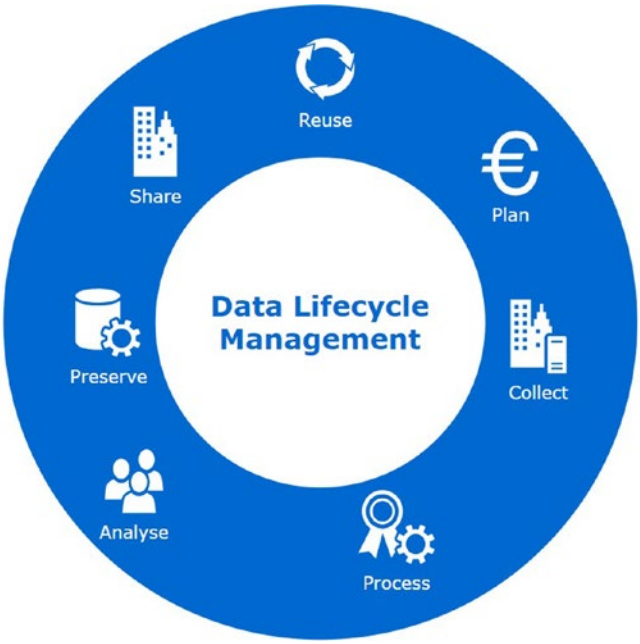

Figure 1-5. *The data lifecycle process*

Data Quality Measurements

Before storing data, it must be evaluated for various data quality factors and it must be validated by the data quality analysts. The analyst then calculates a score indicating the data's overall quality and assigns a percentage grade to the company based on how accurate the data is. Accuracy, completeness, consistency, timeliness, and uniqueness are a few of the data quality dimensions to be considered.

Metadata

Metadata is information that describes another set of information about the data. It helps data users comprehend a data collection more fully. It keeps track of all facets of the data, including how it was gathered

and examined, and it provides insights into the nature, features, and applications of the data. A successful data program cannot be successful without metadata. As shown in Figure 1-6, data and metadata management consists of metadata management, data lineage information, and knowledge graphs.

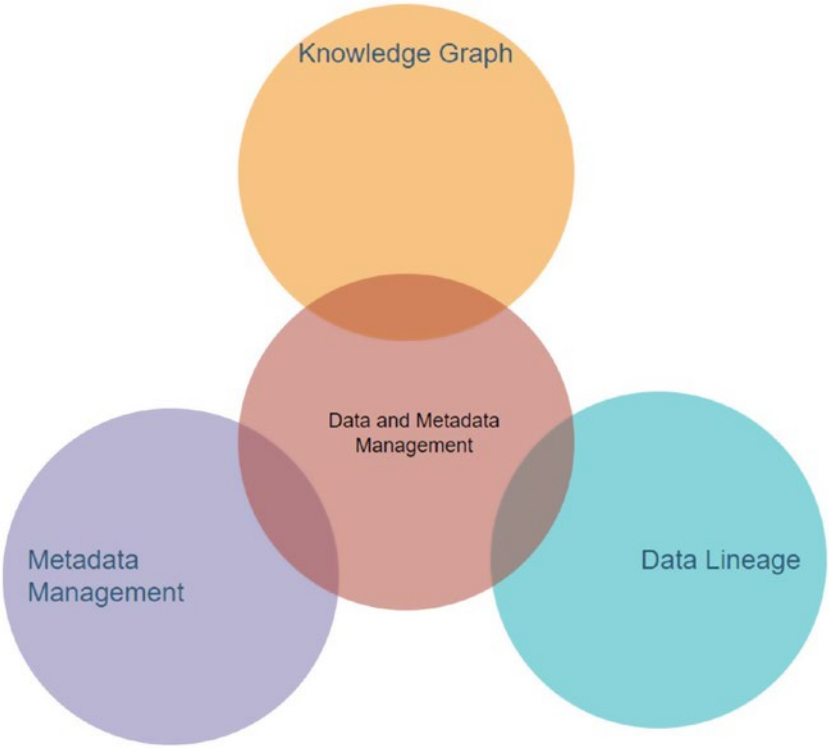

Figure 1-6. *Metadata and data lineage*

Metadata is critical to data management because it provides essential details about data assets:

- What is the data
- When was it created
- Where is it stored

- How has it changed
- Who has access to it
- Who owns it

Maximizing Data Value for Data-Driven Decisions

If a business doesn't make the best possible use of the data it obtains, none of the other data management guidelines matter. Organizations must make sure that data is accessible to everybody who needs it, since it has no value unless it is used. In order to bring the best out of the collected data, companies can take the following actions:

- Define the data strategy and business goals for the data
- Define data standards to create cleansed, user friendly, easily accessible data
- Enable data-driven decision making
- Educate all CXO level people on the potential of using such data to make better decisions

After looking at the fundamentals of data management and its principles, this section turns to explaining the major challenges of any data management project.

Dealing with Substantial Volumes of Data

As you might imagine, managing the complete data lifecycle becomes more difficult as more data is acquired. On top of this, monitoring and validation are needed. By successfully managing large volumes of data, companies can get a deeper understanding of consumer behavior and market trends, enabling them to make smarter actions, improve procedures, and optimize goods and services.

Apart from the volume of the data, the velocity of the data also matters. In order for companies to make better and correct decisions, the data should be used as soon as it is produced and available. Working with old or delayed data can negatively impact the overall decision making process.

Siloed and Varied Data Sources

One of the major challenges faced by organizations is that data is stored and scattered across multiple locations. For example, data is stored in an Azure Cloud, AWS, or on-premises database or application. Having multiple data storage areas leads to data compatibility issues, which must be resolved before you can actually use the data to derive business insights. It also prevents you from having a single source of data; duplicate data will be stored across multiple locations. Data deduplication and having a single source of truth should be a part of strategy of any data-driven organization.

Maintaining the Quality of the Data

One of the biggest challenges facing many firms today is data quality. The majority of organizations use databases to update information, yet it can be challenging to maintain data quality while processing or recording information. Ultimately, you have to remove irrelevant data while keeping hold of high-quality, reliable data that the company will likely need.

Like any other resource, data can be outdated or inaccurate. Making decisions based on this data could cause companies to lose a lot of money. In order to ensure that decisions are based on accurate data, it is crucial to have proper data quality monitoring standards in place.

Data Integration

The end goal of having high-quality data is to enable stakeholders to make better decisions by allowing other business intelligence tools to analyze and process the data. Data integration makes data-driven decisions simpler by seamlessly connecting multiple sources of data.

Data Governance and Security

Data management is the process used to safeguard the value of data, whereas data governance is the process of securing the data. Data management practices are incorporated and specified while developing a data governance strategy. The use of technology and solutions is governed by data governance policies and guidelines, with management relying on these solutions to complete tasks.

The practice of preserving digital information throughout its full lifecycle to defend it against corruption or illegal access is known as data security. It consists of business policies and procedures, as well as technology, software, storage, and user policies.

Data Automation

Data needs to be collected and categorized. Data automation is useful in this situation and will save users a lot of time. It also improves the accuracy by which data is collected. As shown in Figure 1-7, automating the data process simplifies the entire cycle, from data collection to analysis without human intervention.

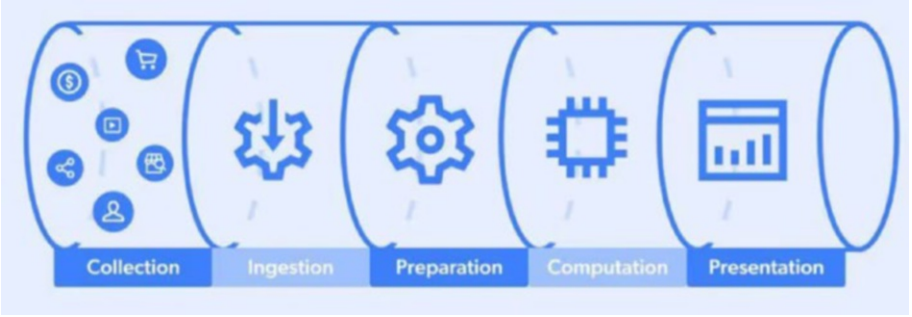

Figure 1-7. *Data pipeline automation*

Data Management Frameworks

Organizations use a data management framework as a set of rules and procedures to manage their data. A framework helps firms ensure that information is accurate, dependable, and consistent. Data governance, data quality, data integration, data security, data privacy, data retention, data architecture, and data analytics are often included in a data management framework.

These three data management frameworks are widely used by various organizations:

- Strategic Alignment Model
- Amsterdam Information Model
- DAMA DMBOK Framework

Let's look at each of them in detail.

The Strategic Alignment Model

One of the most frequently referenced strategic alignment models is the one developed by Henderson and Venkatraman. The two major components of this model are strategic fit and functional integration.

- *Strategic fit*: The alignment of internal and external domains. It shows how the company strategy and the IT strategy are linked.
- *Functional integration*: The two types of integration between the business and IT sectors. The relationship between organizational infrastructure, IT infrastructure, and process is the focus of this integration.

A conceptual model known as SAM has been utilized to comprehend strategic alignment from the viewpoint of four components—business strategy, IT strategy, organizational strategy, and infrastructure strategy. See Figure 1-8.

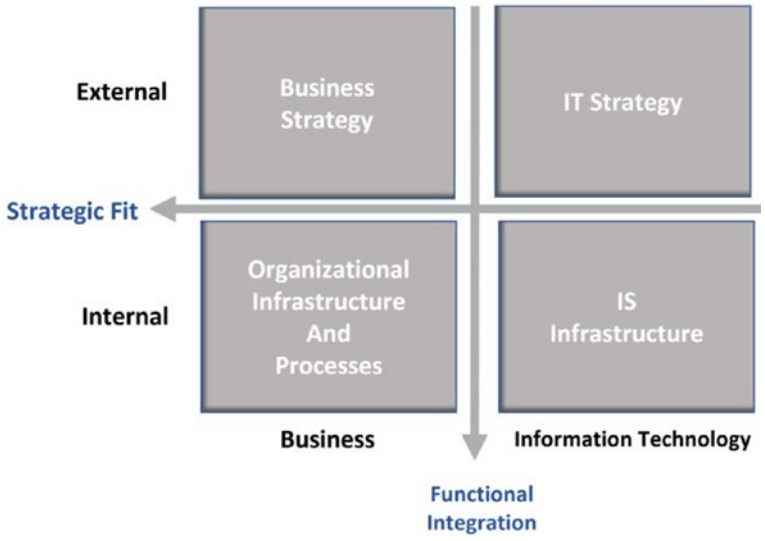

Figure 1-8. *Strategic alignment model (Courtesy: Henderson and Venkatraman)*

Strategic fit and functional integration serve as the model's two main pillars, as shown in Figure 1-8. Strategic fit acknowledges that the IT strategy should be expressed in terms of an internal domain as well as an external domain.

Visual representations of how priorities are connected and aligned both vertically and horizontally in the company are called strategic alignment models. An alignment model is a tool that helps assess how well an organization's longer-term objectives are aligned with its resources and prospects, as well as with risks, vulnerabilities, and management opportunities. An organization will also have a better understanding of how to track the development of strategic alignment.

Clear long-term goals and objectives must be completed to ensure a safe future for the company from the strategic alignment point of view. Each function and department discovers key connections and develops its own initiatives and plans that match and align with division strategies after the enterprise strategy and division strategy have been developed. And last, each manager or department develops its own strategies that relate to its division. This method is frequently described as creating a line of sight across the business or as a strategic fit.

The Amsterdam Information Model

Amsterdam Information Model was founded by Maes, Truijens, and Abcouwer. They created an extended version of the SAM model. The connections between organizations and information are mapped out by the Amsterdam framework for information management.

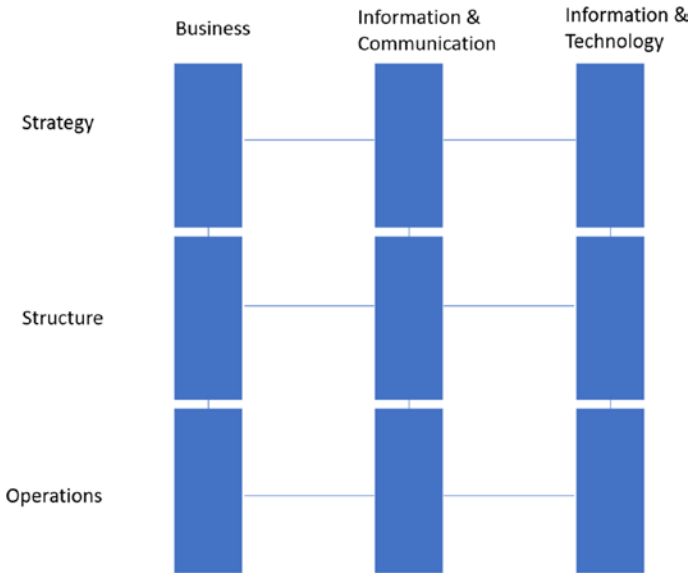

Figure 1-9. *The Amsterdam information model*

As shown in Figure 1-9, by dividing the internal domain into structural and operational levels, this model consists of a core position that addresses business management and design. The additional column is included to separate the technical side from the information use side. The Amsterdam Information Model is a Unified Framework by extending this approach. This is referred to as an effort to turn the idea of alignment into a workable strategy that includes management and design elements.

The DAMA DMBOK Framework

Every business manages data in some capacity. Implementing data management consists of integrating a data management function into the organizational structure of a business. The business must create and deploy a data management framework in order to accomplish this. A data management framework is a group of connected elements that transforms data management into a business function in this context. Models and procedures are the essential elements of the data management framework. A *model* is an illustration of a collection of capabilities for data management. A *method* is a description of how to perform an action.

Design, implementation, maturity assessment, and performance evaluation are the four functions of a data management framework. At different organizational levels—strategic, operational, and functional—you can describe and define data management and a data management role. The functional framework for the DAMA-DMBOK offers a thorough and organized approach to data management. It is made up of a number of parts, each of which represents an important facet of data management.

It is a collection of processes and best practices for managing the data efficiently. Data management describes the process to plan, specify, create, maintain, store, secure, and distribute data over the period of time. The current data management environment might be a complex mix of words, techniques, tools, viewpoints, and hype. The DAMA-DMBOK guide provides concepts and capability maturity models for the standardization of the following activities:

- Data activities, processes, and best practices
- Roles and responsibilities in the data management project
- Deliverables and metrics
- Maturity model

Data management professionals will work more efficiently and consistently if the data management disciplines are standardized. As you can see in Figure 1-10, the data management cycle mainly consists of designing a data architecture followed by storing data in databases such as reference, master, and metadata. Once the data is available, the next step is to focus on the data quality followed by the data integration, data warehousing, and data governance. At the end, data security also matters to securely access and distribute data.

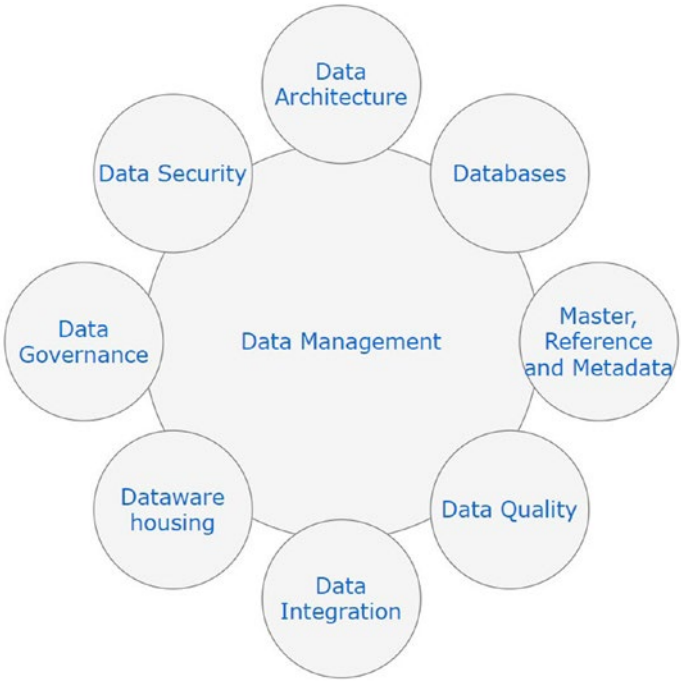

Figure 1-10. *The data management process*

Regardless of the data management framework selected by the company, it should take the same actions to set up a data management function. Businesses should design the necessary capabilities in accordance with the framework they selected. A company's capacity to accomplish tasks or produce results is how the data management framework defines a capability. Each data management capability has five components, as shown in Figure 1-11.

Figure 1-11. *Dimensions of data management capability*

As you can see in Figure 1-11, dimensions of the data management capability start with the input and output data to be used. Once you have the input data, you determine the types of policies and rules to be applied to prepare the data. Once the data is ready, organizational roles should be defined to make sure that access to the data is done via a formal process. Relevant tools and technologies are chosen to make sure that value can be derived out of the data.

There are various types of data management frameworks available in the market. Regardless of any data management framework, the enterprise has to follow these steps for the efficient data management:

1. Determine the scope of the data management framework.

   Depending on various factors, each company's essential data management function has a different scope. The company's needs and resources should be met by the established scope. When developing a formal data management function, a corporation should take into account the following factors:

   - Business needs

     A company's motivation to launch such a project is known as a business need. A company should take both internal and external environmental business factors into account.

     Different laws (like GDPR/BCBS239) and organizational changes (like digital transformation and integration with AI/ML technologies) to improve the customer experience are the most frequent needs.

   - Stakeholder management

     The method by which you manage your relationships with stakeholders is known as stakeholder management. It entails methodically locating stakeholders, assessing their requirements and expectations, and organizing and carrying out various actions to interact with them.

- Enterprise scope

  A business may begin to adopt a data management function for all of its business divisions or just some of them. The chosen business driver will determine the optimal course of action. Data management is a multifaceted field of study.

- Data management capabilities

  A business may require various competencies to satisfy the demands of a certain driver. Data governance, data modeling, data and application architecture, and data quality are typical data management competencies.

2. Execute an organizational maturity assessment.

   Every business manages data in some capacity. Even if there isn't a formal data management division in existence, the organization does have some data management skills. A preliminary maturity evaluation makes it possible to evaluate current talents and identify any future deficiencies.

3. Create a future roadmap for data capabilities.

   A business should determine its long-term strategic vision for data management based on the findings of the gap analysis in the previous step.

4. Implement a data management capability.

   The most effective method for putting in place a data management framework differs from firm to firm and is based on business drivers, size, and resources available to the company. Three main strategies are centralized, decentralized, and hybrid. Each strategy has benefits and drawbacks and is appropriate in different situations.

5. Set up business KPIs.

   A business should set up two different KPIs: one to track the development of the data management capability and the other to gauge operational effectiveness. Different degrees of abstraction can be used to create these KPIs.

The DAMA Wheel

The DAMA-DMBOK framework mainly consists of knowledge areas that help define the scope of the data management framework.

The DAMA wheel has multiple knowledge areas. Data governance is at the center of the wheel, because governance is required to manage the consistency of the overall data management project. Other knowledge areas are around data governance and they manage the balance around the wheel.

Figure 1-12 shows the main components of the DAMA wheel.

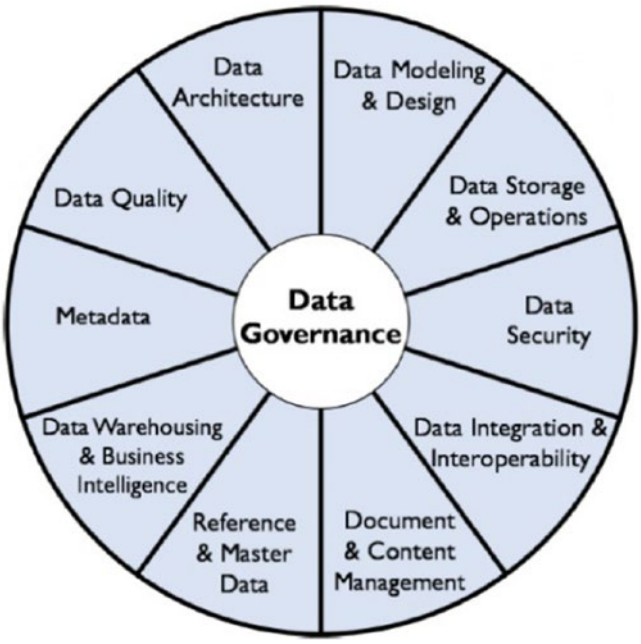

Figure 1-12. *The DAMA wheel (Reference: DAMA International)*

As you can see in Figure 1-12, data governance is at the center of the wheel and other knowledge areas are around data governance. This shows how data governance is correlated with every data management activity.

Data Governance

As explained, data governance defines the policies, procedures, and standards to ensure that data is managed effectively throughout the organization.

As shown in Figure 1-13, data governance help organizations make better decisions to meet the needs of various stakeholders. Data governance defines a set of standard processes that reduce operational efforts.

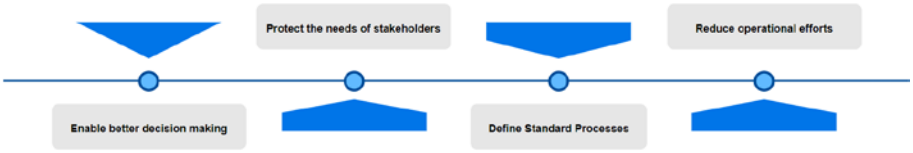

Figure 1-13. *Benefits of data governance*

Data Architecture

The data architecture is used to design and maintain the data infrastructure for data integration, data quality, and data access. Figure 1-14 shows the components of data architecture.

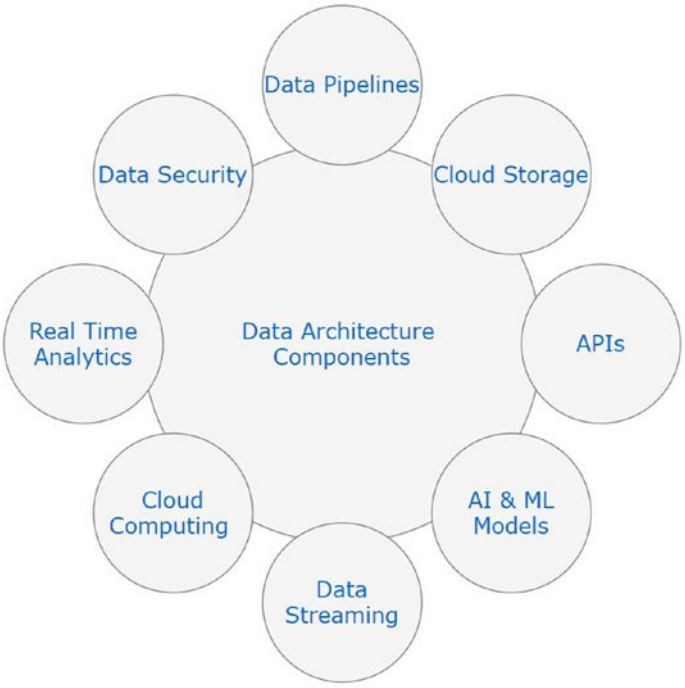

Figure 1-14. *Components of data architecture*

In order to set up a data architecture for any project, data pipelines, data storage setup, and a data ingestion pattern should be designed and implemented. You also need to decide on the cloud computing platform, such as AWS, Azure, or GCP.

Data Modeling and Design

With this data modeling activity, data structures are defined including the relationship needed to enable business process and objectives.

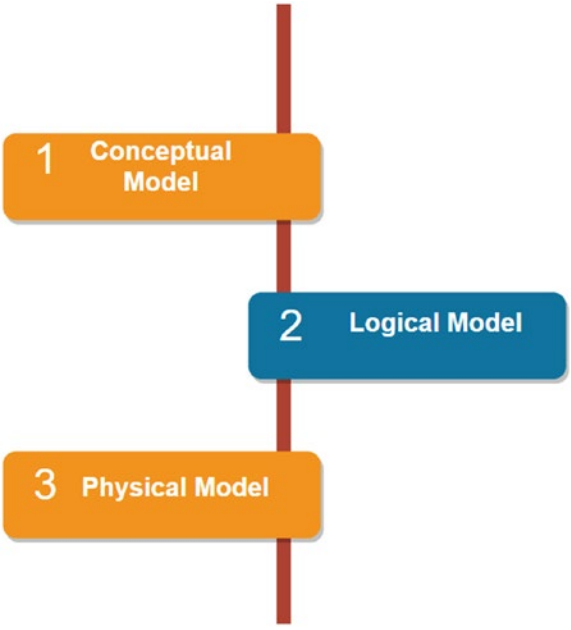

Figure 1-15. *The data modeling stage: conceptual, logical, and physical data model*

As shown in Figure 1-15, in order to start any data modeling activity, you have to start building conceptual data models, followed by logical and physical data models.

Data Storage and Operations

Data storage and operations provide standards to ensure that data is stored efficiently and securely. They also ensure that data is fully available without any performance constraints.

Data Security

Data should be accessible to the end users in a simple and efficient manner. Only authenticated and authorized users should be able to access the data. Data access should also follow any compliance and regulatory requirements.

Figure 1-16. *Data security lifecycle*

As shown in Figure 1-16, data security starts by defining the data lifecycle policies. Based on the data lifecycle, data classification should be done, followed by data minimization. Data access reporting and monitoring should be performed to make sure that access is granted as per the defined standards.

Data Integration and Interoperability

In order to enable data-driven decisions, data should be combined from multiple sources as per the requirements to ensure that data can be exchanged across different systems without any hassle. The following points need to be considered to ensure better data integration:

- Enable and improve data quality
- Refine data integration needs based on requirements/ business scenarios
- Gain confidence from top management to gather correct integration requirements
- Define success metrics
- Perform cost analysis and optimizations
- Enable data governance

Document and Content Management

Companies usually have various forms of data, be it structured, semi-structured, or unstructured data. Unstructured data, such as documents, videos, and so on, should be managed to ensure better accessibility and to meet compliance requirements.

Reference and Master Data

In order to enable informed decisions, all reference and master data should be managed efficiently.

Data Warehousing and Business Intelligence

Data warehouse stores all the incoming data, including historical data, to make sure that the data can be analyzed to capture business insights and make data-driven decisions.

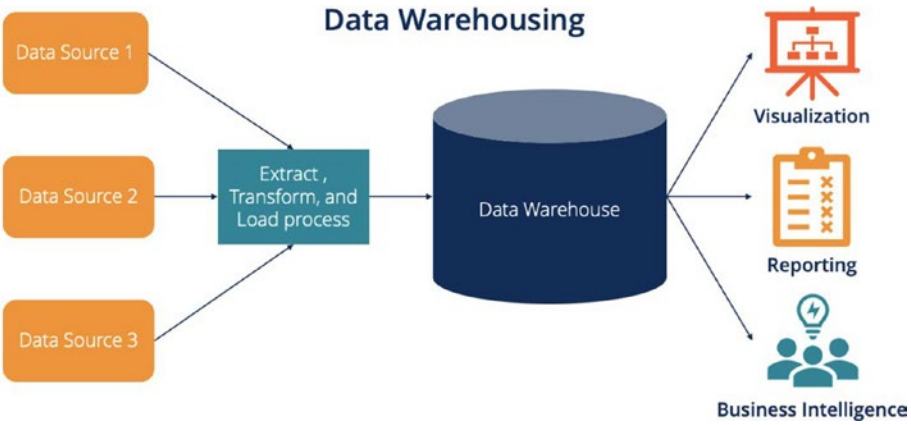

Figure 1-17. *Data warehousing and business intelligence*

As shown in Figure 1-17, data sources from multiple systems/applications will feed the data to the data warehouse after ETL processing. Once the data is available in a data warehouse, reporting and visualizations can be built on the history of data, in order to collect the business insights.

Metadata

Metadata contains data about the data. Metadata about the data should be stored in order to capture data lineage, data classification, and data definitions. Proper storage, management, and capturing of metadata is a must and one of the major pillars of data governance.

Figure 1-18. *Types of metadata*

As shown in Figure 1-18, at a higher level, you can classify metadata into four types: technical metadata, business metadata, operational metadata, and social metadata.

A	B	C
Emp Name	Emp Designation	Emp Salary
Mr. ABC	CA	20000
Mr. XYZ	PMO	50000
Ms. AP	Engineer	70000
Mrs. POP	IT Lead	90000

Metadata

Data

Figure 1-19. *Example of data and metadata*

In the example in Figure 1-19, you see a table with three columns. The header information with a list of columns helps define the data. It is called *metadata*.

Data Quality

The main goal of the data quality process is to make sure that the data is accurate, complete, consistent, and error proof so it can meet business requirements and be used to make good decisions. Figure 1-20 shows the step-by-step process of data quality management.

Figure 1-20. *Step-by-step data quality process*

Understanding the Environmental Factors Hexagon

The Environmental Factors hexagon (see Figure 1-21) shows the relationship between people, process, and technology. Goals and principles are at the center of the diagram, while it also provides recommendations on how people should execute tasks and use tools to set up successful data management projects.

Figure 1-21. *The Environmental Factors hexagon (Reference: www.dama.org)*

Understanding the Knowledge Area Context Diagram

This is based on the concept of the SIPOC diagram used for product management (suppliers, input process, consumers, and so on). Activities in this context diagram can be classified into four phases:

- Plan
- Develop
- Operate
- Control

Each context diagram starts by defining the knowledge areas and goals. On the left side of the contact diagram are the inputs and suppliers. On the right side of the context diagram are the deliverables and consumers of the data.

Conclusion

This chapter explored the fundamentals concepts of data. It explored the data management frameworks process and its relevance to the DAMA-DMBOK Framework. You also took a quick tour of the common data management challenges and principles. At the end of this chapter, you explored the DAMA wheel and its significance to data-driven projects.

In the next chapter, you see how to create relational and nonrelational data solutions using Azure.

CHAPTER 2

Build Relational and Non-Relational Data Solutions on Azure

Many businesses still use outdated or inadequately developed data platform strategies. Moving existing systems to the cloud, creating new apps quickly by utilizing the cloud, and reducing on-premises costs have all become significant trends. The goal of many organizations is to create a strategy for moving their data workloads to the cloud. Administrators are interested in learning how to make their organizations successful.

This chapter explores in depth the possible solutions and best practices of building relational and non-relational data solutions on Azure.

This chapter covers the following topics:

- Data integration using ETL
- Online analytical processing for complex analysis
- Managing transactional data using OLTP
- Managing non-relational data
- Handling time-series and free-form search data
- Working with CSV and JSON files for data solutions

© Sagar Lad 2023
S. Lad, *Modern Data Architecture on Azure*, https://doi.org/10.1007/978-1-4842-9760-5_2

Data Integration Using ETL

Collecting data from various sources and in various formats is a difficulty that businesses frequently encounter. That data often needs to be transferred to one or more types of data storage. The destination is often a different data store than the source. If the formats are different, the data may need to be shaped and cleaned.

Over the years, a number of tools, services, and procedures have been created to assist in addressing these difficulties. Regardless of the method employed, it is always necessary to coordinate the activity and implement some sort of data transformation in the data pipeline. Azure Data Factory and Azure Synapse Analytics can be used to implement ETL pipelines, as shown in Figure 2-1.

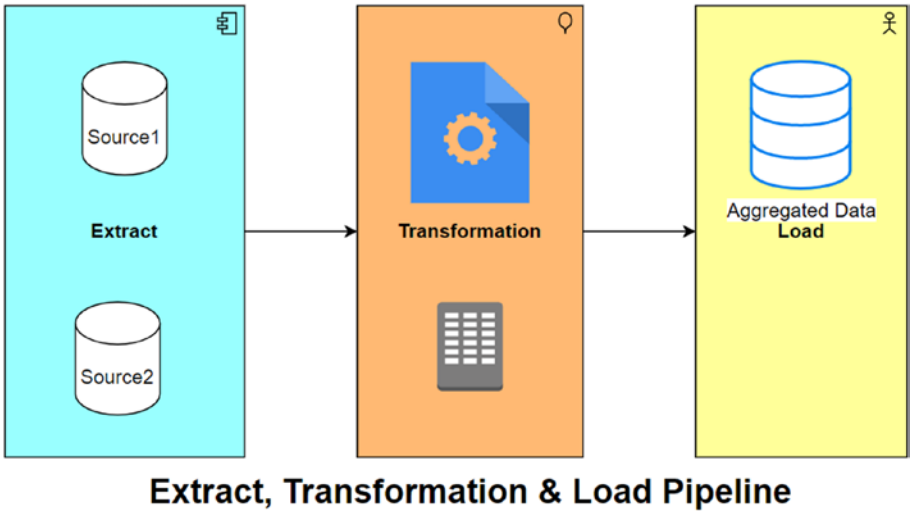

Figure 2-1. *The ETL pipeline*

Data is widely used for data analytics, machine learning, and advanced analytics activities to identify business insights from the data. In this digital era, the volume, velocity, and variety of the data is growing very fast. The raw data needs to be collected, transformed, and processed

using the ETL pipelines before it can be used by business users to create value. Sometimes, the data engineers also use ELT (Extract, Load, and Transform) pipelines instead of ETL pipelines. This is based on the requirements and use case.

