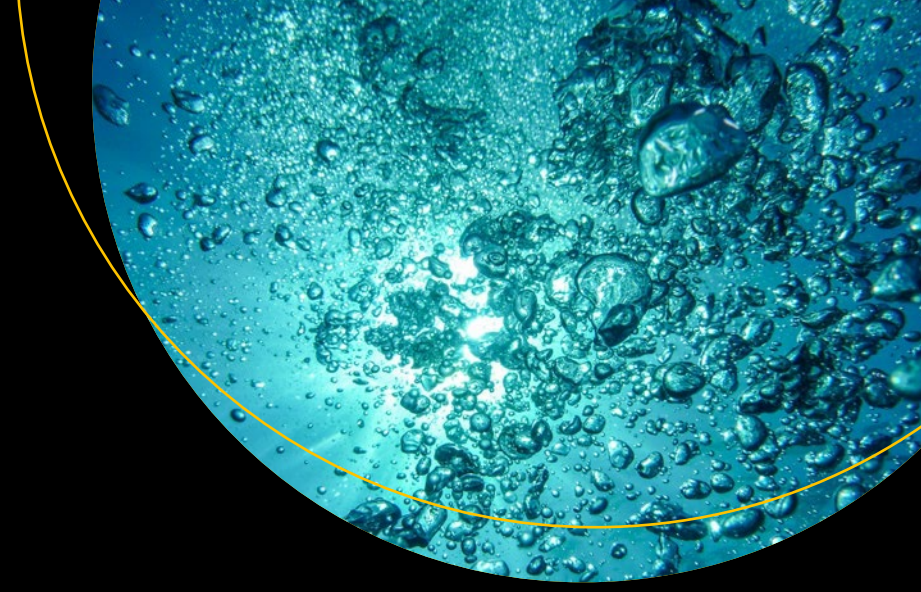

Beginning Azure Synapse Analytics

Transition from Data Warehouse to Data Lakehouse

Bhadresh Shiyal

Apress®

Beginning Azure Synapse Analytics

Transition from Data Warehouse to Data Lakehouse

Bhadresh Shiyal

Apress®

Beginning Azure Synapse Analytics: Transition from Data Warehouse to Data Lakehouse

Bhadresh Shiyal
Mumbai, India

ISBN-13 (pbk): 978-1-4842-7060-8 ISBN-13 (electronic): 978-1-4842-7061-5
https://doi.org/10.1007/978-1-4842-7061-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Development Editor: Laura Berendson
Coordinating Editor: Shrikant Vishwakarma

Cover designed by eStudioCalamar

Cover image designed by Pexels

Distributed to the book trade worldwide by Springer Science+Business Media LLC, 1 New York Plaza, Suite 4600, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, email 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, or visit http://www.apress.com/rights-permissions.

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 via the book's product page, located at www.apress.com/978-1-4842-7060-8. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Dedicated to my wife, Priti,
and my daughter, Jiya.

Table of Contents

About the Author

Bhadresh Shiyal is an Azure data architect and Azure data engineer. For the past seven years, he has been working with a large multinational IT corporation as solutions architect. Prior to that, he spent almost a decade in private- and public-sector banks in India in various IT positions working on several Microsoft technologies. He has 18 years of IT experience, including working for two years on an international assignment from London. He has much experience in application design, development, and deployment.

He has worked on myriad technologies, including Visual Basic, SQL Server, SharePoint Technologies, .NET MVC, O365, Azure Data Factory, Azure Databricks, Azure Synapse Analytics, Azure Data Lake Storage Gen1/Gen2, Azure SQL Data Warehouse, Power BI, Spark SQL, Scala, Delta Lake, Azure Machine Learning, Azure Information Protection, Azure .NET SDK, Azure DevOps, and more.

He holds multiple Azure Certifications, including Microsoft Certified Azure Solutions Architect Expert, Microsoft Certified Azure Data Engineer Associate, Microsoft Certified Azure Data Scientist Associate, and Microsoft Certified Azure Data Analyst Associate.

Bhadresh has worked as solutions architect on large-scale Azure Data Lake implementation projects as well as data transformation projects, in addition to large-scale customized content management systems. He has also worked as technical reviewer for the book *Data Science Using Azure*, prior to authoring this book.

