January 2021

AZ-900 – Azure Data Fundamentals – Microsoft Learn

https://docs.microsoft.com/en-us/learn/certifications/exams/dp-900

Azure Data Fundamentals: Explore core data concepts

https://docs.microsoft.com/en-us/learn/paths/azure-data-fundamentals-explore-core-data-concepts/

Azure Data Fundamentals: Explore core data concepts

I. Introduction

Over the last few decades, the amount of data that systems, applications, and devices have generated has increased significantly. Data is everywhere. Data is available in different structures and formats. Understanding data and exploring it reveals interesting facts, and helps you gain meaningful insights.

In this module, you'll learn about how you can organize and process data. You'll learn about relational and non-relational databases, and how data is handled through transactional processing, and through batch and streaming data processing.

To consider how the tools and techniques learnt can be applied in real world scenarios, imagine you're a data analyst for a large consumer organization. The organization wants to understand customer buying patterns from supermarkets. The organization has a number of datasets from different sources, such as till information (point of sale), weather data, and holiday data. The organization would like to use Azure technologies to understand and analyze these datasets. This scenario will be used throughout the module.

Learning objectives

Identify how data is defined and stored

Identify characteristics of relational and non-relational data

Describe and differentiate data workloads

Describe and differentiate batch and streaming data

II. Identify the need for data solutions

Data is now easier to collect and cheaper to host, making it accessible to nearly every business. Data solutions include software technologies and platforms that can help facilitate the collection, analysis, and storage of valuable information. Every business would like to grow their revenues and make larger profits. In this competitive market, data is a valuable asset. When analyzed properly, data provides a wealth of useful information and inform critical business decisions.

What is data?

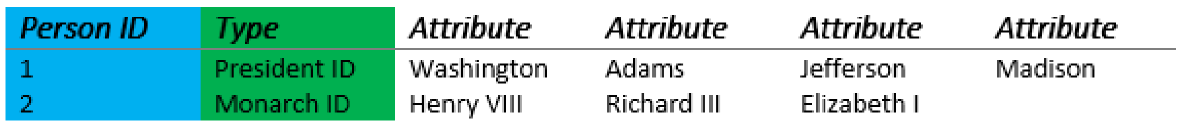
Data is a collection of facts such as numbers, descriptions, and observations used in decision making. You can classify data as structured, semi-structured, or unstructured.

Structured data is typically tabular data that is represented by rows and columns in a database. Databases that hold tables in this form are called relational databases (the mathematical term relation refers to an organized set of data held as a table). Each row in a table has the same set of columns. The image below illustrates an example showing two tables in an ecommerce database. The first table contains the details of customers for an organization, and the second holds information about products that the organization sells.

Semi-structured data is information that doesn't reside in a relational database but still has some structure to it. Examples include documents held in JavaScript Object Notation (JSON) format. The example below shows a pair of documents representing customer information. In both cases, each customer document includes child documents containing the name and address, but the fields in these child documents vary between customers.

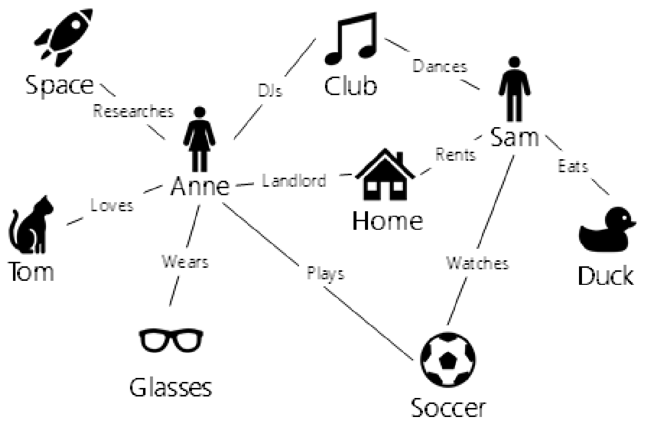
There are other types of semi-structured data as well. Examples include key-value stores and graph databases.

A key-value store is similar to a relational table, except that each row can have any number of columns. The image below shows an example of key-value data.



Person ID	*Type*	*Attribute*	*Attribute*	*Attribute*	*Attribute*
1	President ID	Washington	Adams	Jefferson	Madison
2	Monarch ID	Henry VIII	Richard III	Elizabeth I	

You can use a graph database to store and query information about complex relationships. A graph contains nodes (information about objects), and edges (information about the relationships between objects). The image below shows an example of how you might structure the data in a graph database.



Not all data is structured or even semi-structured. For example, audio and video files, and binary data files might not have a specific structure. They're referred to as unstructured data.

How is data defined, stored, and accessed in cloud computing?

Depending on the type of data such as structured, semi-structured, or unstructured, data will be stored differently. Structured data is typically stored in a relational database such as SQL Server or Azure SQL Database. Azure SQL Database is a service that runs in the cloud. You can use it to create and access relational tables. The service is managed and run by Azure, you just specify that you want a database server to be created. The act of setting up the database server is called provisioning.

You can provision other services as well in Azure. For example, if you want to store unstructured data such as video or audio files, you can use Azure Blob storage (Blob is an acronym for Binary Large Object). If you want to store semi-structured data such as documents, you can use a service such as Azure Cosmos DB.

After your service is provisioned, the service needs to be configured so that users can be given access to the data. You can typically define several levels of access.

- Read-only access means the users can read data but can't modify any existing data or create new data.
- Read/write access gives users the ability to view and modify existing data.
- Owner privilege gives full access to the data including managing the security like adding new users and removing access to existing users.

You can also define which users should be allowed to access the data in the first place. If the data is sensitive (or secret), you may want to restrict access to a few select users.

In the example where you're a data analyst for a large consumer organization you have decided to give read-only access to the whole management team. The management team have no need to modify data, but have security clearance to see any data. Read-write access is given to the app that salespeople use to record sales. The individual users won't need to access the system directly, but will make edits via their app. Data analysts and data managers will have owner privileges because they need to manage the access of other users and administer the system.

Describe data processing solutions

Data processing solutions often fall into one of two broad categories: analytical systems, and transaction processing systems.

What is a transactional system?

A transactional system is often what most people consider the primary function of business computing. A transactional system records transactions. A transaction could be financial, such as the movement of money between accounts in a banking system, or it might be part of a retail system, tracking payments for goods and services from customers. Think of a transaction as a small, discrete, unit of work.

Transactional systems are often high-volume, sometimes handling many millions of transactions in a single day. The data being processed has to be accessible very quickly. The work performed by transactional systems is often referred to as Online Transactional Processing (OLTP).

To support fast processing, the data in a transactional system is often divided into small pieces. For example, if you're using a relational system each table involved in a transaction only contains the columns necessary to perform the transactional task. In the bank transfer example, a table holding information about the funds in the account might only contain the account number and the current balance. Other tables not involved in the transfer operation would hold information such as the name and address of the customer, and the account history. Splitting tables out into separate groups of columns like this is called normalized. The next unit discusses this process in more detail. Normalization can enable a transactional system to cache much of the information required to perform transactions in memory, and speed throughput.

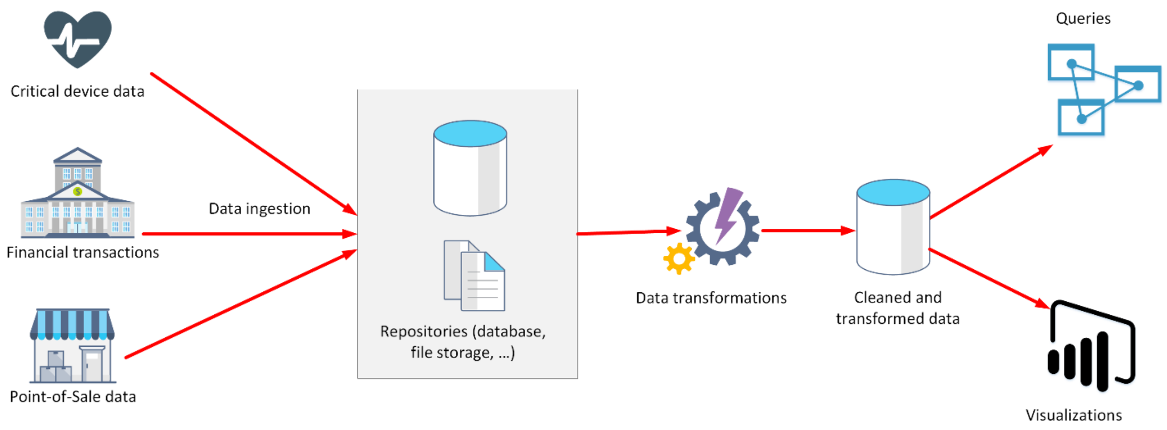
While normalization enables fast throughput for transactions, it can make querying more complex. Queries involving normalized tables will frequently need to join the data held across several tables back together again. This can make it difficult for business users who might need to examine the data.

What is an analytical system?

In contrast to systems designed to support OLTP, an analytical system is designed to support business users who need to query data and gain a big picture view of the information held in a database.

Analytical systems are concerned with capturing raw data, and using it to generate insights. An organization can use these insights to make business decisions. For example, detailed insights for a manufacturing company might indicate trends enabling them to determine which product lines to focus on, for profitability.

Most analytical data processing systems need to perform similar tasks: data ingestion, data transformation, data querying, and data visualization. The image below illustrates the components in a typical data processing system.



Data Ingestion:

Data ingestion is the process of capturing the raw data. This data could be taken from control devices measuring environmental information such as temperature and pressure, point-of-sale devices recording the items purchased by a customer in a supermarket, financial data recording the movement of money between bank accounts, and weather data from weather stations. Some of this data might come from a separate OLTP system. To process and analyze this data, you must first store the data in a repository of some sort. The repository could be a file store, a document database, or even a relational database.

Data Transformation/Data Processing:

The raw data might not be in a format that is suitable for querying. The data might contain anomalies that should be filtered out, or it may require transforming in some way. For example, dates or addresses might need to be converted into a standard format. After data is ingested into a data repository, you may want to do some cleaning operations and remove any questionable or invalid data, or perform some aggregations such as calculating profit, margin, and other Key Performance Metrics (KPIs). KPIs are how businesses are measured for growth and performance.

Data Querying:

After data is ingested and transformed, you can query the data to analyze it. You may be looking for trends, or attempting to determine the cause of problems in your systems. Many database management systems provide tools to enable you to perform ad-hoc queries against your data and generate regular reports.

Data Visualization:

Data represented in tables such as rows and columns, or as documents, aren't always intuitive. Visualizing the data can often be useful as a tool for examining data. You can generate charts such as bar charts, line charts, plot results on geographical maps, pie charts, or illustrate how data changes over time. Microsoft offers visualization tools like Power BI to provide rich graphical representation of your data.

III. Identify types of data and data storage

Describe the characteristics of relational and non-relational data

Relational databases provide probably the most well-understood model for holding data. The simple structure of tables and columns makes them easy to use initially, but the rigid structure can cause some problems. For example, in a database holding customer information, how do you handle customers that have more than one address? Do you add columns to hold the details for each address? If so, how many of these columns should you add? If you allow for three addresses, what happens if a customer has only one address? What do you store in the spare columns? What then happens if you suddenly have a customer with four addresses? Similarly, what information do you store in an address (street name, house number, city, zip code)? What happens if a house has a name rather than a number, or is located somewhere that doesn't use zip codes?

You can solve these problems by using a process called normalization. Typically, the end result of the normalization process is that your data is split into a large number of narrow, well-defined tables (a narrow table is a table with few columns), with references from one table to another, as shown in the image below. However, querying the data often requires reassembling information from multiple tables by joining the data back together at run-time (illustrated by the lines in the diagram). These types of queries can be expensive.

Non-relational databases enable you to store data in a format that more closely matches the original structure. For example, in a document database, you could store the details of each customer in a single document, as shown by the example in the previous unit. Retrieving the details of a customer, including the address, is a matter of reading a single document. There are some disadvantages to using a document database though. If two customers cohabit and have the same address, in a relational database you would only need to store the address information once. In the diagram below, Jay and Frances Adams both share the same address. In a document database, the address would be duplicated in the documents for Jay and Francis Adams. This duplication not only increases the storage required, but can also make maintenance more complex (if the address changes, you must modify it in two documents).

Describe transactional workloads

Relational and non-relational databases are suited to different workloads. A primary use of relational databases is to handle transaction processing.

A transaction is a sequence of operations that are atomic. This means that either all operations in the sequence must be completed successfully, or if something goes wrong, all operations run so far in the sequence must be undone. Bank transfers are a good example; you deduct funds from one account and credit the equivalent funds to another account. If the system fails after deducting the funds, they must be reinstated in the original account (they mustn't be lost). You can then attempt to perform the transfer again. Similarly, you shouldn't be able to credit an account twice with the same funds.

Each database transaction has a defined beginning point, followed by steps to modify the data within the database. At the end, the database either commits the changes to make them permanent, or rolls back the changes to the starting point, when the transaction can be tried again.

A transactional database must adhere to the ACID (Atomicity, Consistency, Isolation, Durability) properties to ensure that the database remains consistent while processing transactions.

Atomicity guarantees that each transaction is treated as a single unit, which either succeeds completely, or fails completely. If any of the statements constituting a transaction fails to complete, the entire transaction fails and the database is left unchanged. An atomic system must guarantee atomicity in each and every situation, including power failures, errors, and crashes.

Consistency ensures that a transaction can only take the data in the database from one valid state to another. A consistent database should never lose or create data in a manner that can't be accounted for. In the bank transfer example described earlier, if you add funds to an account, there must be a corresponding deduction of funds somewhere, or a record that describes where the funds have come from if they have been received externally. You can't suddenly create (or lose) money.

Isolation ensures that concurrent execution of transactions leaves the database in the same state that would have been obtained if the transactions were executed sequentially. A concurrent process can't see the data in an inconsistent state (for example, the funds have been deducted from one account, but not yet credited to another.)

Durability guarantees that once a transaction has been committed, it will remain committed even if there's a system failure such as a power outage or crash.

Database systems that process transactional workloads are inherently complex. They need to manage concurrent users possibly attempting to access and modify the same data at the same time, processing the transactions in isolation while keeping the database consistent and recoverable. Many systems implement relational consistency and isolation by applying locks to data when it is updated. The lock prevents another process from reading the data until the lock is released. The lock is only released when the transaction commits or rolls back. Extensive locking can lead to poor performance, while applications wait for locks to be released.

Distributed databases are widely used in many organizations. A distributed database is a database in which data is stored across different physical locations. It may be held in multiple computers located in the same physical location (for example, a datacenter), or may be dispersed over a network of interconnected computers. When compared to non-distributed database systems, any data update to a distributed database will take time to apply across multiple locations. If you require transactional consistency in this scenario, locks may be retained for a very long time, especially if there's a network failure between databases at a critical point in time. To counter this problem, many distributed database management systems relax the strict isolation requirements of transactions and implement "eventual consistency." In this form of consistency, as an application writes data, each change is recorded by one server and then propagated to the other servers in the distributed database system asynchronously. While this strategy helps to minimize latency, it can lead to temporary inconsistencies in the data. Eventual consistency is ideal where the application doesn't require any ordering guarantees. Examples include counts of shares, likes, or non-threaded comments in a social media system.

Describe analytical workloads

Analytical workloads are typically read-only systems that store vast volumes of historical data or business metrics, such as sales performance and inventory levels. Analytical workloads are used for data analysis and decision making. Analytics are generated by aggregating the facts presented by the raw data into summaries, trends, and other kinds of "Business information."

Analytics can be based on a snapshot of the data at a given point in time, or a series of snapshots. People who are higher up in the hierarchy of the company usually don't require all the details of every transaction. They want the bigger picture.

An example of analytical information is a report on monthly sales. As the head of sales department, you may not need to see all daily transactions that took place (transactional information), but you definitely would like a monthly sales report to identify trends and to make decisions (analytical information).

Transactional information, however, is an integral part of analytical information. If you don't have good records of daily sales, you can't compile a useful report to identify trends. That's why efficient handling of transactional information is very important.

IV. Describe the difference between batch and streaming data

Processing data as it arrives is called streaming. Buffering and processing the data in groups is called batch processing.

Understand batch processing

In batch processing, newly arriving data elements are collected into a group. The whole group is then processed at a future time as a batch. Exactly when each group is processed can be determined in a number of ways. For example, you can process data based on a scheduled time interval (for example, every hour), or it could be triggered when a certain amount of data has arrived, or as the result of some other event.

An example of batch processing is the way that votes are typically counted in elections. The votes are not entered when they are cast, but are all entered together at one time in a batch.

Advantages of batch processing include:

- Large volumes of data can be processed at a convenient time.
- It can be scheduled to run at a time when computers or systems might otherwise be idle, such as overnight, or during off-peak hours.

Disadvantages of batch processing include:

- The time delay between ingesting the data and getting the results.
- All of a batch job's input data must be ready before a batch can be processed. This means data must be carefully checked. Problems with data, errors, and program crashes that occur during batch jobs bring the whole process to a halt. The input data must be carefully checked before the job can be run again. Even minor data errors, such as typographical errors in dates, can prevent a batch job from running.

An example of an effective use of batch processing would be a connection to a mainframe system. Vast amounts of data need to be transferred into a data analysis system and the data is not real-time. An example of ineffective batch-processing would be to transfer small amounts of real-time data, such as a financial stock-ticker.

Understand streaming and real-time data

In stream processing, each new piece of data is processed when it arrives. For example, data ingestion is inherently a streaming process.

Streaming handles data in real time. Unlike batch processing, there's no waiting until the next batch processing interval, and data is processed as individual pieces rather than being processed a batch at a time. Streaming data processing is beneficial in most scenarios where new, dynamic data is generated on a continual basis.

Examples of streaming data include:

- A financial institution tracks changes in the stock market in real time, computes value-at-risk, and automatically rebalances portfolios based on stock price movements.
- An online gaming company collects real-time data about player-game interactions, and feeds the data into its gaming platform. It then analyzes the data in real time, offers incentives and dynamic experiences to engage its players.
- A real-estate website that tracks a subset of data from consumers' mobile devices, and makes real-time property recommendations of properties to visit based on their geo-location.

Stream processing is ideal for time-critical operations that require an instant real-time response. For example, a system that monitors a building for smoke and heat needs to trigger alarms and unlock doors to allow residents to escape immediately in the event of a fire.

Understand differences between batch and streaming data
Apart from the way in which batch processing and streaming processing handle data, there are other differences:

Data Scope: Batch processing can process all the data in the dataset. Stream processing typically only has access to the most recent data received, or within a rolling time window (the last 30 seconds, for example).

Data Size: Batch processing is suitable for handling large datasets efficiently. Stream processing is intended for individual records or micro batches consisting of few records.

Performance: The latency for batch processing is typically a few hours. Stream processing typically occurs immediately, with latency in the order of seconds or milliseconds. Latency is the time taken for the data to be received and processed.

Analysis: You typically use batch processing for performing complex analytics. Stream processing is used for simple response functions, aggregates, or calculations such as rolling averages.

Azure Data Fundamentals: Explore roles and responsibilities in the world of data

I. Explore job roles in the world of data

There are three key job roles that deal with data in most organizations:

Database Administrators manage databases, assigning permissions to users, storing backup copies of data and restore data in case of any failures.

An Azure database administrator is responsible for the design, implementation, maintenance, and operational aspects of on-premises and cloud-based database solutions built on Azure data services and SQL Server. They're responsible for the overall availability and consistent performance and optimizations of the database solutions. They work with stakeholders to implement policies, tools, and processes for backup and recovery plans to recover following a natural disaster or human-made error.

The database administrator is also responsible for managing the security of the data in the database, granting privileges over the data, granting or denying access to users as appropriate.

Data Engineers are vital in working with data, applying data cleaning routines, identifying business rules, and turning data into useful information.

A data engineer collaborates with stakeholders to design and implement data-related assets that include data ingestion pipelines, cleansing and transformation activities, and data stores for analytical workloads. They use a wide range of data platform technologies, including relational and nonrelational databases, file stores, and data streams.

They're also responsible for ensuring that the privacy of data is maintained within the cloud and spanning from on-premises to the cloud data stores. They also own the management and monitoring of data stores and data pipelines to ensure that data loads perform as expected.

Data Analysts explore and analyze data to create visualizations and charts to enable organizations to make informed decisions.

A data analyst enables businesses to maximize the value of their data assets. They're responsible for designing and building scalable models, cleaning and transforming data, and enabling advanced analytics capabilities through reports and visualizations.

A data analyst processes raw data into relevant insights based on identified business requirements to deliver relevant insights.

II. Review tasks and tools for database administration

A database administrator's primary job is to ensure that data is available, protected from loss, corruption, or theft, and is easily accessible as needed.

Database Administrator tasks and responsibilities

Some of the most common roles and responsibilities of a database administrator include:

- Installing and upgrading the database server and application tools.
- Allocating system storage and planning storage requirements for the database system.
- Modifying the database structure, as necessary, from information given by application developers.
- Enrolling users and maintaining system security.
- Ensuring compliance with database vendor license agreement.
- Controlling and monitoring user access to the database.
- Monitoring and optimizing the performance of the database.
- Planning for backup and recovery of database information.
- Maintaining archived data.
- Backing up and restoring databases.
- Contacting database vendor for technical support.
- Generating various reports by querying from database as per need.
- Managing and monitoring data replication.

Common database administrator tools

Most database management systems provide their own set of tools to assist with database administration. For example, SQL Server Database Administrators use SQL Server Management Studio for most of their day-to-day database maintenance activities. Other systems have their own database-specific interfaces, such as pgAdmin for PostgreSQL systems, or MySQL Workbench for MySQL. There are also a number of cross-platform database administration tools available. One example is Azure Data Studio.

What is Azure Data Studio?

Azure Data Studio provides a graphical user interface for managing many different database systems. It currently provides connections to on-premises SQL Server databases, Azure SQL Database, PostgreSQL, Azure SQL Data Warehouse, and SQL Server Big Data Clusters, amongst others. It's an extensible tool, and you can download and install extensions from third-party developers that connect to other systems, or provide wizards that help to automate many administrative tasks.

What is SQL Server Management Studio?

SQL Server Management Studio provides a graphical interface, enabling you to query data, perform general database administration tasks, and generate scripts for automating database maintenance and support operations. The example below shows SQL Server Management Studio being used to back up a database.

A useful feature of SQL Server Management Studio is the ability to generate Transact-SQL scripts for almost all of the functionality that SSMS provides. This gives the DBA the ability to schedule and automate many common tasks.

Use the Azure portal to manage Azure SQL Database

Azure SQL database provides database services in Azure. It's similar to SQL Server, except that it runs in the cloud. You can manage Azure SQL database using Azure portal.

Typical configuration tasks such as increasing the database size, creating a new database, and deleting an existing database are done using the Azure portal.

You can use the Azure portal to dynamically manage and adjust resources such as the data storage size and the number of cores available for the database processing. These tasks would require the support of a system administrator if you were running the database on-premises.

III. Review tasks and tools for data engineering

Data engineers are tasked with managing and organizing data, while also monitoring for trends or inconsistencies that will impact business goals. It's a highly technical position, requiring experience and skills in areas like programming, mathematics, and computer science. But data engineers also need soft skills to communicate data trends to others in the organization and to help the business make use of the data it collects.

Data Engineer tasks and responsibilities

Some of the most common roles and responsibilities of a data engineer include:

- Developing, constructing, testing, and maintaining databases and data structures.
- Aligning the data architecture with business requirements.
- Data acquisition.
- Developing processes for creating and retrieving information from data sets.
- Using programming languages and tools to examine the data.
- Identifying ways to improve data reliability, efficiency, and quality.
- Conducting research for industry and business questions.
- Deploying sophisticated analytics programs, machine learning, and statistical methods.
- Preparing data for predictive and prescriptive modeling.
- Using data to discover tasks that can be automated.

Common data engineering tools

To master data engineering, you'll need to be familiar with a range of tools that enable you to create well-designed databases, optimized for the business processes that will be run. You must have a thorough understanding of the architecture of the database management system, the platform on which the system runs, and the business requirements for the data being stored in the database.

If you're using a relational database management system, you need to be fluent in SQL. You must be able to use SQL to create databases, tables, indexes, views, and the other objects required by the database. Many database management systems provide tools that enable you to create and run SQL scripts. For example, SQL Server Management Studio (described in the previous unit), lets you create and query tables visually, but you can also create your own SQL scripts manually.

In some cases, you may need to interact with a database from the command line. Many database management systems provide a command-line interface that supports these operations. For example, you can use the sqlcmd utility to connect to Microsoft SQL Server and Azure SQL Database, and run ad-hoc queries and commands.

As a SQL Server professional, your primary data manipulation tool might be Transact-SQL. As a data engineer you might use additional technologies, such as Azure Databricks, and Azure HDInsight to generate and test predictive models. If you're working in the non-relational field, you might use Azure Cosmos DB as your primary data store. To manipulate and query the data, you might use languages such as HiveQL, R, or Python.

IV. Review tasks and tools for data visualization and reporting

Data analysts are responsible for understanding what data actually means. A skilled data analyst will explore the data and use it to determine trends, issues, and gain other insights that might be of benefit to the company.

A large part of the data analyst role is concerned with communication and visualization. Data visualization is key to presenting large amounts of information in ways that are universally understandable or easy to interpret and spot patterns, trends, and correlations. These representations include charts, graphs, infographics, and other pictorial diagrams. Data visualization analysts use visualization tools and software to communicate information in these ways, for clients or for their own company. A good data analyst requires experience and skills in reporting tools such as Microsoft Power BI and SQL Server Reporting Services.

Data Analyst tasks and responsibilities
The primary functions of a data analyst usually include the following:

- Making large or complex data more accessible, understandable, and usable.
- Creating charts and graphs, histograms, geographical maps, and other visual models that help to explain the meaning of large volumes of data, and isolate areas of interest.
- Transforming, improving, and integrating data from many sources, depending on the business requirements.
- Combining the data result sets across multiple sources. For example, combining sales data and weather data provides a useful insight into how weather influenced sales of certain products such as ice creams.
- Finding hidden patterns using data.

Common data visualization tools

Traditionally, many data analysts used Microsoft Office Apps such as Microsoft Excel for creating rich visual reports. Many analysts now use Microsoft Power BI, a powerful visualization platform, to create rich, graphical dashboards and reports over data that can vary dynamically.

Power BI is a collection of software services, apps, and connectors that work together to turn your unrelated sources of data into coherent, visually immersive, and interactive insights. Your data might be held somewhere local such as an Excel spreadsheet, or in a collection of cloud-based and on-premises databases, or some other set of data sources. Power BI lets you easily connect to your data sources, discover what's important in that data, and share your findings with others in the organization.

Azure Data Fundamentals: Describe concepts of relational data

I. Explore the characteristics of relational data

In the early years of databases, every application stored data in its own unique structure. When developers wanted to build applications to use that data, they had to know a lot about the particular data structure to find the data they needed. These data structures were inefficient, hard to maintain, and hard to optimize for delivering good application performance. The relational database model was designed to solve the problem of multiple arbitrary data structures. The relational model provided a standard way of representing and querying data that could be used by any application. From the beginning, developers recognized that the chief strength of the relational database model was in its use of tables, which were an intuitive, efficient, and flexible way to store and access structured information.