Data Extraction

Data extraction—from target sources that are typically heterogeneous and include business systems, tools, transaction databases, and others—is the initial step of this process. While some of these data types are likely to be semi-structured, are likely structured outputs of commonly used systems. Extraction can be done in a variety of ways:

- *Full load*: In the full load of the data, the source system always extracts the complete view of the data from the database or data storage. In the full load, when the data is sent, the existing data will be overridden and new data will be loaded to the database.
- *Incremental extraction*: Some systems cannot determine which data has been altered. In this situation, the only way to obtain the data from the system is through a full extract. For this technique to work, you must have a duplicate of the previous extract in the same format so you can track down the modifications that were performed. Azure services—such as SQL dedicated pools on Azure Synapse Analytics, SQL Serverless pools on Azure Synapse Analytics, HDInsight with Hive, Azure Data Factory, and SQL Server Integration Services (SSIS)—can be used to implement the data extraction process as a part of the ETL pipeline.

Data Transformation

Raw data that's collected during the extraction process should be processed and cleansed to create good quality data for the end consumer. The second stage consists of converting the unformatted, raw data that has been cleansed from the sources into a form that can be accessed by various applications. In order to meet operational requirements, data is cleaned, mapped, and converted during this stage, frequently to a particular schema. This procedure involves many sorts of transformations to guarantee the accuracy and reliability of the data. Data is frequently put into a staging database rather than being loaded directly into the destination data source. This procedure guarantees a speedy rollback in the event that things do not proceed as expected. You can create audit reports for legal compliance at this point, as well as identify and fix any data problems.

Some examples of transformation activities in the ETL pipeline are as follows:

- Converting date columns into uniform formats
- Removing unused columns
- Aggregating columns to calculate sum or average values

Data Loading

The process of writing converted data from a staging area to a target database is known as the load process. This procedure can be relatively straightforward or extremely complicated, depending on the requirements of the application. ETL tools and custom code can be used to complete each of these processes.

Designing ELT Pipelines Using the Azure Synapse Server

This section explores in detail on how to design ELT pipelines using the Azure Synapse Server.

Once the Azure Synapse instance has been created, go to the Azure Synapse Studio and click the Develop section, as shown in Figure 2-2.

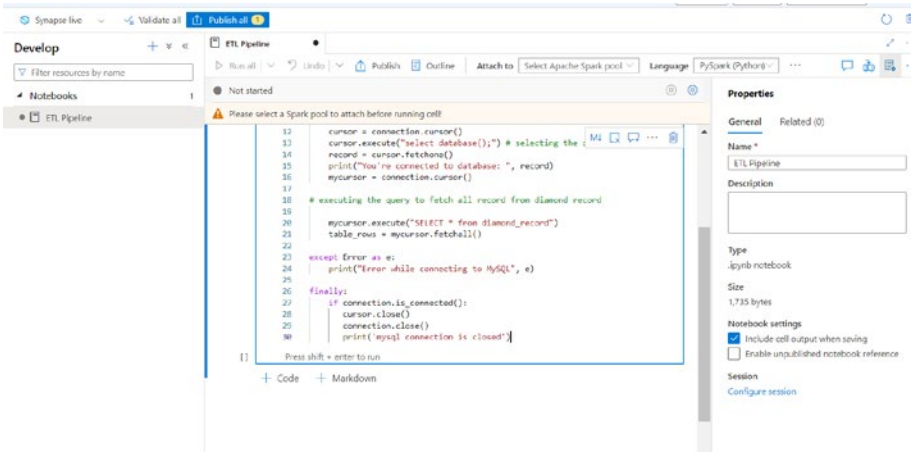

Figure 2-2. *Synapse Server ETL pipeline*

Now click the + button near the Develop section and write Python code for the ETL pipeline in the notebook. Then test and publish the changes. From the Attach To section, select Manage Pools to select existing pools or select Create a New Pool, as shown in Figure 2-3.

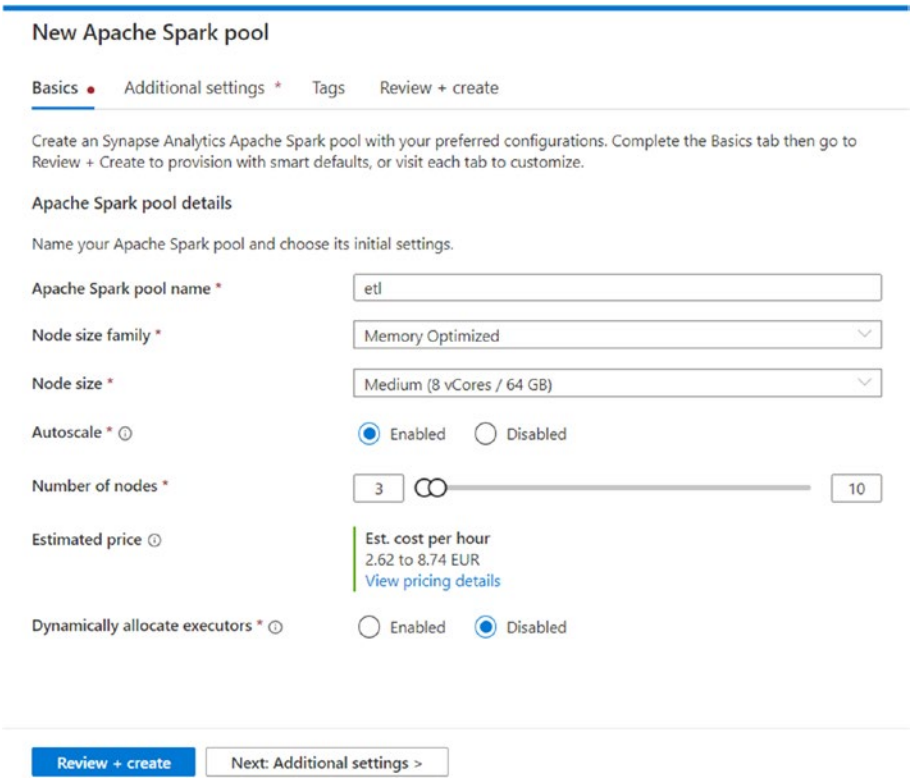

Figure 2-3. *Synapse Server pool creation*

Once it is ready, click the Run All button to run the pipeline in Azure Synapse. You can integrate this pipeline with other pipelines and execute them. You can also track the pipeline execution from the Monitor section of the Azure Synapse Server, as shown in Figure 2-4.

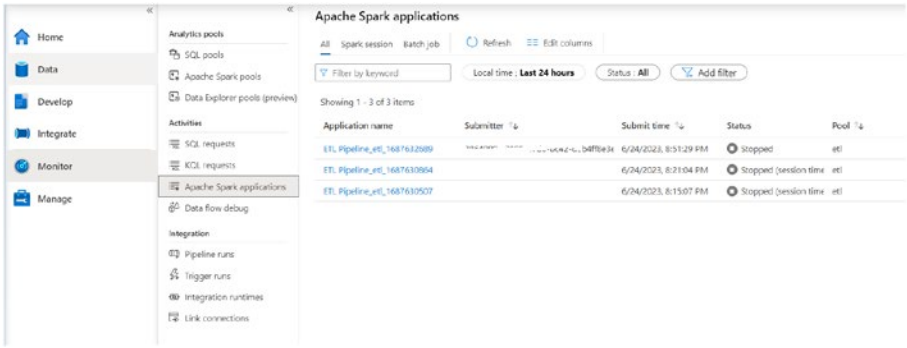

Figure 2-4. *Synapse Server pipeline monitoring*

Online Analytical Processing for Complex Analyses

OLAP, also known as online analytical processing, enables users to quickly and selectively extract and query data in order to examine it from many angles. OLAP business intelligence queries are useful for a variety of planning tasks, including trend analysis, regulatory reporting, budgeting, and so on. For instance, a user can ask for data analysis to display a spreadsheet of all cricket-related products sold by a company during the month of November in India, compare the sales numbers with those for the same products during the following month of December, and then view a comparison of other product sales in another country during the same time frame.

Data is gathered from many sources, stored in data warehouses, cleaned up, and packaged into *data cubes* to enable this type of analysis. Every OLAP cube contains data that has been divided into categories based on dimensions, such as type of clients, geographical regions, and time period. These dimensions are obtained from dimensional tables in

the data warehouses. Then, members that are arranged hierarchically are used to populate the dimensions. Pre-summarizing data across dimensions in OLAP cubes significantly reduces query time compared to relational databases, as shown in Figure 2-5.

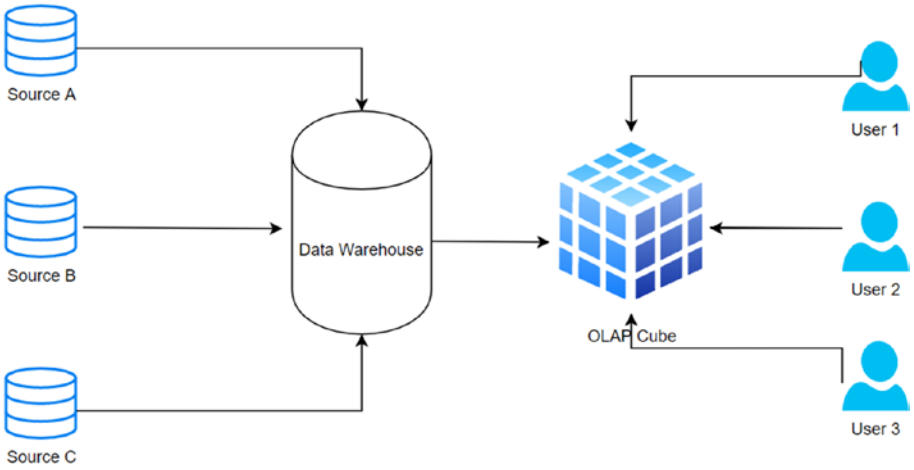

Figure 2-5. *The OLAP process*

Then, analysts can use these multidimensional databases to perform five different OLAP analytical operations:

- *Roll-up*: This procedure, sometimes referred to as drill-up, summarizes the data along the dimension.
- *Drill-down*: In order to measure the product's sales increase, analysts can drill down from "time period" to "years" and "months" using this method.
- *Slice*: This is used by an analyst to exhibit just one level of data, such as "new customers in 2023."

- *Dice*: An analyst can then choose data from various characteristics to evaluate, for example, "sales of cricket balls in Delhi in 2023."
- *Pivot*: Rotating the data axes of the cube gives analysts a fresh perspective of the data.

OLAP databases are different from Online Transaction Processing (OLTP) databases. OLTP databases are used by businesses to hold all of their transactions and records. Records are typically entered one at a time into these databases. However, analysis is not a goal of the databases utilized for OLTP. As a result, it takes time and effort to get responses from these databases. OLAP systems were created to aid in the efficient extraction of business intelligence data.

You can use the Azure Analysis Service to implement OLA cubes for analytics purposes. Once the Azure Analysis Service is created, you have to create a data source based on an existing or a new connection. Then you create a data source view. Based on the data source and the relationship in the underlying data and foreign key match scheme, a model will be created in Azure Analysis Service, as shown in Figure 2-6.

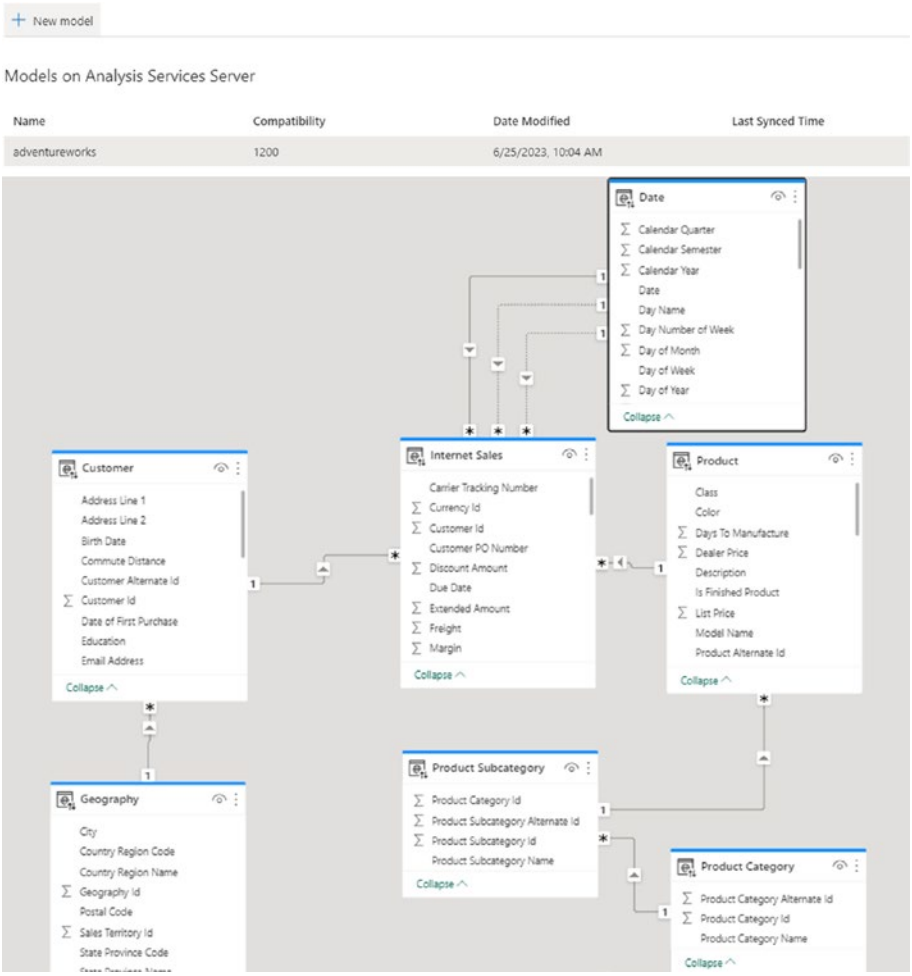

Figure 2-6. *Azure Analysis Service model*

Once the model is created in Azure Analysis Service, you can visualize it in Excel, Power BI Desktop, or Visual Studio. Let's look at semantic modeling in more detail, as it is the basis for designing the OLAP system.

Semantic Data Modeling

A high-level semantics-based database description and organizing database model for databases is known as a *semantic data model*. In comparison to current database models, this database model is intended to capture a wider range of the meaning of an application environment. Its specification explains a database in terms of the categories and groups of the entities present in the application environment, as well as the structural links that connect these items. Semantic data modeling allows the same information to be seen in different ways, by allowing derived information in a database structure definition. This makes it possible to directly satisfy the variety of needs and processing requirements.

The purpose of semantic data modeling is to improve the efficiency and usability of database systems. An semantic data modeling database description can be used as a formal specification and documentation tool for a database as a conceptual database model during the database design process, and as the database model for a database management system. It's common for organizations to have their own terminology, often with synonyms or even with multiple interpretations of the same term. For instance, a customer database might refer to the customer number as the customer ID, whereas an inventory database might monitor a piece of the customer using the customer ID. As shown in Figure 2-7, data warehousing and semantic layer can be designed.

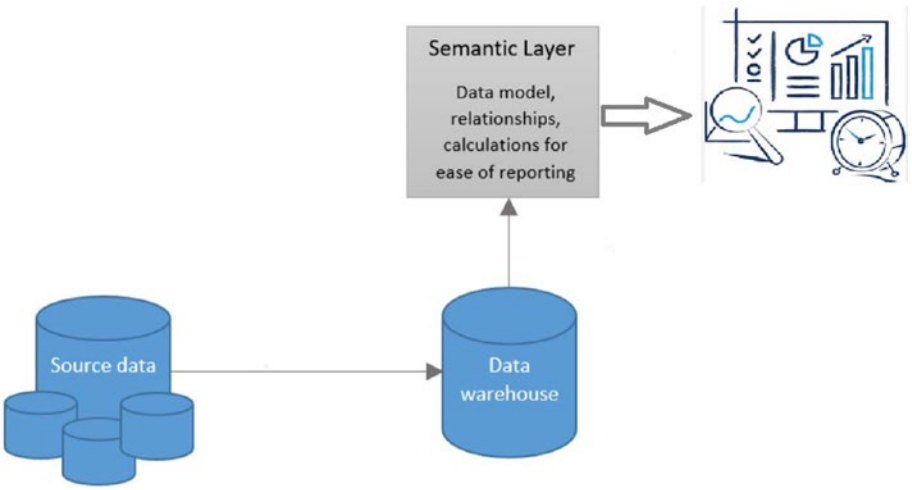

Figure 2-7. *Data warehouse and semantic modeling layer*

The benefits of semantic data modeling include these:

- Integrated data for end users, so no need to worry about joins or connections
- All columns have names that are simple to remember for business purposes
- Calculations and business logic available centrally in the data model
- Consists of powerful time-oriented calculations
- Aggregation behavior has been configured for proper response from reporting tools
- Reporting tools can handle formatting of the data to develop the reports

These are the two main categories of semantic models:

- *Tabular*: Uses the elements of relational modeling, which has models, tables, and columns. Internally, metadata is descended from the cubes, dimensions, and measurements of OLAP modeling. OLAP metadata is used in code and script.
- *Multidimensional*: employs the standard OLAP modeling building blocks, which has cubes, dimensions, and measures.

Let's now consider the requirements in which semantic modeling is beneficial for OLAP systems.

A huge database that belongs to an organization contains a large volume of data. It aims to make this information available to business users so they can build their own reports and conduct their own analyses. Giving such users direct access to the database is one option. However, doing so has a number of disadvantages, including managing security and restricting access. Additionally, a user may find it challenging to comprehend the database's design, particularly the titles of its tables and columns. To acquire the right answers, users need to understand which tables to query, how to join those tables, and other business logic that has to be used.

Challenges of Using OLAP Solutions

The challenges of using OLAP solutions include these:

- OLAP data warehouses are often updated at a slower interval depending on business needs. OLAP systems are better suited for strategic business choices. To keep the OLAP data repositories current, some amount of data cleansing and orchestration must also be planned.

- OLAP data models are typically multidimensional. Direct mapping to entity-relationship or object-oriented models, where each attribute is mapped to one column, becomes challenging as a result. Instead of using the conventional normalization, OLAP systems often employ a star or snowflake schema.

In Azure, there are various services available other than Azure Analysis Services to implement OLAP systems. Key design considerations on when to design OLAP applications can be defined as follows:

- Managed service to run your servers
- Requirements for Azure Active Directory (Azure AD)-based secure authentication
- Need for real time analytics
- Provide semantic models to make analytics more user-friendly and require the use of pre-aggregated data. If so, pick a solution that works with multidimensional cubes
- Choice has to be made where data integration from sources are needed over the OLTP data store

Managing Transaction Data Using OLTP

The quick, reliable data processing that underpins card transactions, online banking, e-commerce, and a plethora of other services we all use on a daily basis is made possible by OLTP (online transactional processing). Large numbers of database transactions can be executed in real time by a large number of users, generally over the Internet. A database transaction

is an operation that modifies, adds, removes, or queries data stored in a database. Many of the financial operations people conduct on a daily basis are powered by OLTP systems, including online banking and ATM transactions, store purchases, flight reservations, and so on. Each of these situations leaves the database transaction as a record of the associated financial transaction.

OLTP systems are mainly useful in the following scenarios:

- You need to process large volume of transactions, such as simple data queries and insertions, updates, and deletions.
- You need to enable multiple users to access the same data while maintaining data integrity.
- You need concurrency algorithms to make sure that no two users can modify the same data at the same time and that all transactions are completed in the correct sequence.
- You need to enable extremely quick processing with millisecond-level response times.
- You need 24/7 accessibility.

OLTP systems handle so many concurrent transactions, that any data loss or outage can have a big impact and be very expensive. Every instant must have access to a full data backup. OLTP systems need ongoing incremental backups as well as frequent regular backups, as shown in Figure 2-8.

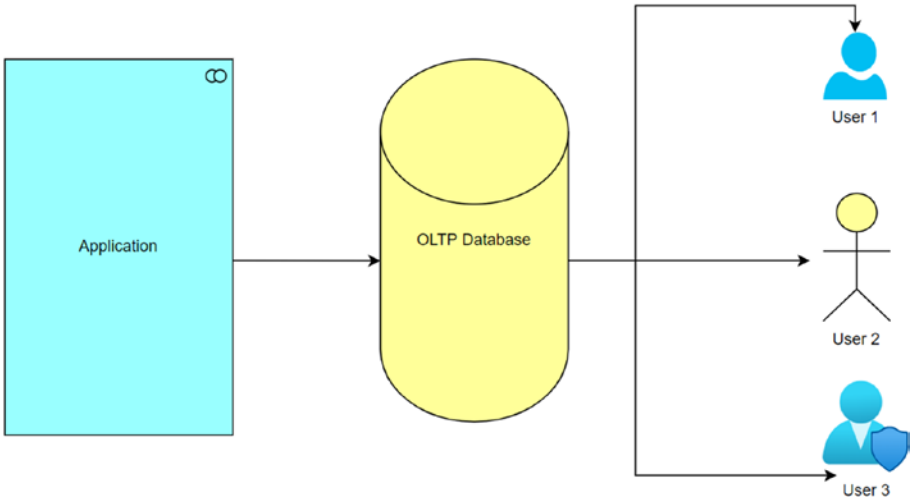

Figure 2-8. *Online transaction processing system (OLTP)*

Transactions must normally be atomic and consistent. Atomicity refers to the fact that a transaction is never left unfinished and always succeeds or fails as a single unit of work. The database system must reverse any actions already taken as part of a transaction if it cannot be completed. If a transaction cannot be completed, this rollback occurs automatically in a conventional RDBMS. Consistency entails that data is consistently left in a usable state after transactions. These qualities have more technical definitions, such as ACID, as shown in Figure 2-9.

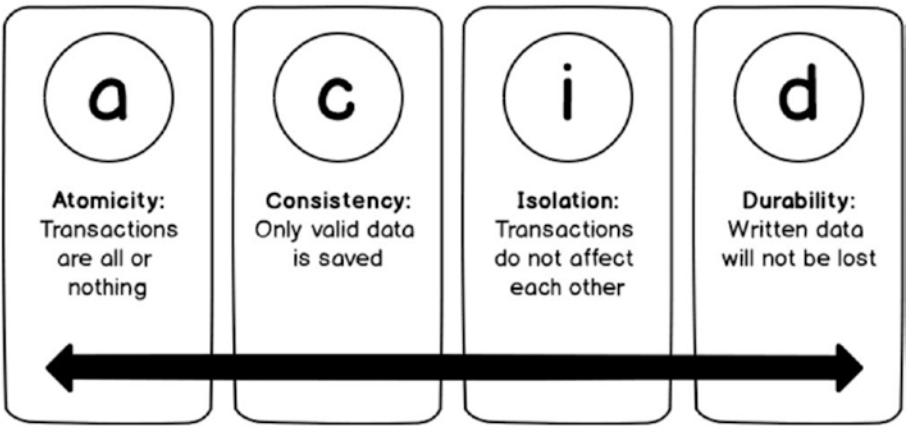

Figure 2-9. *ACID transaction: Online transaction processing system*

- *Atomicity*: When a transaction operates, it behaves as though it were performing just one operation. Either every modification is made or none at all. A transaction fails if even one activity within it fails, and any completed operations are rolled back.
- *Consistency*: The data has a consistent initial state and a consistent final state. Data may be inconsistent during the transaction, but it is left in a consistent condition at the end.
- *Isolation*: A transaction's intermediate state is hidden from other active transactions. Effectively, concurrent transactions are serialized.
- *Durability*: After a transaction is over, modifications are persistently stored. The impact of the completed transaction endures even in the event of a power outage or other system problems.

Typical transactions have the following design considerations.

- Highly normalized database
- Schema on write
- Strong consistency
- Update/append possible
- Heavy writes and light reads
- Relational database
- Highly flexible query execution

In Azure, data warehousing architecture can be implemented in multiple ways. Figure 2-10 shows an example of one of the recommended reference architectures that can be used to design data warehouse applications on Azure.

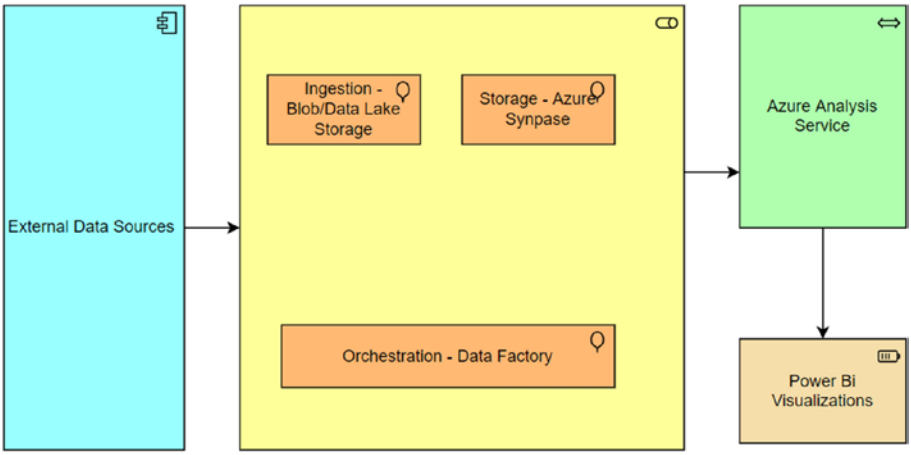

Figure 2-10. *Data warehouse implementation using Azure Services*

This reference architecture in Figure 2-10 demonstrates incremental loading in an ETL pipeline (extract, transform, and load). The ETL pipeline is automatically automated using Azure Data Factory. The pipeline

incrementally uploads the most recent OLTP data to Azure Synapse from an on-premises SQL Server database. A tabular model is created using transactional data for analysis.

This reference architecture workflow has the following components:

- *External data sources*: Data sources can be an on-prem data sources or data stored in the cloud. Integrating different data sources is a frequent use case for data warehouses. This reference design connects the data from the OLTP database to an external dataset that provides city populations by year after loading it.
- Data ingestion and storage:

  Blob storage/ADLS Gen2: Source data is staged in blob storage before being loaded into Azure Synapse. It can be stored into multiple layers as per the layered design approach of blob or ADLS Gen2.

  Azure Synapse: Used to analyze massive amounts of data for massive parallel processing (MPP) for data analysis.

  Data Factory: Used as an orchestrator to automate the whole ETL pipeline starting from getting data to moving the data to Azure Synapse Server for end users.
- *Analysis and reporting*: Azure Analysis service and Power BI services are used to access the data from the Azure Synapse Server and prepare the Power BI reports for analytics purposes.

Managing Non-Relational Data

A non-relational database is one that does not use the typical standard database systems' tabular schema of rows and columns. Non-relational databases, on the other hand, employ a storage model that is tailored to the particular needs of the type of data being stored. Data may be stored, for instance, as straightforward key-value pairs, JSON (JavaScript Object Notation) documents, or a network with edges and vertices.

Data stores that do not employ SQL queries are referred to as NoSQL. Instead, the data stores query the data using various computer languages and techniques. Despite the fact that many of these databases do enable SQL-compatible queries, NoSQL is typically used to refer to "non-relational databases." The fundamental query execution approach is frequently significantly different from how the identical SQL query would be handled by a conventional RDBMS.

There are four types of NoSQL databases, as shown in Figure 2-11:

- Key-value pair
- Document database
- Graph database
- Column family database

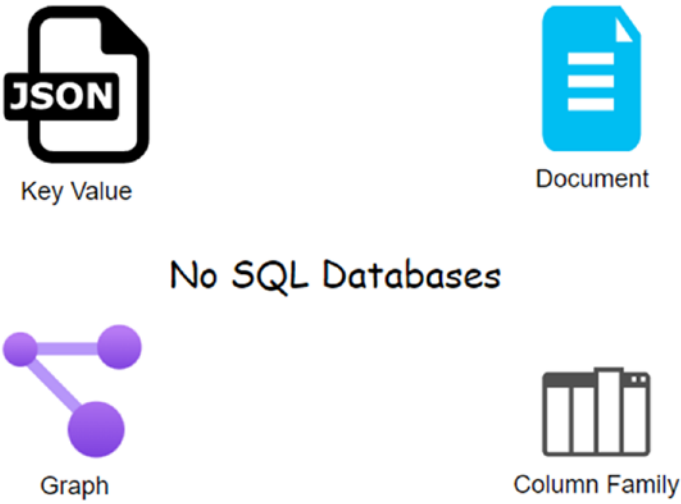

Figure 2-11. *Types of NoSQL databases*

Key-Value Pair Databases

Key-value pairs are used to organize data. The design makes it capable of handling large amounts of data and demanding loads. Each key in a hash table used by key-value pair storage databases must be distinct and the value may be any type of data, including JSON, BLOBs (Binary Large Objects), strings, and so on. Azure Cosmos DB for Table, Azure Cache for Redis, and Azure Table Storage are examples of key-value pair databases. Key-value pairs may contain employee information like this.

```
{
"name" : "Sagar",
"occupation" : "software engineer",
"location" : "seattle"
}
```

Column Family Databases

Based on Google's BigTable document, column-oriented databases operate on columns. Each column is handled individually, as shown in Figure 2-12. Single column databases store their values consecutively. Since the data is easily accessible in a column, these databases provide great performance for aggregation queries like `SUM`, `COUNT`, `AVG`, `MIN`, and so on. Column-based databases with NoSQL queries include Cosmos DB for Cassandra, HBase in HDInsight, Cassandra, Hypertable, and so on.

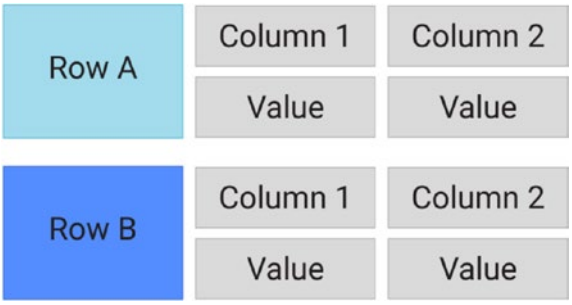

Figure 2-12. *Column family database*

Document Databases

Document-oriented NoSQL databases use a key-value pair for data storage and retrieval, but the value portion is kept as a document, as shown in Figure 2-13. JSON or XML formats are used to store the document. The document format is utilized by content management systems, real-time analytics, and e-commerce applications. Using it for complicated transactions requiring numerous operations or queries against various aggregate structures is not advised. Popular document-originating DBMS systems include Azure Cosmos DB, Amazon SimpleDB, CouchDB, MongoDB, and Amazon SimpleDB.

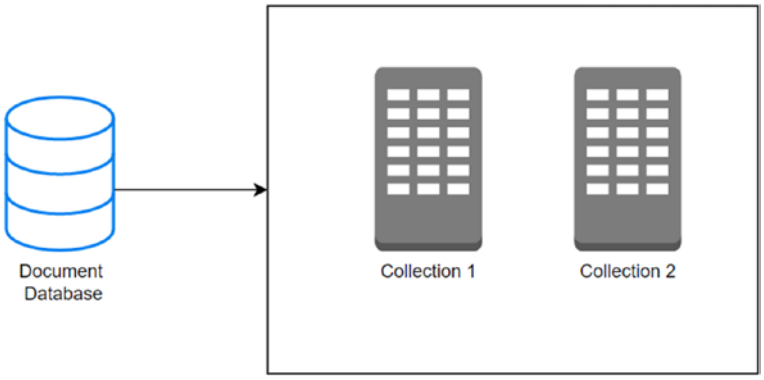

Figure 2-13. *Document database*

Graph Databases

A graph database is multi-relational in nature, in contrast to relational databases, which have loosely coupled tables. Entities and the relationships between them are stored in a graph type database. The relationship between the entity and its edges is represented as a node. An edge reveals the connection between two nodes. There is a distinct identifier for each edge and node, as shown in Figure 2-14. There is no need to calculate them when traversing relationships because they have already been recorded in the database.