About the Technical Reviewer

Massimo Nardone has more than 25 years of experience in security, web/mobile development, cloud, and IT architecture. His true IT passions are security and Android. He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years. He holds a Master of Science degree in computing science from the University of Salerno, Italy.

He has worked as a CISO, CSO, security executive, IoT executive, project manager, software engineer, research engineer, chief security architect, PCI/SCADA auditor, and senior lead IT security/cloud/SCADA architect for many years. Technical skills include security, Android, cloud, Java, MySQL, Drupal, Cobol, Perl, web and mobile development, MongoDB, D3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, Django CMS, Jekyll, Scratch, and more.

He worked as visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML, and Proxy areas). He is currently working for Cognizant as head of cyber security and CISO to help both internally and externally with clients in areas of information and cyber security, like strategy, planning, processes, policies, procedures, governance, awareness, and so forth. In June 2017 he became a permanent member of the ISACA Finland Board.

Massimo has reviewed more than 45 IT books for different publishing companies and is the co-author of *Pro Spring Security, Securing Spring Framework 5 and Boot 2-based Java Applications* (Apress, 2019), *Beginning EJB in Java EE 8* (Apress, 2018), *Pro JPA 2 in Java EE 8* (Apress, 2018), and *Pro Android Games* (Apress, 2015).

Acknowledgments

I always believed that writing a book was a solo exercise, but after completing this book, which is also my first book, I realized that this is not entirely true. You will need support and motivation to continue writing the chapters one after another, tirelessly.

First, I would like to thank my wife, Priti, and my daughter, Jiya, for bearing with me over the many weekends and weekdays, for over six months, on which I devoted my time to this book after completing my day job. The time that was meant to be given to them, I have devoted to writing this book; hence, a very big thank you to both of them. Without their support and constant motivation to complete the chapters on time, it would have not been possible to complete the book.

There are a couple of friends and colleages whom I would like to thank for their support, guiadance, and motivation. I was not initially sure if I could author a book. So, I reached out to a couple of people who have authored books through LinkedIn before deciding to write this book. I would like to thank Mitesh Soni (https://www.linkedin.com/in/mitesh-s-136842b) for help and guidance on this matter. I would also like to thank my colleague and friend Amit Ubale (https://www.linkedin.com/in/amit-ubale) for the constant support and motivation throughout the entire book-writing exercise.

I got constant support from Apress throughout the entire book-writing process. Special thanks go to Smriti Srivastav, acquisition editor, for the initial contract and clarifying all my doubts promptly; Shrikant Vishwakarma, coordinating editor, for genuine follow-ups for each chapter, which kept me on track; and Laura Berendson, development editor, for reviewing each chapter minutely. Finally, I thank Massimo Nardone for his technical review of my book.

Introduction

This book is about beginning to learn Azure Synapse Analytics. It is the most modern data analytics platform offered by Microsoft. It is always challenging to start learning a new IT skill, as you will have to learn new jargon, concepts, methods, tools, and technologies as part of the process. Azure Synapse Anlaytics is no different. I hope that this book will give you a solid technical foundation for starting your journey in the new world of Azure Synapse Analytics.

Anyone with a little IT background, particularly in Azure cloud, will be able to understand the content of the book very easily. The book is targeted to data analysts, data engineers, data scientists, data architects, solution architects, Azure developers, Azure administrators, and so forth who are keen to begin their Azure Synapse Analytics journey. This book will be a good starting point. I have concentrated on some basic concepts in the first two chapters so that it becomes easy for anyone to start learning about Azure Synapse Analytics.

The book has a total of ten chapters, and I recommend that you read them in sequence if you are new to data analytics. If you have a basic understanding of data analytics, data lake, data warehouse, data lakehouse, etc., then you can skip the first two chapters and jump to the third chapter directly. If anyone is interested in any specific components of Azure Synapse Analytics, then one can jump to that component's specific chapter directly as well. Here I have given a very high-level summary of each chapter in a very brief manner:

> Chapter 1: "Core Data and Analytics Concepts" introduces readers to some of the important core data and analytics concepts as a foundation.
>
> Chapter 2: "Modern Data Warehouse and Data Lakehouse" provides conceptual understanding about traditional/legacy data warehouse, modern data warehouse, and finally the most modern data lakehouse.