The simple yet powerful relational model is used by organizations of all types and sizes for a broad variety of information management needs. Relational databases are used to track inventories, process ecommerce transactions, manage huge amounts of mission-critical customer information, and much more. A relational database is useful for storing any information containing related data elements that must be organized in a rules-based, consistent way.

Understand the characteristics of relational data

In a relational database, you model collections of entities from the real world as tables. An entity is described as a thing about which information needs to be known or held. In the ecommerce example, you might create tables for customers, products, and orders. A table contains rows, and each row represents a single instance of an entity. In the ecommerce scenario, each row in the customers table contains the data for a single customer, each row in the products table defines a single product, and each row in the orders table represents an order made by a customer.

The rows in a table have one or more columns that define the properties of the entity, such as the customer name, or product ID. All rows in the same table have the same columns. Some columns are used to maintain relationships between tables. This is where the relational model gets its name from.

You design a relational database by creating a data model. The model below shows the structure of the entities from the previous example. In this diagram, the columns marked PK are the Primary Key for the table. The primary key indicates the column (or combination of columns) that uniquely identify each row. Every table should have a primary key.

The diagram also shows the relationships between the tables. The lines connecting the tables indicate the type of relationship. In this case, the relationship from customers to orders is 1-to-many (one customer can place many orders, but each order is for a single customer). Similarly, the relationship between orders and products is many-to-1 (several orders might be for the same product).

The columns marked FK are Foreign Key columns. They reference, or link to, the primary key of another table, and are used to maintain the relationships between tables. A foreign key also helps to identify and prevent anomalies, such as orders for customers that don't exist in the Customers table. (Referential Integrity) In the model below, the Customer ID and Product ID columns in the Orders table link to the customer that placed the order and the product that was ordered:

The main characteristics of a relational database are:

- All data is tabular. Entities are modeled as tables, each instance of an entity is a row in the table, and each property is defined as a column.
- All rows in the same table have the same set of columns.
- A table can contain any number of rows.
- A primary key uniquely identifies each row in a table. No two rows can share the same primary key.
- A foreign key references rows in another, related table. For each value in the foreign key column, there should be a row with the same value in the corresponding primary key column in the other table.

Most relational databases support Structured Query Language (SQL). You use SQL to create tables, insert, update, and delete rows in tables, and to query data. You use the CREATE TABLE command to create a table, the INSERT statement to store data in a table, the UPDATE statement to modify data in a table, and the DELETE statement to remove rows from a table. The SELECT statement retrieves data from a table. The example query below finds the details of every customer from the sample database shown above.

Rather than retrieve every row, you can filter data by using a WHERE clause. The next query fetches the order ID and product ID for all orders placed by customer 1.

You can combine the data from multiple tables in a query using a join operation. A join operation spans the relationships between tables, enabling you to retrieve the data from more than one table at a time. The following query retrieves the name of every customer, together with the product name and quantity for every order they've placed. Notice that each column is qualified with the table it belongs to:

Explore relational database use cases

You can use a relational database any time you can easily model your data as a collection of tables with a fixed set of columns. In theory, you could model almost any dataset in this way, but some scenarios lend themselves to the relational model better than others.

For example, if you have a collection of music, video, or other media files, attempting to force this data into the relational model could be difficult. You may be better off using unstructured storage, such as that available in Azure Blob storage. Similarly, social networking sites use databases to store data about millions of users, each of whom can be linked to any number of other users in a highly complex web of relationships. This type of data lends itself more to a graph database structure rather than a collection of relational tables.

Relational databases are commonly used in ecommerce systems, but one of the major use cases for using relational databases is Online Transaction Processing (OLTP). OLTP applications are focused on transaction-oriented tasks that process a very large number of transactions per minute. Relational databases are well suited for OLTP applications because they naturally support insert, update, and delete operations. A relational database can often be tuned to make these operations fast. Also, the nature of SQL makes it easy for users to perform ad-hoc queries over data.

Examples of OLTP applications that use relational databases are:

- Banking solutions
- Online retail applications
- Flight reservation systems
- Many online purchasing applications.

II. Explore relational data structures

What is an index?

An index helps you search for data in a table. Think of an index over a table like an index at the back of a book. A book index contains a sorted set of references, with the pages on which each reference occurs. When you want to find a reference to an item in the book, you look it up through the index. You can use the page numbers in the index to go directly to the correct pages in the book. Without an index, you might have to read through the entire book to find the references you're looking for.

When you create an index in a database, you specify a column from the table, and the index contains a copy of this data in a sorted order, with pointers to the corresponding rows in the table. When the user runs a query that specifies this column in the WHERE clause, the database management system can use this index to fetch the data more quickly than if it had to scan through the entire table row by row. In the example below, the query retrieves all orders for customer C1. The Orders table has an index on the Customer ID column. The database management system can consult the index to quickly find all matching rows in the Orders table.

You can create many indexes on a table. So, if you also wanted to find all orders for a specific product, then creating another index on the Product ID column in the Orders table, would be useful. However, indexes aren't free. An index might consume additional storage space, and each time you insert, update, or delete data in a table, the indexes for that table must be maintained. This additional work can slow down insert, update, and delete operations, and incur additional processing charges. Therefore, when deciding which indexes to create, you must strike a balance between having indexes that speed up your queries versus the cost of performing other operations. In a table that is read only, or that contains data that is modified infrequently, more indexes will improve query performance. If a table is queried infrequently, but subject to a large number of inserts, updates, and deletes (such as a table involved in OLTP), then creating indexes on that table can slow your system down.

Some relational database management systems also support clustered indexes. A clustered index physically reorganizes a table by the index key. This arrangement can improve the performance of queries still further, because the relational database management system doesn't have to follow references from the index to find the corresponding data in the underlying table. The image below shows the Orders table with a clustered index on the Customer ID column.

What is a view?

A view is a virtual table based on the result set of a query. In the simplest case, you can think of a view as a window on specified rows in an underlying table.

You can query the view and filter the data in much the same way as a table.

A view can also join tables together. If you regularly needed to find the details of customers and the products that they've ordered, you could create a view based on the join query shown in the previous unit:

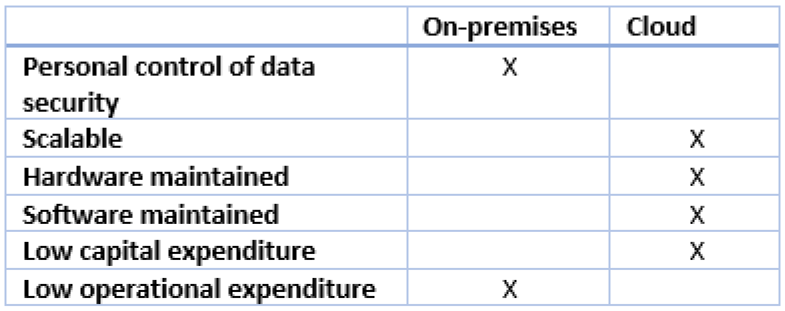
III. Explore relational data workload offerings

Compare on-premises hosting to the cloud

Whether a company places its relational workload in the cloud or whether it decides to keep it on premises, data security will always be paramount. But for those businesses in highly regulated industries, the decision might already be made for them as to whether to host their applications on-premises. Knowing that your data is located within your in-house servers and IT infrastructure might also provide more peace of mind.

Hosting a relational database on-premises requires that an enterprise not only purchases the database software, but also maintains the necessary hardware on which to run the database. The organization is responsible for maintaining the hardware and software, applying patches, backing up databases, restoring them when necessary, and generally performing all the day-to-day management required to keep the platform operational. Scalability is also a concern. If you need to scale your system, you will need to upgrade or add more servers. You then need to expand your database onto these servers. This can be a formidable task that requires you to take a database offline while the operation is performed. In the cloud, many of these operations can be handled for you by the data center staff, in many cases with no (or minimal) downtime. You're free to focus on the data itself and leave the management concerns to others (this is what you pay your Azure fees for, after all).

A cloud-based approach uses virtual technology to host a company's applications offsite. There are no capital expenses, data can be backed up regularly, and companies only have to pay for the resources they use. For those organizations that plan aggressive expansion on a global basis, the cloud has even greater appeal because it allows you to connect with customers, partners, and other businesses anywhere with minimal effort. Additionally, cloud computing gives you nearly instant provisioning because everything is already configured. Thus, any new software that is integrated into your environment is ready to use immediately once a company has subscribed. With instant provisioning, any time spent on installation and configuration is eliminated and users can access the application right away.



	On-premises	Cloud
Personal control of data security	X	
Scalable		X
Hardware maintained		X
Software maintained		X
Low capital expenditure		X
Low operational expenditure	X	

Understand IaaS and PaaS

You generally have two options when moving your operations and databases to the cloud. You can select an IaaS approach, or PaaS.

IaaS is an acronym for Infrastructure-as-a-Service. Azure enables you to create a virtual infrastructure in the cloud that mirrors the way an on-premises data center might work. You can create a set of virtual machines, connect them together using a virtual network, and add a range of virtual devices. In many ways, this approach is similar to the way in which you run your systems inside an organization, except that you don't have to concern yourself with buying or maintaining the hardware. However, you're still responsible for many of the day-to-day operations, such as installing and configuring the software, patching, taking backups, and restoring data when needed. You can think of IaaS as a half-way-house to fully managed operations in the cloud; you don't have to worry about the hardware, but running and managing the software is still very much your responsibility.

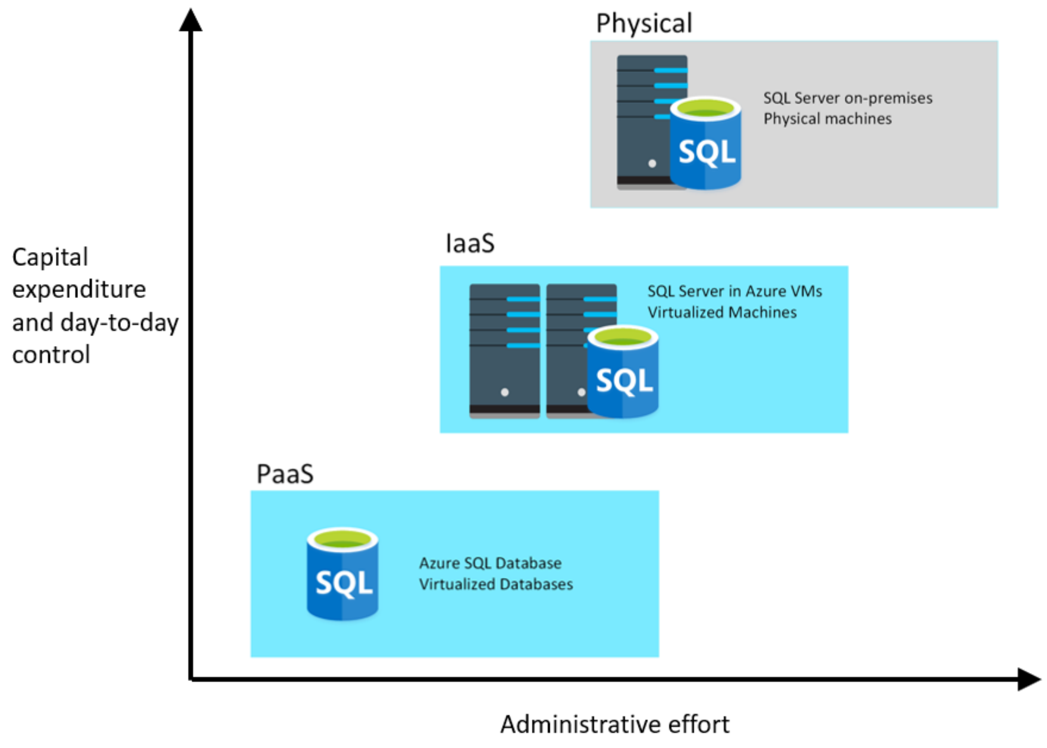
You can run any software for which you have the appropriate licenses using this approach. You're not restricted to any specific database management system.

The IaaS approach is best for migrations and applications requiring operating system-level access. SQL virtual machines are lift-and-shift. That is, you can copy your on-premises solution directly to a virtual machine in the cloud. The system should work more or less exactly as before in its new location, except for some small configuration changes (changes in network addresses, for example) to take account of the change in environment.

PaaS stands for Platform-as-a-service. Rather than creating a virtual infrastructure, and installing and managing the database software yourself, a PaaS solution does this for you. You specify the resources that you require (based on how large you think your databases will be, the number of users, and the performance you require), and Azure automatically creates the necessary virtual machines, networks, and other devices for you. You can usually scale up or down (increase or decrease the size and number of resources) quickly, as the volume of data and the amount of work being done varies; Azure handles this scaling for you, and you don't have to manually add or remove virtual machines, or perform any other form of configuration.

Azure offers several PaaS solutions for relational databases, include Azure SQL Database, Azure Database for PostgreSQL, Azure Database for MySQL, and Azure Database for MariaDB. These services run managed versions of the database management systems on your behalf. You just connect to them, create your databases, and upload your data. However, you may find that there are some functional restrictions in place, and not every feature of your selected database management system may be available. These restrictions are often due to security issues. For example, they might expose the underlying operating system and hardware to your applications. In these cases, you may need to rework your applications to remove any dependencies on these features.

The image below illustrates the benefits and tradeoffs when running a database management system (in this case, SQL Server) on-premises, using virtual machines in Azure (IaaS), or using Azure SQL Database (PaaS). The same generalized considerations are true for other database management systems.



I. Explore the characteristics of non-relational data

What are the characteristics of non-relational data?

You use a database to model some aspect of the real-world. Entities in the real-world often have highly variable structures. For example, in an ecommerce database that stores information about customers, how many telephone numbers does a customer have? A customer might have a landline and a mobile number, but some customers might have a business number, an additional home number, and maybe several mobile numbers. Similarly, the addresses of customers might not always follow the same format; addresses for customers in different states and regions might contain different elements, such as zip codes or postal codes.

In another scenario, if you are ingesting data rapidly, you want to capture the data and save it very quickly. Processing the data and manipulating it into a set of rows in different tables in a relational database might not be appropriate at this point; you can perform these tasks at a later date. At the time of ingestion, you simply need to store the data in its original state and format.

A key aspect of non-relational databases is that they enable you to store data in a very flexible manner. Non-relational databases don't impose a schema on data. Instead, they focus on the data itself rather than how to structure it. This approach means that you can store information in a natural format, that mirrors the way in which you would consume, query and use it.

In a non-relational system, you store the information for entities in collections or containers rather than relational tables. Two entities in the same collection can have a different set of fields rather than a regular set of columns found in a relational table. The lack of a fixed schema means that each entity must be self-describing. Often this is achieved by labeling each field with the name of the data that it represents. For example, a non-relational collection of customer entities might look like this:

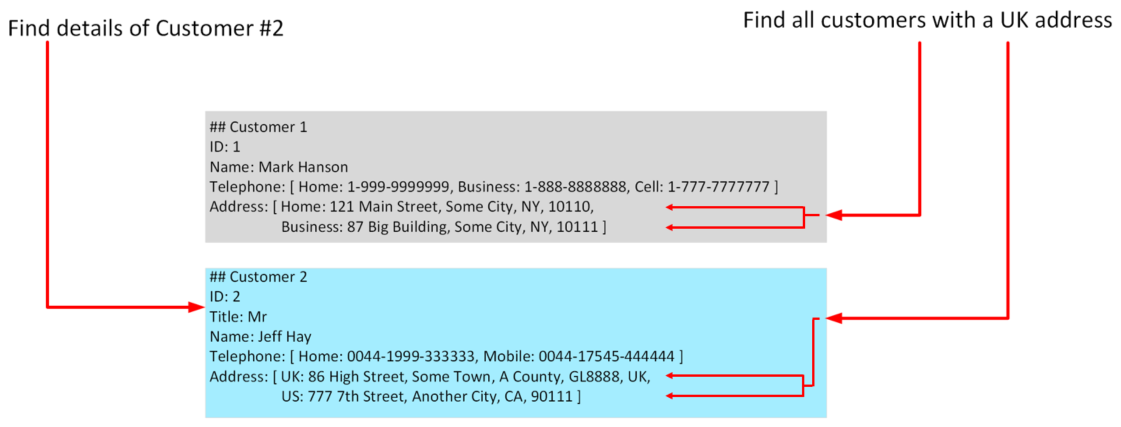
```
## Customer 1
ID: 1
Name: Mark Hanson
Telephone: [ Home: 1-999-9999999,  Business: 1-888-8888888,  Cell: 1-777-7777777  ]
Address: [ Home: 121 Main Street, Some City, NY, 10110,
        Business: 87 Big Building, Some City, NY, 10111 ]

## Customer 2
ID: 2
Title: Mr
Name: Jeff Hay
Telephone: [ Home: 0044-1999-333333,   Mobile: 0044-17545-444444  ]
Address: [ UK: 86 High Street, Some Town, A County, GL8888, UK,
        US: 777 7th Street, Another City, CA, 90111 ]
```

In this example, fields are prefixed with a name. Fields might also have multiple subfields, also with names. In the example, multiple subfields are denoted by enclosing them between square brackets.

Adding a new customer is a matter of inserting an entity with its fields labeled in a meaningful way. An application that queries this data must be prepared to parse the information in the entity that it retrieves.

The data retrieval capabilities of a non-relational database can vary. Each entity should have a unique key value. The entities in a collection are usually stored in key-value order. In the example above, the unique key is the ID field. The simplest type of non-relational database enables an application to either specify the unique key, or a range of keys as query criteria. In the customers example, the database would enable an application to query customers by ID only. Filtering data on other fields would require scanning the entire collection of entities, parsing each entity in turn, and then applying any query criteria to each entity to find any matches. In the example below, a query that fetches the details of a customer by ID can quickly identify which entity to retrieve. A query that attempts to find all customers with a UK address would have to iterate through every entity, and for each entity examine each field in turn. If the database contains many millions of entities, this query could take a considerable time to run.


Find details of Customer #2
Find all customers with a UK address

```
## Customer 1
ID: 1
Name: Mark Hanson
Telephone: [ Home: 1-999-9999999, Business: 1-888-8888888, Cell: 1-777-7777777 ]
Address: [ Home: 121 Main Street, Some City, NY, 10110,
           Business: 87 Big Building, Some City, NY, 10111 ]

## Customer 2
ID: 2
Title: Mr
Name: Jeff Hay
Telephone: [ Home: 0044-1999-333333, Mobile: 0044-17545-444444 ]
Address: [ UK: 86 High Street, Some Town, A County, GL8888, UK,
           US: 777 7th Street, Another City, CA, 90111 ]
```

More advanced non-relational systems support indexing, in a similar manner to an index in a relational database. Queries can then use the index to identify and fetch data based on non-key fields. Non-relational systems such as Azure Cosmos DB (a non-relational database management system available in Azure), support indexing even when the structure of the indexed data can vary from record to record. For more information, read Indexing in Azure Cosmos DB - Overview.

When you design a non-relational database, it's important to understand the capabilities of the database management system and the types of query it will have to support.

Identify non-relational database use cases

Non-relational databases are highly suitable for the following scenarios:

- IoT and telematics. These systems typically ingest large amounts of data in frequent bursts of activity. Non-relational databases can store this information very quickly. The data can then be used by analytics services such as Azure Machine Learning, Azure HDInsight, and Microsoft Power BI. Additionally, you can process the data in real-time using Azure Functions that are triggered as data arrives in the database.

- Retail and marketing. Microsoft uses CosmosDB for its own ecommerce platforms that run as part of Windows Store and Xbox Live. It's also used in the retail industry for storing catalog data and for event sourcing in order processing pipelines.

- Gaming. The database tier is a crucial component of gaming applications. Modern games perform graphical processing on mobile/console clients, but rely on the cloud to deliver customized and personalized content like in-game stats, social media integration, and high-score leaderboards. Games often require single-millisecond latencies for reads and write to provide an engaging in-game experience. A game database needs to be fast and be able to handle massive spikes in request rates during new game launches and feature updates.

- Web and mobile applications. A non-relational database such as Azure Cosmos DB is commonly used within web and mobile applications, and is well suited for modeling social interactions, integrating with third-party services, and for building rich personalized experiences. The Cosmos DB SDKs (software development kits) can be used to build rich iOS and Android applications using the popular Xamarin framework.

A relational database restructures the data into a fixed format that is designed to answer specific queries. When data needs to be ingested very quickly, or the query is unknown and unconstrained, a relational database can be less suitable than a non-relational database.

II. Describe types of non-relational data

What is semi-structured data?

Semi-structured data is data that contains fields. The fields don't have to be the same in every entity. You only define the fields that you need on a per-entity basis. The Customer entities shown in the previous unit are examples of semi-structured data. The data must be formatted in such a way that an application can parse and process it. One common way of doing this is to store the data for each entity as a JSON document. The term JSON stands for JavaScript Object Notation; it's the format used by JavaScript applications to store data in memory, but can also be used to read and write documents to and from files.

A JSON document is enclosed in curly brackets ({ and }). Each field has a name (a label), followed by a colon, and then the value of the field. Fields can contain simple values, or subdocuments (each starting and ending with curly brackets). Fields can also have multiple values, held as arrays and surrounded with square brackets ([and]). Literals, or fixed values, in a field are enclosed in quotes, and fields are separated with commas.

The example below shows the customers from the previous unit, formatted as JSON documents:

JSON

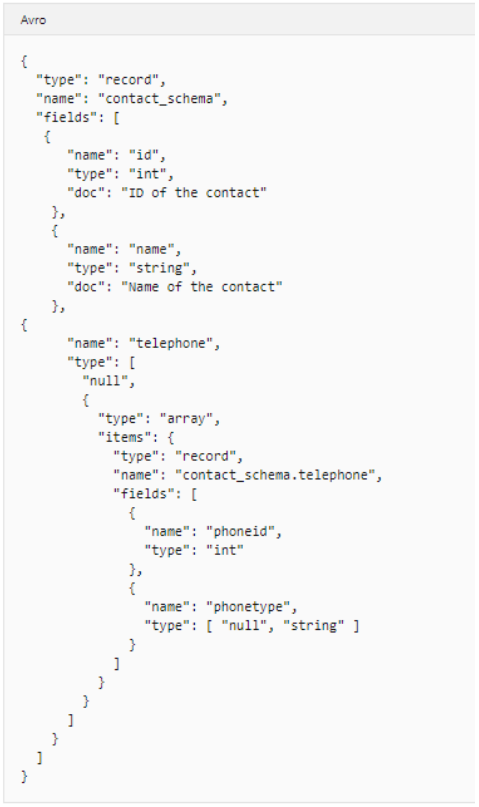
```json
{
  "ID": "1",
  "Name": "Mark Hanson",
  "Telephone": [
    { "Home": "1-999-9999999" },
    { "Business": "1-888-8888888" },
    { "Cell": "1-777-7777777" }
  ],
  "Address": [
    { "Home": [
      { "StreetAddress": "121 Main Street" },
      { "City": "Some City" },
      { "State": "NY" },
      { "Zip": "10110" }
    ] },
    { "Business": [
      { "StreetAddress": "87 Big Building" },
      { "City": "Some City" },
      { "State": "NY" },
      { "Zip": "10111" }
    ] }
  ]
}


{
  "ID": "2",
  "Title": "Mr",
  "Name": "Jeff Hay",
  "Telephone": [
    { "Home": "0044-1999-333333" },
    { "Mobile": "0044-17545-444444" }
  ],
  "Address": [
    { "UK": [
      { "StreetAddress": "86 High Street" },
      { "Town": "Some Town" },
      { "County": "A County" },
      { "Postcode": "GL8888" },
      { "Region": "UK" }
    ] },
    { "US": [
      { "StreetAddress": "777 7th Street" },
      { "City": "Another City" },
      { "State": "CA" },
      { "Zip": "90111" }
    ] }
  ]
}
```

You're free to define whatever fields you like. The important point is that the data follows the JSON grammar. When an application reads a document, it can use a JSON parser to break up the document into its component fields and extract the individual pieces of data.

Other formats you might see include Avro, ORC, and Parquet:

- Avro is a row-based format. It was created by Apache. Each record contains a header that describes the structure of the data in the record. This header is stored as JSON. The data is stored as binary information. An application uses the information in the header to parse the binary data and extract the fields it contains. Avro is a very good format for compressing data and minimizing storage and network bandwidth requirements. This example is a subset of the header information for the previous example, formatted as Avro:



Avro

```
{
  "type": "record",
  "name": "contact_schema",
  "fields": [
   {
      "name": "id",
      "type": "int",
      "doc": "ID of the contact"
    },
    {
      "name": "name",
      "type": "string",
      "doc": "Name of the contact"
    },
{
      "name": "telephone",
      "type": [
        "null",
        {
          "type": "array",
          "items": {
            "type": "record",
            "name": "contact_schema.telephone",
            "fields": [
              {
                "name": "phoneid",
                "type": "int"
              },
              {
                "name": "phonetype",
                "type": [ "null", "string" ]
              }
            ]
          }
        }
      ]
    }
  ]
}
```

- ORC (Optimized Row Columnar format) organizes data into columns rather than rows. It was developed by HortonWorks for optimizing read and write operations in Apache Hive. Hive is a data warehouse system that supports fast data summarization and querying over very large datasets. Hive supports SQL-like queries over unstructured data. An ORC file contains stripes of data. Each stripe holds the data for a column or set of columns. A stripe contains an index into the rows in the stripe, the data for each row, and a footer that holds statistical information (count, sum, max, min, and so on) for each column.
- Parquet is another columnar data format. It was created by Cloudera and Twitter. A Parquet file contains row groups. Data for each column is stored together in the same row group. Each row group contains one or more chunks of data. A Parquet file includes metadata that describes the set of rows found in each chunk. An application can use this metadata to quickly locate the correct chunk for a given set of rows, and retrieve the

data in the specified columns for these rows. Parquet specializes in storing and processing nested data types efficiently. It supports very efficient compression and encoding schemes.

What is unstructured data?

Unstructured data is data that doesn't naturally contain fields. Examples include video, audio, and other media streams. Each item is an amorphous blob of binary data. You can't search for specific elements in this data.

You might choose to store data such as this in storage that is specifically designed for the purpose. In Azure, you would probably store video and audio data as block blobs in an Azure Storage account. (The term blob stands for Binary Large Object*). A block blob only supports basic read and write operations.

You could also consider files as a form of unstructured data, although in some cases a file might include metadata that indicates what type of file it is (photograph, Word document, Excel spreadsheet, and so on), owner, and other elements that could be stored as fields. However, the main content of the file is unstructured.

III. Describe types of non-relational and NoSQL databases

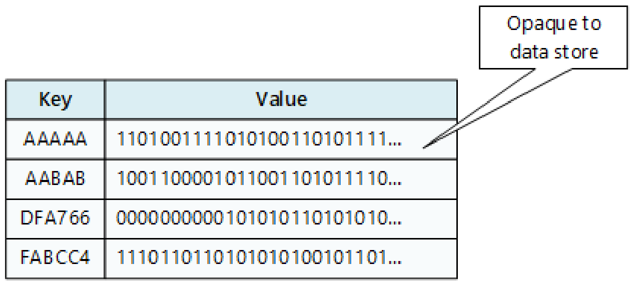
What is NoSQL?

You might see the term NoSQL when reading about non-relational databases. NoSQL is a rather loose term that simply means non-relational. There's some debate about whether it's intended to imply Not SQL, or Not Only SQL; some non-relational databases support a version of SQL adapted for documents rather than tables (examples include Azure Cosmos DB).