Graph-based databases are utilized for spatial data, logistics, and social networks. Popular graph-based databases include Azure Cosmos DB with Graph API, OrientDB, Neo4J, and FlockDB.

Figure 2-14. *Graph database*

Design considerations as to which NoSQL database to use are listed in Figure 2-15.

Requirement	Document data	Column-family data	Key/value data	Graph data
Normalization	Denormalized	Denormalized	Denormalized	Normalized
Schema	Schema on read	Column families defined on write, column schema on read	Schema on read	Schema on read
Consistency (across concurrent transactions)	Tunable consistency, document-level guarantees	Column-family–level guarantees	Key-level guarantees	Graph-level guarantees
Atomicity (transaction scope)	Collection	Table	Table	Graph
Locking Strategy	Optimistic (lock free)	Pessimistic (row locks)	Optimistic (ETag)	
Access pattern	Random access	Aggregates on tall/wide data	Random access	Random access

Figure 2-15. *NoSQL database selection strategy*

Handling Time-Series and Free-Form Search Data

You can enable free-form text processing on documents that have more text in order to assist efficient searching. When performing a text search, a specialized index is built and precomputed against a set of documents.

A client application sends a search query. The result is a list of documents arranged according to how closely they fit the search criteria. This result is returned by the query. The context in which the document satisfies the requirements can also be included in the result set, allowing the application to highlight the phrase that corresponds in the document.

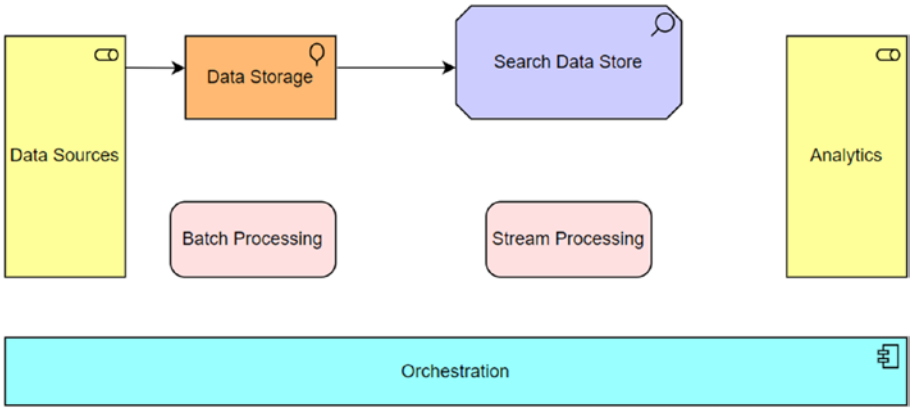

Figure 2-16. *Reference architecture for a search data store*

As you can see in Figure 2-16, most of the documents are stored in Azure Blob Storage or Azure Data Lake Storage Gen2. This data can be stored using the batch pattern or the near real-time ingestion pattern. Elasticsearch, Azure HDInsight with Apache Solr, and Azure Cognitive Search are some of the options for building an external search index.

It takes a lot of time and computation to process a collection of free-form text documents. The search index should allow searching for terms with a similar construction in order to efficiently search free-form text. In order to provide a rich search experience over private, heterogeneous material in online, mobile, and corporate apps, Azure Cognitive Search provides developers with infrastructure, APIs, and tools. Any software that surfaces text to users relies on search and common use cases, including document search and data exploration over proprietary material. Azure cognitive services has the following capabilities:

- Rich indexing with lexical analysis and AI enrichment in a search engine
- Search syntax for text, fuzzy, autocomplete, geo-search, and other searches
- Integration of Azure at the data layer, machine learning layer, and AI (cognitive services)

Additionally, users and apps can execute full-text queries against character-based data stored in SQL Server tables using full-text search in SQL Server and Azure SQL Database. Consider an example of a car company that produces different types of cars for which data is stored in Azure SQL Database. You could search for specific car using the following query:

```
SELECT car_id
FROM cars
WHERE CONTAINS(car_description, '"Audi 100EZ"' OR car_engine
='automatic')   AND car_cost < 200000 ;
```

The time-series data is a set of values arranged chronologically. An important feature of time-series data is temporal ordering, which arranges events in which they occur and are received for processing. For data whose values revolve around changes in an asset or process over time, select a time-series solution. Time-series data can be used to quantify change in the past or to forecast change in the future. Time-series data typically enters data in the repository. Standard online transaction processing data pipelines, in contrast, accept data in any sequence and allow for constant updating. Time is a useful axis for examining or evaluating time-series data since it contains timestamps. A scatter or line chart is the most effective way to display time-series data, as you can see in Figure 2-17.

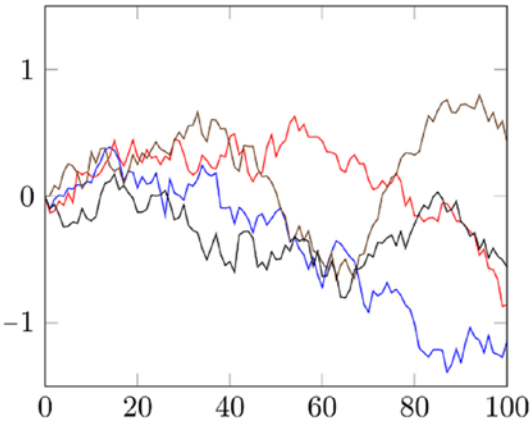

Figure 2-17. *Sample time-series scatter plot*

In order to implement time-series data using Azure Services, various services are available. Data is ingested into the stream processing layer from one or more data sources using Azure IoT Hub, Azure Event Hubs, or Kafka on HDInsight. The data is handled via the stream processing layer, and it can then be sent to a machine learning service for predictive analytics. The processed data is kept in an analytical data store like Azure Data Explorer, HBase, Azure Cosmos DB, or Azure Data Lake. The time-series data can be seen for examination in a reporting and analytics tool or service like Power BI or OpenTSDB for HBase.

To create a full time-series service, you can also use Azure Data Explorer. Multiple time-series creation, manipulation, and analysis with close to real-time monitoring are all natively supported by Azure Data Explorer. Data from numerous platforms and services in a variety of formats can be ingested by Azure Data Explorer. There is no restriction on ingestion, and it is scalable.

You may execute searches and create dashboards for data visualization using the Azure Data Explorer Web UI. Additionally, Azure Data Explorer supports ODBC and JDBC connector-based dashboard applications like Power BI, Grafana, and so on. You can utilize the BI solution that best suits the situation and budget because Azure Data Explorer offers a wide range of BI tools.

Multiple time-series creation, manipulation, and analysis are also supported in the Kusto Query Language (KQL). Let's see how to use KQL to quickly build and analyze thousands of time-series data to capture near real-time monitoring solutions. The first step is to open Azure Data Explorer and select the Query section from the left pane. There is a list of various time-series data, from which you can run the KQL query for the `OccupancyDetection` table, as shown in Figure 2-18.

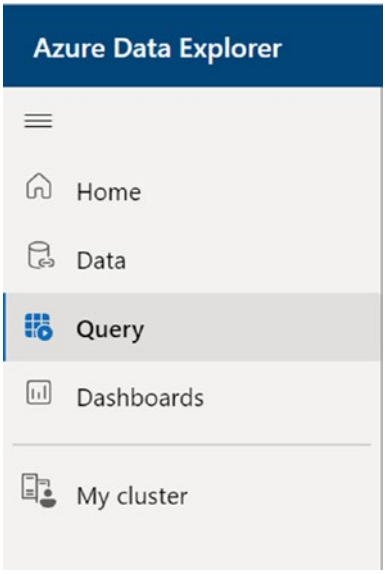

Figure 2-18. *Azure Data Explorer query*

In order to find the top 20 records in the table, you could execute the query, as shown in Figure 2-19. The resulting table contains a timestamp column, temperature, humidity, light, and so on.

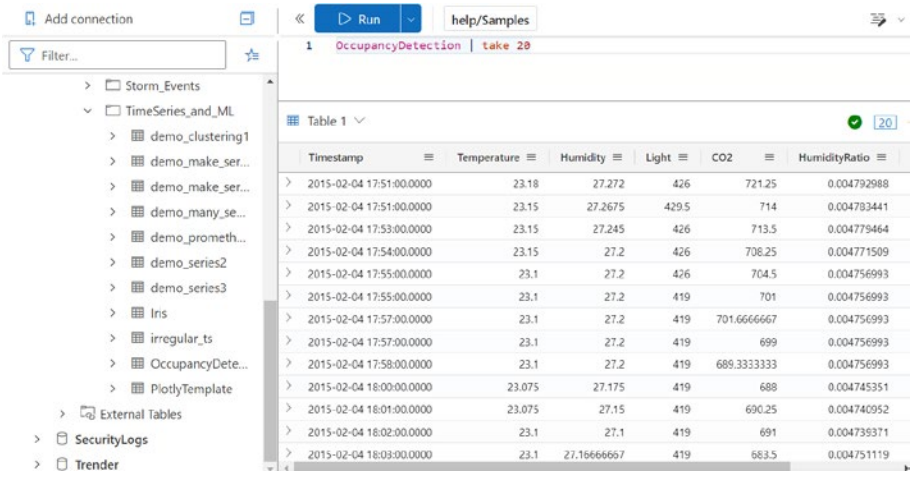

Figure 2-19. *Azure Data Explorer KQL to explore time-series data*

Let's build a new metric with a set of time-series representing the occupancy count itself, partitioned by light and using the query shown in Figure 2-20.

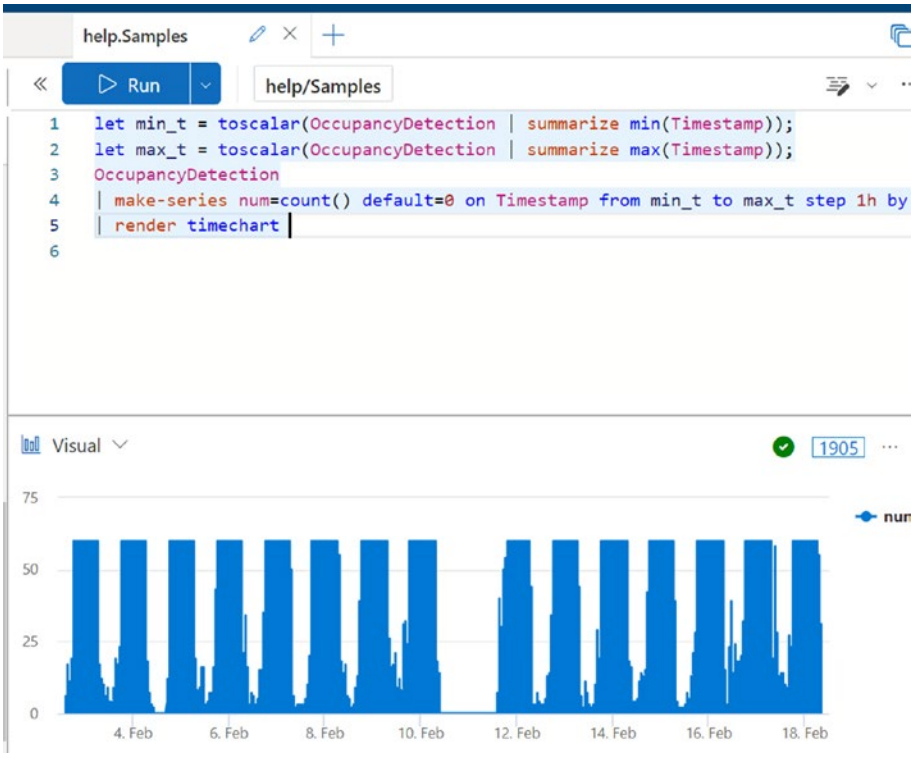

Figure 2-20. *Time-series analysis: KQL with occupancy count*

In the KQL, the `make-series` operator creates a set of three time-series, where:

- `num= count()` is the time-series of traffic
- `from min_t to max_t step 1h` are the oldest and newest timestamps of table records
- `by` is the partition by light

Based on this KQL query, the render bar chart is used for visualization, as shown in Figure 2-20.

Working with CSV and JSON Files for Data Solutions

The most popular formats for absorbing, transferring, and storing unstructured or semi-structured data are probably CSV and JSON. Multiple services can be used to store these CSV or JSON files. You can use Azure Blob Storage or Azure Data Lake Storage Gen2 to store such data, as shown in Figure 2-21.

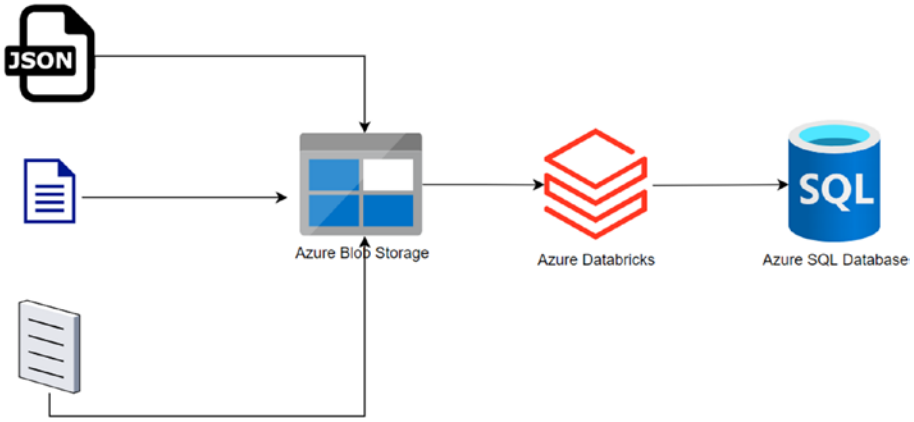

Figure 2-21. *Processing CSV and JSON files using Azure Services*

The following Azure services can be used in Azure to process CSV and JSON files:

- Azure Data Factory
- Azure Logic Apps
- Azure Functions
- App Service
- Azure Data Lake Analytics
- Azure HDInsight

- Azure Synapse
- SQL Data Warehouse
- Azure Machine Learning Workbench
- SQL SSIS

It is possible that the data in CSV files that's read with a certain schema will not match the schema. A field including the name of the city, for instance, will not parse as an integer.

The following example processes CSV files in Azure Synapse. Using the serverless SQL pool, there can be multiple scenarios to read and process CSV files. The simplest way to view the content of your CSV file is to pass the URL to the `OPENROWSET` function together with the 2.0 `PARSER_VERSION` and CSV `FORMAT` options.

SQL Query

```
select top 10 * from openrowset(
bulk 'https://opsdatalake.blob.core.windows.net/landingzone/
processed/logging/logs_usage.csv',
    format = 'csv',
    parser_version = '2.0',
    firstrow = 2 ) as rows
```

Here, you are providing the input path of the CSV file that is stored in the Azure Blob storage. The `firstrow` option is used to skip the first row of the CSV file, which contains header information.

Another way to access a CSV file is to create an external data source with a blob storage location that points to the CSV file.

External Table

```
create external data source ops
with ( location = 'https://opsdatalake.blob.core.windows.net/
public/processed/logginh/logs_ussage' );
```

Once you create a data source, you can use that data source and the relative path of the file in the `OPENROWSET` function:

```
select top 10 *  from openrowset(
        bulk 'latest/logs_usage.csv',
        data_source = 'ops',
        format = 'csv',
        parser_version ='2.0',
        firstrow = 2
    ) as rows
```

Data in JSON is present in the form of semi-structured key-value pairs. JSON and XML are frequently contrasted, since both allow for the storage of data in a hierarchical manner, with child data being displayed alongside its parent. Both are self-descriptive and human-readable, but JSON documents are frequently smaller, which makes them a popular choice for online data exchange, especially with the rise of REST-based web services.

The JSON format has various benefits over CSV:

- Since JSON has hierarchical structures, it is simpler to encode complicated relationships
- Most programming languages has native support for deserializing JSON into objects
- Lists of objects are supported by JSON to help prevent clumsy list translations

If you want to read JSON files stored in Azure Blob Storage, for example, you can use the following query.

SQL Query

```
select top 10 *
from openrowset(
bulk 'https://opsdatalake.blob.core.windows.net/public/
processed/ios_usage.json',
        format = json,
        fieldterminator ='0x0b',
        fieldquote = '0x0b'
    ) with (doc nvarchar(max)) as rows
```

This query reads the JSON files that contain an array of items in their JSON document. The output of the query consists of each object as a separate row in the result set.

Conclusion

This chapter explored in detail how to design and implement relational and non-relational databases with various Azure services. It also did a deep dive into designing the ETL pipeline and you saw how to manage OLTP and OLAP applications. In the end, you took a quick tour of processing the most commonly used file types—CSV and JSON.

In the next chapter, you learn about different types of big data architectures and its use cases.

CHAPTER 3

Building a Big Data Architecture

Big data tools and techniques must be employed to handle large volumes of data and to carry out operations on that data. When you use the phrase "using big data tools and techniques," you are referring to the big data ecosystem. Every use case requires a different solution, which must be developed and produced in an efficient manner in accordance with the business requirements. To satisfy the needs of the business, a big data solution must be created and managed in accordance with corporate requirements.

This chapter covers the following topics:

- Core components of a big data architecture
- Batch processing
- Real-time processing
- Lambda architecture
- Kappa architecture
- Internet of things
- Data mesh principles and the logical architecture

© Sagar Lad 2023
S. Lad, *Modern Data Architecture on Azure*, https://doi.org/10.1007/978-1-4842-9760-5_3

Core Components of a Big Data Architecture

A big data architecture consists of the components, processes, and technology needed to extract, store, and process large volumes of data. As shown in Figure 3-1, the four core components of big data architecture are data ingestion and processing, data analysis, data visualization, and data governance.

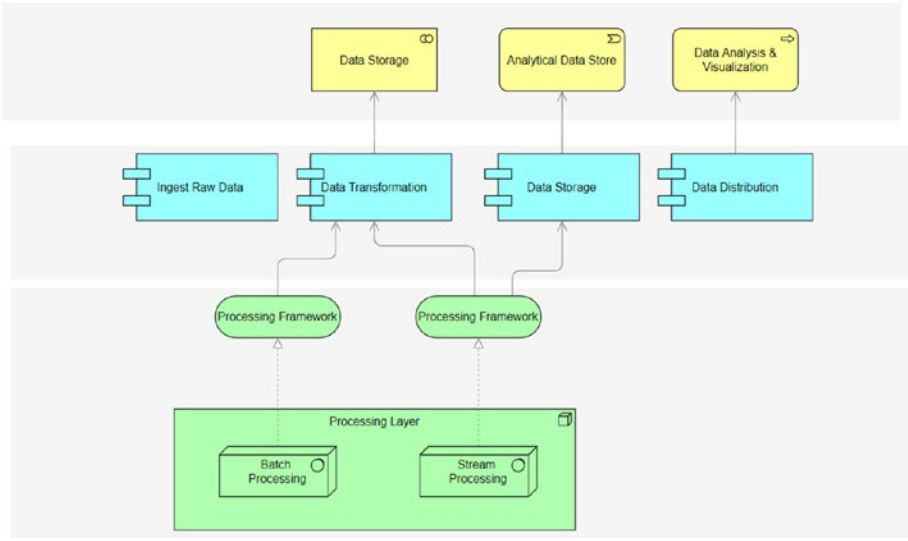

Figure 3-1. *Generic big data processing architecture and components*

A big data architecture must be able to manage the overall lifecycle of a very large volume of data. Data in the big data architecture can be handled via these patterns:

- Batch processing
- Near real-time data processing
- Direct query layers
- Advanced analytics

Let's look at the components of big data architecture in detail.

Data Ingestion and Processing

The most complex activity in the big data application is data ingestion, which is the first step of creating a data pipeline. In this step, data is being collected from thousands of sources into the data lake. In the current digital world, data is being generated in a variety of formats and at different paces. Because of this, it's important to properly process the data in order to make wise business decisions. It is true that "If the start goes well, then half of the work is done."

Connecting to numerous data sources, extracting data, and detecting changes in the data are all part of big data ingestion. This involves transferring data, particularly unstructured data, from its original location to a system that can store and analyze it. Data ingestion can also be defined as the act of collecting data from many sources and storing it in a convenient location. The data pipeline starts here when data is obtained or imported for immediate use.

Data lakes are designed to store large volumes of data in its raw and unprocessed form, allowing organizations to store diverse data types without prior structuring. This raw data can later be transformed and processed as needed for various analytical or reporting purposes. Data lakehouses are an evolution of data lakes that aim to bridge the gap between data lakes and traditional data warehouses. They combine the benefits of data lakes (scalability, flexibility) with the benefits of data warehouses (structured data, SQL-based querying). With Azure Data Lake Storage Gen2, you can store data in its original form and then use Azure services like Azure Databricks, Azure Synapse Analytics, and other analytics and machine learning tools to process and analyze the data in a batch or real-time manner.

Since the volume of data is big for the big data application, big data solutions frequently need to use complex batch processing to process files in order to filter, combine, and get data the ready for analysis. You can use Hive, Pig, or custom Map/Reduce jobs in an HDInsight Hadoop cluster.

Another option is to use Java, Scala, or Python programs in an HDInsight Spark cluster. If the end user is comfortable with SQL, you can also perform U-SQL queries in Azure Data Lake Analytics or Databricks SQL Analytics.

Sometimes on the batch pattern, there are also requirements to process the data without any additional latency. In Azure, streaming ingestion can be enabled using Azure Event Hub, Azure IOT Hub, and Kafka. Once the data is ingested in near real time, it must also be processed by filtering, aggregating, and otherwise getting the data ready for analysis. The stream data that has been processed is then written to an output sink. A managed stream processing solution based on continuously running SQL queries that work with unbounded streams is offered by Azure Stream Analytics.

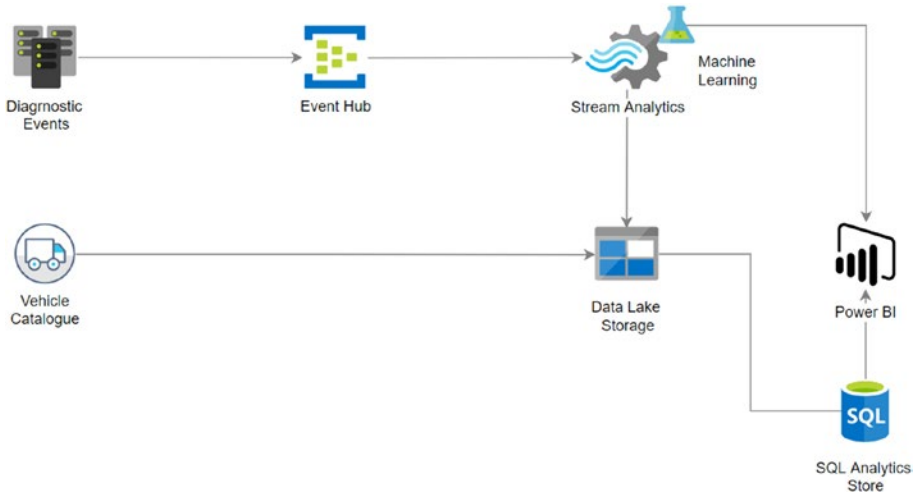

Figure 3-2. *Stream Processing using Azure Services*

As shown in the Figure 3-2, Azure Stream Analytics is one of the reference architectures in Azure that implements stream processing. Stream data is generated in the form of events and, using event hubs, this data is made available to stream analytics for processing. In the stream analytics, you can use various sink connectors to prep the processed data for analytics.

Data Analysis

The process of cleansing, converting, and modeling data in order to find the required data for decision-making is known as *data analysis*. Extracting needed information from data and making decisions based on that information are the main activities of data analysis. You do data analysis every time you make a decision in your daily life. For example, you might consider what happened previously or what would happen if you made that particular choice. This is nothing more than evaluating the future choices based on need. You use your past experiences or your future aspirations to make decisions. Today, data analysis refers to the same process an analyst uses for business goals.

There are various types of tools available in the market for data analysis. The following are some examples of data analysis tools in Azure.

- Azure Analysis Services
- Azure Data Factory
- Azure Data Explorer
- Azure Data Lake Analytics
- Azure Synapse Analytics
- Azure Databricks

There are multiple types of data analytics techniques based on business and technology. Some major data analysis methods are as follows:

- Text analysis
- Statistical analysis
- Predictive analysis

The data analysis process involves more than collecting data using the right tool to study the data and identify patterns.

Phases of data analysis include the following:

- Data collection
- Data filtering
- Data analysis
- Ad hoc data analysis

Data Visualization

In our daily lives, data visualizations are highly prevalent. They are mainly in the form of graphs and charts. Facts presented visually make it simpler for the human brain to comprehend and digest them. Data visualization is frequently used to find new patterns and perform trend analysis. You can discover a strategy to find important information by looking at relationships and contrasting datasets.

The graphical representation of data is known as *data visualization*. Data visualization tools offer an easy approach to observe and analyze trends, outliers, and patterns in data by using visual elements like charts, graphs, and maps. Additionally, data visualization is a great tool for business analysts to clearly deliver data to business stakeholders.

To analyze large volumes of data and make data-driven decisions, data visualization tools and technologies are crucial in the world of big data. Data visualizations offers various advantages:

- Enjoy simplified information exchange
- Investigate possibilities in conversation
- Visualize relationships and patterns

The design might incorporate a data modeling layer to enable users to examine the tabular data model in Azure Analysis Services. Using the modeling and visualization tools in Microsoft Power BI or Microsoft Excel, it might also offer self-service BI. Data scientists and data analysts can

perform interactive data exploration as part of their analysis and reporting. Since many Azure services enable analytical notebooks like Jupyter in these scenarios, users can take advantage of their existing Python or R capabilities.

Data Governance

There are security and privacy constraints in place from the point at which data is ingested through processing, analysis, and storage. Data governance has gradually gained attention and for good reason. Having a data governance structure in place will strengthen your attempts to transform your business into a data-driven one in light of forthcoming mandates on data privacy. You can implement row-level security and data masking in the Azure SQL Database using a built-in feature.

Microsoft's Azure Purview organizes and centralizes a lot of data. You can manage the resources locally, across several clouds and SaaS using this solution. It gives users the option to build a data library that serves as a comprehensive resource map. The data is categorized and tracked in the catalogue, especially the data that is deemed sensitive. As a result, users can access the information they need with ease. Each of the company's data is carefully tracked, with a thorough history of the database actions taken and the access rights obtained. This tool's various capabilities enable data management and automation, as well as simple data classification through integrated and customizable sorting.

Using Batch Processing

You can use the batch processing to regularly finish high-volume, repetitive data processes. Processing individual data transactions might be expensive and inefficient for some data processing operations since the volume of data is big. Data systems, on the other hand, process them

in batches, frequently during off-peak hours when computing resources are more readily available, such as at the end of the day or throughout the course of the night.

Batch processing is used by businesses because it reduces the need for direct manual interaction and boosts the productivity of daily activities. To balance the workload on the system, you can schedule batches of jobs containing millions of records to be processed simultaneously when compute power is most easily available. Additionally, modern batch processing requires little to no human management or monitoring. The system immediately alerts the relevant personnel to address any issues if they arise. Managers adopt a hands-off strategy and rely on their batch-processing tools to complete the task at hand.

Some examples of batch processing are as follows:

- Yearly billing cost
- Inventory management and processing
- Supply chain management
- Payment subscription cycles
- Monthly/yearly report generation

From straightforward data transformations to a more comprehensive ETL pipeline, batch processing is employed in a number of situations. Batch processing may be used in a big data setting where very large datasets are involved and the computation requires a lot of time. Typically, batch processing produces data that is suitable for modeling and machine learning, or it writes the data to a data storage that is designed for analytics and visualization.

Long-running batch processes are frequently used in big data systems to filter, aggregate, and otherwise get the data ready for analysis. These tasks often entail reading source files from scalable storage such as HDFS, Azure Data Lake Store, and Azure Storage, processing them and then writing the output to new files in scalable storage. Such batch processing

engines must be able to scale out computations in order to handle massive amounts of data. Batch processing, in contrast to real-time processing, is anticipated to have latencies that range from minutes to hours.

Once batch processing has started, user interaction is not necessary. This distinguishes batch processing from transaction processing, which needs human input and includes processing transactions one at a time. There are also various technology choices available in Azure to implement the batch processing. Processing transactions in a group or batch is known as batch processing. Although it can be done at any time, batch processing is especially well suited to end-of-cycle processing, such as processing a bank's reports at the end of the day or creating monthly reports.

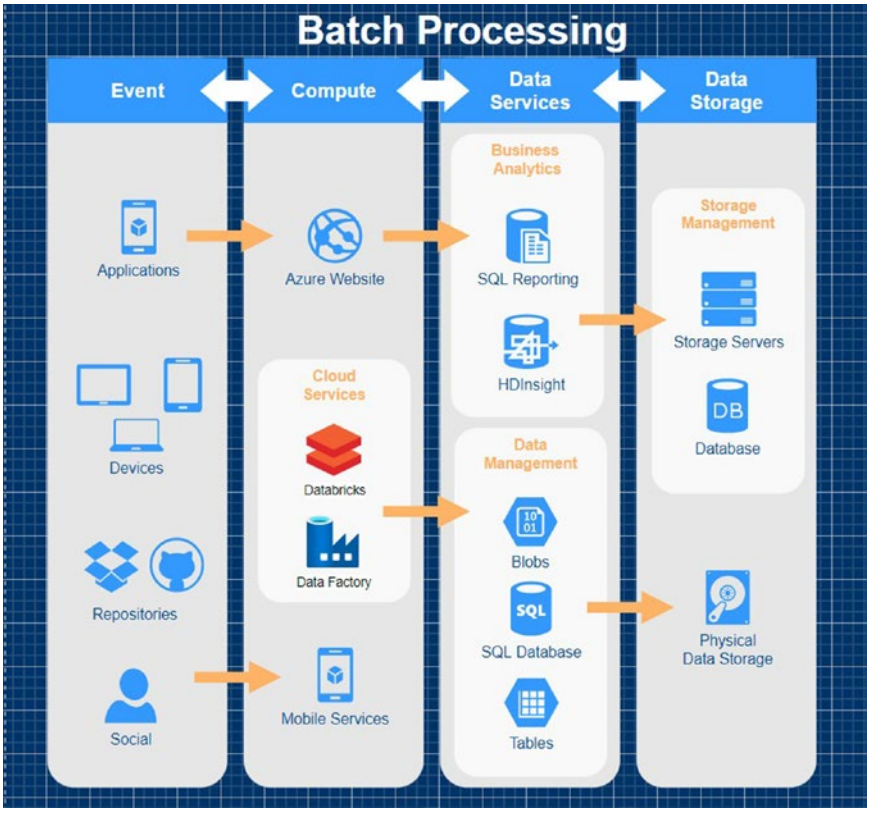

Figure 3-3. *Batch processing using Azure Services*

There are various choices available in Azure to implement batch processing, as shown in Figure 3-3. Let's explore these Azure services in detail.

Azure Synapse Analytics

A distributed system called Azure Synapse was created to analyze massive amounts of data. It can do high-performance analytics since it supports massive parallel processing. As you can see in Figure 3-4, you can implement batch processing using the combination of Synapse Analytics, Power BI, and Azure Data Factory.