Chapter 3: "Introduction to Azure Synapse Analytics" builds foundational knowledge by introducing Azure Synapse Analytics, its main features, and its key services capabilities.

Chapter 4: "Architecture and Its Main Components" explains Azure Synapse Analytics' core architecture and its main components, as it is very different from traditional data warehouse architecture and its components.

Chapter 5: "Synapse SQL" explores Synapse SQL in detail, including its architecture, its main features, and some how-to's to make readers familiar with some important activities that can be carried out for Synapse SQL.

Chapter 6: "Synapse Spark" explains Synapse Spark, its main components, and Delta Lake, along with some how-to's to make readers familiar with important tasks pertaining to Synapse Spark.

Chapter 7: "Synapse Pipelines" introduces Azure Synapse Pipelines and explans various types of pipeline activities with examples.

Chapter 8: "Synapse Workspace and Synapse Studio" familiarizes readers with Synapse Workspace and Synapse Studio, including its main features and its capabilities and how to accomplish some important tasks.

Chapter 9: "Synapse Link" explains the differences between OLTP and OLAP, why HTAP is required and its benefits, and then introduces Synapse Link along with its Cosmos DB integration, its features, and some use cases.

Chapter 10: "Azure Synapse Use Cases and Reference Architectures" discusses some of the industry use cases and reference architectures to introduce readers to real-life usage of Azure Synapse Analytics.

Happy reading !

CHAPTER 1

Core Data and Analytics Concepts

A few years back, oil companies used to rule the business world. That is no longer the case. Now data is considered the new oil, and there are multiple reasons to agree with this idea. Due to the explosion in social media platforms, a high volume of data is generated on a daily basis. Additionally, **I**nternet **o**f **T**hings (**IoT**) devices generate a significant volume of data. Similarly, a variety of data is being generated and stored at a never-before-seen high velocity. As a result, organizations of all sizes across the globe have started to utilize the available data to their advantage. In today's modern era, each organization strives to become a data-driven one by introducing data-driven decision-making processes so as to stay ahead of competition in the market.

The IT industry has adapted to the changes in the data landscape very quickly. Many changes have happened rapidly regarding the kind of work we do in IT, as well as the names of the roles. For example, data analyst, data engineer, data scientist, and so forth are some of the new roles that have gotten popular in recent times. We also started to deal with all types of data, including structured data, semi-structured data, and unstructured data, along with the different ways in which data is made available for processing, including batch and streaming data. You may be familiar with the basic concept of databases, but let us start with core data concepts as a refresher.

Core Data Concepts

Before we jump into data analytics in detail, let us go through some of the core data concepts. If you feel that some or all of the concepts are known to you, then please feel free to jump to the next section or chapter. Here, we will go through some of the basic concepts, like data, structured data, unstructured data, semi-structured data, batch data processing, and streaming or real-time (RT) data processing.

B. Shiyal, *Beginning Azure Synapse Analytics*, https://doi.org/10.1007/978-1-4842-7061-5_1

What Is Data?

Let us look at the definition of "data" before we dive into other core data concepts. The word "data" is a plural of the word "datum." The word "datum" is originally from Latin. It means a piece of information. Thus, data are multiple pieces of information put together. Basically, we are talking about a bunch of information when we talk about data. In today's era of rapidly advancing technology, the word "data" has become very common in usage. "Data science" and "data engineering" have become very popular terms, which has increased the importance of data as well.

In the IT field, we know that any software is divided into two parts. One part is the code of the software, and the other part is the data on which the software code will take some actions. There are various types of data with which we deal regularly. Here, we are not talking about data types in a programing language paradigm. Generally, based on how the data is arranged or stored, we can divide data into the following three categories or types: structured data, semi-structured data, and unstructured data. Let us try to understand each one of them.