NoSQL (non-relational) databases generally fall into four categories: key-value stores, document databases, column family databases, and graph databases. The following sections discuss these types of NoSQL databases.

What is a key-value store?

A key-value store is the simplest (and often quickest) type of NoSQL database for inserting and querying data. Each data item in a key-value store has two elements, a key and a value. The key uniquely identifies the item, and the value holds the data for the item. The value is opaque to the database management system. Items are stored in key order.


Opaque to data store

Key	Value
AAAAA	110100111101010011010111 1...
AABAB	1001100001011001101011110...
DFA766	0000000000101010110101010...
FABCC4	1110110110101010100101101...

A query specifies the keys to identify the items to be retrieved. You can't search on values. An application that retrieves data from a key-value store is responsible for parsing the contents of the values returned.

Write operations are restricted to inserts and deletes. If you need to update an item, you must retrieve the item, modify it in memory (in the application), and then write it back to the database, overwriting the original (effectively a delete and an insert).

The focus of a key-value store is the ability to read and write data very quickly. Search capabilities are secondary. A key-value store is an excellent choice for data ingestion, when a large volume of data arrives as a continual stream and must be stored immediately.

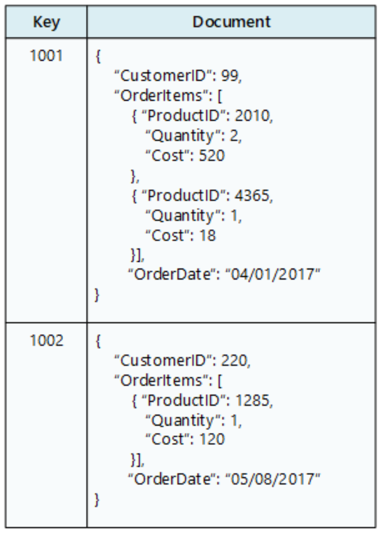
Azure Table storage is an example of a key-value store. Cosmos DB also implements a key-value store using the Table API.

What is a document database?

A document database represents the opposite end of the NoSQL spectrum from a key-value store. In a document database, each document has a unique ID, but the fields in the documents are transparent to the database management system. Document databases typically store data in JSON format, as described in the previous unit, or they could be encoded using other formats such as XML, YAML, JSON, BSON. Documents could even be stored as plain text. The fields in documents are exposed to the storage management system, enabling an application to query and filter data by using the values in these fields.

Typically, a document contains the entire data for an entity. What items constitute an entity are application-specific. For example, an entity could contain the details of a customer, an order, or a combination of both. A single document may contain information that would be spread across several relational tables in an RDBMS (relational database management system).

A document store does not require that all documents have the same structure. This free-form approach provides a great deal of flexibility. Applications can store different data in documents as business requirements change.



Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

An application can retrieve documents by using the document key. The key is a unique identifier for the document. Some document databases create the document key automatically. Others enable you to specify an attribute of the document to use as the key. The application can also query documents based on the value of one or more fields. Some document databases support indexing to facilitate fast lookup of documents based on one or more indexed fields.

Some document database management systems support in-place updates, enabling an application to modify the values of specific fields in a document without rewriting the entire document. Other document database management systems (such as Cosmos DB) can only read and write entire documents. In these cases, an update replaces the entire document with a new version. This approach helps to reduce fragmentation in the database, which can, in turn, improve performance.

Most document databases will ingest large volumes of data more rapidly than a relational database, but aren't as optimal as a key-value store for this type of processing. The focus of a document database is its query capabilities.

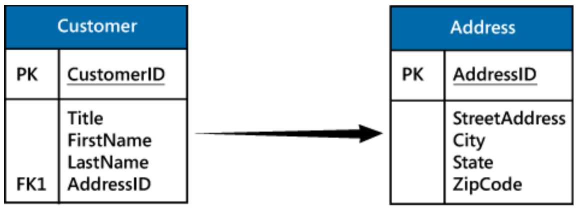
Azure Cosmos DB implements a document database approach in its Core (SQL) API.

What is a column family database?

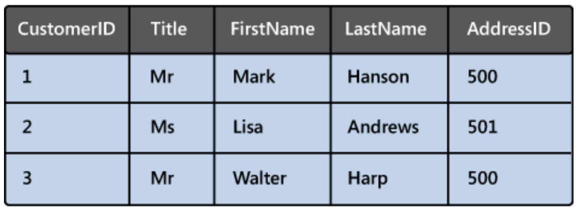
A column family database organizes data into rows and columns. Examples of this structure include ORC and Parquet files, described in the previous unit.

In its simplest form, a column family database can appear very similar to a relational database, at least conceptually. The real power of a column family database lies in its denormalized approach to structuring sparse data.

For example, if you need to store information about customers and their addresses in a relational database (ignoring the need to maintain historical data as described in the previous section), you might design a schema similar to that shown below. This diagram also shows some sample data. In this example, customer 1 and customer 3 share the same address, and the schema ensures that this address information is not duplicated. This is a standard way of implementing a one-to-many relationship.

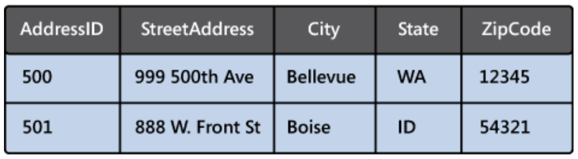


Customer Table



CustomerID	Title	FirstName	LastName	AddressID
1	Mr	Mark	Hanson	500
2	Ms	Lisa	Andrews	501
3	Mr	Walter	Harp	500

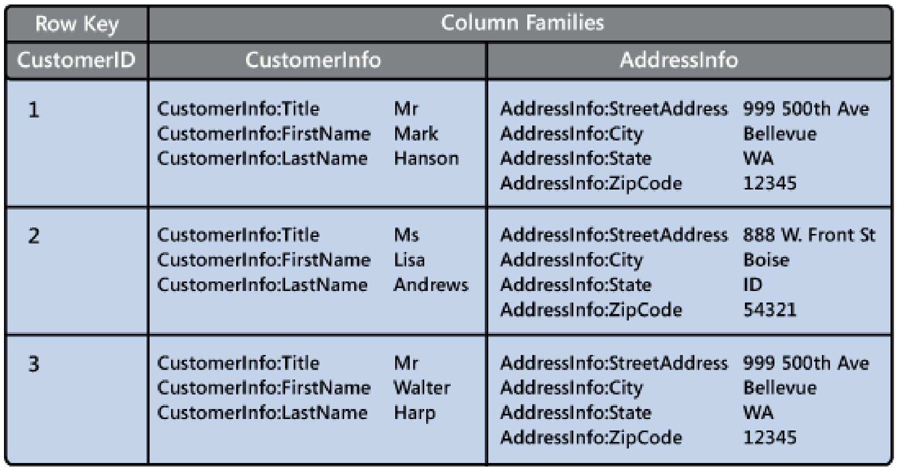
Address Table



AddressID	StreetAddress	City	State	ZipCode
500	999 500th Ave	Bellevue	WA	12345
501	888 W. Front St	Boise	ID	54321

The relational model supports a very generalized approach to implementing this type of relationship, but to find the address of any given customer an application needs to run a query that joins two tables. If this is the most common query performed by the application, then the overhead associated with performing this join operation can quickly become significant if there are a large number of requests and the tables themselves are large.

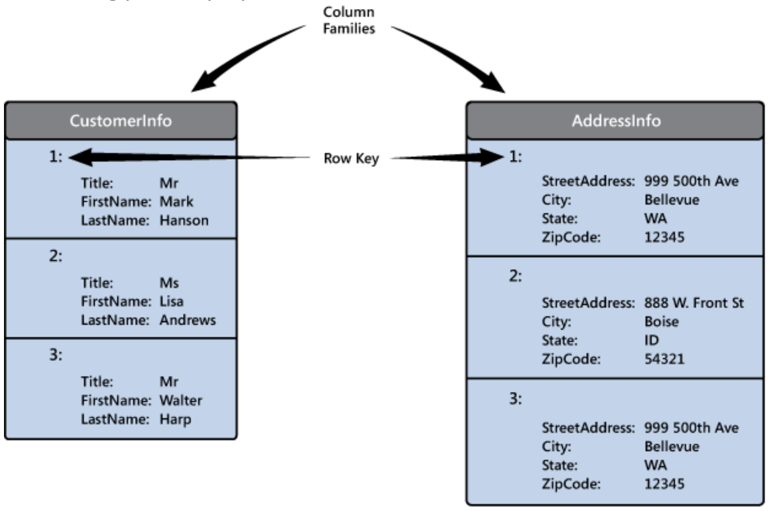
The purpose of a column family database is to efficiently handle situations such as this. You can think of a column family database as holding tabular data comprising rows and columns, but you can divide the columns into groups known as column-families. Each column family holds a set of columns that are logically related together. The image below shows one way of structuring the same information as the previous image, by using a column family database to group the data into two column-families holding the customer name and address information. Other ways of organizing the columns are possible, but you should implement your column-families to optimize the most common queries that your application performs. In this case, queries that retrieve the addresses of customers can fetch the data with fewer reads than would be required in the corresponding relational database; these queries can fetch the data directly from the AddressInfo column family.



Row Key	Column Families	
CustomerID	CustomerInfo	AddressInfo
1	CustomerInfo:Title Mr CustomerInfo:FirstName Mark CustomerInfo:LastName Hanson	AddressInfo:StreetAddress 999 500th Ave AddressInfo:City Bellevue AddressInfo:State WA AddressInfo:ZipCode 12345
2	CustomerInfo:Title Ms CustomerInfo:FirstName Lisa CustomerInfo:LastName Andrews	AddressInfo:StreetAddress 888 W. Front St AddressInfo:City Boise AddressInfo:State ID AddressInfo:ZipCode 54321
3	CustomerInfo:Title Mr CustomerInfo:FirstName Walter CustomerInfo:LastName Harp	AddressInfo:StreetAddress 999 500th Ave AddressInfo:City Bellevue AddressInfo:State WA AddressInfo:ZipCode 12345

The illustration above is conceptual rather than physical, and is intended to show the logical structure of the data rather than how it might be physically organized. Each row in a column family database contains a key, and you can fetch the data for a row by using this key.

In most column family databases, the column-families are stored separately. In the previous example, the CustomerInfo column family might be held in one area of physical storage and the AddressInfo column family in another, in a simple form of vertical partitioning. You should really think of the structure in terms of column-families rather than rows. The data for a single entity that spans multiple column-families will have the same row key in each column family. As an alternative to the conceptual layout shown previously, you can visualize the data shown as the following pair of physical structures.



CustomerInfo

1:
Title: Mr
FirstName: Mark
LastName: Hanson

2:
Title: Ms
FirstName: Lisa
LastName: Andrews

3:
Title: Mr
FirstName: Walter
LastName: Harp

AddressInfo

1:
StreetAddress: 999 500th Ave
City: Bellevue
State: WA
ZipCode: 12345

2:
StreetAddress: 888 W. Front St
City: Boise
State: ID
ZipCode: 54321

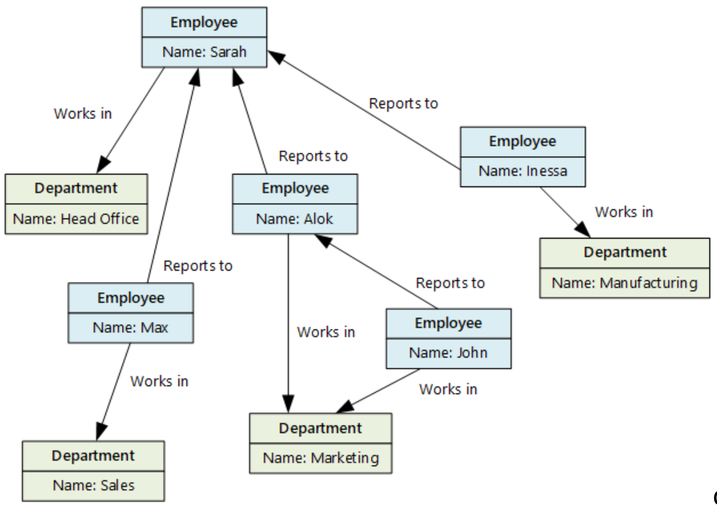
3:
StreetAddress: 999 500th Ave
City: Bellevue
State: WA
ZipCode: 12345

The most widely used column family database management system is Apache Cassandra. Azure Cosmos DB supports the column-familiy approach through the Cassandra API.

What is a graph database?

Graph databases enable you to store entities, but the main focus is on the relationships that these entities have with each other. A graph database stores two types of information: nodes that you can think of as instances of entities, and edges, which specify the relationships between nodes. Nodes and edges can both have properties that provide information about that node or edge (like columns in a table). Additionally, edges can have a direction indicating the nature of the relationship.

The purpose of a graph database is to enable an application to efficiently perform queries that traverse the network of nodes and edges, and to analyze the relationships between entities. The image below shows an organization's personnel database structured as a graph. The entities are the employees and the departments in the organization, and the edges indicate reporting lines and the department in which employees work. In this graph, the arrows on the edges show the direction of the relationships.



c

A structure such as this makes it straightforward to conduct inquiries such as "Find all employees who directly or indirectly work for Sarah" or "Who works in the same department as John?" For large graphs with lots of entities and relationships, you can perform very complex analyses very quickly, and many graph databases provide a query language that you can use to traverse a network of relationships efficiently. You can often store the same information in a relational database, but the SQL required to query this information might require many expensive recursive join operations and nested subqueries.

Azure Cosmos DB supports graph databases using the Gremlin API. The Gremlin API is a standard language for creating and querying graphs.

Azure Data Fundamentals: Explore concepts of data analytics

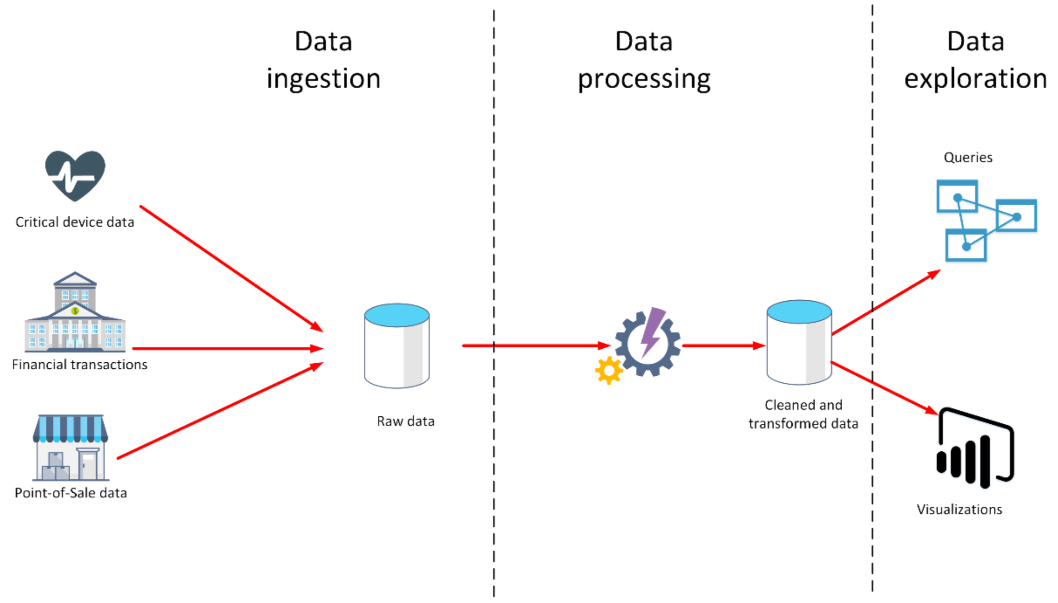
Data analytics is the process of examining raw data to uncover trends, and discover information used to ask and answer questions related to organizational performance.

I. Describe data ingestion and processing

Data analytics is concerned with taking data and finding meaningful information and inferences from it. This could be as wide ranging as selecting the ideal range of products for a retailer, or selecting the best vaccine candidates for a biotechnology company.

For example, in a company data analytics could be concerned with taking the data that your organization produces, and using it to establish a picture of how your organization is performing, and what you can do to maintain business performance. Data analytics could help you to identify strengths and weaknesses in your organization, and enable you to make appropriate business decisions.

The data a company uses can come from many sources. There could be a mass of historical data to comb through, and fresh data continuing to arrive all the time. This data could be the result of customer purchases, bank transactions, stock price movements, real-time weather data, monitoring devices, or even cameras. In a data analytics solution, you combine this data and construct a data warehouse that you can use to ask (and answer) questions about your business operations. Building a data warehouse requires that you can capture the data that you need and wrangle it into an appropriate format. You can then use analysis tools and visualizations to examine the information, and identity trends and their causes.



What is data ingestion?

Data ingestion is the process of obtaining and importing data for immediate use or storage in a database. The data can arrive as a continuous stream, or it may come in batches, depending on the source. The purpose of the ingestion process is to capture this data and store it. This raw data can be held in a repository such as a database management system, a set of files, or some other type of fast, easily accessible storage.

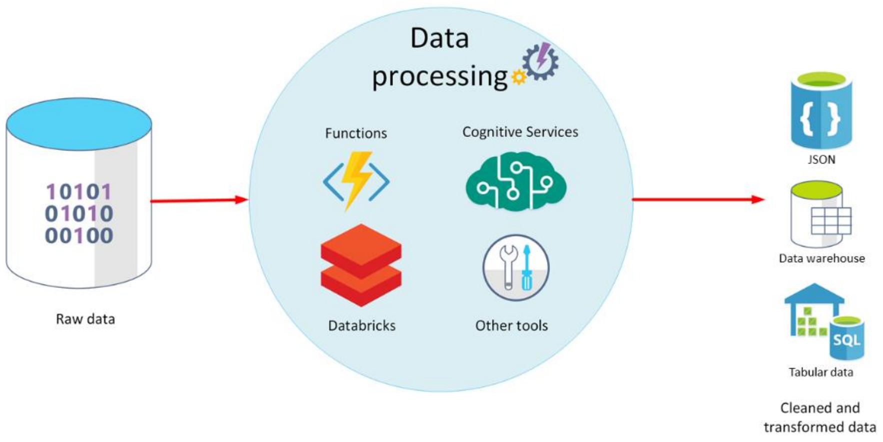
The ingestion process might also perform filtering. For example, ingestion might reject suspicious, corrupt, or duplicated data. Suspicious data might be data arriving from an unexpected source. Corrupt or duplicated data could be due to a device error, transmission failure, or tampering.

It may also be possible to perform some transformations at this stage, converting data into a standard form for later processing. For example, you might want to reformat all date and time data to use the same date and time representations, and convert all measurement data to use the same units. However, these transformations must be quick to perform. Don't attempt to run any complex calculations or aggregations on the data at this stage.

What is data processing?

The data processing stage occurs after the data has been ingested and collected. Data processing takes the data in its raw form, cleans it, and converts it into a more meaningful format (tables, graphs, documents, and so on). The result is a database of data that you can use to perform queries and generate visualizations, giving it the form and context necessary to be interpreted by computers and used by employees throughout an organization.

The aim of data processing is to convert the raw data into one or more business models. A business model describes the data in terms of meaningful business entities, and may aggregate items together and summarize information. The data processing stage could also generate predictive or other analytical models from the data. Data processing can be complex, and may involve automated scripts, and tools such as Azure Databricks, Azure Functions, and Azure Cognitive Services to examine and reformat the data, and generate models. A data analyst could use machine learning to help determine future trends based on these models.



What is ELT and ETL?

The data processing mechanism can take two approaches to retrieving the ingested data, processing this data to transform it and generate models, and then saving the transformed data and models. These approaches are known as ETL and ELT.

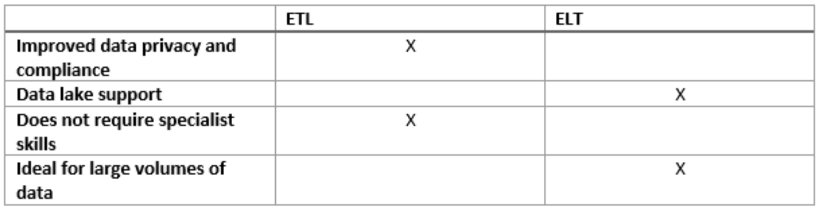
ETL stands for Extract, Transform, and Load. The raw data is retrieved and transformed before being saved. The extract, transform, and load steps can be performed as a continuous pipeline of operations. It is suitable for systems that only require simple models, with little dependency between items. For example, this type of process is often used for basic data cleaning tasks, deduplicating data, and reformatting the contents of individual fields.

An alternative approach is ELT. ELT is an abbreviation of Extract, Load, and Transform. The process differs from ETL in that the data is stored before being transformed. The data processing engine can take an iterative approach, retrieving and processing the data from storage, before writing the transformed data and models back to storage. ELT is more suitable for constructing complex models that depend on multiple items in the database, often using periodic batch processing.

ELT is a scalable approach that is suitable for the cloud because it can make use of the extensive processing power available. The more stream-oriented approach of ETL places more emphasis on throughput. However, ETL can filter data before it's stored. In this way, ETL can help with data privacy and compliance, removing sensitive data before it arrives in your analytical data models.

Azure provides several options that you can use to implement the ELT and ETL approaches. For example, if you are storing data in Azure SQL Database, you can use SQL Server Integration Services. Integration Services can extract and transform data from a wide variety of sources such as XML data files, flat files, and relational data sources, and then load the data into one or more destinations.

This is a simple table showing the advantages of ETL and ELT in most cases.



	ETL	ELT
Improved data privacy and compliance	X	
Data lake support		X
Does not require specialist skills	X	
Ideal for large volumes of data		X

Another more generalized approach is to use Azure Data Factory. Azure Data Factory is a cloud-based data integration service that allows you to create data-driven workflows for orchestrating data movement and transforming data at scale. Using Azure Data Factory, you can create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores. You can build complex ETL processes that transform data visually with data flows, or by using compute services such as Azure HDInsight Hadoop, Azure Databricks, and Azure SQL Database.

II. Explore data visualization

What is reporting?

Reporting is the process of organizing data into informational summaries to monitor how different areas of an organization are performing. Reporting helps companies monitor their online business, and know when data falls outside of expected ranges. Good reporting should raise questions about the business from its end users. Reporting shows you what has happened, while analysis focuses on explaining why it happened and what you can do about it.

What is business intelligence?

The term Business Intelligence (BI) refers to technologies, applications, and practices for the collection, integration, analysis, and presentation of business information. The purpose of business intelligence is to support better decision making.

Business intelligence systems provide historical, current, and predictive views of business operations, most often using data that has been gathered into a data warehouse, and occasionally working from live operational data. Software elements support reporting, interactive "slice-and-dice" pivot table analysis, visualization, and statistical data mining. Applications tackle sales, production, financial, and many other sources of business data for purposes that include business performance management. Information is often gathered about other companies in the same industry for comparison. This process of comparison with other companies in the same industry is known as benchmarking.

What is data visualization?

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to spot and understand trends, outliers, and patterns in data. If you are using Azure, the most popular data visualization tool is Power BI.

Using Power BI, you can connect to multiple different sources of data, and combine them into a data model. This data model lets you build visuals, and collections of visuals you can share as reports, with other people inside your organization.

Explore visualization options to represent data

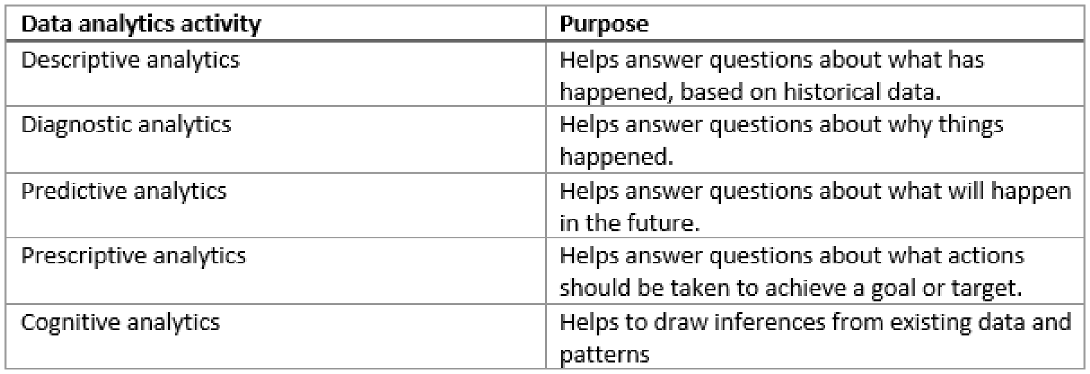
Data visualization helps you to focus on the meaning of data, rather than looking at the data itself. A good data visualization enables you to quickly spot trends, anomalies, and potential issues. The most common forms of visualizations are:

- Bar and column charts: Bar and column charts enable you to see how a set of variables changes across different categories. For example, the first chart below shows how sales for a pair of fictitious retailers vary between store sites.
- Line charts: Line charts emphasize the overall shape of an entire series of values, usually over time.
- Matrix: A matrix visual is a tabular structure that summarizes data. Often, report designers include matrixes in reports and dashboards to allow users to select one or

more element (rows, columns, cells) in the matrix to cross-highlight other visuals on a report page.

- Key influencers: A key influencer chart displays the major contributors to a selected result or value. Key influencers are a great choice to help you understand the factors that influence a key metric. For example, what influences customers to place a second order or why sales were so high last June.
- Treemap: Treemaps are charts of colored rectangles, with size representing the relative value of each item. They can be hierarchical, with rectangles nested within the main rectangles.
- Scatter: A scatter chart shows the relationship between two numerical values. A bubble chart is a scatter chart that replaces data points with bubbles, with the bubble size representing an additional third data dimension. A dot plot chart is similar to a bubble chart and scatter chart, but can plot categorical data along the X-Axis.
- Filled map. If you have geographical data, you can use a filled map to display how a value differs in proportion across a geography or region. You can see relative differences with shading that ranges from light (less-frequent/lower) to dark (more-frequent/more).

III. Explore data analytics



Data analytics activity	Purpose
Descriptive analytics	Helps answer questions about what has happened, based on historical data.
Diagnostic analytics	Helps answer questions about why things happened.
Predictive analytics	Helps answer questions about what will happen in the future.
Prescriptive analytics	Helps answer questions about what actions should be taken to achieve a goal or target.
Cognitive analytics	Helps to draw inferences from existing data and patterns

Descriptive analytics

Descriptive analytics helps answer questions about what has happened, based on historical data. Descriptive analytics techniques summarize large datasets to describe outcomes to stakeholders.

By developing KPIs (Key Performance Indicators), these strategies can help track the success or failure of key objectives. Metrics such as return on investment (ROI) are used in many industries. Specialized metrics are developed to track performance in specific industries.

Examples of descriptive analytics include generating reports to provide a view of an organization's sales and financial data.

Diagnostic analytics

Diagnostic analytics helps answer questions about why things happened. Diagnostic analytics techniques supplement more basic descriptive analytics. They take the findings from descriptive analytics and dig deeper to find the cause. The performance indicators are further investigated to discover why they got better or worse. This generally occurs in three steps:

- Identify anomalies in the data. These may be unexpected changes in a metric or a particular market.
- Collect data that's related to these anomalies.
- Use statistical techniques to discover relationships and trends that explain these anomalies.