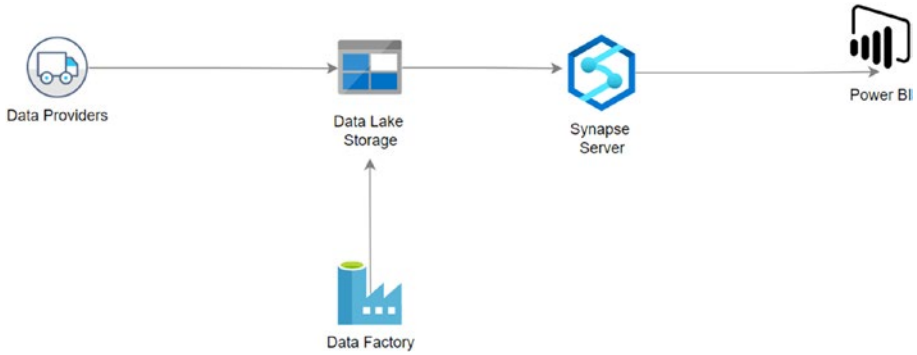

Figure 3-4. *Batch processing using Azure Synapse Analytics*

Azure Data Lake Analytics

An on-demand analytics job service can be performed using Azure Data Lake Analytics. Large datasets kept in the Azure Data Lake Store can be processed in a distributed fashion using this program.

Languages: U-SQL with extensions for Python, R, and C#.

Azure Data Lake Analytics can easily be integrated with Azure SQL Database, Azure Synapse, Azure Data Lake Store, and Azure Storage blobs.

Azure Databricks

The developers of Apache Spark formed the software startup Databricks. The company has also produced well-known software, including Koalas, MLflow, and Delta Lake. These are the well-liked open-source initiatives that cover machine learning, data science, and data engineering. Databricks creates web-based Spark platforms that feature IPython-style notebooks and automated cluster administration.

A data analytics platform tailored for the Microsoft Azure cloud services platform is called Azure Databricks. Three environments are available using Azure Databricks:

- SQL Databricks
- Data science and engineering firm Databricks
- Machine learning with Databricks

For the following scenario, you can choose Azure Databricks to implement the batch processing.

- Do you prefer using a managed service to running your own servers?
- Do you prefer to write declarative or imperative batch processing logic?
- Will you run batch processing amid peaks and valleys? If so, take into account features that allow to automatically end the cluster.
- Do you need to batch process and query relational data repositories, for example, to search reference data?

Azure Data Explorer

You can use Azure Data Explorer to perform the real-time analysis of large volumes of data coming from applications, websites, and IoT devices. It will help optimize client experiences and monitor sensor devices and operations to process real-time data on-the-fly, as shown in Figure 3-5.

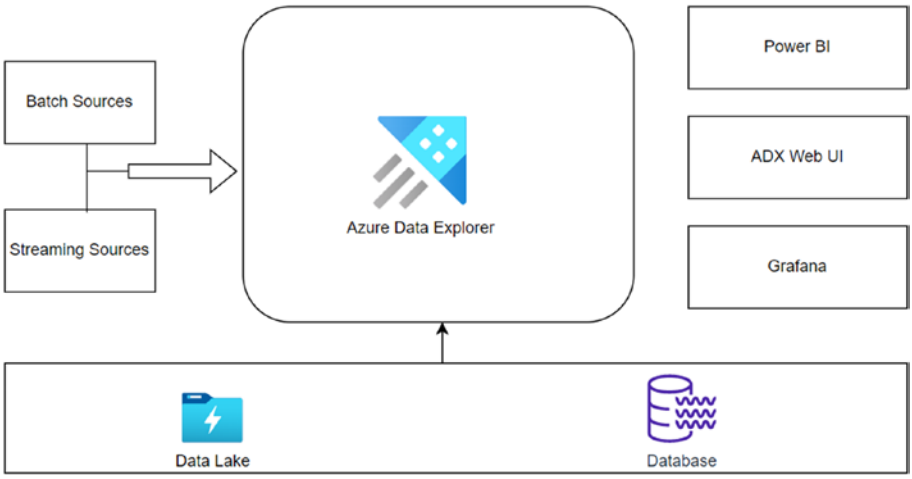

Figure 3-5. *Big data analytics using Azure Data Explorer*

Big data processing using Azure Data Explorer has the following flow:

- You ingest a variety of sources, like structured, semi-structured, and unstructured data into Azure Data Explorer.
- Using its connectors for Data Factory, Event Hubs, Azure IoT Hub, and Kafka, Azure Data Explorer ingests data with low latency and high throughput.
- Pre-aggregated data can be exported from Azure Data Explorer to Azure Storage, where it can then be imported into Synapse Analytics for the creation of data models and reports.

- Build near real-time analytics dashboards using Power BI and Azure Data Explorer to gain insights at lightning speed.
- Create a data warehouse with Azure Synapse Analytics and then mix it with other data to provide business intelligence reports.
- For machine learning, pattern recognition, and time-series analysis, Azure Data Explorer offers native advanced analytics capabilities.

Real-Time Processing

Real-time processing involves processing incoming data from numerous data sources continuously and with extremely little lag. The need to assess this data quickly in order to gain a competitive advantage is more important as businesses gather more and more data. A real-time machine or system receives your input, processes it, and produces a useful result in a matter of milliseconds.

Ingesting, processing, and storing messages in real time, especially in large numbers, is one of the major challenges of real-time processing solutions. Processing must be carried out in order to avoid clogging the ingestion pipeline. High-volume writes must be supported by the data storage. The ability to act fast on the data is another difficulty. Some people also refer to the ingestion pattern as a near real-time ingestion pattern. There is a small difference between near real-time systems and real-time data ingestion systems:

- Real-time ingestion pattern is when the information processing needs to be done immediately. For example, in a banking system or card transaction for an ATM. Real-time processing requires continual input, constant processing, and continual output. See Figure 3-6.

Figure 3-6. *Real-time ingestion process flow*

- Near real-time ingestion requires speed, but does not have to happen right away.

Real-time processing architecture consists of the following logical components.

1. *Real-time data ingestion*

   Operational executives can make decisions in time-sensitive contexts such as compliance monitoring with the use of streaming data input. You can also perform advanced analytics and transactional processes using real-time data ingestion.

2. *Stream processing*

   In order to swiftly evaluate, filter, or improve data in real time, stream processing is a data management technique that entails consuming a continuous data stream. As shown in Figure 3-7, the data is transferred to an application, a data store, or another stream processing engine after it has been processed. The ability for businesses to mix data feed from diverse sources is one reason that stream processing services and architectures are becoming more and more popular. Transactions, website analytics, and weather reports are just a few examples of various sources.

Although the fundamental concepts of stream processing have been around for a while, their implementation is becoming simpler due to a variety of open-source tools and cloud services.

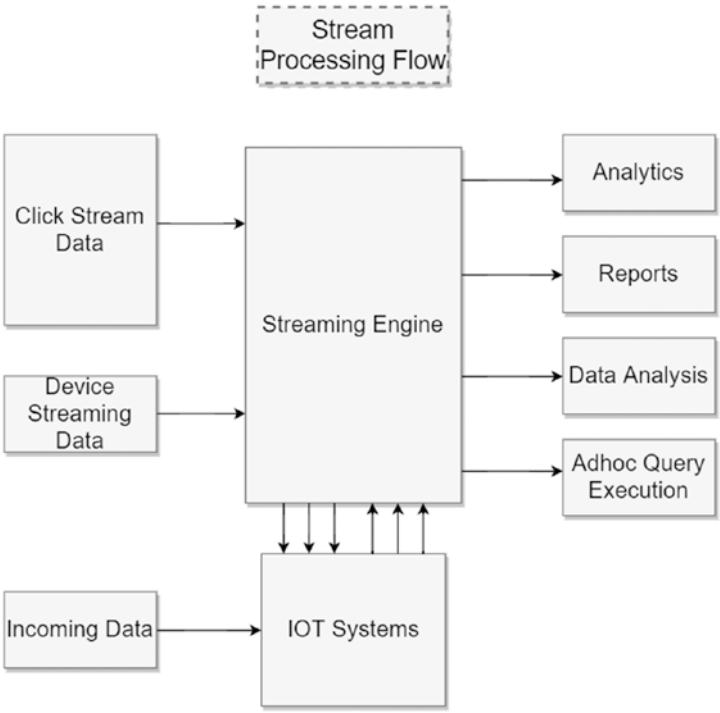

Figure 3-7. *Stream processing flow*

3. *Analytical data store*

   Big data management for commercial applications and services is the area of expertise of analytical database software. Advanced analytics and rapid query response times are features of analytical databases that have been optimized. Additionally, they are generally columnar databases that can

efficiently write and read data to and from hard disk storage. This reduces the time it takes to process a query. They are also more scalable than traditional databases.

4. *Reporting*

   Reporting provides more insights into the streaming data via data analysis and reporting tools. Let's look at the recommended Azure services for the real-time processing in Azure.

Real-Time Data Ingestion

You can do near real-time data ingestion using services like Event Hubs, Kafka and Stream Analytics, and so on.

- *Event hubs:* Used to ingest big data from a wide range of sources. You can quickly and accurately acquire events from a number of sources, and store events with dependability and durability. In order to process data quickly and concurrently, event hubs also support a number of consumers and consumer groups. You can store all incoming data in Azure Storage or alternatively, Azure Functions start up in response to fresh events.
- *Kafka*: Numerous businesses use Apache Kafka, an open-source, distributed event streaming platform, for mission-critical applications, high-performance data pipelines, streaming analytics, and data integration.

Kafka's three main features are as follows:

- Applications can broadcast data or subscribe to it.
- It is precisely in the order in which they occurred and robustly stores records.
- Records are processed in real-time, or as they come in.
- ***Streaming*** *data storage:* Azure Blob Storage or Azure Data Lake Storage Gen2 are used to store incoming real-time data captured via Event Hub or Kafka topics. Streaming data is frequently combined with static reference data that can be kept in a file store in real-time processing systems. Last but not least, real-time data captures may be sent to file storage for archiving or additional batch processing in a Lambda architecture.
- Stream Processing
  - *Azure Stream Analytics*

    Azure Stream Analytics can continuously run queries on data streams. The results of these queries are written to sinks like storage and databases or to reports in Power BI. They consume streams of data from storage or message brokers, and then filter and aggregate the data depending on temporal windows. The query language used by Stream Analytics is SQL-based, providing temporal and geographic components.

  - *Spark Streaming*

    One of the most crucial components of the big data ecosystem is Spark Streaming. To manage big data, the Apache Spark Foundation has created a software framework. In essence, it continuously ingests data from all sources, processes it with functions and algorithms, and then pushes the data out to be stored in databases and other places.

- *Analytical Data Store*

  Spark, Hive, Azure Synapse Analytics, and Azure Data Explorer can be used for distributed storage. Tables can be defined and queried and provide options for storing processed real-time data.

- *Visualization and Reporting*

  Similar to batch processed data, processed real-time data that is saved in an analytical data store can be utilized for historical reporting and analysis. When latency is low enough, Power BI can also be used to publish real-time reports and visualizations from analytical data sources directly from the result of stream processing.

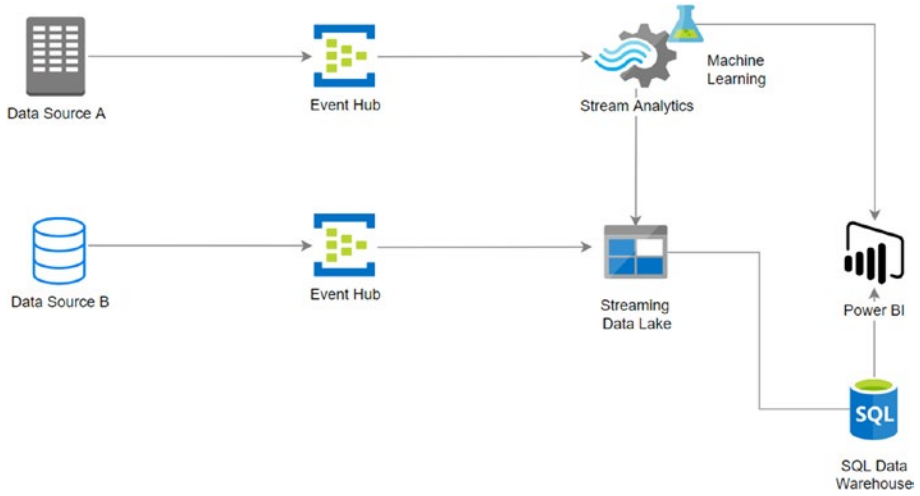

Figure 3-8. *Reference architecture for Azure Streaming Flow*

As you can see in the reference architecture in Figure 3-8, the event hub can be used to stream data in a real-time manner. Azure Stream Analytics can be used to process this streaming data. Once the stream processing is done, you can create visualization reports in Power BI for analytics purposes. This reference architecture is one of the possible architectures that can be used to implement real-time streaming patterns using Azure Services.

The Lambda Architecture

Using the Lambda architecture, you can have a hybrid approach of both batch-processing and stream-processing techniques. Using this approach, you can process enormous amounts of data. Arbitrary function computation is handled using the Lambda architecture.

Two layers make up the Lambda architecture:

- *Batch layer*: With a batch layer, you can continuously feed fresh data into the data system. At some point, the data in the stream layer is corrected after taking a comprehensive look at all the data. Various ETL pipelines as well as a conventional data warehouses can be built using the batch layer. This layer is created according to the schedule, typically once or twice every day. The batch layer serves two crucial purposes:
  - Controls the primary dataset
  - Computes the batch views in advance
- *Speed layer*: Due to the batch layer's latency, this layer manages the data that have not yet been given in the batch view. Additionally, it only works with recent data in order to provide real-time views that give the viewer a complete perspective of the data, as shown in Figure 3-9.

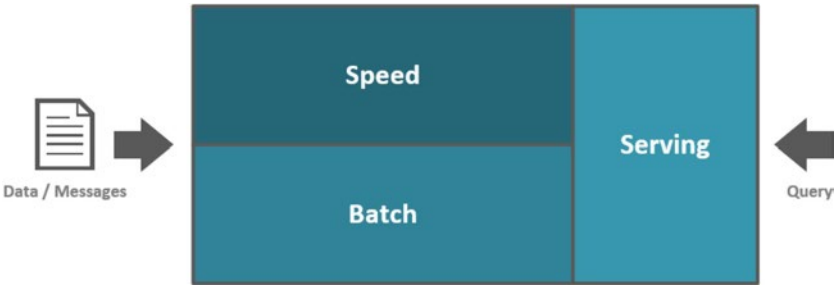

Figure 3-9. *Reference Lambda architecture*

Lambda has always been the preferred method for constructing big data processing architectures, but as technology advances, it is making way for more effective methods like the databricks delta architecture. The Lambda architecture combines batch and real-time processing techniques to handle large amounts of data. This method uses batch processing to provide thorough and accurate views of batch data while also using real-time stream processing to provide views of online data in an effort to balance latency, throughput, and fault-tolerance.

Here are the advantages of implementing the Lambda architecture:

- There is no need to install, maintain, or manage any software on the server.
- The application can be scaled automatically or by modifying its capacity.
- Automated high availability refers to the availability and fault tolerance that serverless applications already have.
- Responds quickly to shifting company and market conditions.

As far as big data computing is concerned, it is a new paradigm. However, Lambda-based apps have use cases for log ingestion and the accompanying analytics. Additionally, log messages are frequently generated quickly. They are also unchangeable. Additionally, you could refer to it as "fast data" because each log message can be ingested without the entity that contributed the data receiving a response. For example, you can track the page hits and page popularity using website click log analytics.

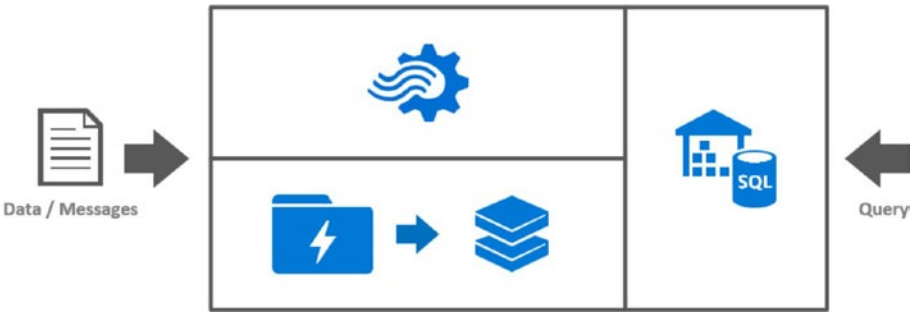

Figure 3-10. *Lambda architecture implementation in Azure*

The serverless, scalable, event processing engine Azure Streaming Analytics—which enables the creation and execution of real-time analytics on numerous streams of data from sources including devices—might be used for the speed layer. You can use Azure Data Lake Storage (ADLS) and Azure Databricks for the batch layer, as shown in Figure 3-10. You can gather data of any size, kind, and ingestion speed in one location for operational and exploratory analyses, thanks to ADLS, an enterprise-wide, hyper-scale repository for big data analytic workloads.

Data scientists, data engineers, and business analysts can collaborate on projects using the interactive workspaces and expedited workflows provided by the Apache Spark-based analytics platform Azure Databricks, which is tailored for the Microsoft Azure cloud services platform. Azure Data Warehouse—a cloud-based enterprise data warehouse that uses massively parallel processing to efficiently conduct sophisticated queries over petabytes of data—can be used as the serving layer.

This design allows for flat files that can be divided into hourly segments to be stored in ADLS after being consumed by Azure Streaming Analytics. Once a file has been processed, you can utilize Azure Databricks to construct a batch process and save the data in the Azure SQL Data Warehouse. You can use PolyBase to query the file being updated and then develop a view to combine the tables in order to acquire the data that was

missed by the batch operation. You have a real-time query with the ability to access the most recent data because the PolyBase table will always receive the most recent streamed data whenever that view is queried.

There are a few limitations of a Lambda big data architecture implementation:

- *Fusion architecture style:* The Lambda architecture requires managing two distinct codebases for batch and stream processing as well as guaranteeing consistency between them, which can make it difficult to implement and maintain.
- *Latency*: The Lambda architecture may still experience some latency difficulties depending on how frequently batches are processed and how accurately streams are processed. It might not be appropriate for applications that need a very high degree of accuracy or very little delay.
- *Resource consumption*: Because the Lambda architecture processes the same data twice through both layers, it can use a lot of resources. Additionally, it might cost more for cloud-based solutions.

The Kappa Architecture

A software architecture for handling streaming data is called the Kappa architecture. It is mainly used for analytics purposes. You can execute real-time and batch processing using the same technology stack. Kappa is built on a streaming architecture where incoming data streams are initially stored in a message engine like Apache Kafka. The data will then be read by a stream processing engine, formatted for analysis, and stored in an analytics database for end users to query. The Kappa architecture is useful for on-demand analytics.

When the data is read and changed right away after being fed into the message engine, the Kappa architecture offers real-time analytics. As a result, end user requests can access recent data rapidly. Additionally, Kappa supports historical analytics by subsequently accessing the batch-processed, stream-stored data from the messaging engine to produce more insightful outputs for different types of analysis.

Since it can manage both real-time stream processing and historical batch processing using the same technological stack as the Lambda architecture, the Kappa architecture is seen as a more straightforward option. For large-scale analytics, both designs require storing historical data. Both architectures can also be used to handle "human fault tolerance," where issues with the processing code can be resolved by changing the code and rerunning it on the historical data. As a result of treating all data in the Kappa architecture as though it were a stream, the stream processing engine serves as the only data transformation engine, as shown in Figure 3-11.

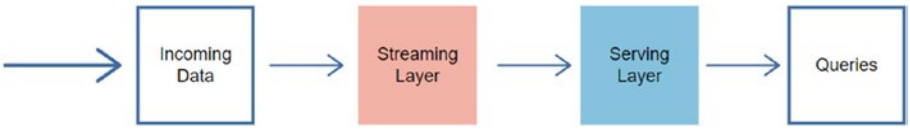

Figure 3-11. *Kappa architecture implementation*

This is a streaming architecture implementation. As a result, it is predicated on the notion of employing a stream processing engine to change data and compute aggregations while permanently storing a succession of incoming data. The stream processing engine's aggregates are delivered to another stream. Finally, you can either create a materialized table with the continuously updated results or ingest the events from an aggregated stream.

Data streams are continuous sequences of data records or events, and stream processing entails ingesting, altering, and consuming these streams of data. A stream processor is a piece of software that can

perform a number of operations on data streams, including complicated event processing, enrichment, aggregation, and filtering. In addition to producing output streams that can be used by other programs or stored in databases or data lakes, stream processors can also produce output streams. Both structured and unstructured data can be processed using stream processing, which can also process data in windows or out of order.

For stream processing, the Kappa architecture has many benefits, including simplicity, scalability, and flexibility. It lessens the complexity and maintenance burden of the data pipeline by utilizing a single processing layer and a single source of truth. Additionally, it does away with the necessity to synchronize and reconcile the batch and real-time layers. Additionally, the Kappa architecture can scale up or down the processing resources in accordance with demand and workload while handling massive quantities and speeds of data with high availability and fault tolerance. Additionally, varied use cases and requirements for stream processing can be supported by using a stream processor that can handle a variety of data formats and functionalities.

Because the event data is immutable and gathered in its entirety rather than just as a selection, there are some similarities to the batch layer of the Lambda architecture. The information is added to a distributed and fault-tolerant unified log as a stream of events. These events are sequential, and only the addition of a new event can alter an event's existing state. The input stream is used for all event processing, much like the speed layer in a Lambda architecture, and the results are saved as a real-time view. You simply replay the stream to recompute the full dataset, generally using parallelism to speed up the computation similar to what the batch layer in Lambda does.

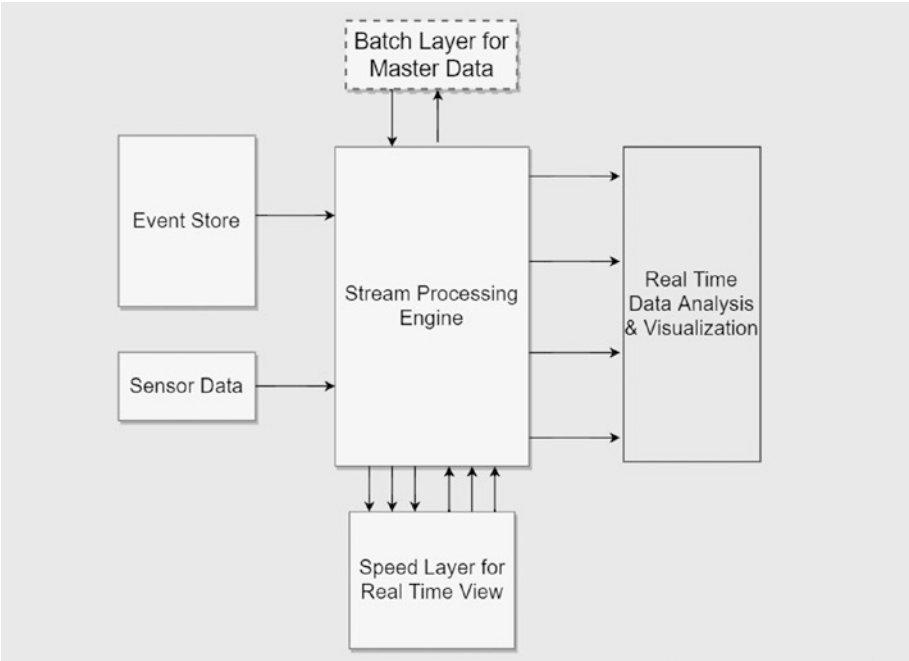

Figure 3-12. *Kappa architecture process flow*

As you can see in Figure 3-12, there is a single stream engine to process the streaming the data. Event data can be generated from the different systems or applications and the batch and streaming processing can both be handled using the same framework. This way, latency can also be reduced to ingest and process data in a near real-time manner. Now that you understand the Kappa architecture with a generic flow, it's time to explore how it can be implemented using various Azure services, as shown in Figure 3-13.

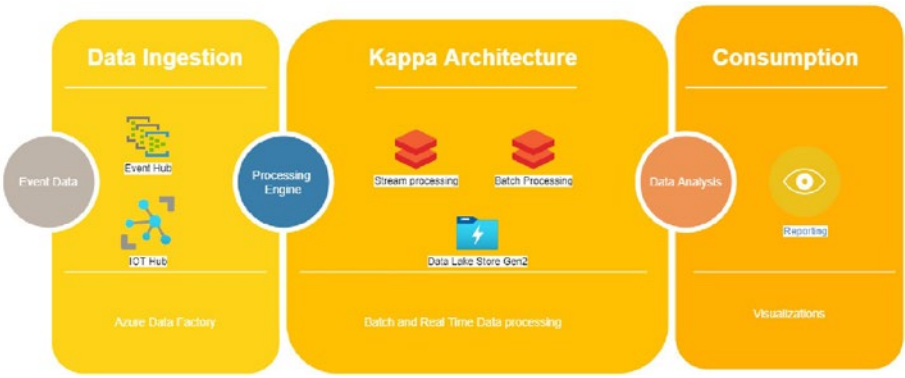

Figure 3-13. *Kappa architecture's Azure implementation*

The ingestion layer is unified and processed by Azure Databricks, as shown in Figure 3-13. A specific kind of storage is required to allow queryable and aggregated data and the delta lake open-source technology can help here. Apache Spark and big data applications now support ACID transactions using the open-source storage layer, known as Delta Lake. You can create Delta Lake tables against the Databricks File System using databricks. Azure storage, such as Azure Blob Storage and Azure Data Lake Storage, can be mounted using the DBFS.

Using setup tools, Delta Lake on Databricks can be designed according to workload patterns and offers optimized layouts and indexes for quick interactive analysis. Data can be processed using different databricks notebooks and saved as a thin layer on top of the data lake, using different tables, thanks to the features of Delta Lake. Data from Delta Lake tables can be accessed using a variety of clients in batches and almost real-time utilizing a single pipeline. As a result of avoiding data administration and numerous storage systems, this integrated strategy is less complex. The primary benefit is the ability to simultaneously run queries on historical and flowing data.

There are few advantages of implementing the Kappa architecture:

- Compared to the Lambda architecture, Kappa architecture is easier to set up and maintain since it uses a single data processing system to manage both batch processing and stream processing workloads. By lowering this complexity, this can make it simpler to improve the data processing pipeline.
- The Kappa architecture allows reprocessing to happen immediately from stream processing jobs, despite the fact that it is not meant for a particular set of difficulties.
- Due to a simple stream-processing pipeline, migrations and reorganizations may be carried out using fresh data streams produced by the canonical data store.
- Without the requirement for a conventional data lake, the Kappa design is a cost-effective and dynamic data processing solution.

In a Kappa architecture system, an append-only immutable log serves as the data store rather than a relational database like SQL or a key-value store like Cassandra. Data is streamed through a computing system and sent into auxiliary stores for serving after being extracted from the log. The Kappa architecture is intended to process enormous amounts of data in real-time. It can handle a variety of data processing workloads with a single technological stack, including continuous data pipelines, real-time data processing, IoT systems, and many other use cases.

There are also some limitations of the Kappa architecture design implementation as follows:

- Even though the Kappa architecture is less complicated than Lambda, it can be challenging to set up and maintain, particularly for companies that are not familiar with Stream.
- Event streaming implementation on the cloud platform can become costlier.

Businesses must carefully consider their choice of data processing architecture because it affects the pipeline's scalability, performance, and flexibility. Businesses should select a big data architecture that satisfies their unique requirements and thoroughly weigh the advantages and disadvantages of each choice before deciding. In general, Kappa is a good place to start when developing a system that requires real-time data access. Based on the maturity of your organization, you can move toward the Kappa architecture implementation.

Internet of Things (IoT)

The Internet of Things is a network of physical objects, including machines, cars, and devices that have sensors, software, and network connectivity built into them to enable data collection and sharing. These items, commonly referred to as "smart objects," include everything from "smart home" gadgets like thermostats to wearables like smartwatches and RFID-enabled clothes, to sophisticated industrial machinery.

This large network of interconnected devices can exchange data and carry out a range of tasks on their own. This is made possible by IoT, which enables these smart gadgets to communicate with one another and with other Internet-enabled devices, like smartphones and gateways. This can involve activities from tracking inventory and shipments in

warehouses to operating machinery and processes in factories, monitoring environmental conditions in farms, regulating traffic patterns with smart cars and other smart automotive devices, and more.

By building Azure PaaS (Platform as a Service) components, you can develop unique IoT applications. Azure services and its components are frequently used by IoT solutions. You can use the following services for management and business integration:

- Power BI creates models and visualizes your data.
- Azure Maps creates location-aware web and mobile applications and services like search, maps, routing, tracking, and traffic.
- Azure Cognitive Search provides a search service. Cognitive Search includes indexing, AI enrichment, and querying capabilities.
- Azure API Management provides a single place to manage all APIs.
- Azure App Service deploys web applications that scale in an organization.
- Azure Mobile Apps builds cross platform and native apps for different OSs.
- Microsoft Power Automate is an SaaS offering for automating workflows across applications and other SaaS services.

  Azure also offers a number of services that can be used to protect and monitor complete IoT solutions. You can control, view, and manage security settings as well as threat detection and response with the use of security services like Azure Active Directory and Microsoft Defender for IoT.

While developing IoT solutions on Azure, a few security considerations should be in place:

- *Device Security*

  IoT devices should be secure during deployment.

  – Hardware requirements for IoT devices

  – Choice of tamper proof hardware

  – Enable security upgrades

  – Use open-source software with care

  – Keep authentication and authorization keys safe and secure

  – Monitor and audit

  – Protect

- *Interface Security*

  Assure the security and integrity of all data transferred between the IoT device and the IoT cloud services.

  – Use TLS 1.2 to secure connection from different devices

  – Keep TLS certificates up to date

  – Use Azure Private Link connections as and when possible

Data Mesh Principles and the Logical Architecture

Nearly all businesses in the era of self-service business intelligence identify as data-first businesses, but not all businesses are approaching their data architecture with the democratization and scalability it requires.

Many data teams wish that there was a simpler method to handle the expanding requirements of the company, from responding to the never-ending flow of ad hoc queries to juggling many data sources through a centralized ETL pipeline.

The data mesh is, in many respects, the data platform version of microservices, much as how software engineering teams changed from monolithic apps to microservice architectures. A *data m*esh is a particular sort of data platform architecture that embraces the pervasiveness of data in the company by using a domain-oriented, self-serve design. This is according to Zhamak Dehghani, who first defined the phrase in 2019. It adapts the domain-driven design theory of Eric Evans, which is a flexible, scalable software development methodology that aligns the structure and language of code with the relevant business domain.

A data mesh supports distributed, domain-specific data consumers and views "data-as-a-product," with each domain handling its own data pipelines (see Figure 3-14). This is in contrast to traditional monolithic data infrastructures, which handle the consumption, storage, transformation, and output of data in one central data lake. The data storage layer belongs to the domain teams. A global interoperability layer that uses the same syntax and data standards serves as the connective interface between these domains and their related data assets. Although some organizations have chosen "data meshy" architectures with platform teams owning a more centralized platform, this can lead to some infrastructure duplication.

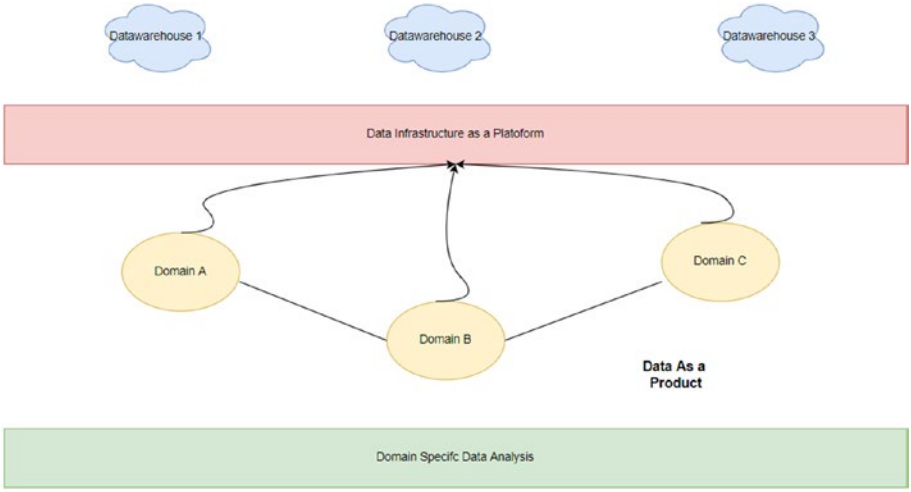

Figure 3-14. *Data mesh implementation*

Data sources, data infrastructure, and domain-oriented data pipelines controlled by functional owners make up a data mesh architecture diagram. A layer of universal interoperability, reflecting standards independent of any particular area, as well as observability and governance, underlies the data mesh architecture. The data mesh architecture moves from a centralized monolithic architecture to distributed architecture.