Structured Data

Structured data is any data that is highly organized and conforms to a specific schema or format. Each data point is placed in a highly organized manner so that it becomes very easy to interpret for both human beings and machines like computers. Since the beginning of the IT era, computers have been highly efficient in understanding and processing structured data. Data that we store in spreadsheet programs like Excel, as well as any data stored in database systems like SQL Server or Azure SQL Database, are basically structured data stored in a tabular format.

Let us look at an example of structured data. Let us assume that you have been asked to create an Excel sheet containing details of all the employees of your organization or of a specific department within your organization. You enter column names like Employee ID, First Name, Last Name, Date of Joining, Date of Birth, and Salary. You also enter details about a few employees. We can say that the data stored in your Excel sheet is structured data because it is highly organized and all records conform to the schema (list of all those columns you defined). Similarly, if you decide to store employee details from that Excel sheet in a database like SQL Server by creating a new table in it, then that would also be considered structured data.

Semi-structured Data

Semi-structured data is any data that is *not* fully organized but still partially conforms to a specific schema or format. Data stored in JSON or **J**ava**S**cript **O**bject **N**otation **(JSON)** documents and XML or e**X**tensible **M**ark-up **L**anguage **(XML)** files are good examples of semi-structured data. Instead of field names or columns, which we use for structured data, here we use tags to define different data points within a record. It is not considered highly organized since the order of the columns as well as the number of columns for each record may vary based on actual requirements. It means that semi-structured data lacks a fixed or rigid schema or data model. Generally, data and schemas are mixed without separation. Key–value stores and graph databases are used to store semi-structured data.

Let us assume that your IT team has given you a JSON file that contains your employees' details. If you open the file and read the data inside it, you will notice that there are tags for each data value. You may also find that for a few employees there are additional tags that are not available for all employees. For example, for a few employees, it may show you a nested structure to store previous roles or positions that they held in the past in your organization. This may not be present for those employees who have not held more than one role or position. Here, the employee data is somewhat organized and partially conforms to some schema or data model, so it is called semi-structured data.

Unstructured Data

This is the most important type of data that you should know about in more detail as around 85 percent of corporate data in this world is unstructured data. Any data that is *not* highly organized and *does not* conform to a specific schema or format is called unstructured data. In the last few years, it has gotten increasing attention due to its volume and the value it brings to data-driven decision-making processes. Businesses face huge challenges in storing, processing, and extracting hidden information from unstructured data.

Documents, image files, audio files, video files, and so forth are all examples of unstructured data. It is very difficult to gain meaningful insights from unstructured data as it is not highly organized, and it does not have a specific schema either. For unstructured data, you may try to store some properties at the file level that allow you to store additional information about the data being stored inside the file containing unstructured data. However, these properties will not be able to give you all the details

about unstructured data. For example, if you have to process a lot of image files, there is no simple way to do so apart from using some advanced machine learning algorithms that can scan through each image and predict what is inside each image.

Storage of unstructured data was also a real challenge as we cannot store such data in a database since it does not conform to any schema. In the last few years, cloud-based blob storage offerings from various public cloud companies have solved this problem to a large extent. Azure has multiple options to store your unstructured data. Azure File Storage, Azure Blob Storage, Azure Data Lake Storage Gen 1, and Azure Data Lake Storage Gen 2 are various types of storage options available to us to store unstructured data in the Azure cloud. Generally, it is preferred to store it on cloud storage, which may provide highly secured unlimited storage capacity and greater scalability with various cost-effective options that you can choose based on your actual storage requirements.

We have looked at various types of data based on how it is organized and stored. Now, let us see different ways of processing the data.

Data Processing Methods

The data that you receive from source systems will be in raw and unprocessed form. To derive meaningful insights from raw data, it needs to be processed first. This process of converting and transforming raw data into a usable and desired form is known as data processing. There are different methods of data processing based on how and when you process the data.

Batch Data Processing

This is the most common and traditional way of data processing. When you collect the newly ingested data into a group before processing it all together, it is known as batch data processing. Data ingestion may happen multiple times before you process it as part of a single batch at a future time. There are multiple pros and cons for batch data processing. Based on your requirements, you can decide if batch data processing will be suitable or not.