Predictive analytics

Predictive analytics helps answer questions about what will happen in the future. Predictive analytics techniques use historical data to identify trends and determine if they're likely to recur. Predictive analytical tools provide valuable insight into what may happe n in the future. Techniques include a variety of statistical and machine learning techniques such as neural networks, decision trees, and regression.

Prescriptive analytics

Prescriptive analytics helps answer questions about what actions should be taken to achieve a goal or target. By using insights from predictive analytics, data-driven decisions can be made. This technique allows businesses to make informed decisions in the face of uncertainty. Prescriptive analytics techniques rely on machine learning strategies to find patterns in large datasets. By analyzing past decisions and events, the likelihood of different outcomes can be estimated.

Cognitive analytics

Cognitive analytics attempts to draw inferences from existing data and patterns, derive conclusions based on existing knowledge bases, and then add these findings back into the knowledge base for future inferences--a self-learning feedback loop. Cognitive analytics helps you to learn what might happen if circumstances change, and how you might han dle these situations.

Inferences aren't structured queries based on a rules database, rather they're unstructured hypotheses gathered from a number of sources, and expressed with varying degrees of confidence. Effective cognitive analytics depends on machine learning algorithms. It uses several NLP (Natural Language Processing) concepts to make sense of previously untapped data sources, such as call center conversation logs and product reviews.

Theoretically, by tapping the benefits of massive parallel/distributed computing and the falling costs of data storage and computing power, there's no limit to the cognitive development that these systems can achieve.

Azure Data Fundamentals: Explore relational data in Azure

https://docs.microsoft.com/en-us/learn/paths/azure-data-fundamentals-explore-relational-data/

Azure Data Fundamentals: Explore relational data services in Azure

I. Explore relational Azure data services

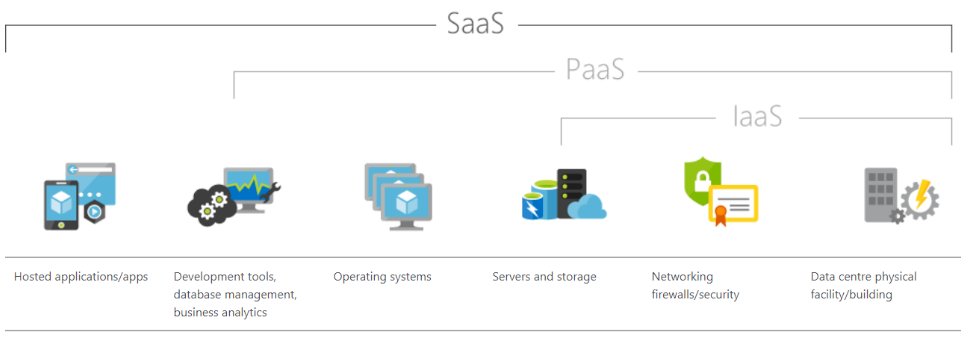
Understand IaaS, PaaS, and SaaS

Before delving into Azure Data Services, you need to understand some common terms used to describe the different ways in which you can host a database in Azure.

IaaS is an acronym for Infrastructure-as-a-Service. Azure enables you to create a virtual infrastructure in the cloud that mirrors the way an on-premises data center might work. You can create a set of virtual machines, connect them together using a virtual network, and add a range of virtual devices. You take responsibility for installing and configuring the software, such as the DBMS, on these virtual machines. In many ways, this approach is similar to the way in which you run your systems inside an organization, except that you don't have to concern yourself with buying or maintaining the hardware.

PaaS stands for Platform-as-a-service. Rather than creating a virtual infrastructure, and installing and managing the database software yourself, a PaaS solution does this for you. You specify the resources that you require (based on how large you think your databases will be, the number of users, and the performance you require), and Azure automatically creates the necessary virtual machines, networks, and other devices for you. You can usually scale up or down (increase or decrease the size and number of resources) quickly, as the volume of data and the amount of work being done varies; Azure handles this scaling for you, and you don't have to manually add or remove virtual machines, or perform any other form of configuration.

SaaS is short for Software-as-a-Service. SaaS services are typically specific software packages that are installed and run on virtual hardware in the cloud. SaaS packages are typically hosted applications rather than more generalized software such as a DBMS. Common SaaS packages available on Azure include Microsoft 365 (formerly Office 365).


SaaS
PaaS
IaaS
Hosted applications/apps
Development tools, database management, business analytics
Operating systems
Servers and storage
Networking firewalls/security
Data centre physical facility/building

What are Azure Data Services?

Azure Data Services fall into the PaaS category. These services are a series of DBMSs managed by Microsoft in the cloud. Each data service takes care of the configuration, day-to-day management, software updates, and security of the databases that it hosts. All you do is create your databases under the control of the data service.

Azure Data Services are available for several common relational database management systems. The most well-known service is Azure SQL Database. The others currently available are Azure Database for MySQL servers, Azure Database for MariaDB servers, and Azure Database for PostgreSQL servers. The remaining units in this module describe the features provided by these services.

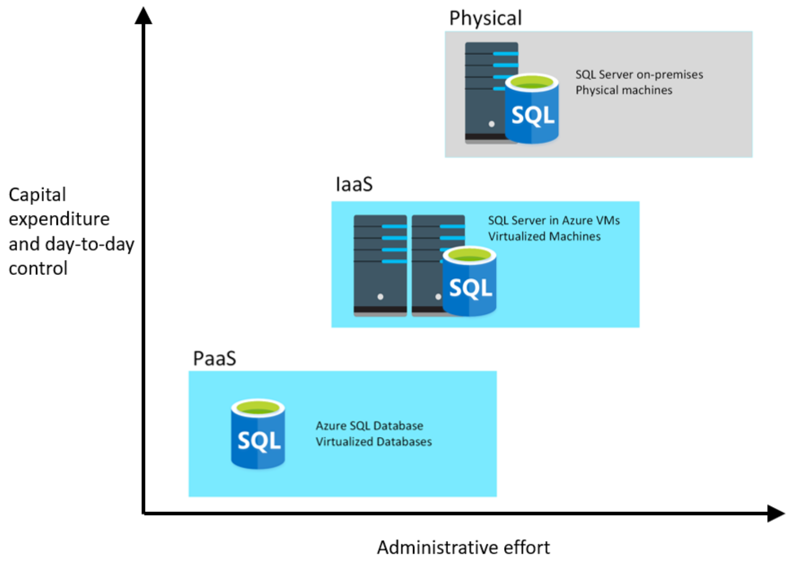
Using Azure Data Services reduces the amount of time that you need to invest to administer a DBMS. However, these services can also limit the range of custom administration tasks that you can perform, because manually performing some tasks might risk compromising the way in which the service runs. For example, some DBMSs enable you to install custom software into a database, or run scripts as part of a database operation. This software might not be supported by the data service, and allowing an application to run a script from a database could affect the security of the service. You must be prepared to work with these restrictions in mind.

Apart from reducing the administrative workload, Azure Data Services ensure that your databases are available for at least 99.99% of the time.

There are costs associated with running a database in Azure Data Services. The base price of each service covers underlying infrastructure and licensing, together with the administration charges. Additionally, these services are designed to be always on. This means that you can't shut down a database and restart it later.

Not all features of a database management system are available in Azure Data Services. This is because Azure Data Services takes on the task of managing the system and keeping it running using hardware situated in an Azure datacenter. Exposing some administrative functions might make the underlying platform vulnerable to misuse, and even open up some security concerns. Therefore, you have no direct control over the platform on which the services run. If you need more control than Azure Data Services allow, you can install your database management system on a virtual machine that runs in Azure. The next unit examines this approach in more detail for SQL Server, although the same issues apply for the other database management systems supported by Azure Data Services.

The image below highlights the different ways in which you could run a DBMS such as SQL Server, starting with an on-premises system in the bottom left-hand corner, to PaaS in the upper right. The diagram illustrates the benefits of moving to the PaaS approach.



II. SQL Server on Azure virtual machines - (IaaS)

What is SQL Server on Azure Virtual Machines?

SQL Server on Virtual Machines enables you to use full versions of SQL Server in the Cloud without having to manage any on-premises hardware. This is an example of the IaaS approach.

SQL Server running on an Azure virtual machine effectively replicates the database running on real on-premises hardware. Migrating from the system running on-premises to an Azure virtual machine is no different than moving the databases from one on-premises server to another.

In the example scenario described in the introduction, the database runs stored procedures and scripts as part of the database workload. If these stored procedures and scripts depend on features that are restricted by following a PaaS approach, then running SQL Server on your own virtual machines might be a good option. However, you remain responsible for maintaining the SQL Server software and performing the various administrative tasks to keep the database running from day-to-day.

This approach is suitable for migrations and applications requiring access to operating system features that might be unsupported at the PaaS level. SQL virtual machines are lift-and-shift ready for existing applications that require fast migration to the cloud with minimal changes.

This approach is optimized for migrating existing applications to Azure, or extending existing on-premises applications to the cloud in hybrid deployments.

You can use SQL Server in a virtual machine to develop and test traditional SQL Server applications. With a virtual machine, you have the full administrative rights over the DBMS and operating system. It's a perfect choice when an organization already has IT resources available to maintain the virtual machines.

These capabilities enable you to:

- Create rapid development and test scenarios when you do not want to buy on-premises non-production SQL Server hardware.
- Become lift-and-shift ready for existing applications that require fast migration to the cloud with minimal changes or no changes.
- Scale up the platform on which SQL Server is running, by allocating more memory, CPU power, and disk space to the virtual machine. You can quickly resize an Azure virtual machine without the requirement that you reinstall the software that is running on it.

Business benefits
Running SQL Server on virtual machines allows you to meet unique and diverse business needs through a combination of on-premises and cloud-hosted deployments, while using the same set of server products, development tools, and expertise across these environments.

It's not always easy for businesses to switch their DBMS to a fully managed service. There may be specific requirements that must be satisfied in order to migrate to a managed service that requires making changes to the database and the applications that use it. For this reason, using virtual machines can offer a solution, but using them does not eliminate the need to administer your DBMS as carefully as you would on-premises.

III. Azure SQL Database – (PaaS)

Azure SQL Database is a PaaS offering from Microsoft. You create a managed database server in the cloud, and then deploy your databases on this server.

Azure SQL Database is available with several options: Single Database, Elastic Pool, and Managed Instance. The following sections describe Single Database and Elastic Pool. Managed Instance is the subject of the next unit.

Single Database
This option enables you to quickly set up and run a single SQL Server database. You create and run a database server in the cloud, and you access your database through this server. Microsoft manages the server, so all you have to do is configure the database, create your tables, and populate them with your data. You can scale the database if you need additional storage space, memory, or processing power. By default, resources are pre-allocated, and you're charged per hour for the resources you've requested. You can also specify a serverless configuration. In this configuration, Microsoft creates its own server, which might be shared by a number of databases belonging to other Azure subscribers. Microsoft ensures the privacy of your database. Your database automatically scales and resources are allocated or deallocated as required. For more information, read What is a single database in Azure SQL Database.

Elastic Pool
This option is similar to Single Database, except that by default multiple databases can share the same resources, such as memory, data storage space, and processing power through multiple-tenancy. The resources are referred to as a pool. You create the pool, and only your databases can use the pool. This model is useful if you have databases with resource requirements that vary over time, and can help you to reduce costs. For example, your payroll database might require plenty of CPU power at the end of each month as you handle payroll processing, but at other times the database might become much less active. You might have another database that is used for running reports. This database might become active for several days in the middle of the month as management reports are generated, but with a lighter load at other times. Elastic Pool enables you to use the resources available in the pool, and then release the resources once processing has completed.

Use cases
Azure SQL Database gives you the best option for low cost with minimal administration. It is not fully compatible with on-premises SQL Server installations. It is often used in new cloud projects where the application design can accommodate any required changes to your applications.

Azure SQL Database is often used for:

- Modern cloud applications that need to use the latest stable SQL Server features.
- Applications that require high availability.
- Systems with a variable load, that need the database server to scale up and down quickly.

Business benefits
Azure SQL Database automatically updates and patches the SQL Server software to ensure that you are always running the latest and most secure version of the service.

The scalability features of Azure SQL Database ensure that you can increase the resources available to store and process data without having to perform a costly manual upgrade.

The service provides high availability guarantees, to ensure that your databases are available at least 99.99% of the time. Azure SQL Database supports point-in-time restore, enabling you to recover a database to the state it was in at any point in the past. Databases can be replicated to different regions to provide additional assurance and disaster recovery

Advanced threat protection provides advanced security capabilities, such as vulnerability assessments, to help detect and remediate potential security problems with your databases. Threat protection also detects anomalous activities that indicate unusual and potentially harmful attempts to access or exploit your database. It continuously monitors your database for suspicious activities, and provides immediate security alerts on potential vulnerabilities, SQL injection attacks, and anomalous database access patterns. Threat detection alerts provide details of the suspicious activity, and recommend action on how to investigate and mitigate the threat.

Auditing tracks database events and writes them to an audit log in your Azure storage account. Auditing can help you maintain regulatory compliance, understand database activity, and gain insight into discrepancies and anomalies that might indicate business concerns or suspected security violations.

SQL Database helps secure your data by providing encryption. For data in motion, it uses Transport Layer Security. For data at rest, it uses Transparent Data Encryption. For data in use, it uses Always Encrypted. For more information on Transport Layer Security, Transparent Data Encryption, and Always Encrypted, see the links in the Summary unit.

In the Wide World Importers scenario, linked servers are used to perform distributed queries. However, neither Single Database nor Elastic Pool support linked servers. If you want to use Single Database or Elastic Pool, you may need to modify the queries that use linked servers and rework the operations that depend on these features.

IV. Azure SQL Database Managed Instance

What is Azure SQL Database managed instance?

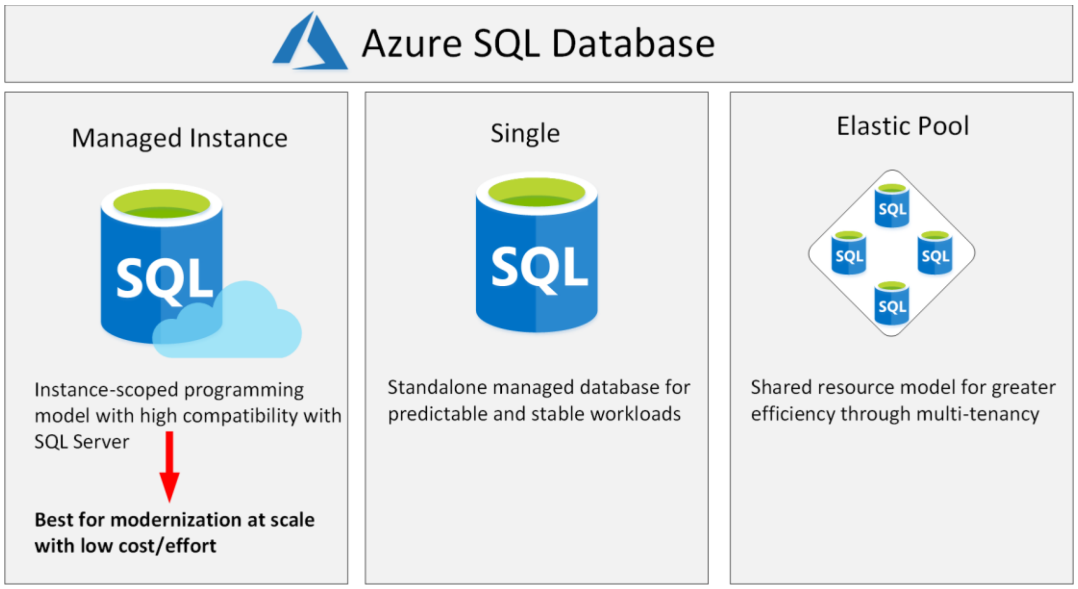
The Single Database and Elastic Pool options restrict some of the administrative features available to SQL Server. Managed instance effectively runs a fully controllable instance of SQL Server in the cloud. You can install multiple databases on the same instance. You have complete control over this instance, much as you would for an on-premises server. The Managed instance service automates backups, software patching, database monitoring, and other general tasks, but you have full control over security and resource allocation for your databases. You can find detailed information at What is Azure SQL Database managed instance?.

Managed instances depend on other Azure services such as Azure Storage for backups, Azure Event Hubs for telemetry, Azure Active Directory for authentication, Azure Key Vault for

Transparent Data Encryption (TDE) and a couple of Azure platform services that provide security and supportability features. The managed instances make connections to these services.

All communications are encrypted and signed using certificates. To check the trustworthiness of communicating parties, managed instances constantly verify these certificates through certificate revocation lists. If the certificates are revoked, the managed instance closes the connections to protect the data.

The following image summarizes the differences between SQL Database managed instance, Single Database, and Elastic Pool


Azure SQL Database
Managed Instance
SQL
Instance-scoped programming model with high compatibility with SQL Server
Best for modernization at scale with low cost/effort
Single
SQL
Standalone managed database for predictable and stable workloads
Elastic Pool
SQL
SQL
SQL
SQL
Shared resource model for greater efficiency through multi-tenancy

Use cases

Consider Azure SQL Database managed instance if you want to lift-and-shift an on-premises SQL Server instance and all its databases to the cloud, without incurring the management overhead of running SQL Server on a virtual machine.

SQL Database managed instance provides features not available with the Single Database or Elastic Pool options. If your system uses features such as linked servers, Service Broker (a message processing system that can be used to distribute work across servers), or Database Mail (which enables your database to send email messages to users), then you should use managed instance. To check compatibility with an existing on-premises system, you can install Data Migration Assistant (DMA). This tool analyzes your databases on SQL Server and reports any issues that could block migration to a managed instance.

Business benefits

SQL Database managed instance provides all the management and security benefits available when using Single Database and Elastic Pool. Managed instance deployment enables a system administrator to spend less time on administrative tasks because the SQL Database service either performs them for you or greatly simplifies those tasks. Automated tasks include operating system and database management system software installation and patching, dynamic instance resizing and configuration, backups, database replication (including system databases), high availability configuration, and configuration of health and performance monitoring data streams.

Managed instance has near 100% compatibility with SQL Server Enterprise Edition, running on-premises.

The SQL Database managed instance deployment option supports traditional SQL Server Database engine logins and logins integrated with Azure Active Directory (AD). Traditional SQL Server Database engine logins include a username and a password. You must enter your credentials each time you connect to the server. Azure AD logins use the credentials associated with your current computer sign-in, and you don't need to provide them each time you connect to the server.

In the Wide World Importers scenario, SQL Database managed instance may be a more suitable choice than Single Database or Elastic Pool. SQL Database managed instance supports linked servers, although some of the other the advanced features required by the database might not be available. If you want a complete match, then running SQL Server on a virtual machine may be your only option, but you need to balance the benefits of complete functionality against the administrative and maintenance overhead required.

V. PostgreSQL, MariaDB, MySQL

What are MySQL, MariaDB, and PostgreSQL?

PostgreSQL, MariaDB, and MySQL are relational database management systems that are tailored for different specializations.

MySQL started life as a simple-to-use open-source database management system. It is the leading open source relational database for Linux, Apache, MySQL, and PHP (LAMP) stack apps. It's available in several editions; Community, Standard, and Enterprise. The Community edition is available free-of-charge, and has historically been popular as a database management system for web applications, running under Linux. Versions are also available for Windows. Standard edition offers higher performance, and uses a different technology for storing data. Enterprise edition provides a comprehensive set of tools and features, including enhanced security, availability, and scalability. The Standard and Enterprise editions are the versions most frequently used by commercial organizations, although these versions of the software aren't free.

MariaDB is a newer database management system, created by the original developers of MySQL. The database engine has since been rewritten and optimized to improve performance.

MariaDB offers compatibility with Oracle Database (another popular commercial database management system). One notable feature of MariaDB is its built-in support for temporal data. A table can hold several versions of data, enabling an application to query the data as it appeared at some point in the past.

PostgreSQL is a hybrid relational-object database. You can store data in relational tables, but a PostgreSQL database also enables you to store custom data types, with their own non-relational properties. The database management system is extensible; you can add code modules to the database, which can be run by queries. Another key feature is the ability to store and manipulate geometric data, such as lines, circles, and polygons.

PostgreSQL has its own query language called pgsql. This language is a variant of the standard relational query language, SQL, with features that enable you to write stored procedures that run inside the database.

MySQL	MariaDB	PostgreSQL
• Very popular • Available as free Community edition or paid-for, and more functional, Standard and Enterprise editions • Azure Database for MySQL is based on the free Community edition, but adds high availability and scalability	• Compatible with Oracle Database • Built-in support for temporal data	• Can store both relational and non-relational data • Can store geometric data • Extensible

What is Azure Database for MySQL?
Azure Database for MySQL is a PaaS implementation of MySQL in the Azure cloud, based on the MySQL Community Edition.

The Azure Database for MySQL service includes high availability at no additional cost and scalability as required. You only pay for what you use. Automatic backups are provided, with point-in-time restore.

The server provides connection security to enforce firewall rules and, optionally, require SSL connections. Many server parameters enable you to configure server settings such as lock modes, maximum number of connections, and timeouts.

Azure Database for MySQL provides a global database system that scales up to large databases without the need to manage hardware, network components, virtual servers, software patches, and other underlying components.

Certain operations aren't available with Azure Database for MySQL. These functions are primarily concerned with security and administration. Azure manages these aspects of the database server itself.

Benefits of Azure Database for MySQL
You get the following features with Azure Database for MySQL:

- High availability features built-in.
- Predictable performance.
- Easy scaling that responds quickly to demand.
- Secure data, both at rest and in motion.
- Automatic backups and point-in-time restore for the last 35 days.
- Enterprise-level security and compliance with legislation.
- The system uses pay-as-you-go pricing so you only pay for what you use.

Azure Database for MySQL servers provides monitoring functionality to add alerts, and to view metrics and logs.

What is Azure Database for MariaDB?
Azure Database for MariaDB is an implementation of the MariaDB database management system adapted to run in Azure. It's based on the **MariaDB Community Edition.**

The database is fully managed and controlled by Azure. Once you've provisioned the service and transferred your data, the system requires almost no additional administration.

Benefits of Azure Database for MariaDB
Azure Database for MariaDB delivers:

- Built-in high availability with no additional cost.
- Predictable performance, using inclusive pay-as-you-go pricing.
- Scaling as needed within seconds.
- Secured protection of sensitive data at rest and in motion.
- Automatic backups and point-in-time-restore for up to 35 days.
- Enterprise-grade security and compliance.

What is Azure Database for PostgreSQL?
If you prefer PostgreSQL, you can choose Azure Database for PostgreSQL to run a PaaS implementation of PostgreSQL in the Azure Cloud. This service provides the same availability, performance, scaling, security, and administrative benefits as the MySQL service.

Some features of on-premises PostgreSQL databases are not available in Azure Database for PostgreSQL. These features are mainly concerned with the extensions that users can add to a database to perform specialized tasks, such as writing stored procedures in various programming languages (other than pgsql, which is available), and interacting directly with the operating system. A core set of the most frequently used extensions is supported, and the list of available extensions is under continuous review.

Azure Database for PostgreSQL has two deployment options: Single-server and Hyperscale.

Azure Database for PostgreSQL single-server
The single-server deployment option for PostgreSQL provides similar benefits as Azure Database for MySQL. You choose from three pricing tiers: Basic, General Purpose, and Memory Optimized. Each tier supports different numbers of CPUs, memory, and storage sizes —you select one based on the load you expect to support.

Azure Database for PostgreSQL Hyperscale (Citus)
Hyperscale (Citus) is a deployment option that scales queries across multiple server nodes to support large database loads. Your database is split across nodes. Data is split into chunks based on the value of a partition key or sharding key. Consider using this deployment option for the largest database PostgreSQL deployments in the Azure Cloud.

Benefits of Azure Database for PostgreSQL
Azure Database for PostgreSQL is a highly available service. It contains built-in failure detection and failover mechanisms.

Users of PostgreSQL will be familiar with the pgAdmin tool, which you can use to manage and monitor a PostgreSQL database. You can continue to use this tool to connect to Azure Database for PostgreSQL. However, some server-focused functionality, such as performing server backup and restore, are not available because the server is managed and maintained by Microsoft.

Azure Database for PostgreSQL servers records information about the queries run against databases on the server, and saves them in a database named azure_sys. You query the query_store.qs_view view to see this information, and use it to monitor the queries that users are running. This information can prove invaluable if you need to fine-tune the queries performed by your applications.

Migrate data to Azure
If you have existing MySQL, MariaDB, or PostgreSQL databases running on premises, and you want to move the data to a database running the corresponding data services in Azure, you can use the Azure Database Migration Service (DMS).

The Database Migration Service enables you to restore a backup of your on-premises databases directly to databases running in Azure Data Services. You can also configure replication from an on-premises database, so that any changes made to data in that database are copied to the database running in Azure Data Services. This strategy enables you to reconfigure users and applications to connect to the database in the cloud while the on-premises system is still active; you don't have to shut down the on-premises system while you transfer users to the cloud.

Azure Data Fundamentals: Explore provisioning and deploying relational data services in Azure

I. Describe provisioning relational data services

What is provisioning?

Provisioning is the act of running a series of tasks that a service provider, such as Azure SQL Database, performs to create and configure a service. Behind the scenes, the service provider will set up the various resources (disks, memory, CPUs, networks, and so on) required to run the service. You'll be assigned these resources, and they remain allocated to you (and charged to you), until you delete the service.

How the service provider provisions resources is opaque, and you don't need to be concerned with how this process works. All you do is specify parameters that determine the size of the resources required (how much disk space, memory, computing power, and network bandwidth). These parameters are determined by estimating the size of the workload that you intend to run using the service. In many cases, you can modify these parameters after the service has been created, perhaps increasing the amount of storage space or memory if the workload is greater than you initially anticipated. The act of increasing (or decreasing) the resources used by a service is called scaling.

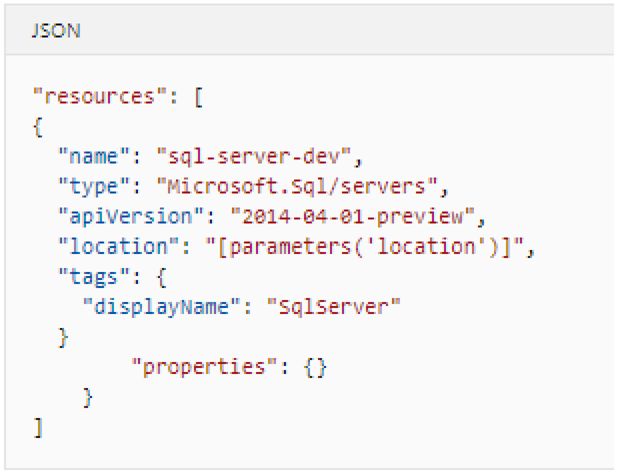
Azure provides several tools you can use to provision services:

The Azure portal. This is the most convenient way to provision a service for most users. The Azure portal displays a series of service-specific pages that prompt you for the settings required, and validates these settings, before actually provisioning the service.

The Azure command-line interface (CLI). The CLI provides a set of commands that you can run from the operating system command prompt or the Cloud Shell in the Azure portal. You can use these commands to create and manage Azure resources. The CLI is suitable if you need to automate service creation; you can store CLI commands in scripts, and you can run these scripts programmatically. The CLI can run on Windows, macOS, and Linux computers. For detailed information about the Azure CLI, read What is Azure CLI.