The data mesh architecture works on the following principles:

- *Decentralize business domain data stores*

  The domain teams are required by the domain ownership principle to be accountable to their data. This principle states that analytical data should be organized into domains, like how team boundaries match the constrained context of the system. Analytical and operational data ownership is transferred from the central data team to the domain teams as a result of the domain-driven distributed architecture.

- *Data as a product*

  The data as a product principle applies the notion of product thinking to analytical data. According to this notion, data has users outside of the domain. The domain team is in charge of supplying high-quality data that satisfies the needs of other domains. Domain data should essentially be handled the same way as any other public API.

- *Self-service data platform*

  Employ a platform-based approach to data infrastructure. To create, implement, and maintain interoperable data products for all domains, a specialized data platform team offers capabilities, tools, and systems that are independent of any particular domain. The data platform team's platform makes it possible for domain teams to easily use and produce data products.

- *Federated data governance*

  The governance group promotes standards throughout the entire data mesh in order to fulfil the principal's goal of achieving interoperability of all data products. The federated governance's primary objective is to establish a data environment that adheres to corporate policies and industry standards.

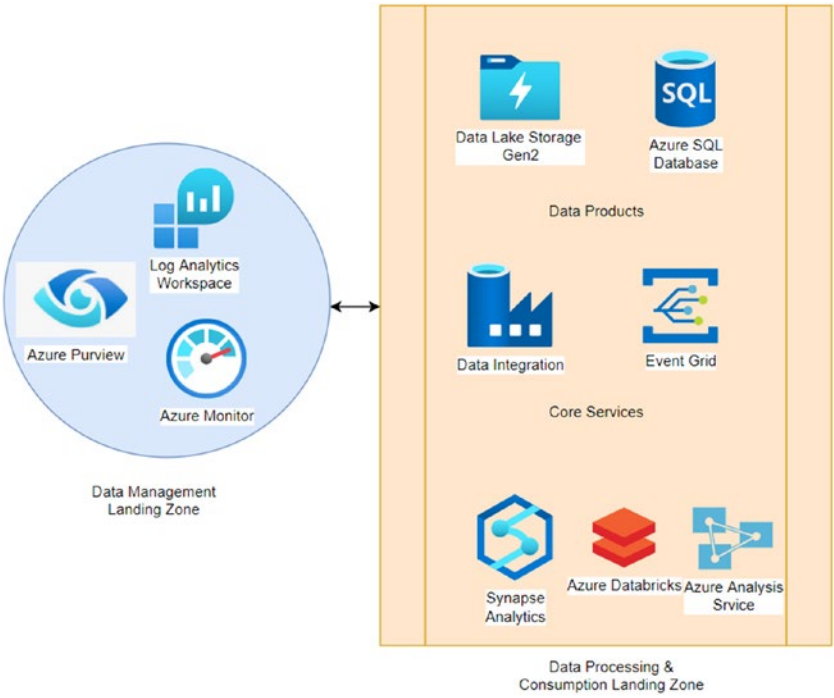

Figure 3-15. *Data mesh implementation using Azure Services*

All functional data domains will congregate in the same data landing zone under this design. For the purposes of separating different data domains and data products, this refers to a single subscription that has a common set of services and resource groups. The domain-specific data has Azure services like Azure Data Factory, Azure Logic Apps, Azure Data Lake Store, and Azure Synapse Analytics for data ingestion and processing purposes, as shown in Figure 3-15.

Therefore, everything is already set up to realize a data mesh in Microsoft Azure. The key is to focus on data integration and consider whether cross-platform software would be a better option. After that, using a variety of tools, you can often build your data mesh and data lakehouse in the first stage. But, as always, this also depends on needs; not every function is necessarily required. In this case, it's likely that the company's

size is crucial. In order to establish proper data governance and ensure that the data only reaches the relevant users with the right data quality, it is also crucial to employ the right tools while building a data mesh.

Conclusion

In this chapter, you explored in detail about the core components of the big data architecture. You read how to design and implement batch and real-time streaming ingestion patterns using Azure services. Later, you explored the Lambda and Kappa architectures in detail. At the end of the chapter, you learned about the role of the data mesh architecture and its significance and also how it can be enabled using Azure services.

In the next chapter, you inspect various data management patterns and related technology choices with respect to an Azure cloud.

CHAPTER 4

Data Management Patterns and Technology Choices with Azure

A *data management pattern* is a repetitive set of solutions that can be used to create and design a software/application that is common for most data management projects. These patterns enable reusable processes, design approaches, and set of codes for the ease of maintenance and consistency across such projects.

This chapter discusses the various data management patterns in detail and explains what Azure services can be used to implement such patterns.

This chapter covers the following topics:

- Data pattern and trends in depth
- Analytical stores for big data analytics
- Building enterprise data lake and data lakehouses
- Data pipeline orchestration
- Real-time stream processing in Azure

© Sagar Lad 2023
S. Lad, *Modern Data Architecture on Azure*, https://doi.org/10.1007/978-1-4842-9760-5_4

Data Patterns and Trends in Depth

Data design patterns are widely used in software engineering to design and implement data-driven solutions. Design patterns are a set of standard practices that enable reusability, create consistent solutions, and accelerate the overall application development to handle and manage data driven challenges for enterprise organizations. Data management is a practice of collecting, storing, securing, and organizing data throughout the data lifecycle. The following sections look in detail at the common data management patterns.

CQRS Pattern

CQRS stands for Command and Query Responsibility Segregation. It is a design pattern that separates reading and writing data from its storage (see Figure 4-1). It has two separate service models, as mentioned here:

- *Query Service Model*: The end users use this service model to read the data from the data storage.
- *Command Service Model:* The technical team uses this service model to update the data in the data store.

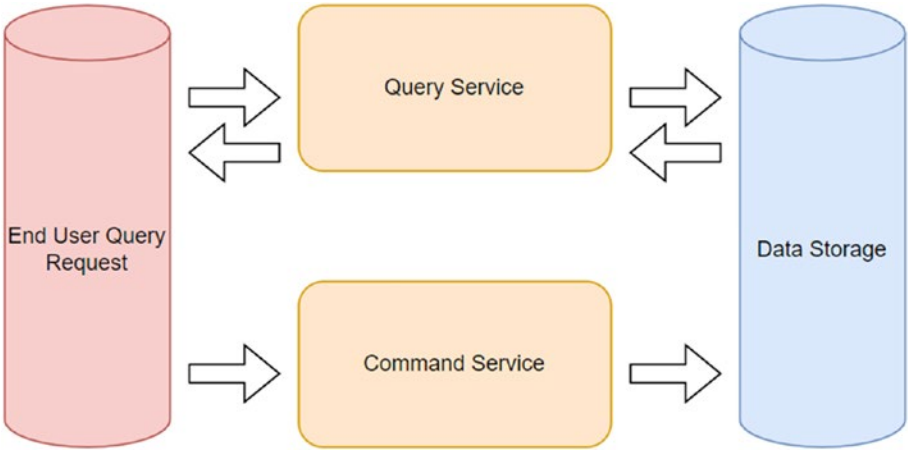

Figure 4-1. *CQRS pattern: command and query responsibility segregation*

The CQRS pattern can be a good choice of implementation in the following scenarios:

- High read operations for downstream data consumption
- Must manage the performance of read and write operations based on the operations needs
- Application is required to change over a period of time using various versions of the data model or changes in business rules
- Integrate the application with an event streaming capability where the temporary failure of the one of the systems doesn't affect the other

With CQRS implementation, the advantage is that read and write requests are separately managed with independent scaling. It results in simpler queries, more security, and optimized database schema for both data read and writes. During the CQRS design and implementation, you have to make sure that the complexity of the code is apt. During some scenarios, the data may become stale while reading from storage.

Let's look at an example of the CQRS pattern and its implementation by understanding the scenarios of a banking system. The customer of a bank requests to withdraw money from their account and when they check the account balance, the correct account balance should be displayed without any delays. You can implement CQRS implementation in such scenarios. The most common example of implementing the CQRS pattern is using Kafka as a service bus and consuming those messages from the Kafka topic via Azure Databricks or Azure Functions in a near real-time manner. See Figure 4-2.

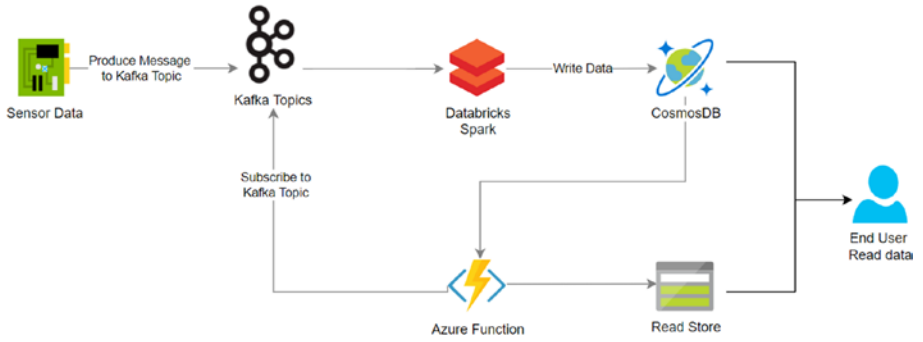

Figure 4-2. *CQRS implementation using Azure Services*

As you can see in Figure 4-2, you can use Kafka as a service bus to receive the messages from various applications in an sync manner. You can use Azure Databricks and/or Azure functions as a sink connector to subscribe to those topics. That way, you can receive the messages, process them, and make them available to the end users in a near real-time

manner. As soon as there is a message in the Kafka topic, spark streaming processing in databricks reads and processes the message to make the data available for downstream consumption.

Event Sourcing

The fundamental idea of the event sourcing pattern is that all the changes to the application are managed in the form of various events. Events will be used to trigger and manage all the communication between application components built using the micro services. It stores data in the form of events in an append only log. In order to get an overall picture of an entity at a specific point of time, you can look at the events in the order in which they arrived. The Event Sourcing architecture offers several benefits:

- *Audit log*: Since data is stored in the form of events in the order in which it arrives, it can be very much useful to form a strong base for audit logs.
- *Data historization and time travel*: All the data is stored in the form of events and can be tracked at any point of time.
- *Event-driven architecture*: In the event-driven architecture, data is processed as soon as it arrives or is available to avoid any delay in using this data from the end consumption point of view. That stream of events creates a notification as soon as the data arrives.

Materialized Views

A *materialized view* is a pre-calculated data based on the query specification and stored so that it can be used later. Data is calculated very early, and querying the materialized view is faster than querying the actual

data. It can also help optimize the query with high compute cost and a small dataset. Materialized views can improve query performance in the following scenarios:

- Streaming data aggregation
- Run queries to execute subsets of data
- Join small and large sets of data

The difference between a normal view and a materialized view is that the normal view is a virtual copy of the data while the materialized view exists physically.

You can use Azure Databricks SQL to create a materialized view on top of data stored in Azure.

For example:

```
CREATE MATERIALIZED VIEW banking_orders
        AS
        SELECT
          c.name,
          sum(c.amount),
          o.orderdate
        FROM trx_orders o
          LEFT JOIN customers c ON
            o.custkey = c.c_custkey
        GROUP BY
          name;
```

Index Table Pattern

In the database, when certain columns are used more frequently, it is best to create an index to those columns to improve the performance of the query. See Figure 4-3.

Figure 4-3. *Table indexing*

When you use an index on a SQL table, it will try to scan the records based on the index instead of scanning all records from the table. One of the best examples of indexing is the index of this book, where high-level topics are listed with the page number so it is easier to find the relevant topic based on your requirements. In SQL server, there are two types of indexes:

- *Clustered index*: In the clustered index table, the whole table data is stored based on the index key, which consists of one or multiple columns.
- *Non-clustered index*: In the non-clustered index table, the index stores only the index key values that are pointing to the actual storage location of the clustered index.

  Design considerations and choices for creating an index on a table is not straightforward. The following pointers should be considered while making design choices for SQL index.

- *Database design*: For OLTP databases, it is not recommended to create indexes in order to avoid overloading the database because of the large number of data insertions and modifications. If the database is designed to be of an OLAP type, then indexing is recommended since most of the end users will be querying the database to read data from such a database.

- *T-SQL query*: It is recommended to use a single query to perform updates and deletes instead of writing multiple queries to perform such transactions on the database. This will reduce the overall overhead of the indexes while executing data modification statements on the SQL database.
- *Index storage*: There is a major impact on how and where data indexes are stored. When the non-cluster index is stored on a file group located in a different disk drive than the disk drive when the main table is created, it will improve the performance of the query.

Analytical Store for Big Data Analytics

While designing and implementing big data architecture applications, you very often need an analytical store that has data and can be used to query and fetch the data from the database. You can use various analytical tools to store and process big data for these analytical purposes. Analytical stores serve as a golden layer for the downstream data consumption for the data stored in an analytical store. This data is ready to use, on top of which visualization reports can be built for analytics purposes and machine learning models can be built using the data stored in the analytical store.

In Azure, you have various options for building an analytical store for the data serving storage. The following sections explore each of them in detail.

Azure Synapse Analytics

Enterprise organizations are increasingly facing challenges with the ever increasing volume of data and the rate at which data is increasing. When the volume of data increases beyond 1TB, it is often called a high volume

of data. In order to process this high volume of data more efficiently, companies must choose the best tools and techniques. The Azure Synapse offering from Microsoft helps you process and store large volumes of data efficiently and make it available to your end users. Fundamental to the Azure Synapse implementation is Azure Data Lake Storage Gen2, which enables decentralized storage of data. See Figure 4-4.

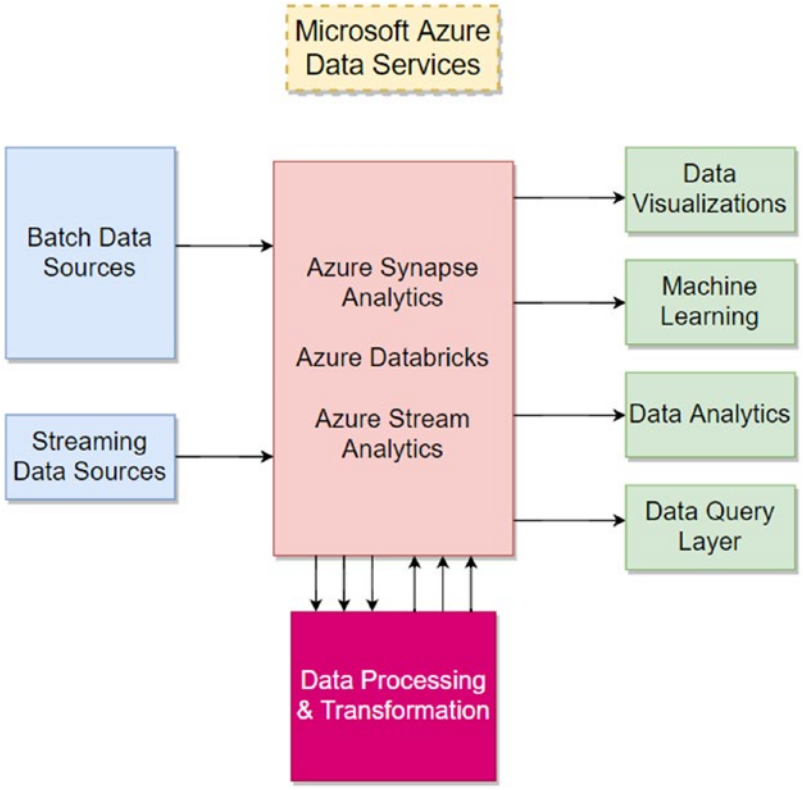

Figure 4-4. *High volume data processing with Azure services*

Using the Spark and SQL pools, you can process the data using Azure Synapse Analytics for distributed data processing. With Apache Spark, you can optimize the data preparation and ETL processing steps using the machine learning and BI tools. It combines the capabilities of both

data warehousing and data analytics. With a serverless SQL pool, you can directly query the data from the data lake. With Synapse, you can process high volumes of data with the following benefits:

1. Create a single source of truth data by collecting and combining data from various sources.
2. Store data securely in a data lake with encryption and threat detection methods.
3. Use the machine learning capabilities of Apache Spark pool for data analytics.
4. Enjoy compatibility with tools like Power BI for the visualizations using the data stored in the data lake.
5. Use a unified tool that has the storage, processing, and analytical capability to tackle the challenge of a large volume of data.

Azure Databricks

The Databricks Lakehouse platform enables data engineering projects for easier ingestion and transformation for both batch and streaming data. It is a one-stop-shop for all the ETL (Extract, Transformation, and Load) work. It enables all business users, data engineers, and platform administrators to work with data-related projects. It simplifies the overall data architecture by breaking down the data silos among various teams and creating a single source of truth for data engineers, machine learning engineers, data scientists, and data analysts. It focuses on addressing the data reliability and ease of managing data constraints by adding the data warehousing capabilities to the data lakehouse:

- ACID compliance
- Open-source tools and technologies
- Support for schema drifts

- Efficient authentication and authorization
- Support of a SQL-like query mechanism
- Feature to upsert data

Native integration of Azure Databricks with Data Factory, Blob Storage, Data Lake Storage Gen2, Azure SQL DB, and Cosmos DB makes it a perfect fit for designing and implementing data-related workloads on Azure.

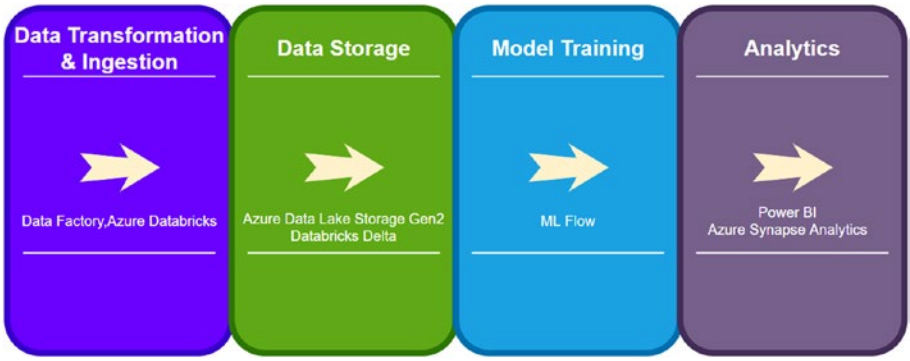

Figure 4-5. *Modern data engineering on Azure*

In order to design your data engineering workload using Azure Databricks, there are four main pillars to consider, as shown in Figure 4-5.

Data Ingestion Process

Data can reside anywhere. It can be in an on-prem system or cloud application. Using an orchestrator like Azure Data Factory, you can fetch the data from various applications. Azure Data Factory has more than 50 connectors to integrate data from various sources. It offers lower latency and serverless compute for the data integration. You can combine the data factory with various other Azure offerings to simplify the data integration.

- Data Factory and Azure Functions
- Data Factory with Azure Databricks

Data Storage

Azure Data Lake Storage Gen2 is where the extracted data is stored. Data Lake Storage Gen2 is an HDFS-compliant storage that has limitless capabilities. Various types of data formats can be stored in Azure Data Lake Storage Gen2, including parquet and delta formats. The advantage of storing data in the delta format is that it supports ACID properties, schema enforcement, data historization, and audit history.

Data Transformation and Model Training

In most data engineering projects, you can process data in the form of batch and real-time processing. Databricks has a built-in Apache Spark class to streamline the data workflow and deploy the ML workflow using Azure Databricks. Databricks also offers a Delta Live Table to simplify the data transformation process. The Delta Live Table helps data engineers and analysts use cleaned data rapidly with simplified pipeline development and maintenance using the pipeline. It also simplifies the operational complexity with automated administration tasks, which have built-in data quality and monitoring measures. In order to implement the real-time streaming pattern, Azure Event Hub can be used for the near real-time data ingestion and for quick data consumption from downstream.

Analytics

In the analytics and serving layer, you can use Synapse Analytics in combination with Azure Databricks to serve the data for downstream data consumption. This combination makes the architecture highly scalable, on demand, and scalable of processing data at lightning speed. In order to use the data for analytics purposes, the databricks serverless SQL can be used to query data using SQL-like queries. There is no need to manage, configure, or scale additional infrastructure to set up the SQL layer.

Azure Data Explorer

Enormous amounts of data is being generated consistently and in a variety of formats. Effective techniques should be used to handle this large volume of data. Azure Data Explorer (also known as ADX) is a highly scalable, fast, and fully managed data analytics service for streaming data. In the Azure Data Explorer, you can use the SQL-like language called KQL to analyze the data from the streaming sources. Azure Data Explorer offers the following features:

- Faster data ingestion with real-time data analytics and reporting
- Integration with various type of data sources, including the structured, semi-structured, and unstructured data
- The advanced and powerful Kusto Query Language (KQL), which supports advanced data processing functions
- Easier integration and compatibility with data visualization tools like Power BI

Businesses can unlock the full potential by integrating Azure Data Explorer with Azure Data Lake Storage Gen2 and enabling the real-time analytics to run complex queries and execute the advanced analytics on the stored data. By integrating Azure Data Explorer with data lake storage, you can enjoy the following benefits:

- *Faster actionable insights*: End users can obtain real-time, quick insights by querying the data directly from Azure Data Lake Storage Gen2.
- *High scalability*: Azure Data Explorer and Azure Data Lake Storage Gen2 are designed to handle massive volumes of data so businesses can scale their data analytics capabilities as required.

- *Simplified data management*: Azure Data Explorer also simplifies its overall data management capabilities by securely accessing and managing the data with authentication and authorization on the stored data.

Azure Data Explorer provides complete and comprehensive solutions for unlocking the full potential of the data and derive meaningful results.

Considering the variety of Azure Services available to build the analytical capabilities, based on different scenarios and requirements, you can make a correct choice by choosing an Azure Service. See Table 4-1.

Table 4-1. *Choice of Analytical Store Database*

Scenario	Choice of Azure Service
Relational database	Azure SQL DB
Relational table with column storage	Azure SQL Synapse Server
Logging and telemetry data	Azure Data Explorer
Semantic models	Azure Analysis Service
Scalability	Azure SQL DB, Azure Synapse and Azure Data Explorer, Cosmos DB
Data encryption and row-column level security	Azure SQL DB, Synapse, Data Explorer and Azure Analysis Service
Columnar database	HBase

Building Enterprise Data Lakes and Data Lakehouses

Each enterprise organization is becoming more and more data-oriented because it helps them make better decisions, improve their products, and provide better customer service. In order to store and use this data,

there are various data solutions available in the market. The best storage solution maximizes the use of this data and creates value out of it. This can be a complex and challenging task, considering the variety of patterns and tools available in the market. In order to select the best solution for your company, it's important that you understand two popular building data solutions:

- Enterprise Data Lake
- Enterprise Data Lakehouse

Enterprise Data Lakes

In enterprise data lakes, you can store all different types of datasets: structured, unstructured, and semi-structured data. You can store raw data centrally in a data lake in the form of batch patterns and near-real-time streaming patterns. Data can be retrieved from storage as necessary and processed for a specific purpose, such as validation, classification, reporting, and so on. Data is being stored in the data lake without any transformation into a raw format.

Data lakes offer a scalable, affordable, and (most crucially) adaptable storage solution. The problem with data lakes is that they are more challenging to maintain in terms of organization, and it might be challenging to locate the necessary piece of data inside them. The main purpose of building a data lake is to handle and process large volumes of data in its original format. You can use the data stored in the data lake in the following scenarios:

- Simplified and well-governed data management and storage
- Secured and fine-grained data access
- Accelerate AI and machine learning applications

- Empower data analytics and business intelligence
- Data democratization using self-service tools
- Seamless data integration from various data sources

Traditional data warehouse implementations of relational databases are not well suited for storing semi-structured and unstructured data. Relational systems can only normally store structured transactional data since they have a fixed data schema. Data lakes don't require any upfront definition and support a variety of schemas. They can manage various types of data in different forms as a result. There are no constraints on the schema of the stored data. Data lakes are therefore a crucial part of a company's data architectures. They are typically used by businesses as a platform for big data analytics and other data science and machine learning applications that involve advanced analytics methods, such as data mining, analytics, and machine learning for massive volumes of data. See Figure 4-6.

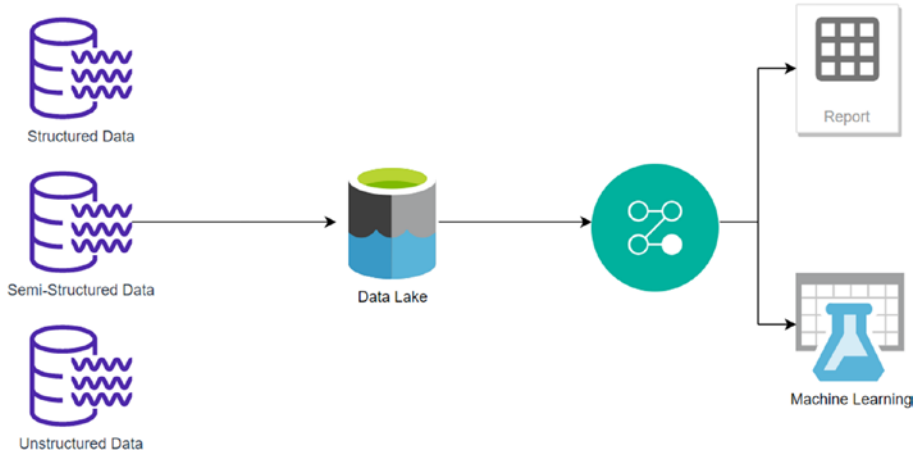

Figure 4-6. *The data lake design pattern*

The major benefits of implementing a data lake is that it can be used for both data storage and computation. You can design and implement a data lake for both on-premises and cloud storage. In contrast to a data

warehouse, data lakes and cloud data lakes are built to handle all types of data: structured, semi-structured, and unstructured data. Similar to databases and warehouses, a data lake can also manage structured data. But unstructured data that is neither processed or arranged can also be handled by data lakes.

Effective data management has become a critical need, as the amount of unstructured data in the organization has increased. Data lakes are a useful tool for storing a variety of data. They can grow data management all the way up to petabytes of data. Furthermore, a precise structure schema is not required to make the data lake implementation more flexible and easy to manage and govern.

Data lake implementations offer various benefits compared to traditional data warehouse implementations, as follows:

- *Data democratization:* Once the data is collected and available in various applications, it is available easily for everyone in the company through a data lake. So, data lakes enable data democratization. This is in contrast to traditional data warehouses, where only the highest executives currently have the luxury of requesting reports from multiple departments to gather insights then make a decision. But people in middle management and working on the actual implementation at the ground level don't have these insights and can't analyze data-driven decisions. They do not have the luxury of asking other departments for the data they require. Even if they manage to obtain the data, it is a tedious process and takes more time. With data lakes, you can overcome this limitation and help everyone make wise decisions at their level using the required information, which is easily accessible.

- *Improved data quality*: You can increase the overall data quality by maintaining data in the data lake due to the following features:
  - Versioning
  - Support of time traveling of the data
  - Data validation
  - Support of policies for data governance
- *Schema drift support*: In the traditional data warehouse implementation, the schema required for storing the data should be in a particular format. But adding these constraints for the OLAP database doesn't help since you want to use the data as is. However, the data lake enables you to establish various schemas for the same data or to be schema-free. In essence, it makes it possible to separate schema from data, which is great for analytics.
- *Scalability*: Data lakehouse implementations are scalable since they can manage both the storage and processing capabilities to manage the data. Data lakehouses can manage different volumes, velocities, and varieties of data.

In order to implement data lake solutions in Azure, you can use these services: Azure Data Lake Storage Gen2, Azure Data Lake Analytics, Azure Databricks.

Enterprise Data Lakehouses

In order to provide enterprises with an optimal data management solution, data lakehouses aim to address the fundamental issues that data warehouses and data lakes share. Data lakehouses bring all the features of data lakes and warehouses into a single data management solution. Although data warehouses can be more expensive and have fewer scalability options compared to data lakes, they are typically more efficient than the data lake. In a data lakehouse, data teams can manage and transform data more quickly. Teams can perform and scale lakehouses to perform advanced analyses like machine learning without having to switch to two different data systems. This is made possible by consolidating these advantages under a single data architecture.

Data lakehouses use data structures from data warehouses and combine them with the adaptability and low cost storage of data lakes, enabling enterprises to store and access big data more rapidly and effectively while also giving them the opportunity to avoid data quality issues. This satisfies the requirements of both business intelligence and data science workstreams by supporting a variety of data datasets, including structured and unstructured data. Typically, they support the powerful SQL, R, and Python programming languages. Data lakehouses also support ACID transactions. Atomicity is the state in which all modifications to data are made as though they are a single action. When data is constant from the beginning to the end of a transaction, it is said to be consistent. Isolation describes a transaction's intermediate state as being hidden from other transactions. Durability is the capacity to maintain data changes even in the event of a system failure after a transaction has successfully completed. As several users read and write data simultaneously, this capability is essential for ensuring data consistency.

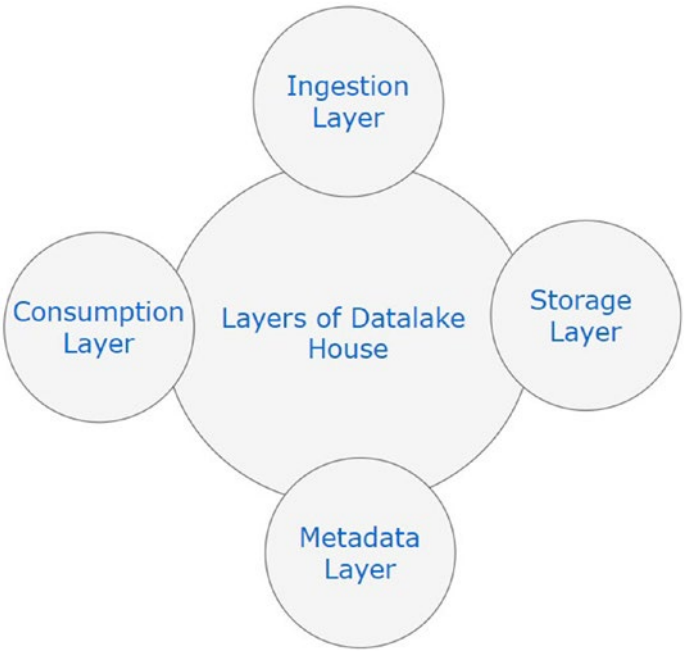

Figure 4-7. *Layers of a data lakehouse*

As shown in Figure 4-7, a data lakehouse consists of these four layers:

- *Ingestion layer*: This layer integrates data from various sources using the orchestrator to bring together all data to the cloud.
- *Storage layer*: Structured, unstructured, and semi-structured data is stored in the storage layer using open-source file formats like Parquet or Optimized Row Columnar (ORC). The system's capacity to handle all data kinds at a reasonable price is a lakehouse's actual utility.

- *Metadata Layer*: The data lakehouse has a metadata layer that is a unified catalogue to provide metadata for each object in the lake storage. Additionally, this layer provides the user with access to management tools like ACID transactions, file caching, and indexing for faster query execution. Within this layer, users can apply preset schemas to allow data governance and auditing features.
- *Consumption layer*: All the data and metadata stored in the data lakehouse will be made available to end users for downstream consumption in a secure manner via efficient authentication and authorization.

Data Pipeline Orchestration

Big data solutions consist of various processes to ingest, integrate, and process data for end users. The process that automates different processes of a data lifecycle is known as an *orchestrator*. An orchestrator can plan tasks, carry out workflows, and manage task dependencies.

As shown in Figure 4-8, a data pipeline orchestrator automates the overall flow of the data lifecycle, starting from collecting data and its processing followed by consuming the data and then using it for machine learning and AI adoption. There are many challenges of handling those steps manually if there is no orchestrator. For example, while doing the manual data entry, a human might enter the wrong data. Manually handling the data lifecycle has many issues, as explained here:

- Possibility of human errors
- Complex and unmanaged data workflows
- Lack of data management
- Potential data security breaches

Manual handling of the data creates data of poor quality and leads to poor business judgments. Handling data workflows via this automated process collects good data and brings value out of it. See Figure 4-8.