It allows you to process the data during non-business hours. It also provides flexibility to process a large volume of data at once. Many large data processing jobs are executed during night hours, when it will have the least impact on the business. Generally, data collected from various **o**n-**l**ine **t**ransaction **p**rocessing (**OLTP**) systems

during business hours will be processed when business hours are over as part of overnight batch data processing jobs. Based on business requirements, you may also decide to run batch data processing jobs more than once during the day. For example, instead of processing all data together during overnight hours, you may decide to process data every four or eight hours.

By now, you will have noticed that batch data processing delays the availability of processed data, as the ingested data is not processed immediately but rather later, in a batch. There are many business scenarios in which this is not an acceptable approach for data processing. Another limitation of batch data processing that you need to consider is that all the required data should be available at the time of batch data processing, without any errors in it. Erroneous data or incomplete data will result in a delay in the batch data processing schedule, which may impact the business outcomes.

Let us assume that you download sales data for all salespersons at the end of each day from your sales transaction system. Based on that, you run a job to update the consolidated sales report containing the sales details of each salesperson for each day. This is an example of batch data processing in which you collect and group the ingested data for a day in a batch and then process it to update the consolidated sales report for that day.

Streaming or Real-Time Data Processing

If each new piece of data being ingested is also immediately being processed, then it is called streaming data processing. It is also known as **r**eal-**t**ime (**RT**) data processing, as it processes the data in real time without any major delay. Unlike batch data processing, there is no waiting after the data is ingested, as the data processing takes place in real time with a delay of only a few seconds or milliseconds. There are business scenarios in which new and dynamic data is generated continuously. These scenarios are the most appropriate in which to use real-time data processing or streaming data processing. Time-critical business applications will require streaming data processing, as any delay of a few seconds or in certain situations a few milliseconds will result in an adverse impact on business outcomes.

While batch data processing has all the data in a batch to be processed later, streaming data processing has only the most recent data or data generated within a rolling time window of less than one minute or so. Batch data processing can handle a very large volume of data in a batch, while streaming data processing processes only a

few records at a time, as it must process the data in real time. These differences imply that you should use batch data processing for large and complex analytics workloads, while streaming data processing should be used for simple calculations or aggregations due to its very limited time window for data processing.

An **a**utomated **t**eller **m**achine (**ATM**) is a good example of the use of streaming data processing. The data processing happens in real time without much delay. It debits your account in real time, and the ATM dispatches the money instantaneously. Many financial institutions will track the stock market price movements in real time and update their risks immediately to avoid any losses. With streaming data processing, they can also provide real-time stock advice to their valuable customers. A lot of sensor data being used by autonomous vehicles (self-driving cars, etc.) requires streaming data processing, as any delay in processing may prove fatal. Industrial IoT devices and wearable smart devices like health bands, smart watches, and so forth use streaming data processing.

Let us assume that you are responsible for updating a sales report that includes the details of each sale made by each salesperson in real time. Here, it will become necessary for you to stream the data and process it in real time as and when a sale is registered in your sales transaction system. This will allow you to see the real-time sales numbers on your dashboard.

Relational Data and Its Characteristics

In bygone days, each application used to store data in its own unique data structure, which made it very difficult to make changes to it. This data structure was hard to design, code, and maintain. To solve this problem, **r**elational **d**ata**b**ase **m**anagement **s**ystems (**RDBMS**) were introduced. Relational databases made life easy for developers as they followed the specific pattern of a relational data model, which was easy to replicate across many applications. It was also very easy to design, code, and maintain such relational database management systems.

There are some important characteristics of relational data that we will go through one by one here:

- **Table:** A real-life object can be considered an entity for which we want to store information or data. This can be directly mapped to a table. Thus, a table is very basic but is the most important characteristic of relational data.