Azure PowerShell. Many administrators are familiar with using PowerShell commands to script and automate administrative tasks. Azure provides a series of commandlets (Azure-specific commands) that you can use in PowerShell to create and manage Azure resources. You can find further information about Azure PowerShell online, at Azure PowerShell documentation. Like the CLI, PowerShell is available for Windows, macOS, and Linux.

Azure Resource Manager templates. An Azure Resource Manager template describes the service (or services) that you want to deploy in a text file, in a format known as JSON (JavaScript Object Notation). The example below shows a template that you can use to provision an instance of Azure SQL Database.



JSON

```
"resources": [
{
  "name": "sql-server-dev",
  "type": "Microsoft.Sql/servers",
  "apiVersion": "2014-04-01-preview",
  "location": "[parameters('location')]",
  "tags": {
    "displayName": "SqlServer"
  }
        "properties": {}
    }
]
```

You send the template to Azure using the az deployment group create command in the Azure CLI, or New-AzResourceGroupDeployment command in Azure PowerShell. For more information about creating and using Azure Resource Manager templates to provision Azure resources, see What are Azure Resource Manager templates?

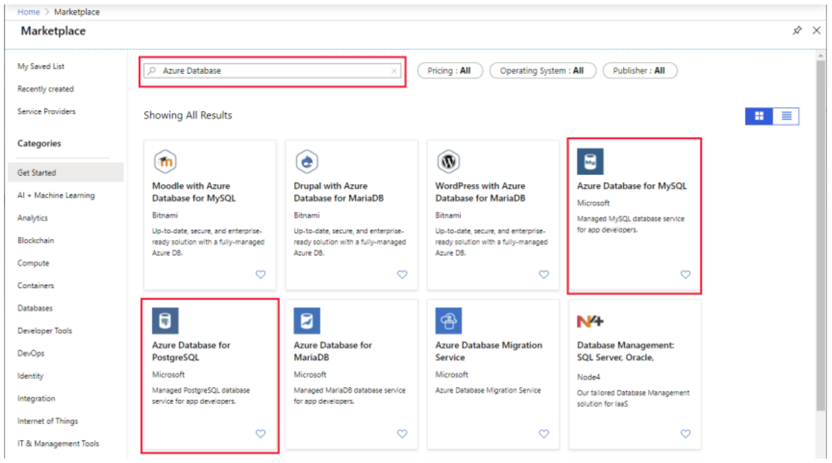
II. Describe provisioning Azure SQL Database

https://docs.microsoft.com/en-us/learn/modules/explore-provision-deploy-relational-database-offerings-azure/3-describe-provision-sql-database

III. Describe provisioning PostgreSQL and MySQL

How to provision Azure Database for PostgreSQL and Azure Database for MySQL

As with Azure SQL Database, you can provision a PostgreSQL or MySQL database interactively using the Azure portal. You can find both of these services in the Azure Marketplace:



Moodle with Azure Database for MySQL
Bitnami
Up-to-date, secure, and enterprise-ready solution with a fully-managed Azure DB.

Drupal with Azure Database for MariaDB
Bitnami
Up-to-date, secure, and enterprise-ready solution with a fully-managed Azure DB.

WordPress with Azure Database for MariaDB
Bitnami
Up-to-date, secure, and enterprise-ready solution with a fully-managed Azure DB.

Azure Database for MySQL
Microsoft
Managed MySQL database service for app developers.

Azure Database for PostgreSQL
Microsoft
Managed PostgreSQL database service for app developers.

Azure Database for MariaDB
Microsoft
Managed MariaDB database service for app developers.

Azure Database Migration Service
Microsoft
Azure Database Migration Service

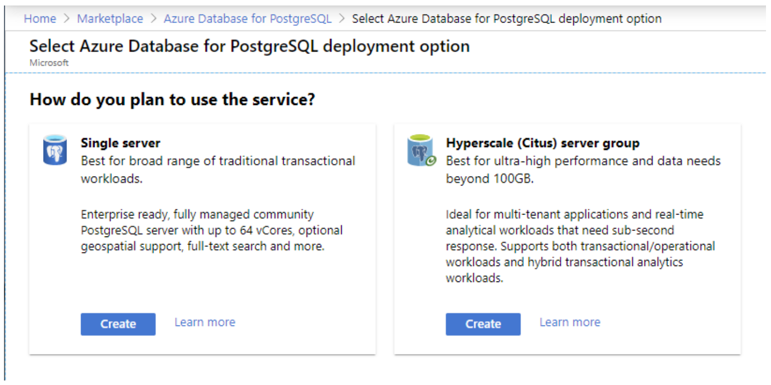
Database Management: SQL Server, Oracle,
Node4
Our tailored Database Management solution for IaaS

The processes for provisioning Azure Database for PostgreSQL and Azure Database for MySQL are very similar.

PostgreSQL Hyperscale

The hyperscale deployment option supports:

- Horizontal scaling across multiple machines. This option enables the service to add and remove computers as workloads increase and diminish.
- Query parallelization across these servers. The service can split resource intensive queries into pieces which can be run in parallel on the different servers. The results from each server are aggregated back together to produce a final result. This mechanism can deliver faster responses on queries over large datasets.
- Excellent support for multi-tenant applications, real time operational analytics, and high throughput transactional workloads


Home > Marketplace > Azure Database for PostgreSQL > Select Azure Database for PostgreSQL deployment option

Select Azure Database for PostgreSQL deployment option
Microsoft

How do you plan to use the service?

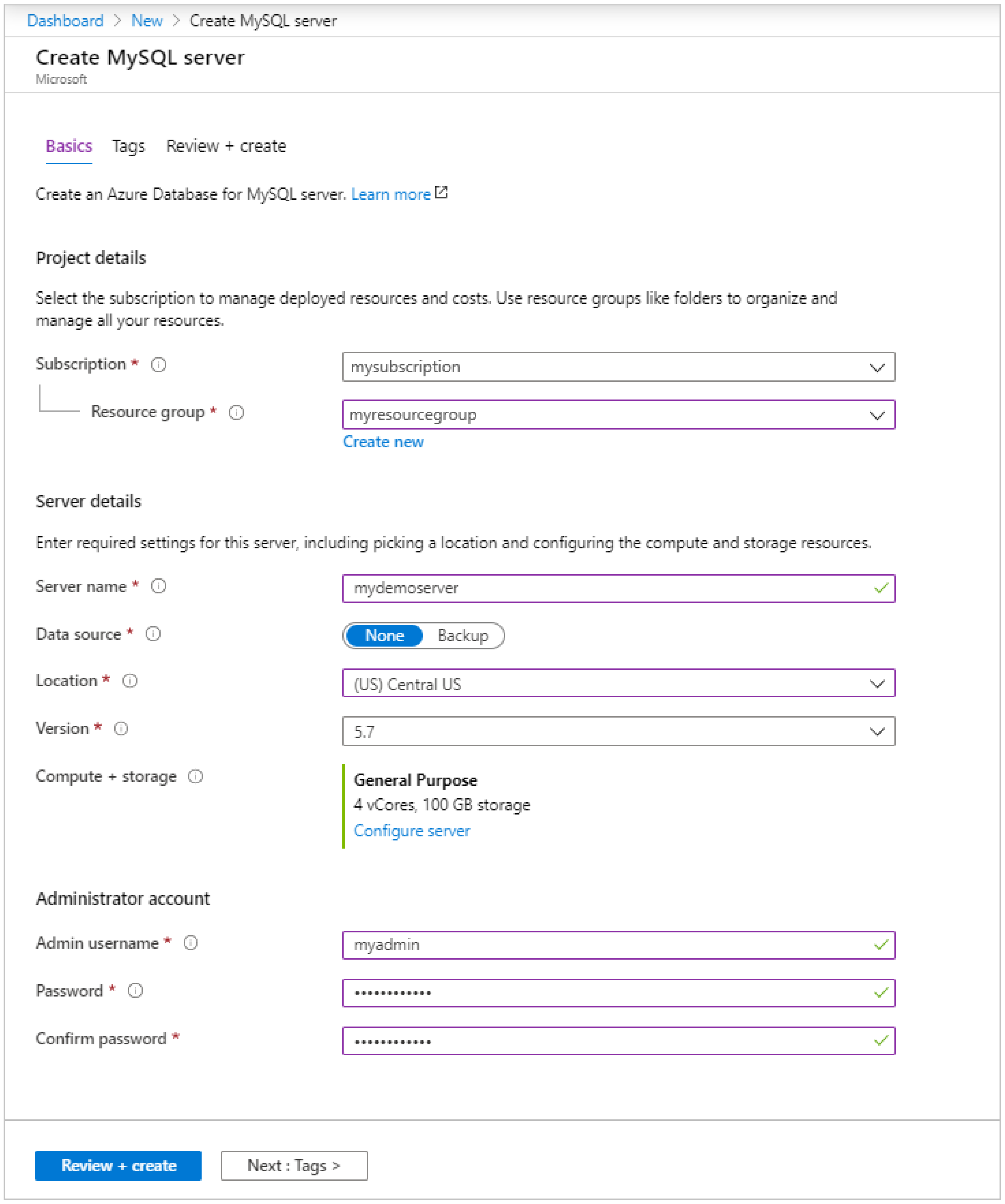
Single server
Best for broad range of traditional transactional workloads.

Enterprise ready, fully managed community PostgreSQL server with up to 64 vCores, optional geospatial support, full-text search and more.

Hyperscale (Citus) server group
Best for ultra-high performance and data needs beyond 100GB.

Ideal for multi-tenant applications and real-time analytical workloads that need sub-second response. Supports both transactional/operational workloads and hybrid transactional analytics workloads.

Provisioning a PostgreSQL or a MySQL database service:



Dashboard > New > Create MySQL server

Create MySQL server
Microsoft

Basics | Tags | Review + create

Create an Azure Database for MySQL server. Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	mysubscription
Resource group *	myresourcegroup (Create new)

Server details

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

Server name *	mydemoserver
Data source *	[✓ None] Backup
Location *	(US) Central US
Version *	5.7
Compute + storage	General Purpose 4 vCores, 100 GB storage Configure server

Administrator account

Admin username *	myadmin
Password *	••••••••••••
Confirm password *	••••••••••••

The **Create MySQL Server** tab, prompts for the following details:

- Subscription. Select your Azure subscription.
- Resource Group. Either pick an existing resource group, or select Create new to build a new one.

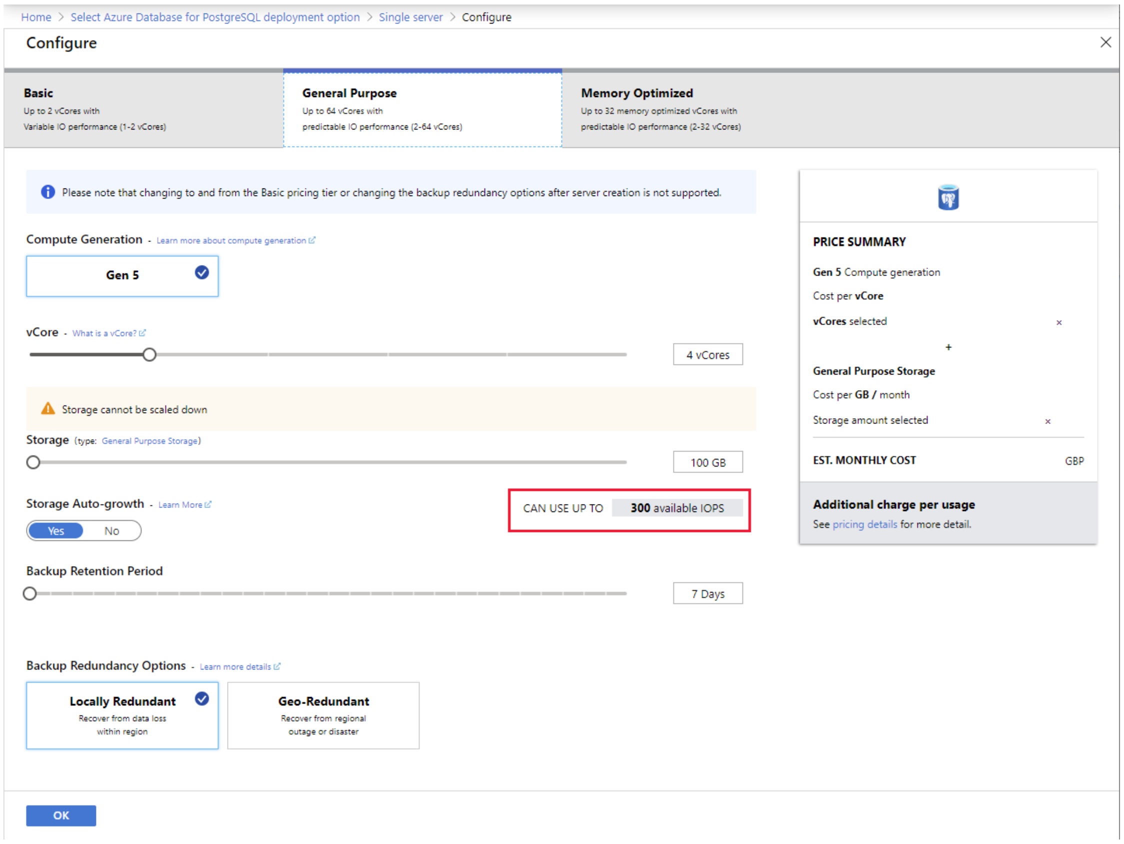
- Server Name. Each MySQL or PostgreSQL database must have a unique name that hasn't already been used by someone else. The name must be between 3 and 31 characters long, and can only contain lower case letters, digits, and the "-" character.

- Data Source. Select None to create a new server from scratch. You can select Backup if you're creating a server from a geo-backup of an existing Azure Database for MySQL server.

- Location. Either select the region that is nearest to you, or the region nearest to your users.

- Version. The version of MySQL or PostgreSQL to deploy.

- Compute + storage. The compute, storage, and backup configurations for your new server. The Configure server link enables you to select the resources required to support you database workloads. These resources include the amount of computing power, memory, backups, and redundancy options (for high availability).



Home > Select Azure Database for PostgreSQL deployment option > Single server > Configure

Configure

Basic
Up to 2 vCores with
Variable IO performance (1-2 vCores)

General Purpose
Up to 64 vCores with
predictable IO performance (2-64 vCores)

Memory Optimized
Up to 32 memory optimized vCores with
predictable IO performance (2-32 vCores)

Please note that changing to and from the Basic pricing tier or changing the backup redundancy options after server creation is not supported.

Compute Generation - Learn more about compute generation

Gen 5 ✓

vCore - What is a vCore?

4 vCores

Storage cannot be scaled down

Storage (type: General Purpose Storage)

100 GB

Storage Auto-growth - Learn More

[✓ Yes] No

CAN USE UP TO 300 available IOPS

Backup Retention Period

7 Days

Backup Redundancy Options - Learn more details

Locally Redundant ✓
Recover from data loss within region

Geo-Redundant
Recover from regional outage or disaster

OK

PRICE SUMMARY

Gen 5 Compute generation
Cost per **vCore**
vCores selected ×
+
General Purpose Storage
Cost per **GB /** month
Storage amount selected ×

EST. MONTHLY COST GBP

Additional charge per usage
See pricing details for more detail.

You can select between three pricing tiers, each of which is designed to support different workloads:

- Basic. This tier is suitable for workloads that require light compute and I/O performance. Examples include servers used for development or testing or small-scale, infrequently used applications.

- General Purpose. Use this pricing tier for business workloads that require balanced compute and memory with scalable I/O throughput. Examples include servers for hosting web and mobile apps and other enterprise applications.

- Memory Optimized This tier supports high-performance database workloads that require in-memory performance for faster transaction processing and higher concurrency. Examples include servers for processing real-time data and high-performance transactional or analytical apps.

Admin username. A sign-in account to use when you're connecting to the server. The admin sign-in name can't be azure_superuser, admin, administrator, root, guest, or public.

Password. Provide a new password for the server admin account. It must contain from 8 to 128 characters. Your password must contain characters from three of the following categories: English uppercase letters, English lowercase letters, numbers (0-9), and non-alphanumeric characters (!, $, #, %, and so on).

IV. Describe configuring relational data services

Configure connectivity and firewalls

The default connectivity for Azure relational data services is to disable access to the world.

Configure connectivity to virtual networks and on-premises computers

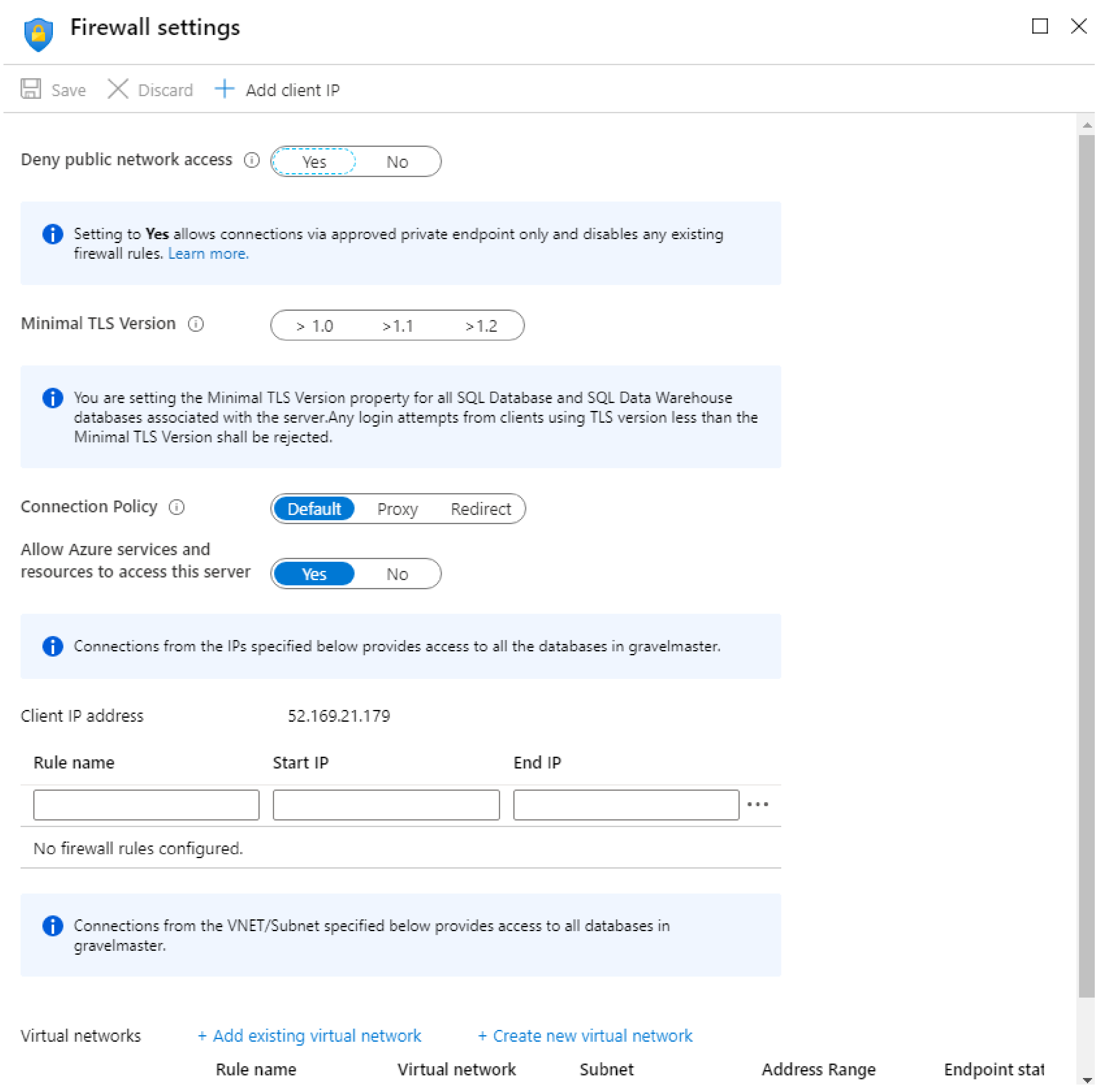
To enable connectivity, use the Firewalls and virtual networks page for a service. To enable connectivity, choose Selected networks. Three further sections will appear, labeled Virtual network, Firewall, and Exceptions.

In the Virtual networks section, you can specify which virtual networks are allowed to route traffic to the service. When you create items such as web applications and virtual machines, you can add them to a virtual network. If these applications and virtual machines require access to your resource, add the virtual network containing these items to the list of allowed networks.

If you need to connect to the service from an on-premises computer, in the Firewall section, add the IP address of the computer. This setting creates a firewall rule that allows traffic from that address to reach the service.

The Exceptions setting allows you to enable access to any other of your services created in your Azure subscription.

The image below shows the Firewalls and virtual networks page for an Azure SQL database. MySQL and PostgreSQL have a similar page.


Firewall settings

Save | Discard | Add client IP

Deny public network access: [✓ Yes] No

Setting to **Yes** allows connections via approved private endpoint only and disables any existing firewall rules. Learn more.

Minimal TLS Version: > 1.0 | >1.1 | >1.2

You are setting the Minimal TLS Version property for all SQL Database and SQL Data Warehouse databases associated with the server.Any login attempts from clients using TLS version less than the Minimal TLS Version shall be rejected.

Connection Policy: [✓ Default] Proxy Redirect

Allow Azure services and resources to access this server: [✓ Yes] No

Connections from the IPs specified below provides access to all the databases in gravelmaster.

Client IP address: 52.169.21.179

Rule name	Start IP	End IP

No firewall rules configured.

Connections from the VNET/Subnet specified below provides access to all databases in gravelmaster.

Virtual networks: + Add existing virtual network | + Create new virtual network

Rule name	Virtual network	Subnet	Address Range	Endpoint stat

Configure connectivity from private endpoints.
Azure Private Endpoint is a network interface that connects you privately and securely to a service powered by Azure Private Link. Private Endpoint uses a private IP address from your virtual network, effectively bringing the service into your virtual network. The service could be an Azure service such as Azure App Service, or your own Private Link Service. For detailed information, read What is Azure Private Endpoint?.

The Private endpoint connections page for a service allows you to specify which private endpoints, if any, are permitted access to your service. You can use the settings on this page, together with the Firewalls and virtual networks page, to completely lock down users and applications from accessing public endpoints to connect to your Azure SQL Database account.

Configure authentication
With Azure Active Directory (AD) authentication, you can centrally manage the identities of database users and other Microsoft services in one central location. Central ID management provides a single place to manage database users and simplifies permission management.

You can use these identities and configure access to your relational data services.

For detailed information on using Azure AD with Azure SQL database, visit the page What is Azure Active Directory authentication for SQL database on the Microsoft website. You can also authenticate users connecting to Azure Database for PostgreSQL and Azure Database for MySQL with AD.

Configure access control
Azure AD enables you to specify who, or what, can access your resources. Access control defines what a user or application can do with your resources once they've been authenticated.

Access management for cloud resources is a critical function for any organization that is using the cloud. Azure role-based access control (Azure RBAC) helps you manage who has access to Azure resources, and what they can do with those resources. For example, using RBAC you could:

- Allow one user to manage virtual machines in a subscription and another user to manage virtual networks.
- Allow a database administrator group to manage SQL databases in a subscription.
- Allow a user to manage all resources in a resource group, such as virtual machines, websites, and subnets.
- Allow an application to access all resources in a resource group.

You control access to resources using Azure RBAC to create role assignments. A role assignment consists of three elements: a security principal, a role definition, and a scope.

- A security principal is an object that represents a user, group, service principal, or managed identity that is requesting access to Azure resources.

- A role definition, often abbreviated to role, is a collection of permissions. A role definition lists the operations that can be performed, such as read, write, and delete. Roles can be given high-level names, like owner, or specific names, like virtual machine reader. Azure includes several built-in roles that you can use, including:

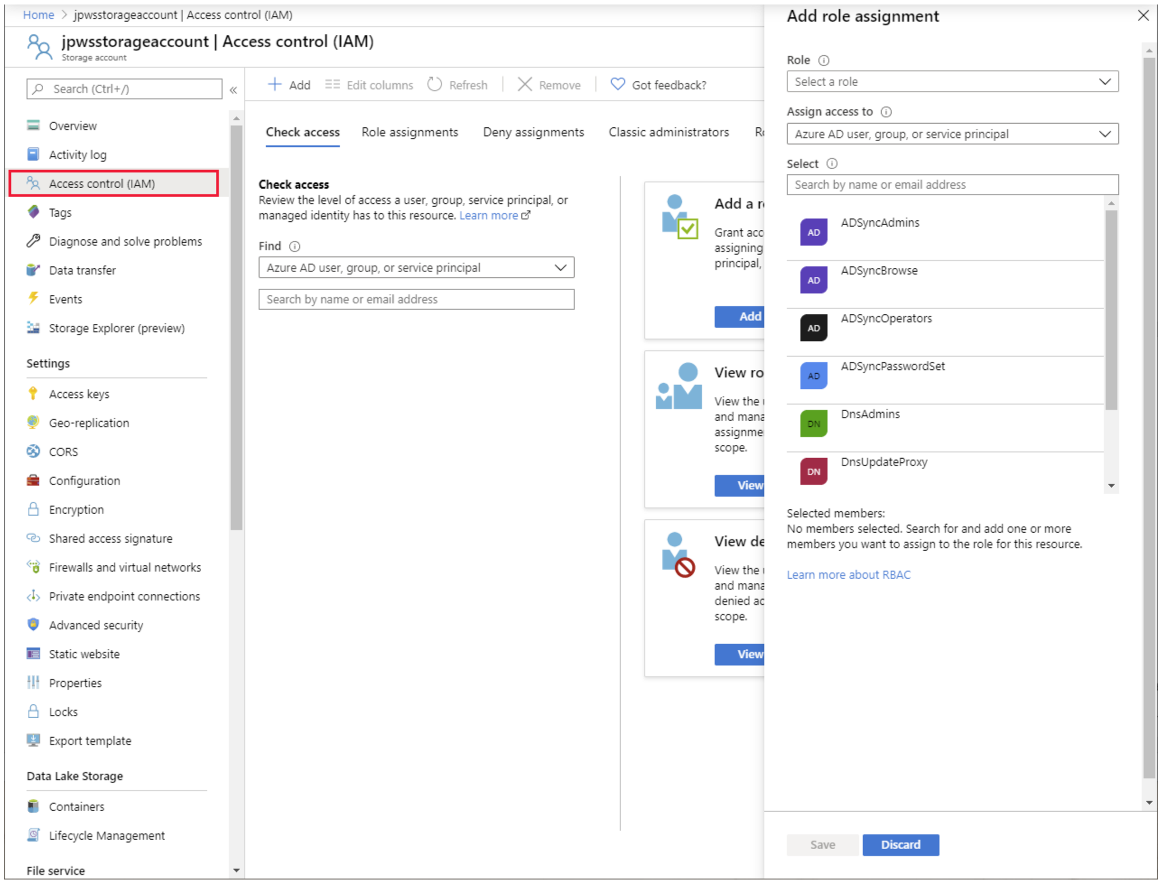
    - Owner - Has full access to all resources including the right to delegate access to others.

    - Contributor - Can create and manage all types of Azure resources but can't grant access to others.