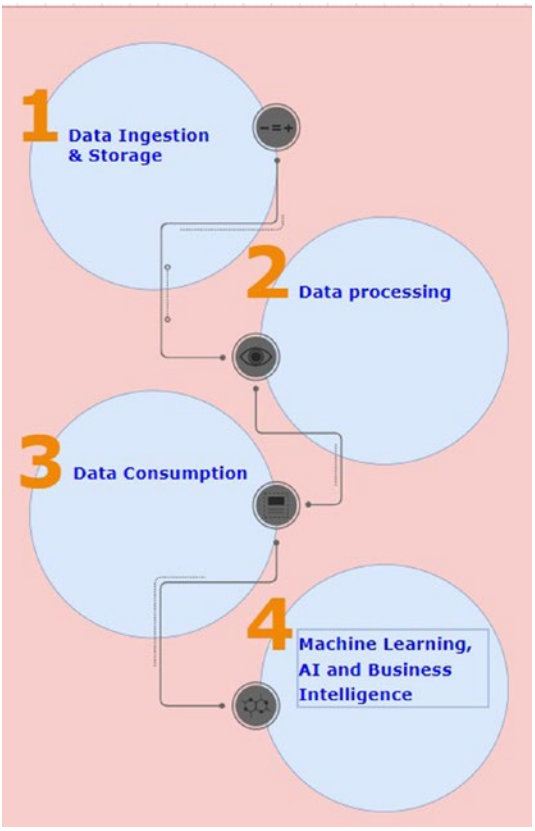

Figure 4-8. *Data pipeline orchestration*

Enabling an automated workflow using the orchestration system has the following benefits:

- *Reduction of manual errors:* Manual intervention of data processing has been a frequent problem when the data workflow is not fully automated. Such mistakes are avoided by programming the entire data pipeline procedure. You can use Azure Data Factory or Airflow to orchestrate the overall data lifecycle.
- *Faster and more efficient decision making*: Data engineers and analysts can focus more on building the actual data capabilities to generate more useful insights and automate each stage, which speeds up the decision-making process. So, using an orchestrator for data flow can avoid latency in using the data and its decision making.
- *Better data availability and improved data quality:* Data orchestration prevents data silos and increases the availability and accessibility of data for analytics purposes. Companies can trust the quality of their data since each step in the data orchestration process runs smoothly and detects mistakes at each level.

In order to choose an overall data orchestrator, consider the following criteria:

- should help move, transform, and manipulate data as it travels from its point of origin to its final destination and they should do this efficiently and faster.

- Orchestrators should be able to handle the data's structure such as its schema, statistics, and data quality metrics. This will help audit the platform and detect any mistakes across the data workflow. Storage and information about the metadata is crucial to managing the data workflow.
- In order to manage the workflow across multiple environments, a data orchestrator tool should be able to easily write, test, and deploy, taking into account the DevOps tooling layer.

When you look at the options of using an orchestrator in Azure, you have the following options (see Table 4-2):

- *Azure Data Factory*

  You can connect to a wide variety of data sources supported by Azure Data Factory by creating a *linked service*. This makes it possible to ingest data from a data source and prepare it for analysis or transformation. Linked services can also launch computational services immediately. For instance, you can launch a Hadoop cluster on-demand to process data using a Hive query. As a result, linked services enable you to specify the data sources needed to ingest and prepare data. They also gather data through a datasets object once the associated service is specified. Datasets represent the data structures found in the data store that the linked service object is referring to. You can use Azure Data Factory as an orchestrator in various scenarios. For example, to start or stop on-demand services like Hadoop clusters. When a cluster is needed, it may

be built using ADF, and it can also be shut down once your task is done. If so, you can manage how the Azure resources are used to process the data and only pay for what you utilize. It can be used to feed data into machine learning models. This is very helpful when you want to automate the process of feeding data to the machine learning model so it can conduct its analyses after a model has been productionized. This task can be accomplished easily via ADF.

- *Azure SSIS*

  Although conventional data warehouses have been using r ETL solutions like SQL Server Integration Services (SSIS) in order to use data in the cloud, it requires the scalability that the cloud can provide due to the rising number of data sources and processing requirements. Azure Data Factory has an integration runtime for SSIS. Using an Azure-SSIS integration runtime, you can upload, run, and keep track of packages through Azure Data Factory, like how you do it using tools like SQL Server Management Studio (SSMS).

- *Apache Oozie on Azure HDInsight*

  Apache Oozie is a workflow scheduler that manages the Apache Hadoop jobs. You can use Apache Oozie on Azure HD Insights to schedule Hadoop jobs hosted on cloud or on-prem. Directed acyclic graphs (DAGs) of activities make up Oozie workflow jobs. Oozie coordinator jobs are recurring Oozie workflow tasks that are started based on the

availability of data and time. Oozie is integrated with the rest of the Hadoop stack and supports both system-specific jobs (like Java programs and shell scripts) and streaming map-reduce, Pig, Hive, Sqoop, and Distcp.

Table 4-2. *Choice of Data Orchestrator*

Scenario	Azure Service
Cloud based	Azure Data Factory and Oozie on HDInsight
Copy data and custom transformations	Azure Data Factory, Oozie in HDInsight, and SSIS
Spark processing	Azure Data Factory
Access on-prem datasets	Azure Data Factory, SSIS
Azure machine learning	Azure Data Factory
Scalability	Azure Data Factory

As you can see in Table 4-2, you can choose orchestrators from the available Azure services based on various scenarios.

Real-Time Stream Processing in Azure

Stream processing is a very important component of the big data technology implementation that processes a lot of data. There are various fully managed frameworks available today that can be used to set up a cloud-based end-to-end streaming data pipeline. It can be difficult to make sense of the pertinent terminology so that you can choose an appropriate framework.

Real-time analytics, event processing, and streaming analytics are all closely related to stream processing. These use cases are currently all implemented using stream processing as the main basis. Runtime libraries called *stream processing engines* enable programmers to process streaming data without having to deal with the underlying streaming mechanics.

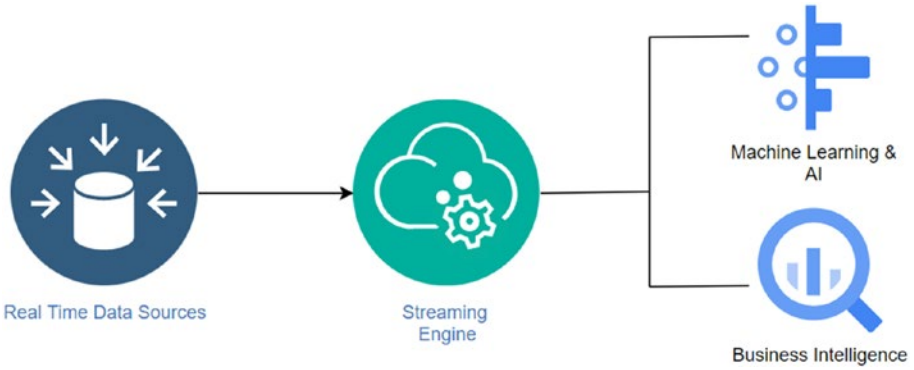

Figure 4-9. *Real-time stream processing*

As shown in Figure 4-9, streaming data architectures are created to handle and manage real-time stream data from diverse sources. They have many advantages over batch processing because they enable quick responses to new data, resulting in insights that are more precise and timely. Infrastructure for real-time streaming data typically consists of the following elements:

- *Stream sources*: Real-time data can be streamed from various sources, including databases, mobile, web apps, and other sources.
- *Streaming ingestion*: When there needs to be no delay between data input and query, streaming ingestion is helpful for loading data. Streaming ingestion is useful when there is a need to use the recent data immediately.

- *Streaming storage and processing*: Taking immediate action on the event streaming data is known as *stream processing*. Data professionals have used the term "real-time processing" to refer to data that is processed as frequently as required for a specific use case. However, stream processing is now used in a more precise sense because of the development and widespread acceptance of stream processing tools and frameworks. Multiple jobs can be carried out in parallel or both as part of stream processing, which frequently involves numerous activities on the incoming series of data. A stream processing pipeline consists of the creation of stream data, its processing, and its transmission to a destination.

- *Stream destination*: Depending on the use case, the destination layer in the streaming destination resembles a destination that has been specifically created. You can choose an event-driven application, data lake, data warehouse, or database for the streaming destination.

  In order to implement real-time streaming in Azure, you have various choices available.

- *Azure Event Hub*: Big data streaming platforms and event ingestion services use Azure Event Hub. It is used to gather, process, and analyze in a streaming manner, quickly without any delays. You can process this data and store it in storage services like Storage Account and Azure SQL Database. Companies enable automated pipelines to process, ingest, and store the data to the data lake, where the unprocessed data will be copied to a table using an orchestrator like Azure Data Factory.

- *Kafka*: In the Azure Event Hub, there is an option to use an Apache Kafka endpoint, which enables end users to connect to the Event Hub using the Kafka protocol. Conceptually, Kafka and the Event Hub are the same. Kafka supports a public-subscribe mechanism, which enables data producers and consumers to have async communication.
- *Azure Stream Analytics*: With Azure Stream Analytics, you can continuously run queries on an infinite data stream, which is processed in the form of events. The results of these queries are written to sinks like storage, databases, and visualization reports in Power BI. They consume streams of data from the message brokers, and then filter and aggregate the data depending on temporal windows. The query language used by Stream Analytics is SQL-based, provides temporal and geographic components, and may be expanded using JavaScript.

  The stream processing and message ingestion components handle the majority of the processing orchestration in a real-time streaming solution.

Conclusion

This chapter explored in detail the various data patterns and trends. You also learned about the various scenarios and how to choose an analytical store to perform data analysis on top of a data lake and data lakehouse. In the last section of this chapter, you also read about the importance of orchestration to manage the overall data flow and learned how to choose Azure services and implement real-time streaming patterns.

The next chapter inspects how to manage and govern data lakes for ad hoc queries from a data governance point of view.

CHAPTER 5

Data Architecture Process

In today's data-driven world, data is the new oil. Companies need good data to stay ahead of their competitors. In order to manage data securely, data architecture processes must be defined. Because data often contains personal identifiable information, it should be stored securely and the platform should be compliant with external regulators and audit requirements.

This chapter covers the following topics:

- Guide to data modeling
- Data lake for ad hoc queries
- Enterprise data governance: data scrambling, obfuscation, DataOps
- Master data management and storage optimization
- Data encryption patterns

Guide to Data Modeling

Before starting with the actual technical implementation (such as setting up databases and warehouses), many stakeholders, including developers, data architects, and business analysts, must agree on the data they will

© Sagar Lad 2023
S. Lad, *Modern Data Architecture on Azure*, https://doi.org/10.1007/978-1-4842-9760-5_5

capture and how they intend to use it. In order to set up a data foundation, you must start with a data model in order to understand the data and determine the approach for implementation.

A data model, like a house's architecture, specifies what to build and how it looks before construction has started, when it is easier to make changes. Building the data model enables you to avoid mistakes in database design and development, to avoid collecting pointless data, and to avoid data duplication across various sites. This involves utilizing words and symbols to describe the data and explain how it flows in order to create a simplified picture of a software system and the data pieces it includes. Data models offer a blueprint for creating a new database or re-engineering an old application.

Information systems and the databases they rely on are designed considering various aspects, including IT, business analysts, management, and others. Most of them can be validated by developing data models before doing the actual implementation. Data models provide a clear understanding of the data that is needed and how it will be structured to support the intended business activities. These systems require correctly defined and prepared data. These models serve a range of use cases, including database modeling, information system design, and process development. Explicitly defining the structure of your data ensures a consistent, clean exchange of data.

As shown in Figure 5-1, the enterprise typically goes through stages of data modeling, starting from the conceptual model followed by the logical and physical models.

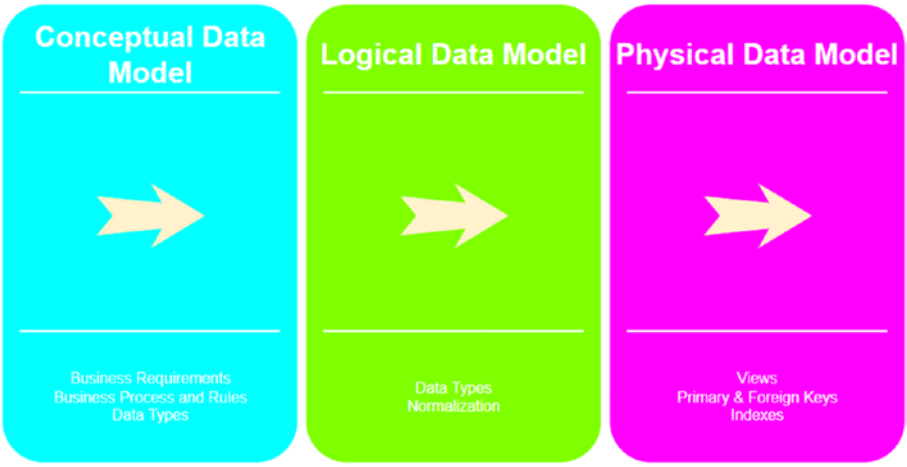

Figure 5-1. *Types of data models*

Conceptual Data Model

The conceptual data model provides a well-organized and simplified business view of the data required to support business processes, document business events, and monitor related performance indicators. With the conceptual data model, it will help to identify the data that is used in the business, but not how that data is processed or how it will be implemented technically. The viewpoint of this model is unaffected by any underlying business applications.

Conceptual data models use a standardized set of symbols to create a simple language model that defines the data being modeled. This straightforward visual language is useful for conveying the business users' perspective on the data they use. Entity relationship diagrams (ER-diagrams) contain entities, characteristics, and relationships and are the core modeling components from which the conceptual data model's symbol system is derived.

The conceptual data model is used by IT and business to specify the following:

- Scope of the required data
- Business KPIs used by various business units and agreed on across the entire organization
- Entity and attribute names, data types, and characteristics

Logical Data Model

The structure of the overall data and the connections between different data sources are established by the logical data model. The information describing how the data will be implemented is separate from the actual physical and technical implementation. The logical data model acts as a design guide for the technical data implementation that is going to be developed. By incorporating more details, the logical data model expands upon the fundamental concepts of the conceptual data model. For any application or program, the most important component is data, hence the quality of the data processing and data storage systems must be constructed on a solid and precise data structure. No matter how sophisticated or technically advanced the system is, it must adhere to the regulations and meet organizational objectives and needs in order to be useful. Logical data modeling combines the two most important principles of application development:

- Quality of the data
- Business objectives and requirements

Physical Data Model

A physical data model is a technical, database-specific model that shows the relational data objects and their associations, such as tables, columns, primary keys, and foreign keys. Data Definition language (DDL) statements can be generated from the physical data model and then can be used in the actual database to physically implement it.

There are various approaches to establishing a physical data model:

- The top-down approach creates a physical data model from scratch using tools like Erwin
- Create a physical model from an existing template
- The bottom-approach creates a physical data model from the existing database using the DDL statements
- Export a physical data model file
- Create a physical model from the logical data model

After understanding the importance and concepts of data modeling, it's important to understand the best practices for implementing data modeling, since it is critical to business growth and the maturity of the organization and allows it to collect insights based on the data and therefore gain an edge over the market competition.

Focus on Business Objectives and its Requirements

The main goal of the data modeling expert is to accurately capture business needs and determine which data to prioritize, capture, and store, so it's easily accessible to the end users. Data modeling activities need to comprehend the users' needs by interviewing stakeholders and users about the results they need from the data. It is preferable to start with the datasets that are well-organized with stakeholders and users in mind.

- *Visualize the data and information to be modeled before starting with the actual implementation:* You can clean up datasets using data visualization to make them more accurate and error-free. This also helps to find different categories of data records that relate to the actual entities, change those categories, and then use uncomplicated fields and formats to combine data sources with ease.
- *Refine dimensions, facts, filters, and orders from the business requirements:* Well-organized datasets make the business questions easier to answer. For example, if a retail business has locations across the globe then it is good to identify which of those locations have performed well over the past years. The facts in this case would be historical sales datasets, the dimensions would be product and store country, the filter would be the last 12 months, and the order would be the best five stores in declining order of sales.
- *Use relevant data instead of accessing all the data:* In today's digital world, data is generated at various frequencies in huge volumes. Instead of accessing all the data, which will consume lot of time and processing costs, try to data that is relevant to your needs.
- *Validate the data model in an incremental manner:* Once the data model is ready that doesn't mean that you can keep it as it is and just continue with the implementation. During every phase, the data model should be updated to incorporate new sources and correctly manage all existing relationships and constraints like primary and foreign keys.

- *Implement the data-driven model:* The data model provides a consistent view of the overall data used and how it is related. Once the data model is ready, you have to make sure that the physical and technical implementations are also implemented based on the data model to ensure consistency and integrity.
- *Standardize the data model:* While developing a data model, you have to make sure that all the data source and table names, including columns, data types, and so on, follow the same naming conventions. This enables end users to access consistent data, including data type and naming conventions.

In order to build data-centric solutions, data modeling is an essential step. It serves as the foundation for creating components for the data access layer, business layer, and service tier to efficiently manage, store, and distribute data. A solid data model must be built when designing data-centered enterprise applications in order to make it easier to enhance and improve performance. Small or enterprise-level organizations can anticipate that data modeling will significantly increase business value after all the criteria are met.

There are various benefits of implementing data models:

- Establishing communication between business and IT departments
- Identifying the need for data to optimize company operations
- Reduce the processing time and cost by properly planning IT and process investments
- Reduce errors while enhancing data integrity and consistency

- Set up efficient and faster data retrieval and analytics processes
- Identify and monitor target key performance indicators that are suited to the company's goals

You can use the following tools to create data models:

- Erwin Data Modeler
- ER Studio
- Archi
- Sparx
- IBM Infosphere DataStage

Data Lake for Ad Hoc Queries

Data is becoming more and more critical nowadays. In order to make better business decisions and be able to fully realize the value of the data, it must be readily available, accurate, and current. Stakeholders won't be able to completely trust reports and analyses unless data is properly managed and governed.

Using ad hoc query and analysis processes, everyone in the company can gain access to critical organizational insights so they can quickly respond to business queries and make proactive, well-informed decisions. Not only technical team members but also business stakeholders should be able to perform this ad hoc analysis.

As you can see from Figure 5-2, in order to set up an ad hoc analysis process, the data should be prepared and made available to the data lake.

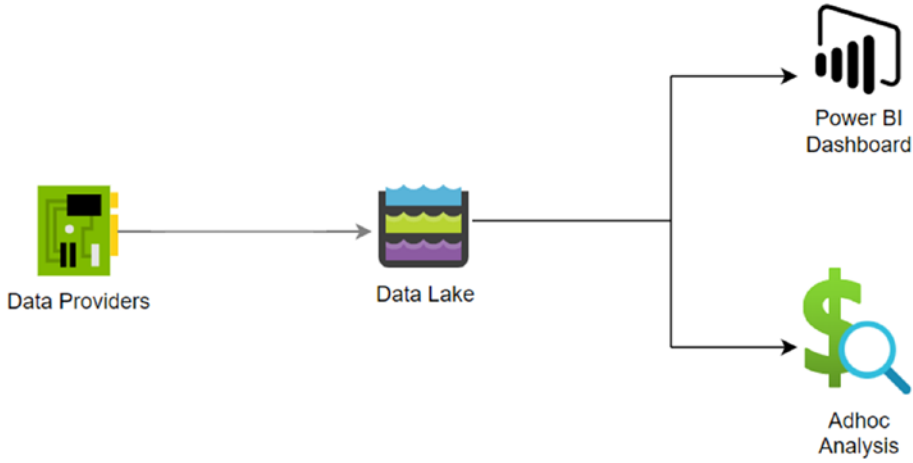

Figure 5-2. *Ad hoc analysis: data lake*

From the data lake, the data can then be distributed among the different data consumers based on the needs and agreed upon pattern. One of the powerful use cases to implement the ad hoc analysis is to use Databricks SQL, which provides various features and a range of technical skills based on the end user experience. For example, business data analysts may be very comfortable with the SQL-like language, which can be enabled using Databricks SQL. Data engineers might be much comfortable doing ad hoc analysis based on Python, so they can use Python or Scala to perform ad hoc analysis on the databricks delta tables.

Using Databricks SQL for ad hoc data analyses has the following features:

- *Ease of use and process data workflows from anywhere:* No matter where your data is, the ultimate goal is to use it as much as possible to bring value to the data. SQL-like user interfaces and features make it simple for the business analysts and engineers to import data into enterprise apps or perform ad hoc analysis from the cloud storage or data lakehouse. It also offers best-

in-class performance, and you can use your preferred tools like dbt on Databricks SQL or the built-in ETL capabilities on the lakehouse to manage dependencies and convert data in-place.

- *Speed up analysis and business intelligence:* Databricks SQL is compatible with and able to easily integrate into most business intelligence tools, including Tableau, Power BI, and Qlik, and so on. You can easily analyze and create visualization reports using the most recent and comprehensive data available. Data and business analysts can utilize their preferred techniques to find their business insights. Every analyst has the freedom and capability to query, discover, and share insights through collaboration with Databricks SQL in a self-service manner, without doing any additional development. As shown in Figure 5-3, Databricks SQL offers various features to speed up ad hoc analysis.

Figure 5-3. *Databricks SQL features to speed up ad hoc analysis*

- *Single source of truth for ad hoc data analysis and ease of governance:* Using the databricks delta table on top of data stored in the Azure data lake storage enables companies to have a single copy of all the data without moving data or duplicating it. You can easily manage and govern the data stored in the data lake. This will also ensure that data is easily discoverable and more secure. As you can see in Figure 5-4, the Unity catalog enables you to control and manage permissions at catalog, schema, table, and view levels.

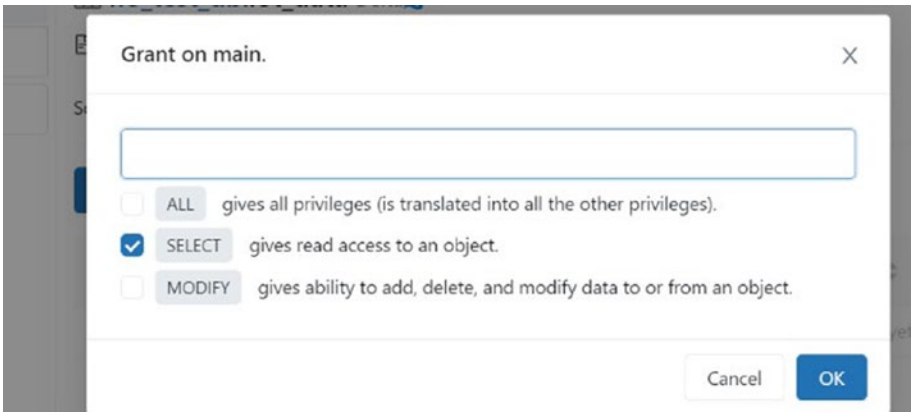

Figure 5-4. *Permission management with Unity catalog*

Without Unity catalog, you have to manage role-based and access control mechanisms for each workspace centrally. But as shown Figure 5-5, by using the Unity catalog, you can centrally manage and govern user management and access control across multiple workspaces.

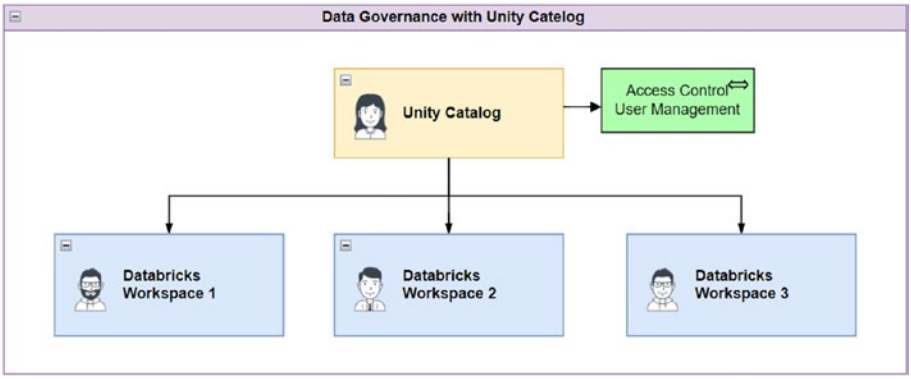

Figure 5-5. *Data governance with the Unity catalog and a data lake*

In order to perform ad hoc analysis, you can also create materialized views in Databricks SQL. It stores the query output physically, which contains precomputed data that is updated on demand. This precomputation of the data enables you to execute queries faster, with low response times and enhanced performance. Materialized views should be used when the data doesn't change often. Databricks SQL's Unity catalog is used to manage these materialized views. The materialized views can be refreshed manually or automatically at a scheduled frequency.

Data processing pipelines heavily use the SQL materialized views. Using the following syntax, you can create a materialized view in databricks SQL.

```
CREATE MATERIALIZED VIEW payments_orders AS
SELECT c.name, sum(p.amount),p.transaction_date
FROM payments p
LEFT JOIN company c ON
p.custkey = c.c_custkey
GROUP BY
p.name,
p.orderdate;
```

It is possible to configure automatic refresh for a materialized view based on a predetermined schedule in Databricks SQL. It is also possible to change the refresh frequency schedule using the ALTER VIEW statement. During the materialized view creation process, you can also set up a refresh schedule using the SCHEDULE clause. A databricks task is immediately created to handle the updates after a schedule has been defined. If you want to check if the materialized view has been configured with the refresh frequency, you can execute the DESCRIBE EXTENDED command. With this command, you can analyze the schedule in order to control and monitor the automated refresh schedule for the Databricks SQL materialized view.

Enterprise Data Governance: Data Scrambling, Obfuscation, and DataOps

Data governance is defined as a set of processes, responsibilities, and standards that create a reliable and efficient use of data. Additionally, it helps you set the foundation for the data management procedures to maintain the accuracy, security, and privacy of data across its entire data lifecycle. Companies use data to promote organizational growth, enhance decision-making, and guarantee outcomes to comply with the data governance policy. Data governance helps companies protect against misuse of the data and ensures the reliability and consistency of the data. Data governance is becoming more and more important since the volume of data is increasing and businesses are becoming more reliant on their data to make decisions.

In order to start any data governance program, you should have a steering committee that will drive the complete program, a governance team, and a group of data stewards. They collaborate to develop the guidelines that govern the data, as well as the methods for implementing and enforcing that are generally handled by the data stewards. In addition

to the IT and data management teams, executives from business operations should also participate. As you can see in Figure 5-6, the pillars of data governance are data processes, data control, data policies, and consistent structure of the data. Data inconsistencies in various systems throughout an organization might not be handled when data governance is not implemented properly.

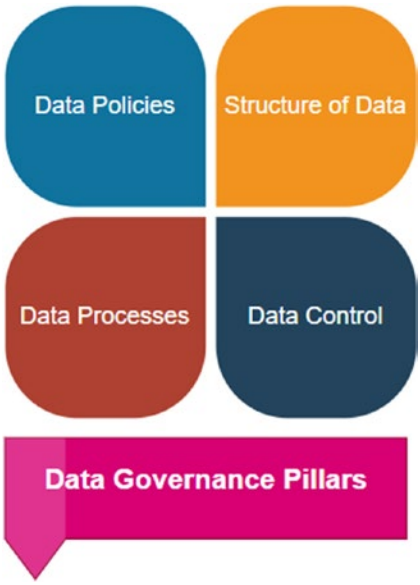

Figure 5-6. *Pillars of data governance*

As a result, data integration projects might become more challenging and problems with data integrity may occur that will impair the accuracy of business intelligence, reporting, and analytics systems. Additionally, data inaccuracies might not be found and corrected, which would reduce the accuracy of BI and analytics.

You can manage on-premises, multi-cloud, and software-as-a-service (SaaS) data with the combination of Microsoft Azure Purview's solutions using the governance portal. Azure Purview offers automated data

discovery, sensitive data classification, and end-to-end data lineage. You can also create a comprehensive, up-to-date map of the data platform and environment.

Azure Purview enables companies to store, manage, and safeguard data for the whole enterprise. Using Purview, you can classify related data assets, manage and govern access to files, identify and label sensitive information, and unify data assets. Purview provides a dependable data governance solution that is both straightforward and user-friendly and pricing is pay as per use. Azure Purview's major goal is to provide the tools to manage data while giving customers access to reliable and relevant information. There are four main key pillars of data governance supported by Microsoft Purview:

- Automatic detection, classification, and identification of data lineage
- Overview of data resources and interrelation between the resources for better data governance
- Common glossary of business and technical terms
- Information about sensitive data and how it moves throughout the data estate

The major goals of implementing a data governance initiative is to achieve the following:

- Data is proactively managed
- Businesses are responsible for data governance efforts
- Data owner takes full responsibility of the critical data
- Business domains take care of maintaining the quality of the data and have related priorities to focus on the relevant data

- Software engineering and information systems supports and enables data owners and data stewards maintain the quality of the data
- Each individual in the organization is responsible and involved in the data governance activities

Data Masking Techniques

Data masking is a technique for generating or creating data from sensitive data that is architecturally comparable but isn't physically visible in its actual form. Software testing and machine learning datasets are a few use cases of masked data. When the original data is not required, a useful replacement is created in order to protect the real data. Many banking and financial sectors have strong data security measures in place to safeguard their production data, but these measures are significantly less strict when data is utilized for non-production purposes. In particular, when data is used by third parties outside of the organization's control, it can create serious security and compliance problems. The solution is to use data masking, which ensures that any time data is transmitted outside of production environments, it is disguised to guard against compromise.

To secure sensitive information or personal data, many different techniques change data into a different form. They are together referred to as *obfuscation*. Encryption, tokenization, and data masking are three of the used methods used to obscure data. They all function differently. Both encryption and tokenization are reversible since it is possible to deduce the original values from the obfuscated data. On the other hand, if data masking is carried out appropriately, it is irreversible. Figure 5-7 shows the overall process of data masking on a production dataset.

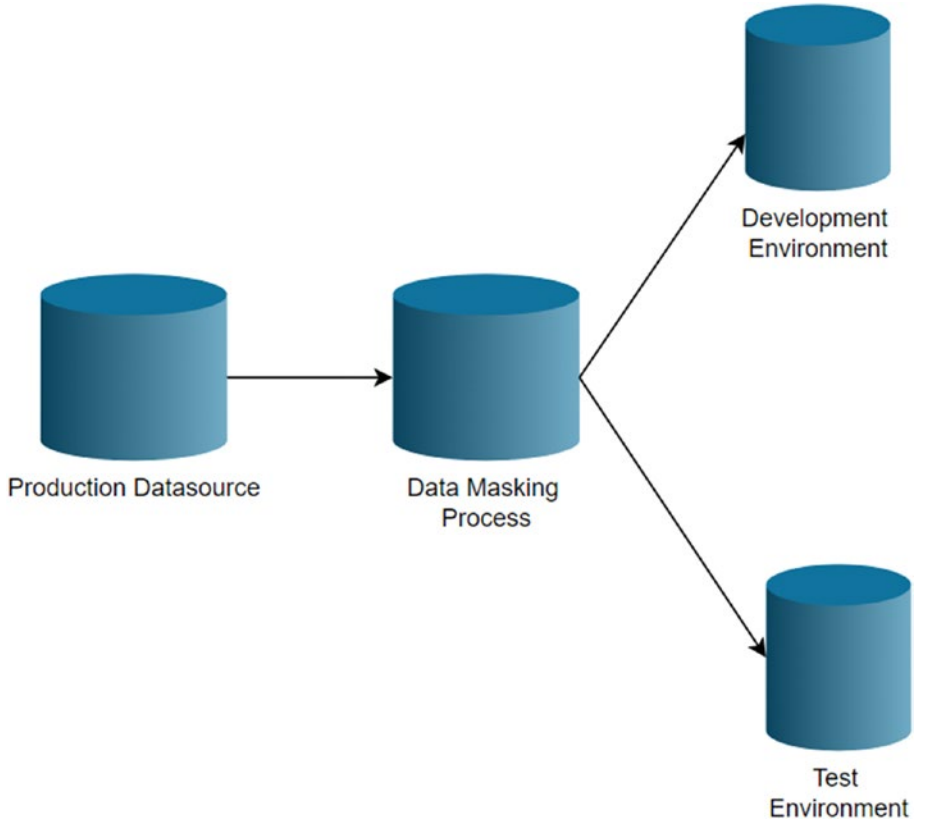

Figure 5-7. *Data masking process on production data*

Data masking is essential to data security since it reduces the effects of a data leakage and related concerns. Consider a database or a cloud storage that contains sensitive financial data or client data—not all sensitive data should be accessible without any specific purpose. Using the data masking process, the underlying facts of the data won't change, but data is prepared in a way that personally identified information (PII) won't be exposed to anyone. Nowadays, there are even requirements from the regulators like General Data Protection Regulation (GDPR). As per the GDPR rule, personally identifiable information (PII) must be protected using methods like anonymization and *pseudonymization* (replacing

data with comparable data that conceals a real person's identity). As you can see in Figure 5-8, there are several different types of data masking techniques.