- **Rows and columns:** A table consists of one or multiple rows, which allow us to store multiple records or information about multiple similar objects or entities in a single table. Each row will have one or more columns to store different types of information about the real-life object or entity. Each column can store only one type of value for each row. Each row will have the same number and types of columns and in the same order as well.
- **Primary key:** To maintain uniqueness within a table or to identify each record within a table uniquely, at least one column across all rows must have different and unique values. That column is known as the primary key of the table. There may be a scenario in which you will have to combine more than one column to have uniqueness within the table. These columns are collectively called the composite key.
- **Relationship:** This is a very important characteristic from which the relational database derives its name, as it allows you to establish relationships among tables within the database. We can define a relationship between two tables by setting the primary key from one table to be a foreign key in another table. This gives us the ability to implement real-life objects' relationships very easily in relational data.
- **Foreign Key:** To create relationships between two tables, it is necessary that the first table's primary key is present in the second table, and that column in the second table must be defined as the foreign key. This way, a foreign key is the column, or the combination of columns, present in the second table, and the same column or combination of columns is defined as the primary key in the first table. This plays a crucial role in defining relationships between two tables.
- **Structured Query Language (SQL):** It is necessary to have the ability to easily query the stored data from relational tables. Apart from querying data, SQL also supports various other operations, like creating a table, inserting records into a table, updating records into a table, and deleting records from a table. It also allows you to

filter rows based on where conditions, which you can specify in your queries. It also makes it possible to combine or join two or more tables together, allowing you to retrieve data from two or more tables at the same time using a single query statement.

- **ACID Properties:** ACID stands for **a**tomicity, **c**onsistency, **i**solation, and **d**urability. Any changes to the database are done via a transaction, which is a single logical unit of work to be carried out on the database. ACID properties are followed in most of the relational databases to maintain consistency before and after the transaction. Atomicity ensures that either the full transaction succeeds, or it fails. There is no partial success or failure allowed. Consistency ensures that the correctness of the database is maintained before and after each transaction. It enforces integrity constraints in the database. Isolation ensures that multiple transactions can occur simultaneously without compromising consistency in the database. The changes being carried out by one transaction will not be visible to the other transaction until the first transaction is committed successfully. Durability ensures that once the transactions are committed to the database, they are permanently stored in persistent storage, which can retrieved easily in the case of any system failure.

Azure provides multiple options for storing relational data in various RDBMSs. Azure SQL Database, Azure Database for PostgreSQL, Azure Database for MySQL, and Azure Database for MariaDB are various RDBMS options available on Azure for storing relational data. Each option has its own pros and cons, so it is necessary to explore each in detail to check which one meets most of your business and technical requirements.

Non-Relational Data and Its Characteristics

There are many real-life scenarios that will not have a relational data structure. For example, you cannot store video, audio, images, text data, and so forth in a relational database, as these are not relational data, but you still need to deal with these types of data. These are non-relational data, and you will have to store them in non-relational databases or repositories. These data will not have a relational structure, so they cannot be stored in a table in a relational database.

Generally, non-relational data is bigger in volume than relational data, and it also comes in a variety of data formats. It is necessary to understand the important characteristics of non-relational data. So, let us check a few of them out here:

- **Containers:** As we know, tables are used to store relational data. Here, we can use containers to store non-relational data. This allows us to store many entities or objects with different schema or different fields. Thus, containers support varied schema, which is not possible for a table in a relational database.
- **Storage Flexibility:** Non-relational data will not have a fixed schema. It is necessary to have storage flexibility to store data with varied formats, like audio, video, image, text, and so on. Non-relational data is stored in its native format, which is an important characteristic as it will allow you to deal with non-relational data in its raw format.
- **Data Retrieval:** For non-relational data, generally a unique key–value pair is used to identify an entity or object. Many non-relational databases support key–value pairs, which helps in retrieving non-relational data easily. It is recommended you use key–value pairs only to iterate through or filter the entities or objects stored in containers. Non-relational databases will provide their own SQL-like data-retrieval mechanisms or may have their own procedures for data retrieval.
- **Indexing:** This is not supported by all non-relational databases. This characteristic is similar to relational databases to a great extent. If the non-relational database you are using supports indexing, then your queries should be able to use the benefits of indexing to identify and fetch data for fields other than key–value pairs.

When you plan to use non-relational databases, it is very important to understand the preceding characteristics and check which of these are supported by your choice of non-relational database. Azure provides multiple options for storing various types of non-relational data. Azure Cosmos DB, Azure Blob Storage, Azure File Storage, and so forth are some of the non-relational data storage options provided by Azure.