- Reader- Can view existing Azure resources.
- User Access Administrator - Lets you manage user access to Azure resources.
- You can also create your own custom roles. For detailed information, see Create or update Azure custom roles using the Azure portal on the Microsoft website.

- A scope lists the set of resources that the access applies to. When you assign a role, you can further limit the actions allowed by defining a scope. This is helpful if, for example, you want to make someone a Website Contributor, but only for one resource group.

You add role assignments to a resource in the Azure portal using the Access control (IAM) page. The Role assignments tab enables you to associate a role with a security principal, defining the level of access the role has to the resource. For further information, read Add or remove Azure role assignments using the Azure portal.



Check access
Review the level of access a user, group, service principal, or managed identity has to this resource. Learn more

Find: Azure AD user, group, or service principal
Search by name or email address

Role: Select a role
Assign access to: Azure AD user, group, or service principal
Select: Search by name or email address

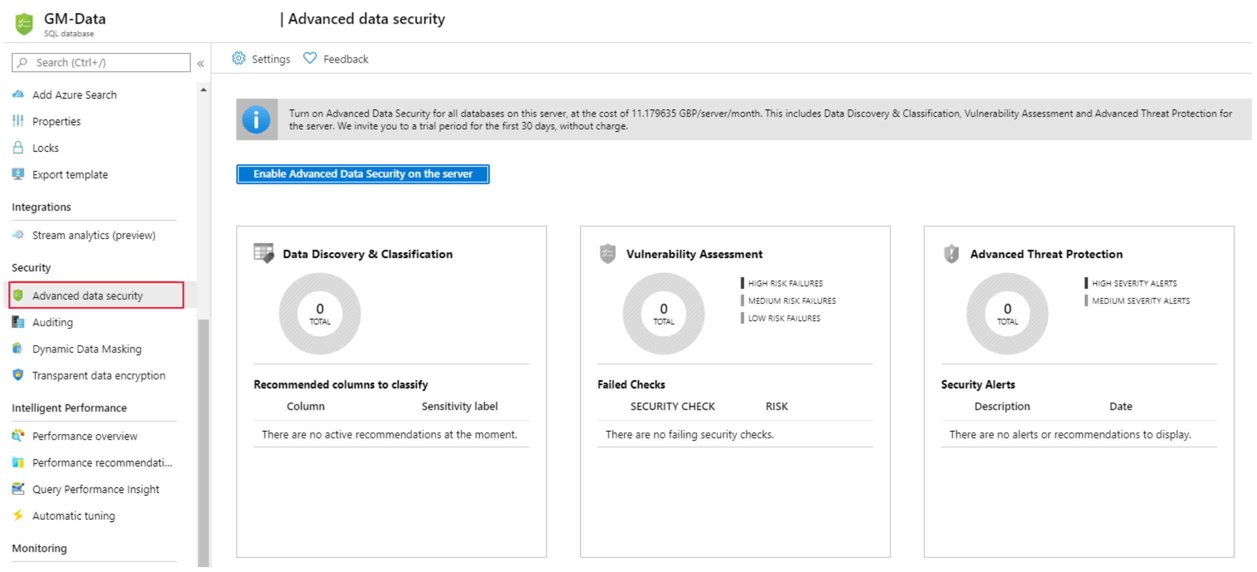
Selected members:
No members selected. Search for and add one or more members you want to assign to the role for this resource.

Configure advanced data security

Apart from authentication and authorization, many services provide additional protection through advanced data security.

Advanced data security implements threat protection and assessment. Threat protection adds security intelligence to your service. This intelligence monitors the service and detects unusual patterns of activity that could be harmful, or compromise the data managed by the service. Assessment identifies potential security vulnerabilities and recommends actions to mitigate them.

The image below shows the Advanced data security page for SQL database. The corresponding pages for MySQL and PostgreSQL are similar.



Turn on Advanced Data Security for all databases on this server, at the cost of 11.179635 GBP/server/month. This includes Data Discovery & Classification, Vulnerability Assessment and Advanced Threat Protection for the server. We invite you to a trial period for the first 30 days, without charge.

Recommended columns to classify

Column	Sensitivity label
There are no active recommendations at the moment.	

Failed Checks

SECURITY CHECK	RISK
There are no failing security checks.	

Security Alerts

Description	Date
There are no alerts or recommendations to display.	

V. Describe configuring Azure SQL Database, Azure Database for PostgreSQL and Azure Database for MySQL

Configure Azure SQL Database

The overarching principle for network security of the Azure SQL Database offering is to allow only the connection and communication that is necessary to allow the service to operate. All other ports, protocols, and connections are blocked by default. Virtual local area networks (VLANs) and access control lists (ACLs) are used to restrict network communications by source and destination networks, protocols, and port numbers.

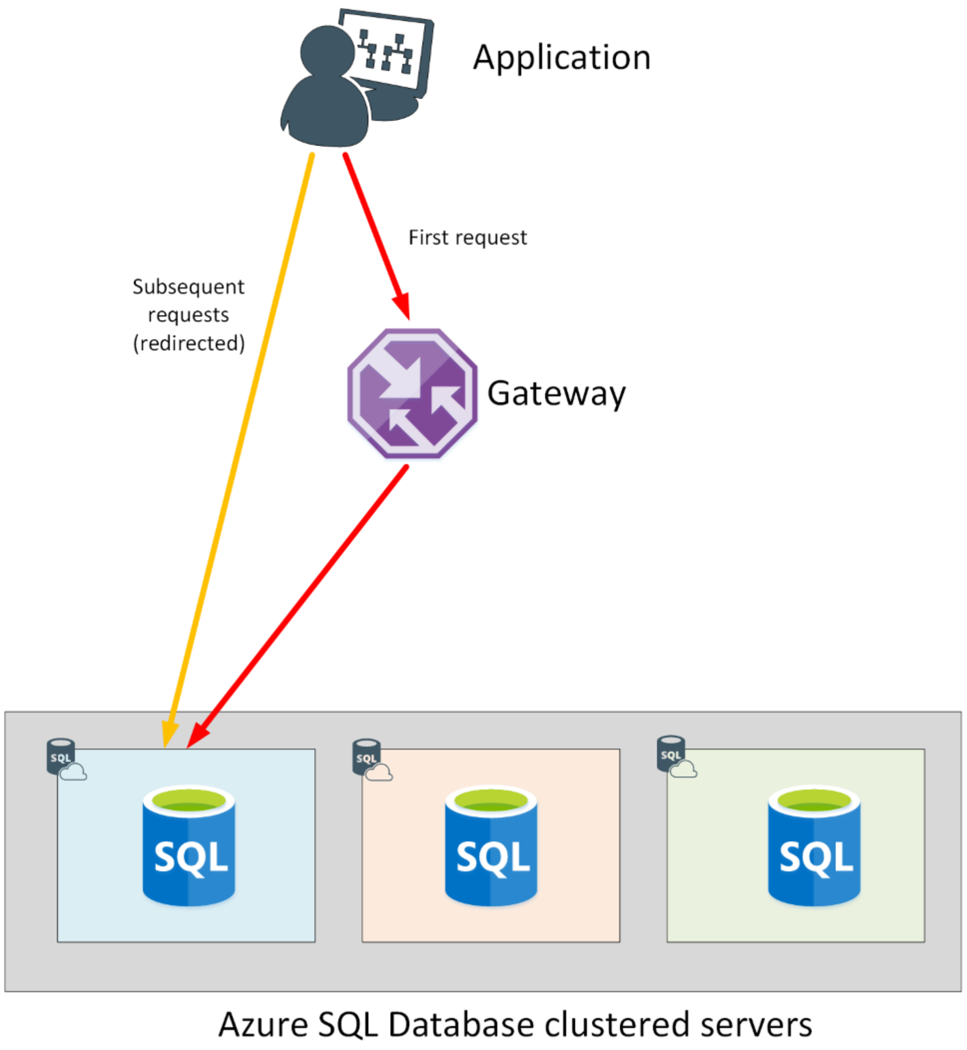
Items that implement network-based ACLs include routers and load balancers. You control traffic flow through these items by defining firewall rules.

The following steps describe how a connection is established to an Azure SQL database:

- Clients connect to a gateway that has a public IP address and listens on port 1433.
- Depending on the effective connection policy, the gateway either redirects traffic to the database cluster, or acts as a proxy for the database cluster.
- Inside the database cluster, traffic is forwarded to the appropriate Azure SQL database.

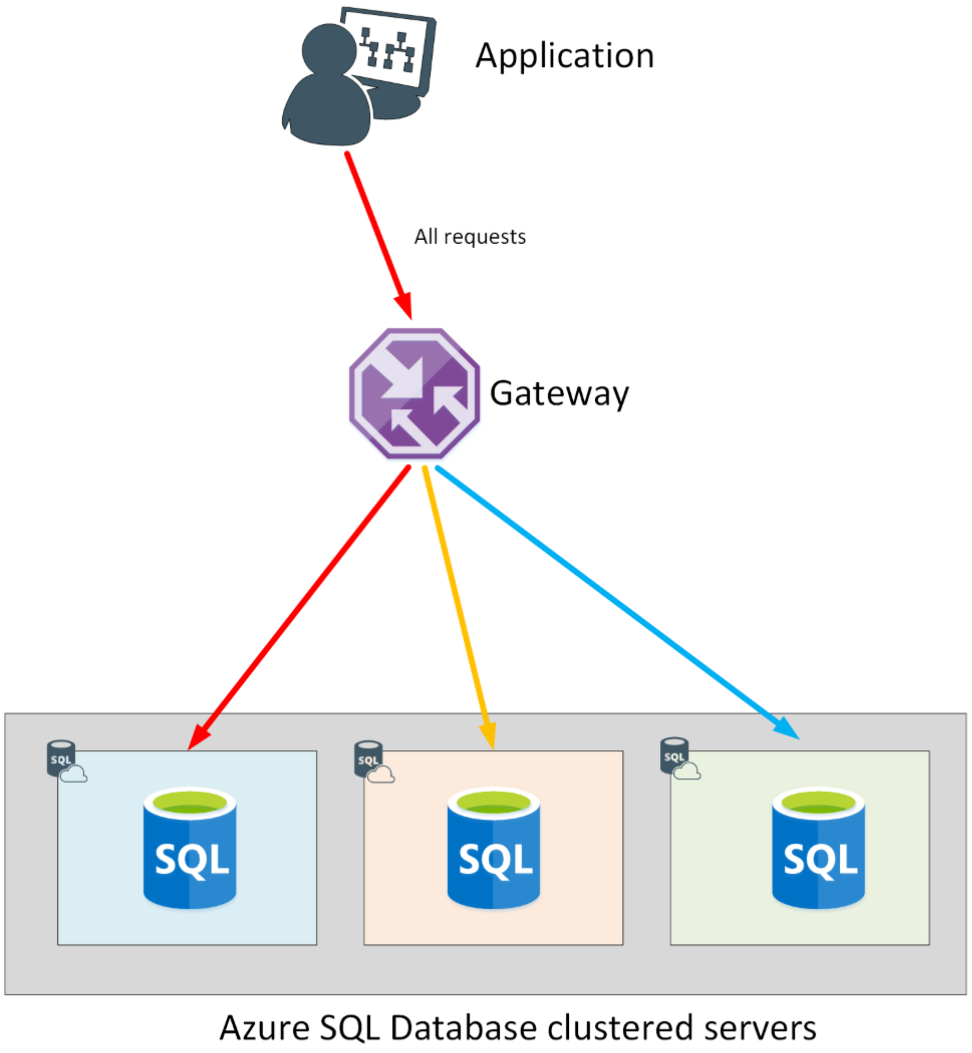
Connectivity from within Azure

If you're connecting from within another Azure service, such as a web application running under Azure App Service, your connections have a connection policy of Redirect by default. A policy of Redirect means that after your application establishes a connection to the Azure SQL database through the gateway, all following requests from your application will go directly to the database rather than through the gateway. If connectivity to the database subsequently fails, your application will have to reconnect through the gateway, when it might be directed to a different copy of the database running on another server in the cluster.


Application
First request
Subsequent requests (redirected)
Gateway
SQL
SQL
SQL
Azure SQL Database clustered servers

Connectivity from outside of Azure

If you're connecting from outside Azure, such as an on-premises application, your connections have a connection policy of Proxy by default. A policy of Proxy means the connection is established via the gateway, and all subsequent requests flow through the gateway. Each request could (potentially) be serviced by a different database in the cluster.



Configure DoSGuard

Denial of service (DoS) attacks are reduced by a SQL Database gateway service called DoSGuard. DoSGuard actively tracks failed logins from IP addresses. If there are multiple failed logins from a specific IP address within a period of time, the IP address is blocked from accessing any resources in the service for a short while.

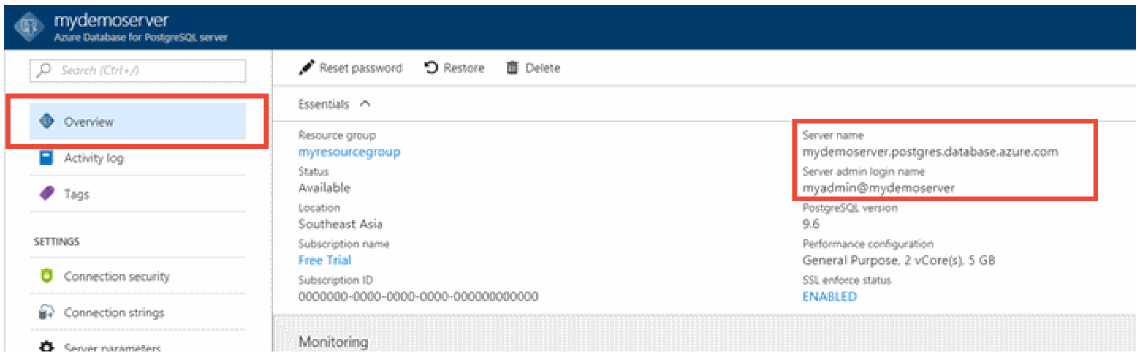
In addition, the Azure SQL Database gateway performs the following tasks:

- It validates all connections to the database servers, to ensure that they are from genuine clients.

- It encrypts all communications between a client and the database servers.
- It inspects each network packet sent over a client connection. The gateway validates the connection information in the packet, and forwards it to the appropriate physical server based on the database name that's specified in the connection string.

Configure Azure Database for PostgreSQL

When you create your Azure Database for PostgreSQL server, a default database named postgres is created. To connect to your database server, you need your full server name and admin sign-in credentials. You can easily find the server name and sign in information on the server Overview page in the portal. This page contains the Server name and the Server admin sign-in name. Port 5432



Resource group
myresourcegroup
Status
Available
Location
Southeast Asia
Subscription name
Free Trial
Subscription ID
0000000-0000-0000-0000-000000000000
Server name
mydemoserver.postgres.database.azure.com
Server admin login name
myadmin@mydemoserver
PostgreSQL version
9.6
Performance configuration
General Purpose, 2 vCore(s), 5 GB
SSL enforce status
ENABLED

Configure server parameters and extensions

A PostgreSQL database server has a number of configuration parameters that you can set. These parameters support fine-tuning of the database, and debugging of code in the database. You can modify these parameters using the Server parameters page in the Azure portal.

If you're familiar with PostgreSQL, you'll find that not all parameters are supported in Azure. The Server parameters page on the Microsoft website describes the PostgreSQL parameters that are available.

PostgreSQL also provides the ability to extend the functionality of your database using extensions. Extensions bundle multiple related SQL objects together in a single package that can be loaded or removed from your database with a single command. After being loaded in the database, extensions function like built-in features. You install an extension in your database before you can use it. To install a particular extension, run the CREATE EXTENSION command from psql tool to load the packaged objects into your database. Not all PostgreSQL extensions are supported in Azure. For a full list, read PostgreSQL extensions in Azure Database for PostgreSQL - Single Server.

Configure read replicas

You can replicate data from an Azure Database for PostgreSQL server to a read-only server. Azure Database for PostgreSQL supports replication from the master server to up to five replicas. Replicas are updated asynchronously with the PostgreSQL engine native replication technology.

Read replicas help to improve the performance and scale of read-intensive workloads. Read workloads can be isolated to the replicas, while write workloads can be directed to the master.

A common scenario is to have BI and analytical workloads use read replicas as the data source for reporting.

Because replicas are read-only, they don't directly reduce the burden of write operations on the master. This feature isn't targeted at write-intensive workloads.

Replicas are new servers that you manage similar to regular Azure Database for PostgreSQL servers. For each read replica, you're billed for the provisioned compute in vCores and storage in GB/month.

Use the Replication page for a PostgreSQL server in the Azure portal to add read replicas to your database:

Configure Azure Database for MySQL
In order to connect to the MySQL database you've provisioned, you'll need to enter the connection information. This information includes fully qualified server name and sign-in credentials. You can find this information on the Overview page for your server.

Configure server parameters
Like PostgreSQL, a MySQL database server has a number of configuration parameters that you can set. You can modify these parameters using the Server parameters page in the Azure portal.

Configure read replicas
This feature is similar to that available for PostgreSQL. You can create up to five read replicas for a MySQL database. This feature enables you to geo-replicate data across regions and distribute the overhead associated with read-intensive workloads. Replication is asynchronous from the master server, so there may be some lag between records being written at the master and becoming available across all replicas.

Read replication isn't intended to support write-heavy workloads.

Use the Replication page for a MySQL server in the Azure portal to add read replicas to your database.

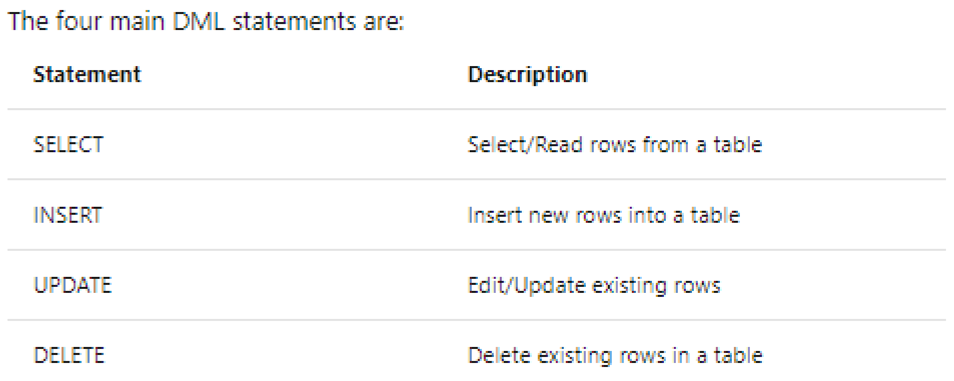
I. Introduction to SQL

Some popular dialects of SQL include:

- **Transact-SQL (T-SQL).** This version of SQL is used by Microsoft SQL Server and Azure SQL Database.
- **pgSQL.** This is the dialect, with extensions implemented in PostgreSQL.
- **PL/SQL.** This is the dialect used by Oracle. PL/SQL stands for Procedural Language/SQL.

SQL statements are grouped into two main logical groups, and they are:

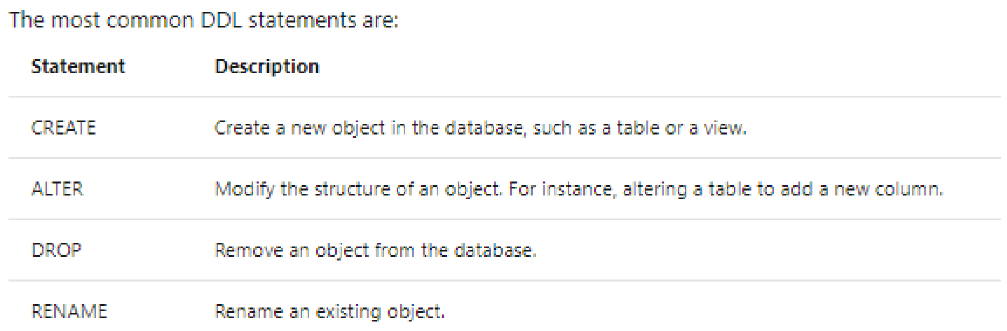
Data Manipulation Language (DML)



The four main DML statements are:

Statement	Description
SELECT	Select/Read rows from a table
INSERT	Insert new rows into a table
UPDATE	Edit/Update existing rows
DELETE	Delete existing rows in a table

Data Definition Language (DDL)



The most common DDL statements are:

Statement	Description
CREATE	Create a new object in the database, such as a table or a view.
ALTER	Modify the structure of an object. For instance, altering a table to add a new column.
DROP	Remove an object from the database.
RENAME	Rename an existing object.

The datatypes available for columns in a table will vary between database management systems. However, most database management systems support numeric types such as INT (an integer, or whole number), and string types such as VARCHAR (*VARCHAR* stands for variable length character data). For more information, see the documentation for your selected database management system.

II. Query relational data in Azure SQL Database

Retrieve connection information for Azure SQL Database

You can use any of these tools to query data held in Azure SQL Database:

- The query editor in the Azure portal
- The sqlcmd utility from the command line or the Azure Cloud Shell
- SQL Server Management Studio
- Azure Data Studio
- SQL Server Data Tools

To use these tools, you first need to establish a connection to the database. You'll require the details of the server to connect to, an Azure SQL Database account (a username and password) that has access to this server, and the name of the database to use on this server. You can find the server name for a database using the Azure portal: go to the page for your database, and on the Overview page note the fully qualified server name in the Server name field.

Some tools and applications require a connection string that identifies the server, database, account name, and password. You can find this information from the Overview page for a database in the Azure portal: select Show database connection strings.

Use the Azure portal to query a database

To access the query editor in the Azure portal, go to the page for your database and select Query editor. You'll be prompted for credentials. You can set the Authorization type to SQL Server authentication and enter the user name and password that you set up when you created the database. Or you can select Active Directory password authentication and provide the credentials of an authorized user in Azure Active Directory. If Active Directory single sign-on is enabled, you can connect by using your Azure identity.

You enter your SQL query in the query pane and then click Run to execute it. Any rows that are returned appear in the Results pane. The Messages pane displays information such as the number of rows returned, or any errors that occurred:

You can also enter INSERT, UPDATE, DELETE, CREATE, and DROP statements in the query pane.

Use SQLCMD to query a database

The sqlcmd utility runs from the command line and is also available in the Cloud Shell. You specify parameters that identify the server, database, and your credentials. The code below shows an example. Replace <server> with the name of the database server that you created, <database> with the name of your database, and <user name> and <password> with your credentials.

If the sign-in command succeeds, you'll see a 1> prompt. You can enter SQL commands, then type GO on a line by itself to run them.

Use Azure Data Studio
Azure Data Studio is a graphical utility for creating and running SQL queries from your desktop. For download and installation instructions, visit the Download and install Azure Data Studio page on the Microsoft website.

The first time you run Azure Data Studio the Welcome page should open. If you don't see the Welcome page, select Help, and then select Welcome. Select Create a connection to open the Connection pane:

a. Fill in the following fields using the server name, user name, and password for your Azure SQL Server:

b. Select Connect.

    i. If your server doesn't have a firewall rule allowing Azure Data Studio to connect, the Create new firewall rule form opens. Complete the form to create a new firewall rule. For details, see Create a server-level firewall rule using the Azure portal.

c. After successfully connecting, your server is available in the SERVERS sidebar on the Connections page. You can now use the New Query command to create and run scripts of SQL commands.

Use SQL Server Management Studio
Use SQL Server Data Tools in Visual Studio

III. Query relational data in Azure Database for PostgreSQL

Retrieve connection information for Azure Database for PostgreSQL
To connect to a PostgreSQL database, you require the name of the server, and the credentials for an account that has access rights to connect to the server. You can find the server name and the name of the default administrator account on the Overview page for the Azure Database for PostgreSQL instance in the Azure portal. Contact your administrator for the password. As with Azure SQL Database, you must open the PostgreSQL firewall to enable client applications to connect to the service.

Use psql to query a database
The psql utility is available in the Azure Cloud Shell. You can also run it from a command prompt on your desktop computer, but you must download and install the psql client. You can find the psql client on the postgresql.org website.

To connect to Azure Database for PostgreSQL using psql, perform the following operations:

a. Run the following command. Make sure to replace the server name and admin name with the values from the Azure portal.

```
psql --host=<server-name>.postgres.database.azure.com --username=<admin-user>@<server-name> --dbname=postgres
```

b. If your connection is successful, you'll see the prompt postgres=>.

c. You can create a new database with the following SQL command:

```
CREATE DATABASE "Adventureworks";
```

d. Inside psql, you can run the command \c Adventureworks to connect to the database.

e. You can create tables and insert data using CREATE and INSERT commands

f. You can retrieve the data you just added using the following SQL commands:

```
SELECT * FROM PEOPLE;
SELECT * FROM LOCATIONS;
```

g. Other psql commands include:

\l to list databases.
\dt to list the tables in the current database.
\q command to quit psql.

Connect to PostgreSQL database using Azure Data Studio
To connect to Azure Database for PostgreSQL from Azure Data Studio, you must first install the PostgreSQL extension.

On the Extensions page, search for postgresql, Select the PostgreSQL extension, and then select Install.

1. In Azure Data Studio, go to the SERVERS sidebar, and select New Connection.

2. In the Connection dialog box, in the Connection type drop-down list box, select PostgreSQL.

3. Fill in the remaining fields using the server name, user name, and password for your PostgreSQL server.

4. Select Connect to establish the connection. After successfully connecting, your server opens in the SERVERS sidebar. You can expand the Databases node to connect to

databases on the server and view their contents. Use the New Query command in the toolbar to create and run queries.

IV. Query relational data in Azure Database for MySQL

Retrieve connection information for Azure Database for MySQL
Like SQL Database and PostgreSQL, you require the name of the server, and the credentials for an account that has access rights to connect to the server. You can find the server name and the name of the default administrator account on the Overview page for the Azure Database for MySQL instance in the Azure portal. Contact your administrator for the password. You must also open the MySQL firewall to enable client applications to connect to the service.

You can download and install MySQL Workbench from the MySQL Community Downloads page.

To connect to Azure MySQL Server by using MySQL Workbench, perform the following steps:

1. Start MySQL Workbench on your computer.

2. On the Welcome page, select Connect to Database.

3. In the Connect to Database dialog box, enter the following information on the Parameters tab:

4. Select OK to create the connection. If the connection is successful, the query editor will open.

5. You can use this editor to create and run scripts of SQL commands.

6. To run the sample SQL Code, select the lightning bolt icon in the toolbar
7. The query results appear in the Result Grid section in the middle of the page. The Output list at the bottom of the page shows the status of each command as it is run.

Azure Data Fundamentals: Explore non-relational data in Azure

https://docs.microsoft.com/en-us/learn/paths/azure-data-fundamentals-explore-non-relational-data/
Azure Data Fundamentals: Explore non-relational data services in Azure

Data comes in all shapes and sizes, and can be used for a large number of purposes. Many organizations use relational databases to store this data. However, the relational model might not be the most appropriate schema. The structure of the data might be too varied to easily model as a set of relational tables. For example, the data might contain items such as video, audio, images, temporal information, large volumes of free text, encrypted information, or other types of data that aren't inherently relational. Additionally, the data processing requirements might not be best suited by attempting to convert this data into the relational format. In these situations, it may be better to use non-relational repositories that can store data in its original format, but that allow fast storage and retrieval access to this data.