Figure 5-8. *Different data masking techniques*

Data Scrambling

Hiding sensitive information or removing sensitive data is done through the process of data scrambling. Data scrambling is irreversible; scrambled data cannot be reconstructed. Data encryption is only possible during the cloning process.

Data Encryption

During the data encryption process, data is converted from one form to another form so that only those with a secret or decryption key or password can decipher it. Unencrypted data is referred to as plaintext, while encrypted data is frequently referred to as ciphertext. Data encryption is one of the most common and successful data security techniques used to safely store and use sensitive data. Asymmetric encryption, commonly referred to as public-key encryption, and symmetric encryption are the two basic methods of data encryption.

Data Ageing

Data ageing refers to the removal of data from outdated media. Data ageing first ages data in accordance with the storage policy copy's retention guidelines before deleting it in accordance with the associated medium's media recycling guidelines.

Data Substitution

The process of hiding data by changing it to another value is known as data substitution. This is one of the best data masking techniques for keeping the original appearance and feel of the data. In order to do the data substitution, you have to create substitution rules. Based on the substitution rules, data will be transformed from the original data.

Data Shuffling

Data is mixed up using shuffling techniques, which can also keep logical connections between columns. Shuffling randomly rearranges data from a dataset inside an attribute or collection of attributes. Sensitive data can be replaced with other values of the same attribute from a separate record by shuffling it.

These data shuffling methods are a strong fit for analytics use cases since they ensure that the KPI generated on the input dataset will still be accurate, so the underlying data value can be used without exposing it to end users or business users. As a result, production data may be used safely for activities like testing and training because the statistics distribution is maintained throughout. One of the major uses of this method is to create "test data" that resembles actual production data as inputs for a system.

Pseudonymization

The General Data Protection Regulation (GDPR) of the European Union encourages pseudonymization as a data management technique. When data is anonymized or pseudonymized, the details that reveal a subject's identity are cleaned out. This makes it impossible for the data to identify the user in particular. On the left side of Figure 5-9, you can see the original data of the customer, including the name and customer balance. Once the pseudonymization technique is implemented on the original data, the masked data doesn't contain any sensitive information about the customer, but the underlying data value will remain as it is. Figure 5-9 shows the data pseudonymization process with reference examples.

Customer Name	Customer Balance
Mike	30000
Jack	60000
Prince	20000
Gauti	25000

Original Data

Name	Account type
Person 17332	10
Person 12348	5
Person 74738	3
Person 78383	2

Masked Data

Figure 5-9. *Example of data pseudonymization*

In the data anonymization process, data becomes entirely anonymous. All personally identifying information is removed and once it is created it is not ideally irreversible. On the other hand in the pseudonymization process, it is possible to reverse the original data from the pseudonymization data.

Master Data Management and Storage Optimization

Master data is the fundamental information required to manage an organizational unit's operations. The context of business operations and transactions is provided by data about important business entities. Data that is considered master data can differ between and within sectors. Figure 5-10 shows various enterprise data types.

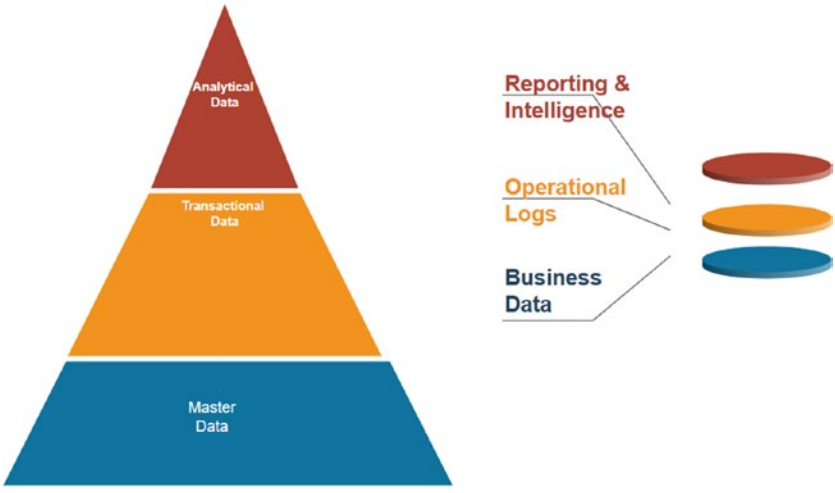

Figure 5-10. *Enterprise data types*

For any company or organization, there are three types of enterprise data captured for various purposes:

- *Transactional data*: Information that is gathered from transactions. It keeps track of the date and location of the transaction, the time the transaction took place, the price ranges of the purchased products, the mode of payment, and other quantities and characteristics related to the transaction.
- *Master data*: All the information that is essential for a company to run. This data is typically shared throughout the company and is used by multiple departments and employees to make decisions. Depending on the type of organization, different master data types exist but all of them have the following characteristics:
  - Complex
  - Non-transactional
  - Mission-critical
- *Analytical data*: This is statistical or analytic data and is stored in an analytical data store, commonly referred to as a data warehouse. It can be used to explore massive datasets and has reliable ways to analyze, extract, or otherwise deal with datasets that are too complicated to be handled by conventional data-processing tools.

Master Data Management

Now it's time to explore master data in more detail and look at the various design patterns that set up master data. Figure 5-11 shows a high-level overview of the master data management process in enterprise organizations.

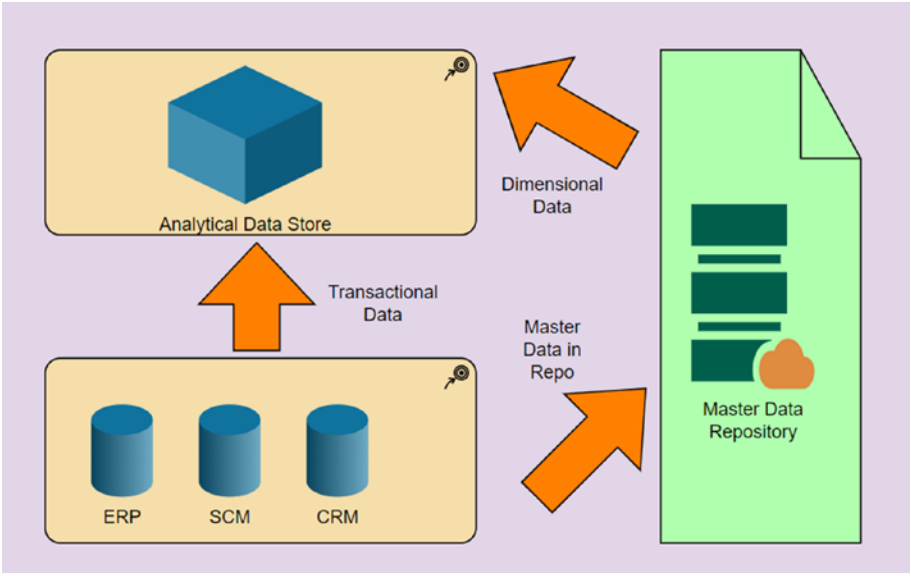

Figure 5-11. *Master data management design considerations and technology*

It is obvious that master data is essential for the captioning, processing, and comprehension of data in businesses. Various types of master data exist and it will change depending on the type of company and business. Master data is used in many applications and business processes throughout an organization. There can be many critical issues if this data is not carefully handled and this can have a significant business impact.

There are many different architecture models available for managing master data, because every firm is different and has its own challenges, IT environments, and business processes. Making the best pick based on certain trends and patterns won't work because each of these trends or patterns can have its own benefits and disadvantages. When selecting the correct type of master data management for your business, you need to take into account the available budget, IT landscape, the current

organizational structure, people and process involved, and their skill sets. There are three basic types of master data management architectures that should be considered:

- *Registry architecture*: This is best for a downstream application that only wants to read the data and not to modify it. This design offers a read-only perspective. Duplicate or redundant data should be removed and this implementation architecture offers a reliable access route to the master data. Attributes stored in the master data must guarantee its uniqueness and cross-reference data to the application system that consists of the full master data record, which are frequently not represented by the data in the master data management system. In this case, all master data attributes—apart from those retained in the master data management system—remain in the application systems with inadequate quality and lack of harmonization. As a result, with reference to all properties in the master data management system, the master data is neither consistent nor full. Figure 5-12 shows the registry architecture for master data management.

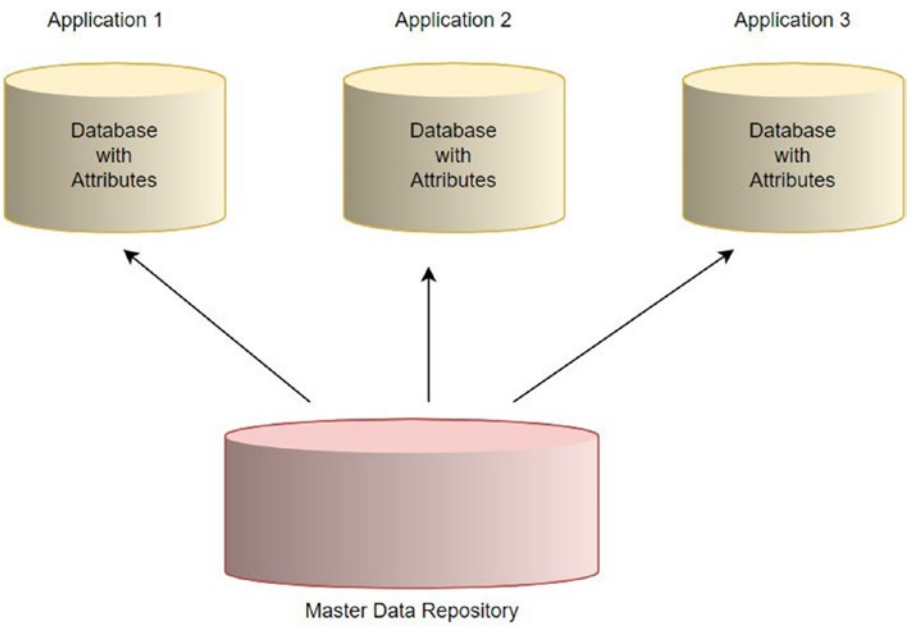

Figure 5-12. *Registry architecture for master data management*

- *Hybrid architecture*: All master data properties in the master data management system are fully materialized by this hybrid design. The master data management system and the application systems are both capable of authoring the master data. From a completeness standpoint, all characteristics are present. However, only convergent consistency is provided in terms of consistency. The synchronization of updates to master data in the application systems transmitted to the master data management system is delayed, which is the cause of this type of consistency. The more this implementation architecture leans toward absolute consistency, the smaller the window of propagation. The master data integration phase of this architecture

requires more money since all master data model attributes must be synchronized and cleaned before being put into the master data management system. Additionally, there is additional cost associated with synchronizing the application systems that alter the master data with master data management systems. However, this strategy offers a number of advantages over the registry architecture implementation:

- Increased quality of the master data
- Quick and rapid access to the data
- Less complexity and simpler to implement with collaborative authoring
- Centralized master data attributes and reporting

Figure 5-13 shows the hybrid architecture for master data management.

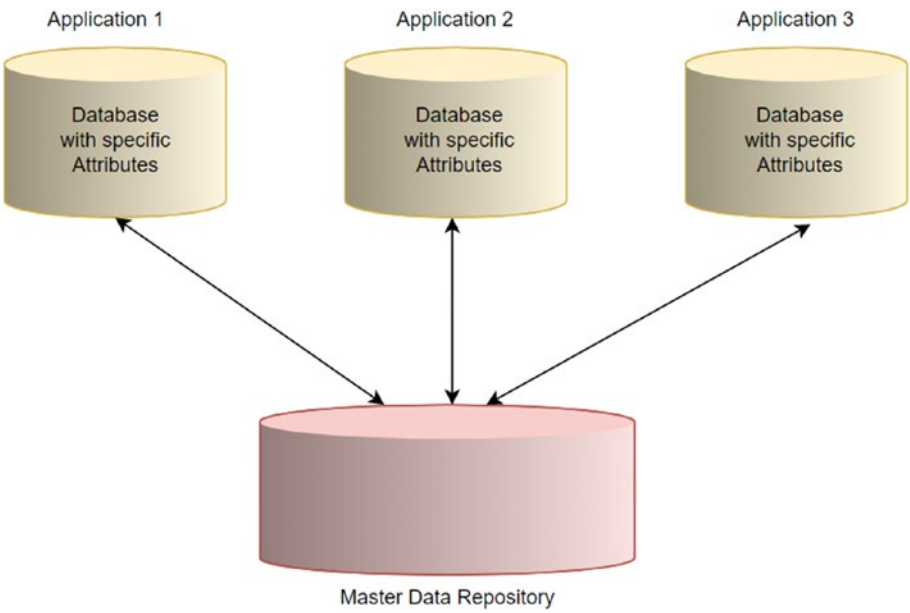

Figure 5-13. *Hybrid architecture for master data management*

- *Repository architecture*: In the repository architecture implementation, the master data is always consistent, accurate, and complete. The master data management system can handle read and write activities on the master data, which is a significant advantage over the hybrid architecture. In order to accomplish this benefit, any application that requires changes to the master data must use the master data management services provided by the master data management system.

  When using this architecture, deploying a master data management solution can necessitate extensive penetration into the application systems, intercepting business activities so that they engage the master data management system

for any changes in the master data. Figure 5-14 shows the repository architecture for master data management.

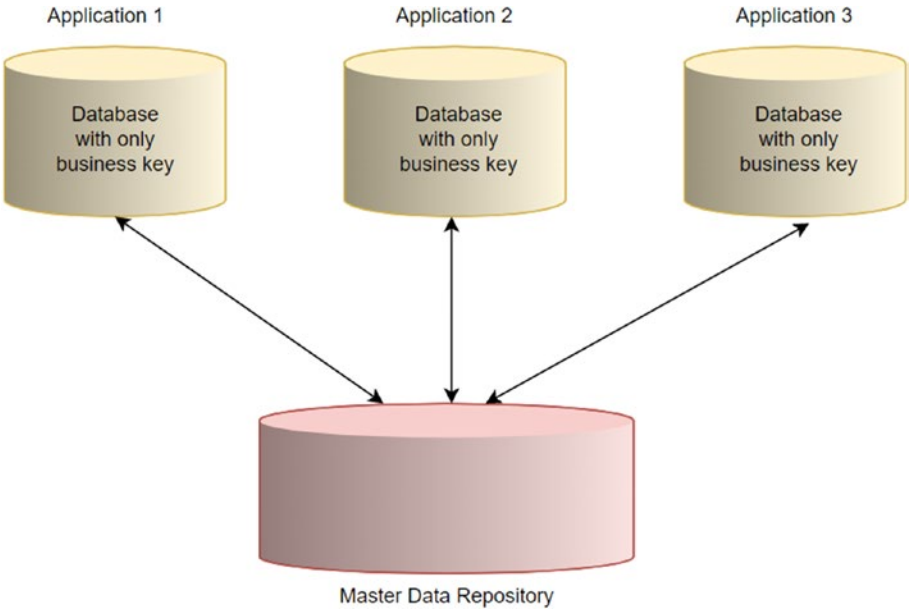

Figure 5-14. *Repository architecture for master data management*

Data Encryption Patterns

In the digital world of expanding and ever-changing technologies with increasing volumes of data, securing and protecting important data becomes very challenging and difficult. In the modern world, data encryption is essential since sensitive and personal information cannot be exchanged or transferred in plaintext to avoid any issues of data leakage and to meet various compliance and regulatory requirements. Simple text data is transformed into an encrypted text or intelligent data, by the process of encryption. Data is transformed into a ciphertext using the encryption process, making it difficult for any random user to read the data

in plaintext form. The integrity of highly sensitive information must be protected at all costs and this is the most important way to store and secure the data.

There are two types of encryption:

- Symmetric encryption
- Asymmetric encryption

Depending on the requirements and various other business-related factors, organizations adopt one of these encryption types. Larger companies utilize encryption to protect sensitive information transported from one user to another globally, ensuring robust encryption between the client and the server. Any data that is not meant to be shared with unauthorized users can be shared between users if the data encryption process is not in place. Different ciphers and encryption algorithms, both powerful and weak, have been developed and have a theoretical framework. With the appropriate tools and techniques, weak algorithms and ciphers can be cracked and further abused. Data encryption protects confidential information held by an individual or organizations by encrypting or encoding the data in a way that only the designated person with an access key can decrypt it. Whenever a user tries to access the data without the required rights, the request-response appears random or unmatched. Data encryption transforms extremely sensitive information into unreadable formats that is difficult for any arbitrary user to access but is essential to the person or organization that owns it. Documents, emails, communications, KYC information, and other types of data can all be encrypted to prevent unauthorized access.

To enable large scaling, modern access through REST APIs, and isolation across tenants, storage in a cloud service like Azure is designed and implemented fundamentally differently than systems that are installed physically. The cloud services vendors offer various options for access

control on storage resources. Shared keys, shared signatures, anonymous access, and identity provider-based techniques are a few examples of such encryptions. Azure Cloud Storage has various built-in encryption features:

- *Identity Based Access Control*: Using Azure Active Directory (Azure AD) and key-based authentication techniques like Shared Access Signature (SAS) or Symmetric Shared Key Authentication supports built-in encryption. Azure storage encrypts all of the data stored in the cloud. A tenant cannot read data that was not created by that tenant. Cross-tenant data leakage is controlled by this feature. Figure 5-15 shows how role-based access management should be implemented for different entities like admins or moderators.

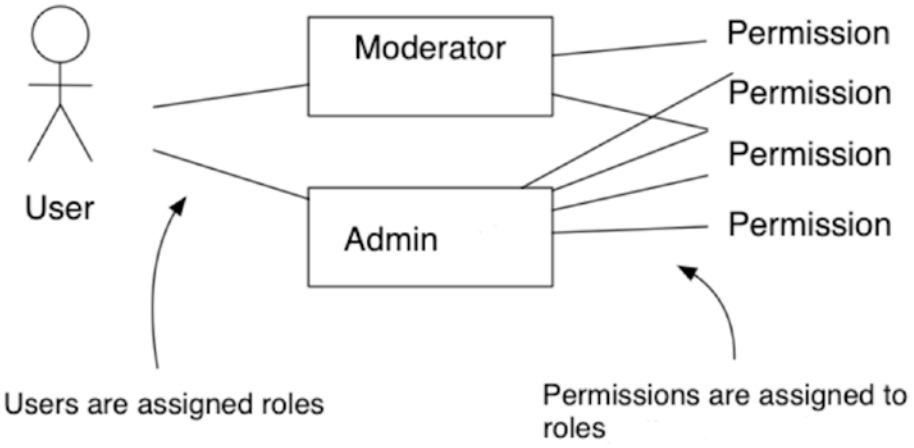

Figure 5-15. *Role-based access management*

- *Region-based access controls*: Data remains only in the selected region and three synchronous copies of data are maintained within that region. Azure storage

provides detailed activity logging that is available on an usage basis. Figure 5-16 shows how access control should be implemented on Azure storage.

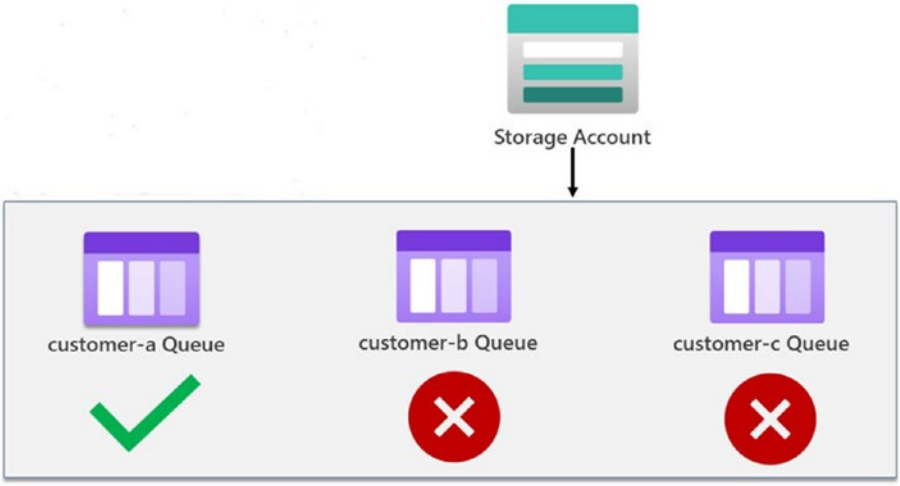

Figure 5-16. *Access control management for Azure storage*

- *Firewall features*: The firewall provides an additional layer of access control and storage threat protection to detect anomalous access and activities. Azure Firewall has the following features:
  - Threat intelligence based filtering
  - Deploys in quicker manner
  - Unified access and security management

Conclusion

This chapter explored in detail the data model and its significance with respect to data architecture. It also explored how to execute ad hoc queries on data lakes and enterprise data governance with data scrambling and DataOps functionalities. You also learn in detail about master data management and various data encryption patterns.

The next chapter explores the various data architecture frameworks and standards in the market, including DAMA-DMBOK and the Zachman Framework.

CHAPTER 6

Data Architecture Framework Explained

According to the Open Group Architecture Framework (TOGAF), a data architecture describes an organization's overall logical and physical design of its data entities and data management resources. The models, policies, rules, and standards that control the gathering, storing, arranging, integrating, and use of data are a subset of enterprise architecture. Data architects are responsible for an organization's data architecture.

This chapter covers the following topics:

- Fundamentals of data modeling
- The Open Group Architecture Framework
- The DAMA DMBOK
- The Zachman Framework

Fundamentals of Data Modeling

As you learned in the previous chapter, data modeling is a crucial activity for setting up the foundations and fundamentals of data-driven projects. In today's world, every industry and business relies on data to grow their business. Data is used for everything, including studies improving medical treatments, business revenue strategies, the development of efficient buildings and construction, and social media platforms.

© Sagar Lad 2023
S. Lad, *Modern Data Architecture on Azure*, https://doi.org/10.1007/978-1-4842-9760-5_6

Information that is machine-readable as opposed to human-readable is critical because it enables ingestion and storage of the data to create a better decision-making process. For instance, if consumer data cannot be synced to the individual product transactions, it is not very useful for the product team. The same data won't be useful to a marketing team if the unique IDs don't correspond to particular price points during the buying process. A data model simplifies data into valuable information that businesses can utilize for planning and decision-making.

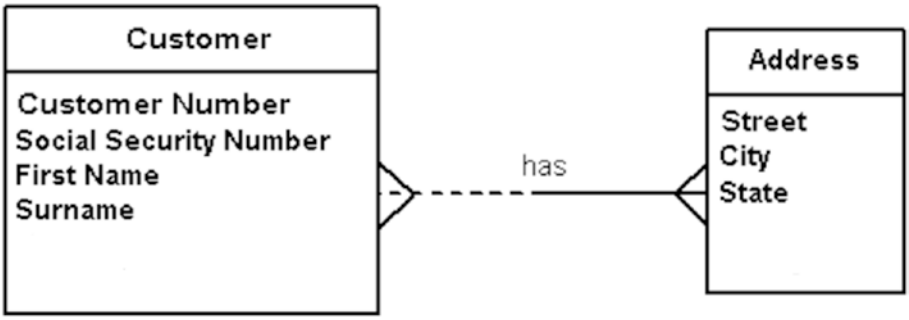

Figure 6-1. *Basic logical data model: customer information model*

As you can see in Figure 6-1, there are two entities: Customer and Address. Each entity has different attributes and there is a one-to-one relationship between the two entities. Each customer has address information associated with it.

Organizations can set benchmarks and targets using good data in order to keep moving forward. Stored data must be organized through the data description, data semantics, and data consistency requirements in order for this measuring to be possible. A data model is an abstract model that establishes links between data items and further develops conceptual models. As you can see in Figure 6-2, there are seven data modeling techniques.

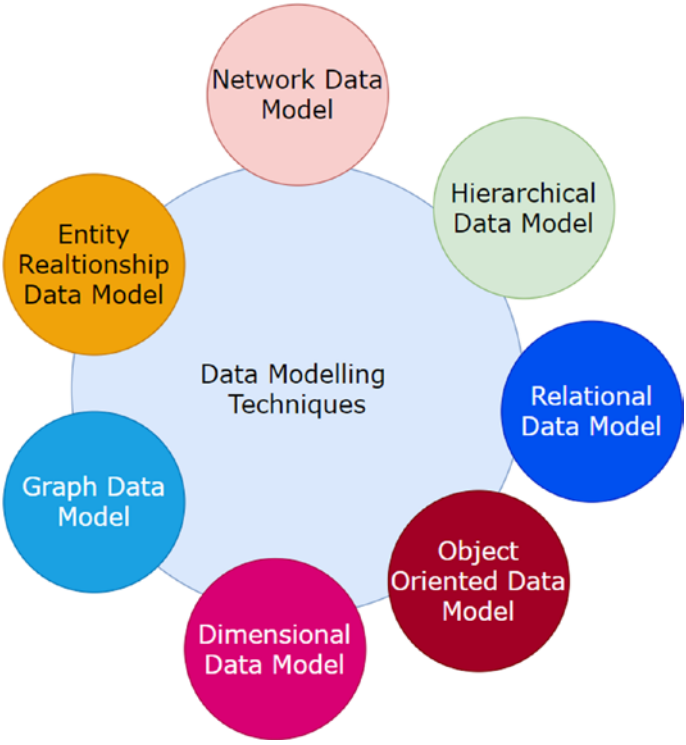

Figure 6-2. *Data modeling techniques*

The following sections explore each of these data modeling techniques in detail.

The Network Data Model

A network model consists of a number of records that are linked to each other. In many ways, a record and an entity in an E-R diagram are similar. Every record consists of a set of characteristics, each of which has a single data value. The link is the exact association between the two records. Thus, in the meaning of the E-R model, a link can be thought of as a binary kind of relationship. The Network Data model frequently uses

diagrams with boxes and arrows to depict data and relationships. Each box represents a certain record type and each arrow a particular kind of relationship. A record type will have fields with individual values intended to store information about the real-world thing that the record is meant to represent. Figure 6-3 shows a network data model for the store. Each store has clerks and customers. Each customer and clerk executes transactions to purchase or sell various items.

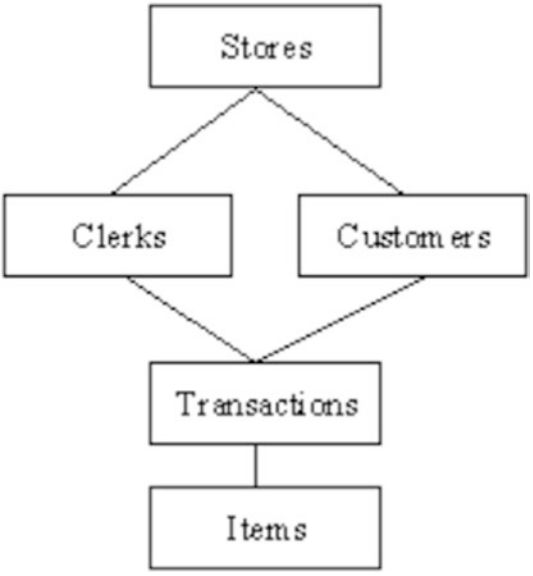

Figure 6-3. *Network Data model for a store*

The Hierarchical Data Model

The Hierarchical Data model shows data in the form of a tree-like structure. In this tree-like structure, each child only has a single parent. The Information Management System from IBM is one of the mainframe database management systems for which hierarchical data models were mainly created. This model's structure helps you create relationships

between different types of data—both one-to-one and one-to-many. Figure 6-4 shows a Hierarchical Data model where each child is related to only one parent.

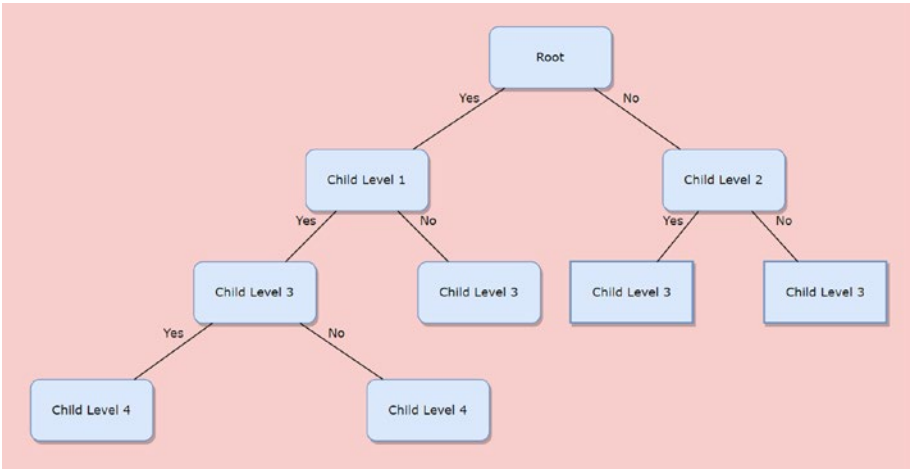

Figure 6-4. *Hierarchical Data model*

In the Hierarchical Data model, a child record can only have one parent, but a parent can have more than one child records.

The Relational Data Model

The Relational Data modeling activity is a technique to conceptually represent and manage data stored in a database for database management. The data in this model is arranged into a set of two-dimensional, mutually exclusive tables or relations. Each relation is made up of a set of columns and rows, where the columns correspond to an entity's properties and the rows, or tuples, correspond to its records. Figure 6-5 shows a relational data model that consists of two entities: teachers and classes. They are related to each other based on the common key, called teacherId.

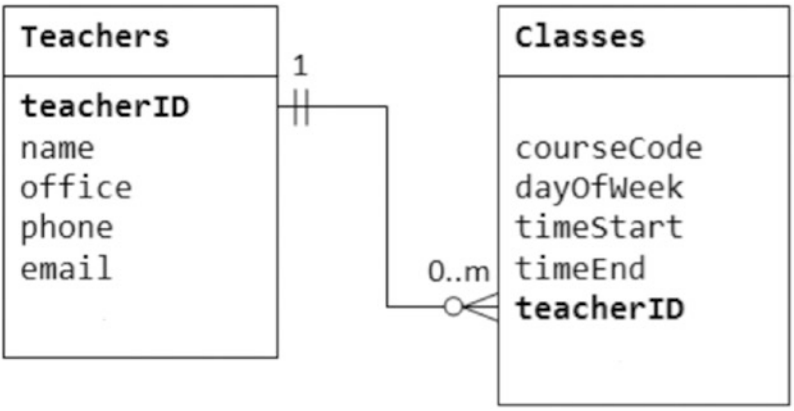

Figure 6-5. *Relational Data model*

The Object-Oriented Data Model

The data model in which data is stored as objects is known as the object-oriented model in database management systems. Real-world entities are represented using the Object-Oriented Data model approach. In the Object-Oriented Model, both the data and the data relationships are kept in a single unit, called an *object*. The object-oriented database management system is constructed on top of the object-oriented model. Figure 6-6 shows an example Object-Oriented Data model.

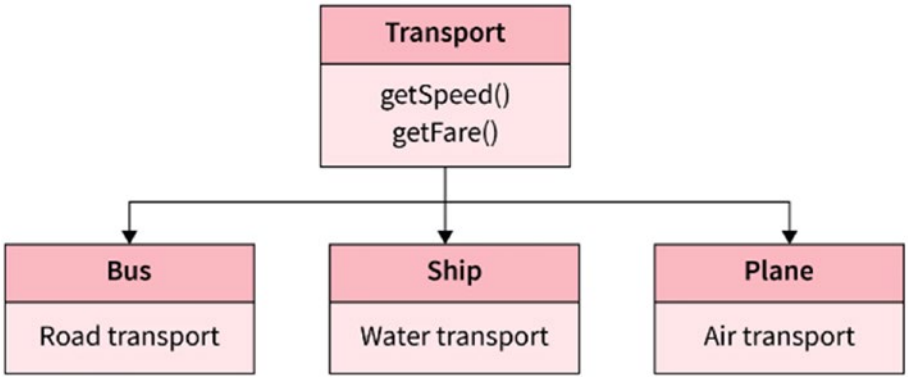

Figure 6-6. *Object-Oriented Data model*

The Dimensional Data Model

Dimensional modeling is designed specifically when you want to store the history of the data. Dimensional modeling is used to enhance databases for efficient data retrieval. The "fact" and "dimension" tables that make up the Dimensional Data model were created by Ralph Kimball. A dimensional model is a tool used in data warehouses to read and analyze numerical data such as values, balances, weights, and so on. Relational models, on the other hand, are designed for the write operations in the transactional systems. Figure 6-7 shows a Dimensional Data example where one fact table is surrounded by various dimensional tables.

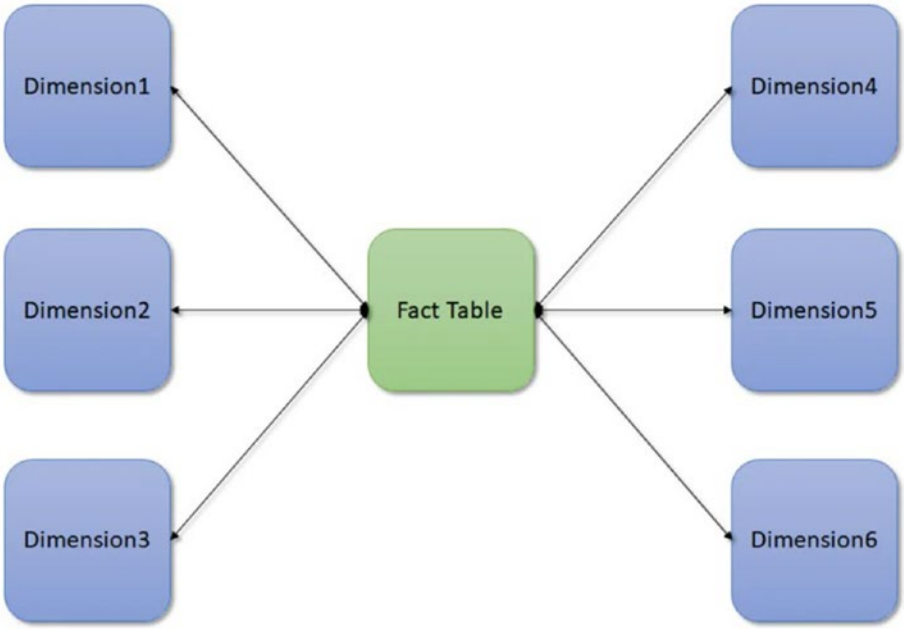

Figure 6-7. *Dimensional data model*

The Graph Data Model

The Graph Data model is also known as "whiteboard-friendly." Choosing whether items in the dataset should be nodes, links, or deleted is a step in the Graph Data modeling process. A blueprint of the entities, relationships, and properties in the data is the end result of the Graph Data modeling process. You can build a visualization model for charts using that blueprint. Although the process is laborious and frequently involves trial and error, it is worthwhile to execute it correctly. A single dataset can be modeled in a variety of ways, although some are more beneficial than others. Using the appropriate model greatly simplifies the life of your engineers and end users. Figure 6-8 shows an example of a Graph Data modeling process where Node A is linked to Node B. The relationship between the nodes is defined by the edge.

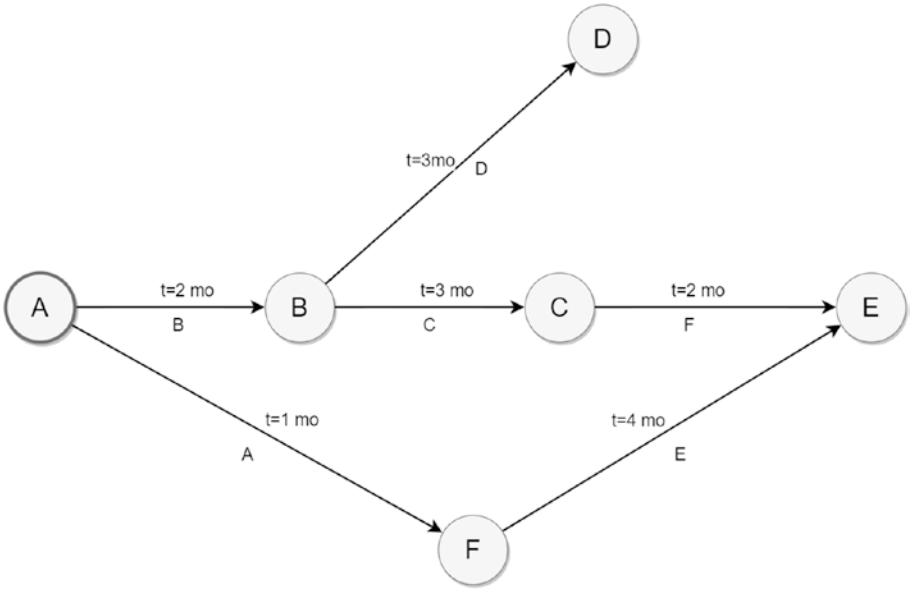

Figure 6-8. *Example of graph data modeling*

The Entity Relationship Data Model

An Entity Relationship Diagram (ER diagram) shows the relationships between entity sets stored in a database. In other words, ER diagrams are also helpful in describing the logical organization of databases. Entities, attributes, and relationships are the three fundamental ideas that serve as the foundation for ER diagrams. Rectangles represent entities in ER diagrams, ovals indicate attributes, and diamond shapes represent relationships. Figure 6-9 shows an entity relationship diagram with three entities: customers, orders, and shipments. Each entity has various attributes. Customers have one-to-many relationships with orders. Each order has many shipments, which have one-to-many relationships as well.

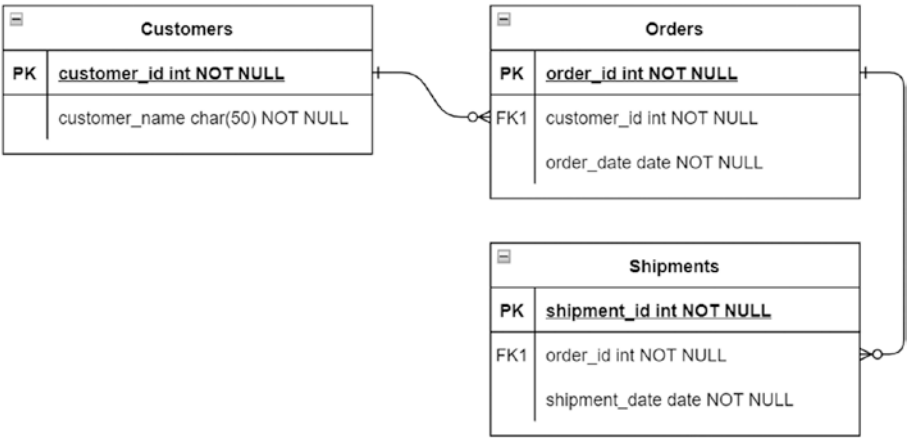

Figure 6-9. *Example of an entity relationship diagram*

In order to make a better decision on which data modeling technique to use when, consider the following points:

- Consider the business processes in order to identify all the components of the systems and entities related to the use case.
- Denormalize the tables as much as possible, which will enable the end users to scan and read faster, which can impact how well the data model performs, especially when working with several tables or integrating various tables.
- A data model should always be built around a business process. With this method, analysts will have enough time navigating the data model and finding the solutions to their problems.
- Each dataset can be inserted into the data models more properly with more information in order to analyze data at a detailed level.

The Open Group Architecture Framework

The Open Group Architecture Framework (TOGAF) is a high-level framework for enterprise software development. TOGAF is a methodology for enterprise architecture design and development. It helps reduce errors, manage deadlines, efficiently manage budgets, and integrate IT with business units to generate excellent output.

The main deliverables of a TOGAF implementation are as follows:

- Standardize the terminologies and glossaries so everyone in the organization and outside the organization uses the same terminologies.
- Using the common methodologies for the enterprise architecture, prevent vendor lock-in and dependency on the third-party software.
- Save money and time and make better use of the available resources.
- Gain a comprehensive overview of the organizational landscape to serve as a flexible, scalable foundation for organizational transformation.
- Use with all different sizes of businesses, regardless of the domain, to adhere to the common architecture standards.

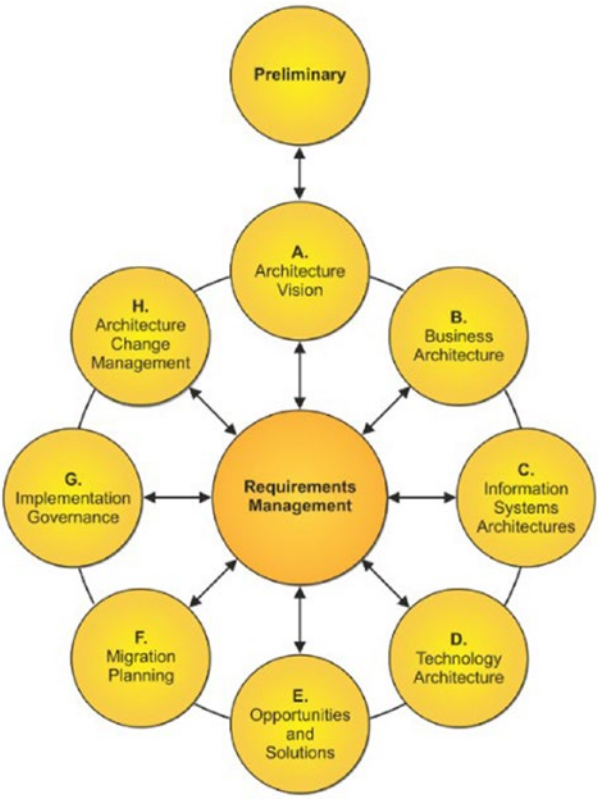

Figure 6-10. *Enterprise Architecture Development Method phases*

One of the major components of the TOGAF is the architecture development method, which streamlines the overall architecture development process.

As you can see in Figure 6-10, ADM has various phases. The TOGAF ADM is a methodical strategy where each phase lists the essential tasks and data inputs needed to gain the understanding needed to create an enterprise architecture. The foundation of the TOGAF standard is TOGAF ADM. The TOGAF ADM is used worldwide. It scales to provide enterprise architecture suitable for strategies, portfolios, and projects.

Preliminary Phase

The enterprise architecture team is developed during the preliminary phase. It focuses on the most important problems that the enterprise architecture team must solve. These consist of asking these questions:

- Who is the data intended for?
- What will the data be used for ?
- The application of the model
- Why do you need the data?

These actions help the stakeholder understand the notion. It also simplifies its use and facilitates application and execution. Once you finalize the initial stage of the preliminary phase, the following can be completed:

- Define the business
- Recognize and comprehend the key components and motivators of the organization
- Describe the requirements for the work of architects
- Determine the organization's most workable framework
- Explain the connections between the various management frameworks

Defining the Architecture Vision

Phase A of the architecture vision must be the starting point for all architecture development. An enterprise architect cannot be complete, even when they are working on the appropriate problem, with the proper constraints, with the right stakeholders, but without the setup of Phase A.

In this phase, the request for Architecture Work will be created. Confirming the architecture scope, stakeholders, stakeholder concerns, and constraints from superior architecture are crucial activities.

Business Architecture

During this phase of ADM, the current company structure, goals, and requirements of all the stakeholders are developed, which also creates the business architecture domain. Organizational design, enterprise process, information flows, business capabilities, and strategic business planning are the main focus areas of the business architecture. Here, you can implement a capability model to concentrate attention on the areas that need improvement within the application development process.

Information System Architecture

The construction of the information systems architecture is the step after the business architecture is finalized in Phase B and a deeper comprehension of the enterprise is needed. This mainly focuses on the data and applications and is quite comprehensive. The domains of data architecture and application architecture are also handled separately.

Technology Architecture

The next step following the information system architecture in the ADM phase is establishing the technology architecture to define, describe, and design the technology. It takes into account the following:

- Finding points of view, objectives, and models to use as a guide and tools
- Defining a baseline technology architecture definition

- Defining and developing the appropriate target technology architecture
- Performing gap analysis
- Defining the components of a roadmap
- Building resolution environmental effects
- Reviewing the stakeholder input
- Finalizing the technology architecture

Opportunities and Solutions

The enterprise architecture team spots possibilities and seeks out the best answers to current or past problems. The phase addresses important phases, changing parameters, and major projects. A company that assists a business in building the appropriate infrastructure will frequently use the TOGAF ADM cycle. By doing this, you guarantee that the model you produce includes all the essential components. During this phase, you create an architecture roadmap in TOGAF ADM Phase E.

Migration Planning

An effective migration strategy guarantees a smooth transfer. It outlines the mode of implementation according to priorities and includes all the necessary components. Some of the actions taken into consideration are cost savings, benefit analysis, and dependency evaluation. Phase F of migration planning implements the deliverables, implementation plan strategies, and tools.

Governance Implementation

Implementing the modifications in the architecture roadmap and implementation plan is the main goal of the Governance Implementation phase. Additionally, it incorporates useful data from additional implementation initiatives. A project can be successfully managed with less risk when it has good architecture governance.

Architecture Change Management

In the Architecture Change Management process, the target enterprise design is realized. Additionally, you ensure that the target architect is compatible with the organization's current environment and ecosystem. This details the main characteristics, the objectives, and how to do the exercise. The phase is ongoing and integrates with the Architecture Requirements Management process. TOGAF Phase H illustrates how enterprise agility is implemented in real projects under the direction of best-practice enterprise architects. The ability to respond to unanticipated opportunities and threats is at the core of enterprise agility.

DAMA DMBOK

The DAMA Guide to the Data Management Body of Knowledge (DMBOK) is a standard set of best practices for how the data world is changing and its best implementation across the industry. DAMA has two books. In the second book, published in 2017, more than 120 data experts revised their contributions. It also identifies 11 fundamental "knowledge areas" for data management. The DMBOK can be used in the workplace by knowledge workers, project managers, executives, technical employees, data analysts, and data architects.

The main goals of the DAMA DMBOK is as follows:

- Provide details about maturity models, deliverables, analytics, and best practices for data management
- Harmonize management techniques throughout the industry for putting these approaches into effect in any corporate setting and create a formal language
- Make clear the extent of what these techniques can and cannot do to offer a vendor-neutral review of management techniques and suitable substitutes for particular circumstances.
- Effectively manage data for essential success of the organization and to maximize the value of the data.
- Set up trustworthy data management techniques with the increasing amount of data.

DMBOK2 is a comprehensive reference book that is easily understood, authoritative, and prepared by top experts in the industry after being thoroughly evaluated by DAMA members. It consists of all the materials that fully define the difficulties of data management and how to address them using the following metrics:

- Define a set of guiding principles for data management techniques and determine how these concepts might be used in different functional areas of data management.
- Provide a practical framework for putting enterprise data management into effect, including commonly used practices, methodologies, and approaches, as well as roles, functions, deliverables, and metrics.

- Set up the foundation for best practices for data management professionals and create a consistent vocabulary for concepts in data management.

The DAMA-DMBOK architecture has been adopted and modified by numerous organizations, especially major businesses and governmental institutions, to enhance their data management processes. These businesses operate across a range of sectors, including the public sector, healthcare, banking, and telecommunications.

Businesses that have successfully implemented DAMA-DMBOK frequently follow these steps:

1. Evaluate the level of their data management maturity.

   This entails assessing the data management strategies across the ten core knowledge domains.

2. Identify areas that need improvement.

   Organizations can identify areas where current data management processes need to be changed or brought into line with DAMA-DMBOK principles.

3. Create an implementation schedule.

   This involves selecting the areas that will have the biggest effect on the organization's data management capabilities as well as creating goals and milestones for implementing DAMA-DMBOK practices.

4. Define roles.

   The duties and responsibilities of data stewards, data architects, and data governance leads must be clearly defined inside organizations.

5. Learn and train.

   Ensure that staff members are knowledgeable of the DAMA-DMBOK structure and have access to the materials, equipment, and training required for its successful implementation.

6. Track and evaluate development.

   Organizations can modify their strategy and ensure continuous improvement by routinely evaluating the progress and effects of putting DAMA-DMBOK methods into practice.

   When developing a data governance plan for your company, take into account a number of well-known frameworks and models for data governance. Here are some noteworthy examples:

   – COBIT

     The ISO 8000 (Data Quality) Model, the Data Governance Institute Data Governance Framework, and the CMMI Data Management Maturity Model Framework

     COBIT stands for Control Objectives for Information and Related Technologies.

     ISACA created the COBIT IT governance framework, which focuses on coordinating IT procedures with organizational objectives. It contains components relevant to data management and governance, even though it is not simply a data governance framework.

  - Data Governance Framework from the Data Governance Institute

    The Data Governance Institute's DGI Framework is a thorough approach to data governance that addresses a number of issues, including data quality, data architecture, data privacy, and data security. It emphasizes the value of stakeholder cooperation and offers helpful advice on carrying out data governance efforts.

- The DMM Model for CMMI Data Management

The CMMI Institute created the Data Maturity Model (DMM), a framework for process improvement that aids organizations in assessing and enhancing their data management capabilities. It offers a structured method to evaluate and improve data management methods and is divided into six process categories, including data governance.

- ISO 8000 (Data Quality)

This international standard series, ISO 8000, focuses on data quality and offers a set of values, prescriptions, and specifications for data management. Although it is not a comprehensive framework for data governance, it can support other frameworks by offering advice on data quality management.

The Zachman Framework

Enterprise Architecture (EA) has developed to structure businesses and ensure that they are in sync with IT systems. The Zachman Framework, an enterprise ontology and core building block of enterprise architecture, offers a method to view an enterprise and its information systems from many angles and demonstrates the interrelationships between the various parts of the enterprise.

Instead of depending on implicit ideas or understanding individual managers' brains, companies use enterprise architecture as a technique to develop explicit representations of enterprise operations and resources. Many major firms struggle greatly to adapt to changes in the complicated business settings of today. This challenge is partly brought on by a lack of internal understanding of the intricate structure of various parts of the organization, where heritage knowledge of the company is hidden in the brains of particular individuals or business units without being made apparent.

The Zachman Framework offers a way to categorize the architecture of an organization. It is a proactive technique for managing business transformation that can be used to model an organization's current functions, elements, and processes. The framework is based on John Zachman's knowledge of how change is handled in intricate goods like buildings and airplanes.

The stages of the system development lifecycle and the actions needed to construct systems throughout each of these stages are the organizing principles for many software techniques. Strategy, analysis, design, construction, transition, and testing are the components. John Zachman presented an alternative perspective on the components of system development in 1987. He arranged the process around the different stakeholders' points of view rather than expressing it as a series of phases. This gave enterprises a useful approach to evaluate how well software development process models meet their information needs.

As shown in Figure 6-11, a matrix of 36 cells, each of which focuses on a different aspect or perspective of the enterprise, makes up the Zachman Framework, a two-dimensional classification method for descriptive representations of an enterprise. While columns reflect the many perspectives of the stakeholders participating in the organization, rows are frequently depicted as the various viewpoints involved in the systems development process.

	Why	How	What	Who	Where	When
Contextual	Goal List	Process List	Material List	Organisational Unit & Role List	Geographical Locations List	Event List
Conceptual	Goal Relationship	Process Model	Entity Relationship Model	Organisational Unit & Role Relationship Model	Locations Model	Event Model
Logical	Rules Diagram	Process Diagram	Data Model Diagram	Role Relationship Diagram	Locations Diagram	Event Diagram
Physical	Rules Specification	Process Function Specification	Data Entity Specification	Role Specification	Location Specification	Event Specification
Detailed	Rules Details	Process Details	Data Details	Role Details	Location Details	Event Details

Figure 6-11. *The Zachman Framework*

The Zachman Framework's rows concentrate on describing the company from six different stakeholder perspectives. These six viewpoints are based on the W5H (what, where, who, when, why, and how) interrogatives in the English language.

A set of artefacts that describe an enterprise from a particular perspective of a group of stakeholders make up the frameworks columns. The many stakeholders are typically categorized as owners, planners, designers architects, implementers, users, or occasionally

as viewpoints such as scope context, business concepts, system logic, technology, physics, component assemblies, and operational classes. The interrogatives or questions posed to the company are represented by the columns:

- What data, information, or objects are relevant to the business ?
- What are the business' processes, or how does the business operate?
- Where are the operations of the company located?
- Who manages the company, and what are the business divisions and their hierarchies?
- When do business processes take place, i.e., what are the business workflows and schedules?
- Why was the solution adopted in the first place? Where did it come from? What drives people to perform particular tasks?

Each row depicts a different perspective on the organization from the viewpoint of many stakeholders. These are arranged in the order of importance. Each of the following stakeholders is given a row:

- *Planner's view*: This view outlines the goals and strategies of the company, setting the groundwork for the other perspectives. The other views are developed and handled within this context.
- *Owner's view*: This section gives an overview of the company that the information system must operate within. This perspective can be examined to determine which areas of the business can be automated.

- *Designer's view*: This view describes how the system will meet the information requirements of the organization. There are no production-specific restrictions or solution-specific elements in the representation.
- *Implementer's view*: This view shows how the system will be put into practice. It addresses production limitations and clarifies specific solutions and technologies.
- *Sub-constructor's view*: These illustrations highlight implementation-specific information for some system components, which must be clarified before production can start. Due to its focus on a particular aspect of the system rather than the entire system, this view is less structurally relevant than the others.
- *User's view*: This is a view of the operational environment in which the system is used.

Conclusion

This chapter explored the fundamentals of data modeling. It also explored in detail an overview of the Open Group Architecture Framework and DAMA, which is the data management body of knowledge. In the end, you also learned about the Zachman Framework.

Index

A

Ad hoc query/analysis process
  databricks SQL features, 152
  data lake, 150
  features, 151
  materialized view, 154, 155
  permission management, 153
  unity catalog/data lake, 154
Architecture process
  ad hoc query and analysis (*see* Ad hoc query/analysis process)
  DAMA DMBOK, 191–195
  definition, 143
  governance (*see* Data governance)
  models (*see* Data modeling)
  TOGAF implementation, 185–191
  Zachman Framework, 195–198
Azure Data Explorer (ADX), 68, 69, 88–89, 125, 126
Azure Data Factory (ADF), 42, 43, 58, 72, 86, 111, 123, 136, 137
Azure Data Lake Storage (ADLS), 98, 103, 153

B

Big data tools/techniques
  analytical store, 120–126
  batch processing tools
    databricks, 87
    data explorer, 88, 89
    interaction/boosts, 84
    lake analytics, 86
    long-running processes, 84
    services, 85
    Synapse analytics, 86
    transaction processing, 85
    transactions, 83
  components, 78
  data analysis, 81, 82
  data mesh platform
    cross-platform software, 111
    data storage layer, 108
    domain data stores, 109
    federated governance, 110
    implementation, 109, 111
    infrastructure duplication, 108
    microservices, 108
    principles, 109
    principles/logical architecture, 108

© Sagar Lad 2023
S. Lad, *Modern Data Architecture on Azure*, https://doi.org/10.1007/978-1-4842-9760-5

Big data tools/techniques (*cont.*)
product principle, 110
self-service data platform, 110
governance, 83
Internet of Things, 105–107
processing/ingestion, 79–81
real-time processing, 89–105
requirements, 77
stream processing, 80
visualizations, 82, 83
Binary Large Objects (BLOBs), 61

C

Command and Query Responsibility Segregation (CQRS)
implementation, 115, 116
pattern, 115
service models, 114
services, 116
Comma Separated values (CSV), 72–75
Control Objectives for Information and Related Technologies (COBIT), 193
CSV or JSON files, 72–75

D

Data Definition language (DDL), 147
Data governance
definition, 155
encryption process
features, 172
firewall features, 173
meaning, 170
region-based access controls, 172
requirements, 171
role-based access management, 172
storage, 173
types, 171
implementation, 157
management process
architecture models, 165, 166
considerations/technology, 164, 165
hybrid architecture, 167–169
registry architecture, 166, 167
repository architecture, 169, 170
masking techniques
ageing, 161
encryption, 161
meaning, 158
obfuscation, 158
production data, 159
pseudonymization, 159, 162
scrambling, 160
shuffling techniques, 161
substitution, 161
types, 160
pillars, 156
Purview, 157
storage optimization, 163, 164
Data Management Association (DAMA)

architecture, 31, 32
capabilities, 29
components, 25, 29
concepts/capability maturity models, 24
conceptual/logical/physical data model, 32
DAMA International (DAMA-I), 2, 3
data capabilities, 28
definition, 29
dimensions, 26
document/content management, 34
framework, 23, 25, 26
governance, 30
integration/interoperability, 34
KPIs, 29
metadata, 35, 36
modeling/design, 32
organizational levels, 23
organizational maturity assessment, 28
quality process, 36, 37
reference/master data, 34
security, 24
security lifecycle, 33
storage/operations, 33
warehousing/business intelligence, 35
Data Management Body of Knowledge (DMBOK)
architecture, 192
business implementation, 192
CMMI data management, 194
COBIT, 193
data management strategies, 192
duties/responsibilities, 192
implementation, 190
metrics, 191
track/evaluate development, 193 (*see also* Data Management Association (DAMA))
Data management solutions
Amsterdam information model, 22, 23
area context diagram, 38, 39
batch *vs*. streaming pattern, 9
competitive advantage, 12
consistency models, 8
DAMA, 2
definition, 1
environmental factors hexagon, 37, 38
frameworks, 20
fundamentals, 3
ingestion, 7
ingestion patterns, 8
lakehouse, 11
lakes, 11
lifecycle, 7, 8
lifecycle process, 14, 15
paradigms, 10
platform paradigm, 9–11
principles, 6, 7
processing/analysis, 8

Data management solutions (*cont.*)
qualitative
definition, 4
nominal data, 4
ordinal data, 5
quality measurements, 15
data-driven decisions, 17
metadata/lineage, 15, 16
quantitative
continuous, 6
discrete, 6
graphs/charts, 5
roles/responsibilities, 13, 14
storage, 7
strategic alignment
models, 20–22
strategy pillars, 13, 14
substantial volumes
governance/security, 19
integration, 18
monitoring/validation, 17
pipeline automation, 19
quality, 18
siloed/data sources, 18
type of, 3
visualization, 8
warehouse, 10
Data Maturity Model (DMM), 194
Data modeling
benefits, 149
business objectives/
requirement, 147–150
conceptual data model, 145, 146
customer/address, 176
customer information model, 176
data-centric solutions, 149
data models, 144
dimensional tables, 181
entity relationship diagram
(ER diagram), 183, 184
foundations/fundamentals, 175
graph, 182
hierarchical model, 178, 179
logical data model, 146
network model, 177, 178
object-oriented model, 180
physical data model, 147
relational data model, 179, 180
stakeholders, 147
technical implementation, 143
techniques, 177
workflow, 144
Design patterns/trends
analytical store
analytics/serving layer, 124
Azure Data Explorer, 125, 126
benefits, 122
database, 126
databricks lakehouse
platform, 122, 123
data processing, 121
integration process, 123
modern data engineering, 123
serving storage, 120
storage, 124
Synapse analytics, 121–123
transformation/model
training process, 124

clustered index, 119
CQRS pattern, 114–117
database design, 119
data driven, 114
data lakes
  benefits, 128
  datasets, 127
  democratization, 129
  design pattern, 128
  implementations, 129
  quality, 130
  relational systems, 128
  scalability, 130
  scenarios, 127
  schema drift, 130
  warehouses/lake houses, 131–133
enterprise organization, 126
event sourcing architecture, 117
index storage, 120
materialized views, 117
non-clustered index, 119
pipeline orchestrator
  Apache Oozie, 137
  Azure Data Factory, 136
  benefits, 135
  criteria, 135
  data orchestrator, 138
  handling data workflows, 134
  lifecycle, 133
  SSIS solutions, 137
real-time patterns
  Azure Stream Analytics, 141
  destination layer, 140
  elements, 139
  Event Hub, 140
  implementation, 140
  Kafka endpoint, 141
  stream processing, 138–140
table indexing, 119–121
T-SQL query, 120
Directed acyclic graphs (DAGs), 137

E, F

Enterprise Architecture (EA), 175, 184, 186, 187, 189, 195
Extract, transform, and load (ETL), 58
  databricks lakehouse platform, 122
  data integration, 42
  extraction, 43
  incremental extraction, 43
  load process, 44
  pipelines, 42
  Synapse server
    develop section, 45
    pipeline monitoring, 47
    pool creation, 46
  transformations, 44

G, H

General Data Protection Regulation (GDPR), 159, 162

I

Internet of Things (IoT), 88, 105–107

J

JavaScript Object Notation (JSON), 60
  CSV data
    benefits, 74
    blob storage, 75
    external table, 74
    OPENROWSET function, 74
    services, 72
    SQL query, 73

K

Kappa architecture
  advantages, 104
  benefits, 101
  design implementation, 105
  event data, 101
  historical analytics, 100
  implementation, 100, 103
  ingestion layer, 103
  meaning, 99
  process flow, 102
  records/events, 100
  setup tools, 103
  transformation engine, 100
Kusto Query Language (KQL), 69–71, 125

L

Lambda architecture
  advantages, 97
  batch layer, 96
  implementation, 98, 99
  interactive/expedited workspaces, 98
  meaning, 95
  reference batch, 96
  speed layer, 96

M

Massive parallel processing (MPP), 59, 86

N

Non-relational database
  column family database, 62
  document, 63, 64
  graph, 63–65
  key-value pairs, 61
  NoSQL databases, 60
  NoSQL selection strategy, 65
  storage model, 60

O

Online analytical processing (OLAP)
  analysis service model, 50
  analytical operations, 48
  complex analysis, 47

cubes, 47, 48
databases, 49
data warehouse, 52
key design considerations, 54
planning tasks, 47
semantic data model, 51–53
solutions, 53
transaction data, 54–59
Online transactional processing (OLTP)
ACID transaction, 56, 57
analysis and reporting, 59
data ingestion and storage, 59
design considerations, 58
external data sources, 59
incremental/regular backups, 55
meaning, 54
reference architecture workflow, 59
scenarios, 55
technical definitions, 56
warehousing architecture, 58

P, Q

Personally identified information (PII), 159
Platform as a Service (PaaS), 106

R

Real-time processing
big data tools
analytical data store, 91
ingestion, 92, 93
ingestion process flow, 89, 90
Kappa architecture, 99–105
Lambda architecture, 95–99
logical components, 90
processing/storing messages, 89
reference architecture, 95
reporting tools, 92
spark streaming, 94
streaming data, 93–95
stream processing, 90, 91
design patterns/trends, 139–142
Relational database, 41
ETL pipeline, 42–47
non-relational database (*see* Non-relational database)
OLAP (*see* Online analytical processing (OLAP))

S

Semantic data model, 51–53
Shared Access Signature (SAS), 172
Software-as-a-service (SaaS), 156
SQL Server Integration Services (SSIS), 43, 137
SQL Server Management Studio (SSMS), 137
Structured Query Language (SQL)
JSON/CSV data, 73
non-relational database, 60
time-series/free-form text processing, 67

T, U, V, W, X, Y

The Open Group Architecture Framework (TOGAF)
architecture change management, 190
architecture vision, 187
business architecture, 188
enterprise development method, 186
governance implementation phase, 190
high-level framework, 184
implementation, 185
information systems architecture, 188
migration planning, 189
opportunities/solutions, 189
preliminary phase, 187
technology architecture, 188
Time-series/free-form text processing
analysis, 70, 71
cognitive services, 66
data explorer, 69, 70
full-text queries, 67
KQL query, 69, 71
reference architecture, 66
search data, 65
time-series data, 67, 68
visualization, 69

Z

Zachman Framework
artefacts, 196
categories, 195
meaning, 195
stakeholders, 197, 198
systems development process, 196