Azure Data Fundamentals: Explore non-relational data in Azure

I. Explore Azure Table storage

Azure Table Storage implements the NoSQL key-value model. In this model, the data for an item is stored as a set of fields, and the item is identified by a unique key.

What is Azure Table Storage?
Azure Table Storage is a scalable key-value store held in the cloud. You create a table using an Azure storage account.

In an Azure Table Storage table, items are referred to as rows, and fields are known as columns. However, don't let this terminology confuse you by thinking that an Azure Table Storage table is like a table in a relational database. An Azure table enables you to store semi-structured data. All rows in a table must have a key, but apart from that the columns in each row can vary. Unlike traditional relational databases, Azure Table Storage tables have no concept of relationships, stored procedures, secondary indexes, or foreign keys. Data will usually be denormalized, with each row holding the entire data for a logical entity. For example, a table holding customer information might store the forename, lastname, one or more telephone numbers, and one or more addresses for each customer. The number of fields in each row can be different, depending on the number of telephone numbers and addresses for each customer, and the details recorded for each address. In a relational database, this information would be split across multiple rows in several tables. In this example, using Azure Table Storage provides much faster access to the details of a customer because the data is available in a single row, without requiring that you perform joins across relationships.

To help ensure fast access, Azure Table Storage splits a table into partitions. Partitioning is a mechanism for grouping related rows, based on a common property or partition key. Rows that share the same partition key will be stored together. Partitioning not only helps to organize data, it can also improve scalability and performance:

Partitions are independent from each other, and can grow or shrink as rows are added to, or removed from, a partition. A table can contain any number of partitions.

When you search for data, you can include the partition key in the search criteria. This helps to narrow down the volume of data to be examined, and improves performance by reducing the amount of I/O (reads and writes) needed to locate the data.

The key in an Azure Table Storage table comprises two elements; the partition key that identifies the partition containing the row (as described above), and a row key that is unique to each row in the same partition. Items in the same partition are stored in row key order. If an application adds a new row to a table, Azure ensures that the row is placed in the correct position in the table. In the example below, taken from an IoT scenario, the row key is a date and time value.

This scheme enables an application to quickly perform Point queries that identify a single row, and Range queries that fetch a contiguous block of rows in a partition.

In a point query, when an application retrieves a single row, the partition key enables Azure to quickly hone in on the correct partition, and the row key lets Azure identify the row in that partition. You might have hundreds of millions of rows, but if you've defined the partition and row keys carefully when you designed your application, data retrieval can be very quick. The partition key and row key effectively define a clustered index over the data.

In a range query, the application searches for a set of rows in a partition, specifying the start and end point of the set as row keys. This type of query is also very quick, as long as you have designed your row keys according to the requirements of the queries performed by your application.

The columns in a table can hold numeric, string, or binary data up to 64 KB in size. A table can have to 252 columns, apart from the partition and row keys. The maximum row size is 1 MB.

Use cases and management benefits of using Azure Table Storage

Azure Table Storage tables are schemaless. It's easy to adapt your data as the needs of your application evolve. You can use tables to hold flexible datasets such as user data for web applications, address books, device information, or other types of metadata your service requires. The important part is to choose the partition and row keys carefully.

The primary advantages of using Azure Table Storage tables over other ways of storing data include:

- It's simpler to scale. It takes the same time to insert data in an empty table, or a table with billions of entries. An Azure storage account can hold up to 500 TB of data.
- A table can hold semi-structured data
- There's no need to map and maintain the complex relationships typically required by a normalized relational database.
- Row insertion is fast
- Data retrieval is fast, if you specify the partition and row keys as query criteria

There are disadvantages to storing data this way though, including:

- Consistency needs to be given consideration as transactional updates across multiple entities aren't guaranteed
- There's no referential integrity; any relationships between rows need to be maintained externally to the table
- It's difficult to filter and sort on non-key data. Queries that search based on non-key fields could result in full table scans

Azure Table Storage is an excellent mechanism for:

- Storing TBs of structured data capable of serving web scale applications. Examples include product catalogs for eCommerce applications, and customer information, where the data can be quickly identified and ordered by a composite key. In the case of a product catalog, the partition key could be the product category (such as footwear), and the row key identifies the specific product in that category (such as climbing boots).
- Storing datasets that don't require complex joins, foreign keys, or stored procedures, and that can be denormalized for fast access. In an IoT system, you might use Azure Table Storage to capture device sensor data. Each device could have its own partition, and the data could be ordered by the date and time each measurement was captured.
- Capturing event logging and performance monitoring data. Event log and performance information typically contain data that is structured according to the type of event or performance measure being recorded. The data could be partitioned by event or performance measurement type, and ordered by the date and time it was recorded. Alternatively, you could partition data by date, if you need to analyze an ordered series of events and performance measures chronologically. If you want to analyze data by type and date/time, then consider storing the data twice, partitioned by type, and again by date. Writing data is fast, and the data is static once it has been recorded.

Azure Table Storage is intended to support very large volumes of data, up to several hundred TBs in size. As you add rows to a table, Azure Table Storage automatically manages the partitions in a table and allocates storage as necessary. You don't need to take any additional steps yourself.

Azure Table Storage provides high-availability guarantees in a single region. The data for each table is replicated three times within an Azure region. For increased availability, but at additional cost, you can create tables in geo-redundant storage. In this case, the data for each table is replicated a further three times in another region several hundred miles away. If a replica in the local region becomes unavailable, Azure will transparently switch to a working replica while the failed replica is recovered. If an entire region is hit by an outage, your tables are safe in a remote region, and you can quickly switch your application to connect to that remote region.

Azure Table Storage helps to protect your data. You can configure security and role-based access control to ensure that only the people or applications that need to see your data can actually retrieve it.

Create and view a table using the Azure portal

The simplest way to create a table in Azure Table Storage is to use the Azure portal. Follow these steps:

1. Sign into the Azure portal using your Azure account.
2. On the home page of the Azure portal, select +Create a resource.
3. On the New page, select Storage account - blob, file, table, queue
4. On the Create storage account page, enter the following details, and then select Review + create.
5. On the validation page, click Create, and wait while the new storage account is configured.
6. When the Your deployment is complete page appears, select Go to resource.
7. On the Overview page for the new storage account, select Tables.
8. On the Tables page, select + Table.
9. In the Add table dialog box, enter testtable for the name of the table, and then select OK.
10. When the new table has been created, select Storage Explorer.
11. On the Storage Explorer page, expand Tables, and then select testtable. Select Add to insert a new entity (rows) into the table.
12. In the Add Entity dialog box, enter your own values for the PartitionKey and RowKey properties, and then select Add Property. Add a String property called Name and set the value to your name. Select Add Property again, and add a Double property (this is numeric) named Age, and set the value to your age. Select Insert to save the entity.
13. Verify that the new entity has been created. The entity should contain the values you specified, together with a timestamp that contains the date and time that the entity was created.
14. If time allows, experiment with creating additional entities. Not all entities must have the same properties. You can use the Edit function to modify the values in entity, and add or remove properties. The Query function enables you to find entities that have properties with a specified set of values.

II. Explore Azure Blob Storage

What is Azure Blob storage?

Azure Blob storage is a service that enables you to store massive amounts of unstructured data, or blobs, in the cloud. Like Azure Table storage, you create blobs using an Azure storage account.

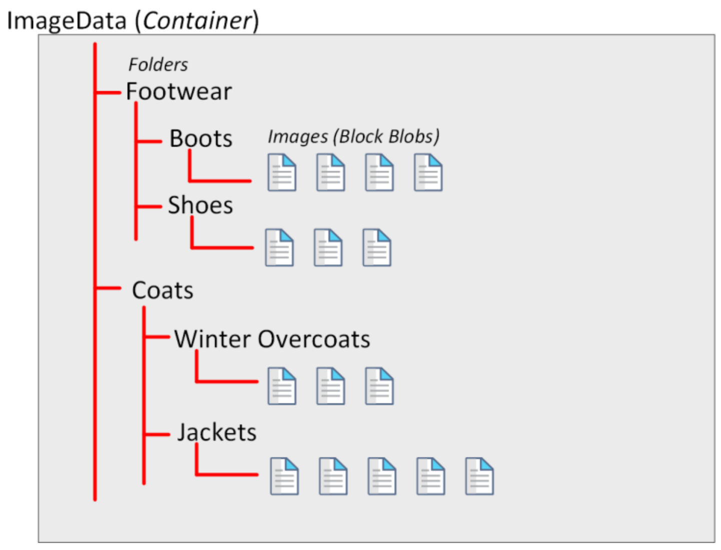
Azure currently supports three different types of blob:

- Block blobs. A block blob is handled as a set of blocks. Each block can vary in size, up to 100 MB. A block blob can contain up to 50,000 blocks, giving a maximum size of over 4.7 TB. The block is the smallest amount of data that can be read or written as an individual unit. Block blobs are best used to store discrete, large, binary objects that change infrequently.

- Page blobs. A page blob is organized as a collection of fixed size 512-byte pages. A page blob is optimized to support random read and write operations; you can fetch and store data for a single page if necessary. A page blob can hold up to 8 TB of data. Azure uses page blobs to implement virtual disk storage for virtual machines.

- Append blobs. An append blob is a block blob optimized to support append operations. You can only add blocks to the end of an append blob; updating or deleting existing blocks isn't supported. Each block can vary in size, up to 4 MB. The maximum size of an append blob is just over 195 GB.

Inside an Azure storage account, you create blobs inside containers. A container provides a convenient way of grouping related blobs together, and you can organize blobs in a hierarchy of folders, similar to files in a file system on disk. You control who can read and write blobs inside a container at the container level.



Blob storage provides three access tiers, which help to balance access latency and storage cost:

- The Hot tier is the default. You use this tier for blobs that are accessed frequently. The blob data is stored on high-performance media.

- The Cool tier. This tier has lower performance and incurs reduced storage charges compared to the Hot tier. Use the Cool tier for data that is accessed infrequently. It's common for newly created blobs to be accessed frequently initially, but less so as time passes. In these situations, you can create the blob in the Hot tier, but migrate it to the Cool tier later. You can migrate a blob form the Cool tier back to the Hot tier.

- The Archive tier. This tier provides the lowest storage cost, but with increased latency. The Archive tier is intended for historical data that mustn't be lost, but is required only rarely. Blobs in the Archive tier are effectively stored in an offline state. Typical reading latency for the Hot and Cool tiers is a few milliseconds, but for the Archive tier, it can take hours for the data to become available. To retrieve a blob from the Archive tier, you must change the access tier to Hot or Cool. The blob will then be rehydrated. You can read the blob only when the rehydration process is complete.

You can create lifecycle management policies for blobs in a storage account. A lifecycle management policy can automatically move a blob from Hot to Cool, and then to the Archive tier, as it ages and is used less frequently (policy is based on the number of days since modification). A lifecycle management policy can also arrange to delete outdated blobs.

Use cases and management benefits of using Azure Blob Storage
Common uses of Azure Blob Storage include:

- Serving images or documents directly to a browser, in the form of a static website. Visit Static website hosting in Azure storage for detailed information.
- Storing files for distributed access
- Streaming video and audio
- Storing data for backup and restore, disaster recovery, and archiving
- Storing data for analysis by an on-premises or Azure-hosted service

To ensure availability, Azure Blob storage provides redundancy. Blobs are always replicated three times in the region in which you created your account, but you can also select geo-redundancy, which replicates your data in a second region (at additional cost).

Other features available with Azure Blob storage include:

- Versioning. You can maintain and restore earlier versions of a blob.

- Soft delete. This feature enables you to recover a blob that has been removed or overwritten, by accident or otherwise.

- Snapshots. A snapshot is a read-only version of a blob at a particular point in time.

- Change Feed. The change feed for a blob provides an ordered, read-only, record of the updates made to a blob. You can use the change feed to monitor these changes, and perform operations such as:

    - Update a secondary index, synchronize with a cache, search-engine, or any other content-management scenarios.
    - Extract business analytics insights and metrics, based on changes that occur to your objects, either in a streaming manner or batched mode.
    - Store, audit, and analyze changes to your objects, over any period of time, for security, compliance or intelligence for enterprise data management.
    - Build solutions to back up, mirror, or replicate object state in your account for disaster management or compliance.
    - Build connected application pipelines that react to change events or schedule executions based on created or changed objects.

Create and view a block blob using the Azure portal

You can create block blobs using the Azure portal. Remember that blobs are stored in containers, and you create a container using a storage account. The following steps assume you've created the storage account described in the previous unit.

1. In the Azure portal, on the left-hand navigation menu, select Home.
2. On the home page, select Storage accounts.
3. On the Storage accounts page, select the storage account you created in the previous unit.
4. On the Overview page for your storage account, select Storage Explorer.
5. On the Storage Explorer page, right-click BLOB CONTAINERS, and then select Create blob container.
6. In the New Container dialog box, give your container a name, accept the default public access level, and then select Create.
7. In the Storage Explorer window, expand BLOB CONTAINERS, and then select your new blob container.
8. In the blobs window, select Upload.
9. In the Upload blob dialog box, use the files button to pick a file of your choice on your computer, and then select Upload
10. When the upload has completed, close the Upload blob dialog box. Verify that the block blob appears in your container.
11. If you have time, you can experiment uploading other files as block blobs. You can also download blobs back to your computer using the Download button.

III. Explore Azure File Storage

What is Azure File Storage?

Azure File Storage enables you to create files and shares in the cloud, and access these file shares from anywhere with an internet connection. Azure File Storage exposes file shares using the Server Message Block 3.0 (SMB) protocol. This is the same file sharing protocol used by many existing on-premises applications. These applications should continue to work unchanged if you migrate your file shares to the cloud. The applications can be running on-premises, or in the cloud. You can control access to shares in Azure File Storage using authentication and authorization services available through Azure Active Directory Domain Services.

You create Azure File storage in a storage account. Azure File Storage enables you to share up to 100 TB of data in a single storage account. This data can be distributed across any number of file shares in the account. The maximum size of a single file is 1 TiB, but you can set quotas to limit the size of each share below this figure. Currently, Azure File Storage supports up to 2000 concurrent connections per shared file.

Once you've created a storage account, you can upload files to Azure File Storage using the Azure portal, or tools such as the AzCopy utility. You can also use the Azure File Sync service to synchronize locally cached copies of shared files with the data in Azure File Storage.

Azure File Storage offers two performance tiers. The Standard tier uses hard disk-based hardware in a datacenter, and the Premium tier uses solid-state disks. The Premium tier offers greater throughput, but is charged at a higher rate.

Use cases and management benefits of using Azure File Storage

Azure File Storage is designed to support many scenarios, including the following:

1. Migrate existing applications to the cloud.

   Many existing applications access data using file-based APIs, and are designed to share data using SMB file shares. Azure File Storage enables you to migrate your on-premises file or file share-based applications to Azure without having to provision or manage highly available file server virtual machines.

2. Share server data across on-premises and cloud.

   Customers can now store server data such as log files, event data, and backups in the cloud to leverage the availability, durability, scalability, and geo redundancy built into the Azure storage platform. With encryption in SMB 3.0, you can securely mount Azure File Storage shares from anywhere. Applications running in the cloud can share data with on-premises applications using the same consistency guarantees implemented by on-premises SMB servers.

3. Integrate modern applications with Azure File Storage.

By leveraging the modern REST API that Azure File Storage implements in addition to SMB 3.0, you can integrate legacy applications with modern cloud applications, or develop new file or file share-based applications.

4. Simplify hosting High Availability (HA) workload data.

Azure File Storage delivers continuous availability so it simplifies the effort to host HA workload data in the cloud. The persistent handles enabled in SMB 3.0 increase availability of the file share, which makes it possible to host applications such as SQL Server and IIS in Azure with data stored in shared file storage.

Azure Files Storage is a fully managed service. Your shared data is replicated locally within a region, but can also be geo-replicated to a second region.

Azure aims to provide up to 300 MB/second of throughput for a single Standard file share, but you can increase throughput capacity by creating a Premium file share, for additional cost.

All data is encrypted at rest, and you can enable encryption for data in-transit between Azure File Storage and your applications.

Create an Azure storage file share using the Azure portal

You can create Azure storage file shares using the Azure portal. The following steps assume you've created the storage account described in unit 2.

1. In the Azure portal, on the hamburger menu, select Home.
2. On the home page, select Storage accounts.
3. On the Storage accounts page, select the storage account you created in the unit 2.
4. On the Overview page for your storage account, select Storage Explorer.
5. On the Storage Explorer page, right-click FILE SHARES, and then select Create file share.
6. In the New file share dialog box, enter a name for your file share, leave Quota empty, and then select Create.
7. In the Storage Explorer window, expand FILE SHARES, and select your new file share, and then select Upload.

8. In the Upload files dialog box, use the files button to pick a file of your choice on your computer, and then select Upload
9. When the upload has completed, close the Upload files dialog box. Verify that the file appears in file share.

IV. Explore Azure Cosmos DB

What is Azure Cosmos DB?

Azure Cosmos DB is a multi-model NoSQL database management system. Cosmos DB manages data as a partitioned set of documents. A document is a collection of fields, identified by a key. The fields in each document can vary, and a field can contain child documents. Many document databases use JSON (JavaScript Object Notation) to represent the document structure. In this format, the fields in a document are enclosed between braces, { and }, and each field is prefixed with its name. The example below shows a pair of documents representing customer information. In both cases, each customer document includes child documents containing the name and address, but the fields in these child documents vary between customers.

A document can hold up to 2 MB of data, including small binary objects. If you need to store larger blobs as part of a document, use Azure Blob storage, and add a reference to the blob in the document.

Cosmos DB provides APIs that enable you to access these documents using a set of well-known interfaces.

The APIs that Cosmos DB currently supports include:

- SQL API. This interface provides a SQL-like query language over documents, enable to identify and retrieve documents using SELECT statements. The example below finds the address for customer 103248 in the documents shown above:
- Table API. This interface enables you to use the Azure Table Storage API to store and retrieve documents. The purpose of this interface is to enable you to switch from Table Storage to Cosmos DB without requiring that you modify your existing applications.
- MongoDB API. MongoDB is another well-known document database, with its own programmatic interface. Many organizations run MongoDB on-premises. You can use the MongoDB API for Cosmos DB to enable a MongoDB application to run unchanged against a Cosmos DB database. You can migrate the data in the MongoDB database to Cosmos DB running in the cloud, but continue to run your existing applications to access this data.
- Cassandra API. Cassandra is a column family database management system. This is another database management system that many organizations run on-premises. The Cassandra API for Cosmos DB provides a Cassandra-like programmatic interface for Cosmos DB. Cassandra API requests are mapped to Cosmos DB document requests. As with the MongoDB API, the primary purpose of the Cassandra API is to enable you to quickly migrate Cassandra databases and applications to Cosmos DB.
- Gremlin API. The Gremlin API implements a graph database interface to Cosmos DB. A graph is a collection of data objects and directed relationships. Data is still held as a set

of documents in Cosmos DB, but the Gremlin API enables you to perform graph queries over data. Using the Gremlin API you can walk through the objects and relationships in the graph to discover all manner of complex relationships, such as "What is the name of the pet of Sam's landlord?" in the graph shown below.

Documents in a Cosmos DB database are organized into containers. The documents in a container are grouped together into partitions. A partition holds a set of documents that share a common partition key. You designate one of the fields in your documents as the partition key. You should select a partition key that collects all related documents together. This approach helps to reduce the amount of I/O (disk reads) that queries might need to perform when retrieving a set of documents for a given entity. For example, in a document database for an ecommerce system recording the details of customers and the orders they've placed, you could partition the data by customer ID, and store the customer and order details for each customer in the same partition. To find all the information and orders for a customer, you simply need to query that single partition:

There's a superficial similarity between a Cosmos DB container and a table in Azure Table storage: in both cases, data is partitioned and documents (rows in a table) are identified by a unique ID within a partition. However, the similarity ends there. Unlike Azure Table storage, documents in a Cosmos DB partition aren't sorted by ID. Instead, Cosmos DB maintains a separate index. This index contains not only the document IDs, but also tracks the value of every other field in each document. This index is created and maintained automatically. This index enables you to perform queries that specify criteria referencing any fields in a container, without incurring the need to scan the entire partition to find that data. For a detailed description of how Cosmos DB indexing works, read Indexing in Azure Cosmos DB - Overview.

Use cases and management benefits of using Azure Cosmos DB

Cosmos DB is a highly scalable database management system. Cosmos DB automatically allocates space in a container for your partitions, and each partition can grow up to 10 GB in size. Indexes are created and maintained automatically. There's virtually no administrative overhead.

To ensure availability, all databases are replicated within a single region. This replication is transparent, and failover from a failed replica is automatic. Cosmos DB guarantees 99.999% high availability.

Additionally, you can choose to replicate data across regions, at additional cost. This feature enables you to place copies of data anywhere in the world, and enable applications to connect to the copy of the data that happens to be the closest, reducing query latency. All replicas are synchronized, although there may be a small window while updates are transmitted and applied. The multi-master replication protocol supports five well-defined consistency choices - strong, bounded staleness, session, consistent prefix, and eventual.

Cosmos DB guarantees less than 10-ms latencies for both reads (indexed) and writes at the 99th percentile, all around the world. This capability enables sustained ingestion of data and fast queries for highly responsive apps.

Cosmos DB is certified for a wide array of compliance standards. Additionally, all data in Cosmos DB is encrypted at rest and in motion. Cosmos DB provides row level authorization and adheres to strict security standards.

Cosmos DB is a foundational service in Azure. Cosmos DB has been used by many of Microsoft's products for mission critical applications at global scale, including Skype, Xbox, Microsoft 365, Azure, and many others. Cosmos DB is highly suitable for the following scenarios:

- IoT and telematics. These systems typically ingest large amounts of data in frequent bursts of activity. Cosmos DB can accept and store this information very quickly. The data can then be used by analytics services, such as Azure Machine Learning, Azure HDInsight, and Power BI. Additionally, you can process the data in real-time using Azure Functions that are triggered as data arrives in the database.
- Retail and marketing. Microsoft uses CosmosDB for its own e-commerce platforms that run as part of Windows Store and Xbox Live. It's also used in the retail industry for storing catalog data and for event sourcing in order processing pipelines.
- Gaming. The database tier is a crucial component of gaming applications. Modern games perform graphical processing on mobile/console clients, but rely on the cloud to deliver customized and personalized content like in-game stats, social media integration, and high-score leaderboards. Games often require single-millisecond latencies for reads and write to provide an engaging in-game experience. A game database needs to be fast and be able to handle massive spikes in request rates during new game launches and feature updates.
- Web and mobile applications. Azure Cosmos DB is commonly used within web and mobile applications, and is well suited for modeling social interactions, integrating with third-party services, and for building rich personalized experiences. The Cosmos DB SDKs can be used to build rich iOS and Android applications using the popular Xamarin framework.

Azure Data Fundamentals: Explore provisioning and deploying non-relational data services in Azure

I. Describe provisioning non-relational data services

What is provisioning?

Provisioning is the act of running a series of tasks that a service provider, such as Azure Cosmos DB, performs to create and configure a service. Behind the scenes, the service provider will set up the various resources (disks, memory, CPUs, networks, and so on) required to run the service. You'll be assigned these resources, and they remain allocated to you (and charged to you), until you delete the service.

How the service provider provisions resources is opaque, and you don't need to be concerned with how this process works. All you do is specify parameters that determine the size of the resources required (how much disk space, memory, computing power, and network bandwidth). These parameters are determined by estimating the size of the workload that you intend to run using the service. In many cases, you can modify these parameters after the service has been created, perhaps increasing the amount of storage space or memory if the workload is greater than you initially anticipated. The act of increasing (or decreasing) the resources used by a service is called scaling.

Can use Azure Portal, Azure CLI , Azure PowerShell and Azure Resource manager Templates.

II. Provision Azure Cosmos DB

How to provision a Cosmos DB account

You can provision a Cosmos DB account interactively using the Azure portal, or you can perform this task programmatically through the Azure CLI, Azure PowerShell, or an Azure Resource Manager template.

If you prefer to use the Azure CLI or Azure PowerShell, you can run the following commands to create a Cosmos DB account. The parameters to these commands correspond to many of the options you can select using the Azure portal. The examples shown below create an account for the Core(SQL) API, with geo-redundancy between the EastUS and WestUS regions, and support for multi-region writes.

Azure CLI

```
az cosmosdb create \
  --subscription <your-subscription> \
  --resource-group <resource-group-name> \
  --name <cosmosdb-account-name> \
  --locations regionName=eastus failoverPriority=0 \
  --locations regionName=westus failoverPriority=1 \
  --enable-multiple-write-locations

## Azure PowerShell

New-AzCosmosDBAccount `
  -ResourceGroupName "<resource-group-name>" `
  -Name "<cosmosbd-account-name>" `
  -Location @("West US", "East US") `
  -EnableMultipleWriteLocations
```

The other deployment option is to use an Azure Resource Manager template. The template for Cosmos DB can be rather lengthy, because of the number of parameters. To make life easier, Microsoft has published a number of example templates for handling different configurations.

How to create a database and a container

First Provision the DB and then provision the container(s) – think of the containers as tables
An Azure Cosmos DB account by itself doesn't really provide any resources other than a few pieces of static infrastructure. Databases and containers are the primary resource consumers. Resources are allocated in terms of the storage space required to hold your databases and containers, and the processing power required to store and retrieve data. Azure Cosmos DB uses the concept of Request Units per second (RU/s) to manage the performance and cost of databases. This measure abstracts the underlying physical resources that need to be provisioned to support the required performance.

You can think of a request unit as the amount of computation and I/O resources required to satisfy a simple read request made to the database. Microsoft gives a measure of approximately one RU as the resources required to read a 1-KB document with 10 fields. So a throughput of one RU per second (RU/s) will support an application that reads a single 1-KB document each second. You can specify how many RU/s of throughput you require when you create a database or when you create individual containers in a database. If you specify throughput for a database, all the containers in that database share that throughput. If you specify throughput for a container, the container gets that throughput all to itself.

If you underprovision (by specifying too few RU/s), Cosmos DB will start throttling performance. Once throttling begins, requests will be asked to retry later when hopefully there are available resources to satisfy it. If an application makes too many attempts to retry a throttled request, the request could be aborted. The minimum throughput you can allocate to a database or container is 400 RU/s. You can increase and decrease the RU/s for a container at any time. Allocating more RU/s increases the cost. However, once you allocate throughput to a database or container, you'll be charged for the resources provisioned, whether you use them or not.

If you prefer to use the Azure CLI or Azure PowerShell, you can run the following commands to create documents and containers. The code below shows some examples:

```
## Azure CLI - create a database

az cosmosdb sql database create \
  --account-name <cosmos-db-account-name> \
  --name <database-name> \
  --resource-group <resource-group-name> \
  --subscription <your-subscription> \
  --throughput <number-of-RU/s>

## Azure CLI - create a container

az cosmosdb sql container create \
  --account-name <cosmos-db-account-name> \
  --database-name <database-name> \
  --name <container-name> \
  --resource-group <resource-group-name> \
  --partition-key-path <key-field-in-documents>

## Azure PowerShell - create a database

Set-AzCosmosDBSqlDatabase `
   -ResourceGroupName "<resource-group-name>" `
   -AccountName "<cosmos-db-account-name>" `
   -Name "<database-name>" `
   -Throughput <number-of-RU/s>

## Azure PowerShell - create a container

Set-AzCosmosDBSqlContainer `
   -ResourceGroupName "<resource-group-name>" `
   -AccountName "<cosmos-db-account-name>" `
   -DatabaseName "<database-name>" `
   -Name "<container-name>" `
   -PartitionKeyKind Hash `
   -PartitionKeyPath "<key-field-in-documents>"
```

III. Provision other non-relational data services

How to create a storage account

Use the Azure portal

Use the Create storage account page to set up a new storage account using the Azure portal.

On the Basics tab, provide for the following details:

- Subscription. Select your Azure subscription.
- Resource Group. Either select an existing resource group, or create a new one, as appropriate.
- Storage account name. As with a Cosmos DB account, each storage account must have a unique name that hasn't already been used by someone else.
- Location. Select the region that is nearest to you if you're in the process of developing a new application, or the region nearest to your users if you're deploying an existing application.
- Performance. This setting has two options:
    - Standard storage accounts are based on hard disks. They're the lowest cost of the two storage options, but have higher latency. This type of storage account is suitable for applications that require bulk storage that is accessed infrequently, such as archives.
    - Premium storage uses solid-state drives, and has much lower latency and better read/write performance than standard storage. Solid-state drives are best used for I/O intensive applications, such as databases. You can also use premium storage to hold Azure virtual machine disks. A premium storage account is more expensive than a standard account.
- Account kind. Azure storage supports several different types of account:
    - General-purpose v2. You can use this type of storage account for blobs, files, queues, and tables, and is recommended for most scenarios that require Azure Storage. If you want to provision Azure Data Lake Storage, you should specify this account type.
    - General-purpose v1. This is a legacy account type for blobs, files, queues, and tables. Use general-purpose v2 accounts when possible.
    - BlockBlobStorage. The type of storage account is only available for premium accounts. You use this account type for block blobs and append blobs. It's

    recommended for scenarios with high transaction rates, or that use smaller objects, or require consistently low storage latency.
  - FileStorage. This type is also only available for premium accounts. You use it to create files-only storage accounts with premium performance characteristics. It's recommended for enterprise or high-performance scale applications. Use this type if you're creating an account to support File Storage.
  - BlobStorage. This is another legacy account type that can only hold blobs. Use general-purpose v2 accounts instead, when possible. You can use this account type for Azure Data Lake storage, but the General-purpose v2 account type is preferable.

- Replication. Data in an Azure Storage account is always replicated three times in the region you specify as the primary location for the account. Azure Storage offers two options for how your data is replicated in the primary region:

  - Locally redundant storage (LRS) copies your data synchronously three times within a single physical location in the region. LRS is the least expensive replication option, but isn't recommended for applications requiring high availability.

  - Geo-redundant storage (GRS) copies your data synchronously three times within a single physical location in the primary region using LRS. It then copies your data asynchronously to a single physical location in the secondary region. This form of replication protects you against regional outages.

  - Read-access geo-redundant storage (RA-GRS) replication is an extension of GRS that provides direct read-only access to the data in the secondary location. In contrast, the GRS option doesn't expose the data in the secondary location, and it's only used to recover from a failure in the primary location. RA-GRS replication enables you to store a read-only copy of the data close to users that are located in a geographically distant location, helping to reduce read latency times.

    - *** NOTE: To maintain performance, premium storage accounts only support LRS replication. This is because replication is performed synchronously to maintain data integrity. Replicating data to a distant region can increase latency to the point at which any advantages of using premium storage are lost.

- Access tier. This option is only available for standard storage accounts. You can select between Hot and Cool.

  The hot access tier has higher storage costs than cool and archive tiers, but the lowest access costs. Example usage scenarios for the hot access tier include:
  - Data that's in active use or expected to be accessed (read from and written to) frequently.
  - Data that's staged for processing and eventual migration to the cool access tier.

The cool access tier has lower storage costs and higher access costs compared to hot storage. This tier is intended for data that will remain in the cool tier for at least 30 days. Example usage scenarios for the cool access tier include:

- Short-term backup and disaster recovery datasets.
- Older media content not viewed frequently anymore but is expected to be available immediately when accessed.
- Large data sets that need to be stored cost effectively while more data is being gathered for future processing. For example, long-term storage of scientific data, or raw telemetry data from a manufacturing facility.

Use the Azure CLI

If you're using the Azure CLI, run the az storage account command to create a new storage account. The example below summarizes the options available:

```
az storage account create \
  --name <storage-account-name> \
  --resource-group <resource-group> \
  --location <your-location> \
  --sku <sku> \
  --kind <kind> \
  --access-tier <tier>
```

The sku is combination of the performance tier and replication options. It can be one of Premium_LRS, Premium_ZRS, Standard_GRS, Standard_GZRS, Standard_LRS, Standard_RAGRS, Standard_RAGZRS, or Standard_ZRS.

The kind parameter should be one of BlobStorage, BlockBlobStorage, FileStorage, Storage, or StorageV2.

The access-tier parameter can either be Cool or Hot.

Use Azure PowerShell

You use the New-AzStorageAccount PowerShell cmdlet to create a new storage account, as follows:

```
New-AzStorageAccount `
  -Name "<storage-account-name>" `
  -ResourceGroupName "<resource-group-name>" `
  -Location "<your-location>" `
  -SkuName "<sku>" `
  -Kind "<kind>" `
  -AccessTier "<tier>"
```

How to provision Data Lake storage in a storage account

Use the Azure portal

If you're provisioning a Data Lake storage, you must specify the appropriate configuration settings when you create the storage account. You can't configure Data Lake storage after the storage account has been set up.

In the Azure portal, on the Advanced tab of the Create storage account page, in the Data Lake Storage Gen2 section, select Enabled for the Hierarchical namespace option.

After the storage account has been created, you can add one or more Data Lake Storage containers to the account. Each container supports a directory structure for storing Data Lake files.

Use the Azure CLI

Run the az storage account command with the enable-hierarchical-namespace parameter to create a new storage account that supports Data Lake Storage:

```
az storage account create \
  --name <storage-account-name> \
  --resource-group <resource-group> \
  --location <your-location> \
  --sku <sku> \
  --kind <kind> \
  --access-tier <tier> \
  --enable-hierarchical-namespace true
```

Use Azure PowerShell

Use the New-AzStorageAccount PowerShell cmdlet with the EnableHierarchicalNamespace parameter, as follows:

```
New-AzStorageAccount `
  -Name "<storage-account-name>" `
  -ResourceGroupName "<resource-group-name>" `
  -Location "<your-location>" `
  -SkuName "<sku>" `
  -Kind "<kind>" `
  -AccessTier "<tier>" `
  -EnableHierarchicalNamespace $True
```

How to provision Blob storage in a storage account

Use the Azure portal

Blobs are stored in containers, and you create containers after you've created a storage account. In the Azure portal, you can add a container using the features on the Overview page for your storage account.

The Containers page enables you to create and manage containers. Each container must have a unique name within the storage account. You can also specify the access level. By default, data held in a container is only accessible by the container owner. You can set the access level to Blob to enable public read access to any blobs created in the container, or Container to allow read access to the entire contents of the container, including the ability to list all blobs. You can also configure role-based access control for a blob if you need a more granular level of security.

Once you've provisioned a container, your applications can upload blobs into the container.

Use the Azure CLI

The az storage container create command establishes a new blob container in a storage account.

```
az storage container create \
  --name <container-name> \
  --account-name <storage-account-name> \
  --public-access <access>
```

The public-access parameter can be blob, container, or off (for private access only).

Use Azure PowerShell

Use the New-AzStorageContainer cmdlet to add a container to a storage account. You must first retrieve a storage account object with the Get-AzStorageAccount cmdlet. The code below shows an example:

```
Get-AzStorageAccount `
  -ResourceGroupName "<resource-group>" `
  -Name "<storage-account-name>" | New-AzStorageContainer `
    -Name "<container-name>" `
    -Permission <permission>
```

The Permission parameter accepts the values Blob, Container, or Off.

How to provision File storage in a storage account

Use the Azure portal

You provision File storage by creating one or more file shares in the storage account. In the Azure portal, select File shares on the Overview page for the account.

Using the File shares page, create a new file share. Give the file share a name, and optionally set a quota to limit the size of files on the share. The total size of all files across all file shares in a storage account can't exceed 5120 GB.

Use the Azure CLI

The Azure CLI provides the az storage share create to create a new file share i n a storage account:

```
az storage share create \
  --name <share-name> \
  --account-name <storage-account-name>
```

Use Azure PowerShell

The New-AzStorageShare cmdlet creates a new file share in a storage account. You must retrieve the storage account details first.

```
Get-AzStorageAccount `
  -ResourceGroupName "<resource-group>" `
  -Name "<storage-account-name>" |New-AzStorageShare `
    -Name "<share-name>"
```

IV. Describe configuring non-relational data services

Configure connectivity and firewalls

The default connectivity for Azure Cosmos DB and Azure Storage is to enable access to the world at large. You can connect to these services from an on-premises network, the internet, or from within an Azure virtual network. Although this level of access sounds risky, most Azure services mitigate this risk by requiring authentication before granting access. Authentication is described later in this unit.

Configure connectivity to virtual networks and on-premises computers

To restrict connectivity, use the Firewalls and virtual networks page for a service. To limit connectivity, choose Selected networks. Three further sections will appear, labeled Virtual Network, Firewall, and Exceptions.

In the Virtual networks section, you can specify which virtual networks are allowed to route traffic to the service. When you create items such as web applications and virtual machines, you can add them to a virtual network. If these applications and virtual machines require access to your resource, add the virtual network containing these items to the list of allowed networks.

If you need to connect to the service from an on-premises computer, in the Firewall section, add the IP address of the computer. This setting creates a firewall rule that allows traffic from that address to reach the service.

The Exceptions setting allows you to enable access to any other of your services created in your Azure subscription.

Configure connectivity from private endpoints

Azure Private Endpoint is a network interface that connects you privately and securely to a service powered by Azure Private Link. Private Endpoint uses a private IP address from your VNet, effectively bringing the service into your VNet. The service could be an Azure service such as Azure Storage, Azure Cosmos DB, SQL, or your own Private Link Service.

The Private endpoint connections page for a service allows you to specify which private endpoints, if any, are permitted access to your service.

Configure authentication

Many services include an access key that you can specify when you attempt to connect to the service. If you provide an incorrect key, you'll be denied access. The image below shows how to find the access key for an Azure Storage account; you select Access Keys under Settings on the main page for the account. Many other services allow you to view the access key in the same way from the Azure portal. If your key is compromised, you can generate a new access key.

Any user or application that knows the access key for a resource can connect to that resource. However, access keys provide a rather coarse-grained level of authentication. Additionally, if you need to regenerate an access key (after accidental disclosure, for example), you may need to update all applications that connect using that key.

Azure Active Directory (Azure AD) provides superior security and ease of use over access key authorization. Microsoft recommends using Azure AD authorization when possible to minimize potential security vulnerabilities inherent in using access keys.

Azure AD is a separate Azure service. You add users and other security principals (such as an application) to a security domain managed by Azure AD.

Configure access control
Azure AD enables you to specify who, or what, can access your resources. Access control defines what a user or application can do with your resources after they've been authenticated.

Access management for cloud resources is a critical function for any organization that is using the cloud. Azure role-based access control (Azure RBAC) helps you manage who has access to Azure resources, and what they can do with those resources. For example, using RBAC you could:

- Allow one user to manage virtual machines in a subscription and another user to manage virtual networks.
- Allow a database administrator group to manage SQL databases in a subscription.
- Allow a user to manage all resources in a resource group, such as virtual machines, websites, and subnets.
- Allow an application to access all resources in a resource group.

You control access to resources using Azure RBAC to create role assignments. A role assignment consists of three elements: a security principal, a role definition, and a scope.

- A security principal is an object that represents a user, group, service, or managed identity that is requesting access to Azure resources.

- A role definition, often abbreviated to role, is a collection of permissions. A role definition lists the operations that can be performed, such as read, write, and delete. Roles can be given high-level names, like owner, or specific names, like virtual machine reader. Azure includes several built-in roles that you can use, including:

    - Owner - Has full access to all resources including the right to delegate access to others.

    - Contributor - Can create and manage all types of Azure resources but can't grant access to others.

    - Reader- Can view existing Azure resources.

    - User Access Administrator - Lets you manage user access to Azure resources.

- You can also create your own custom roles. For detailed information, see Create or update Azure custom roles using the Azure portal on the Microsoft website.

- A scope lists the set of resources that the access applies to. When you assign a role, you can further limit the actions allowed by defining a scope. This is helpful if, for example, you want to make someone a Website Contributor, but only for one resource group.

You add role assignments to a resource in the Azure portal using the Access control (IAM) page. The Role assignments tab enables you to associate a role with a security principal, defining the level of access the role has to the resource. For further information, read Add or remove Azure role assignments using the Azure portal.

Configure advanced security
Apart from authentication and authorization, many services provide additional protection through advanced security.

Advanced security implements threat protection and assessment. Threat protection adds security intelligence to your service. This intelligence monitors the service and detects unusual patterns of activity that could be harmful, or compromise the data managed by the service. Assessment identifies potential security vulnerabilities and recommends actions to mitigate them.

You're charged an additional fee for this feature. The image below shows the Advanced security page for Azure storage. The corresponding page for other non-relational services, such as Cosmos DB, is similar.

V. Configure Azure Cosmos DB, and Azure Storage

Configure Cosmos DB

Configure replication
Azure Cosmos DB enables you to replicate the databases and containers in your account across multiple regions. When you initially provision an account, you can specify that you want to copy data to another region. You don't have control over which region is used as the next nearest region is automatically selected. The Replicate data globally page enables you to configure replication in more detail. You can replicate to multiple regions, and you select the regions to use. In this way, you can pick the regions that are closest to your consumers, to help minimize the latency of requests made by those consumers.

You can also use this page to configure automatic failover to help ensure high availability. If the databases in the primary region (the region in which you created the account) become unavailable, one of the replicated regions will take over processing and become the new primary region.

By default, only the region in which you created the account supports write operations; the replicas are all read-only. However, you can enable multi-region writes. Multi-region writes can cause conflicts though, if applications running in different regions modify the same data. In this case, the most recent write will overwrite changes made earlier when data is replicated, although you can write your own code to apply a different strategy.

Replication is asynchronous, so there's likely to be a lag between a change made in one region, and that change becoming visible in other regions.

Configure consistency

Within a single region, Cosmos DB uses a cluster of servers. This approach helps to improve scalability and availability. A copy of all data is held in each server in the cluster. The following video explains how this works, and the effects it can have on consistency:

Cosmos DB enables you to specify how such inconsistencies should be handled. It provides the following options:

- Eventual. This option is the least consistent. It's based on the situation just described. Changes won't be lost, they'll appear eventually, but they might not appear immediately. Additionally, if an application makes several changes, some of those changes might be immediately visible, but others might be delayed; changes could appear out of order.

- Consistent Prefix. This option ensures that changes will appear in order, although there may be a delay before they become visible. In this period, applications may see old data.

- Session. If an application makes a number of changes, they'll all be visible to that application, and in order. Other applications may see old data, although any changes will appear in order, as they did for the Consistent Prefix option. This form of consistency is sometimes known as read your own writes.

- Bounded Staleness. There's a lag between writing and then reading the updated data. You specify this staleness either as a period of time, or number of previous versions the data will be inconsistent for.

- Strong: In this case, all writes are only visible to clients after the changes are confirmed as written successfully to all replicas. This option is unavailable if you need to distribute your data across multiple global regions.

Eventual consistency provides the lowest latency and least consistency. Strong consistency results in the highest latency but also the greatest consistency. You should select a default consistency level that balances the performance and requirements of your applications.

You can change the default consistency for a Cosmos DB account using the Default consistency page in the Azure portal. Applications can override the default consistency level for individual read operations. However, they can't increase the consistency above that specified on this page; they can only decrease it.

Configure Storage accounts

General configuration

The Configuration page for a storage account enables you to modify some general settings of the account. You can:

- Enable or disable secure communications with the service. By default, all requests and responses are encrypted by using the HTTPS protocol as they traverse the Internet. You can disable encryption if required, although this isn't recommended.
- Switch the default access tier between Cool and Hot.
- Change the way in which the account is replicated.
- Enable or disable integration with Azure AD for requests that access file shares.

Other options, such as the account kind and performance tier, are displayed on this page for information only; you can't change them.

Configure encryption

All data held in an Azure Storage account is automatically encrypted. By default, encryption is performed using keys managed and owned by Microsoft. If you prefer, you can provide your own encryption keys.

To use your own keys, add them to Azure Key Vault. You then provide the details of the vault and key, or the URI of the key in the vault. All new data will be encrypted as it's written. Existing data will be encrypted using a process running in the background; this process may take a little time.

Configure shared access signatures

You can use shared access signatures (SAS) to grant limited rights to resources in an Azure storage account for a specified time period. This feature enables applications to access resources such as blobs and files, without requiring that they're authenticated first. You should only use SAS for data that you intend to make public.

A SAS is a token that an application can use to connect to the resource. The application appends the token to the URL of the resource. The application can then send requests to read or write data using this URL and token.

You can create a token that grants temporary access to the entire service, containers in the service, or individual objects such as blobs and files.

Use the Shared access signature page in the Azure portal to generate SAS tokens. You specify the permissions (you could provide read-only access to a blob, for example), the period for which the SAS token is valid, and the IP address range of computers allowed to use the SAS token. The SAS token is encrypted using one of the access keys; you specify which key to use (key1 or key2).

Cosmos DB Hierarchy

Database Account > Database > Container – Partitioned > Item

Azure Data Fundamentals: Manage non-relational data stores in Azure

I. Manage Azure Cosmos DB

Review above section on Into to Cosmos DB

Perform data operations in Cosmos DB
Cosmos DB provides several options for uploading data to a Cosmos DB database, and querying that data. You can:

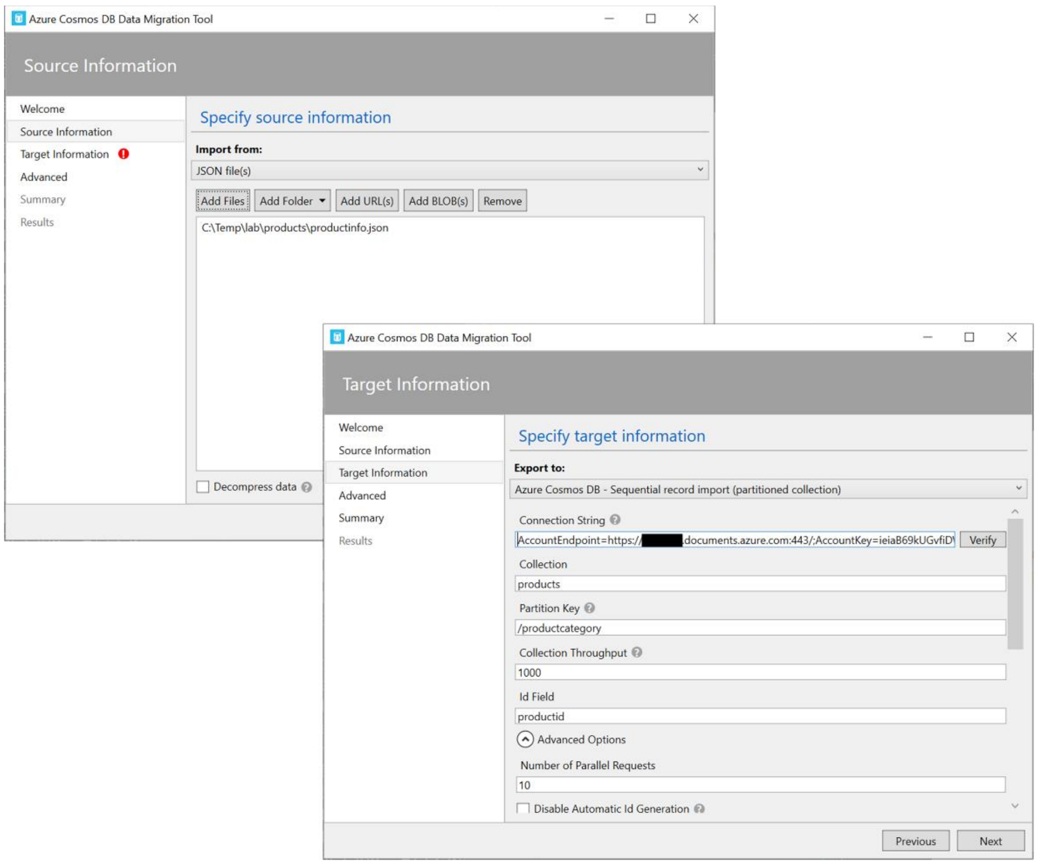
- Use Data Explorer in the Azure portal to run ad-hoc queries. You can also use this tool to load data, but you can only load one document at a time. The data load functionality is primarily aimed at uploading a small number of documents (up to 2 MB in total size) for test purposes, rather than importing large quantities of data.
- Use the Cosmos DB Data Migration tool to perform a bulk-load or transfer of data from another data source.
- Use Azure Data Factory to import data from another source.
- Write a custom application that imports data using the Cosmos DB BulkExecutor library. This strategy is beyond the scope of this module.
- Create your own application that uses the functions available through the Cosmos DB SQL API client library to store data. This approach is also beyond the scope of this module.

Load data using the Cosmos DB Data Migration tool

You can use the Data Migration tool to import data to Azure Cosmos DB from a variety of sources, including:

- JSON files
- MongoDB
- SQL Server
- CSV files
- Azure Table storage
- Amazon DynamoDB
- HBase
- Azure Cosmos containers

The Data Migration tool is available as a download from GitHub. The tool guides you through the process of migrating data into a Cosmos DB database. You're prompted for the source of the data (one of the items listed above), and the destination (the Cosmos DB database and container). The tool can either populate an existing container, or create a new one if the specified container doesn't already exist.



Import from: JSON file(s)

C:\Temp\lab\products\productinfo.json

Export to: Azure Cosmos DB - Sequential record import (partitioned collection)

Connection String: AccountEndpoint=https://[redacted].documents.azure.com:443/;AccountKey=ieiaB69kUGvfiD'

Collection: products

Partition Key: /productcategory

Collection Throughput: 1000

Id Field: productid

Number of Parallel Requests: 10

Disable Automatic Id Generation

Configure Cosmos DB to support bulk loading
If you have a large amount of data, the Data Migration Tool can make use of multiple concurrent threads to batch your data into chunks and load the chunks in parallel. Each thread acts as a separate client connection to the database. Bulk loading can become a write-intensive task.

When you upload data to a container, if you have insufficient throughput capacity configured to support the volume of write operations occurring concurrently, some of the upload requests will fail. Cosmos DB reports an HTTP 429 error (Request rate is large). Therefore, if you're planning on performing a large data import, you should increase the throughput resources available to the target Cosmos container. If you're using the Data Migration Tool to create the container as well as populate it, the Target information page enables you to specify the throughput resources to allocate.

If you've already created the container, use the Scale settings of the database in the Data Explorer page for your database in the Azure portal to specify the maximum throughput, or set the throughput to Autoscale.

Once the data has been loaded, you may be able to reduce the throughput resources to lower the costs of the database.

II. Query Azure Cosmos DB

Although Azure Cosmos DB is described as a NoSQL database management system, the SQL API enables you to run SQL-like queries against Cosmos DB databases. These queries use a syntax similar to that of SQL, but there are some differences. This is because the data in a Cosmos DB is structured as documents rather than tables.

Use the SQL API to query documents
The Cosmos DB SQL API supports a dialect of SQL for querying documents using SELECT statements that will be familiar if you have written SELECT statements in a relational database using an ANSI SQL compliant database engine. The SQL API returns results in the form of JSON documents. All queries are executed in the context of a single container.

Understand a SQL API query
A SQL API SELECT query includes the following clauses:

Note: A query can also contain a JOIN clause. In a relational database management system, such as Azure SQL Database, JOIN clauses are used to connect data from different tables. In the SQL API, you use JOIN clauses to connect fields in a document with fields in a subdocument that is part of the same document. You can't perform joins across different documents.

- SELECT clause. The clause starts with the keyword SELECT followed by a comma-separated list of properties to return. The keyword "*" means all the properties in the document.

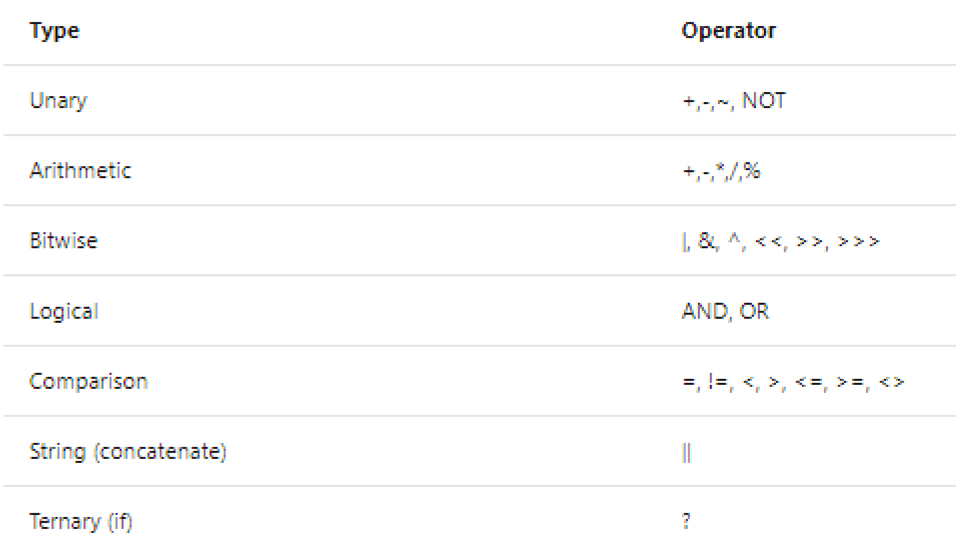
- FROM clause. This clause starts with the keyword FROM followed by an identifier, representing the source of the records, and an alias that you can use for this identifier in other clauses (the alias is optional). In a relational database query, the FROM clause would contain a table name. In the SQL API, all queries are limited to the scope of a container, so the identifier represents the name of the container.

- WHERE clause. This clause is optional. It starts with the keyword WHERE followed by one or more logical conditions that must be satisfied by a document returned by the query. You use the WHERE clause to filter the results of a query.

- ORDER BY clause. This clause is also optional. It starts with the phrase ORDER BY followed by one or more properties used to order the output result set.

Understand supported operators

The SQL API includes many common mathematical and string operations, in addition to functions for working with arrays and for checking data types. The operators supported in SQL API queries include:



Type	Operator
Unary	+,-,~, NOT
Arithmetic	+,-,*,/,%
Bitwise	|, &, ^, <<, >>, >>>
Logical	AND, OR
Comparison	=, !=, <, >, <=, >=, <>
String (concatenate)	||
Ternary (if)	?

The SQL API also supports:

- The DISTINCT operator that you use as part of the SELECT clause to eliminate duplicates in the result data.

- The TOP operator that you can use to retrieve only the first few rows returned by a query that might otherwise generate a large result set.

- The BETWEEN operation that you use as part of the WHERE clause to define an inclusive range of values. The condition field BETWEEN a AND b is equivalent to the condition field >= a AND field <= b.

- The IS_DEFINED operator that you can use for detecting whether a specified field exists in a document.

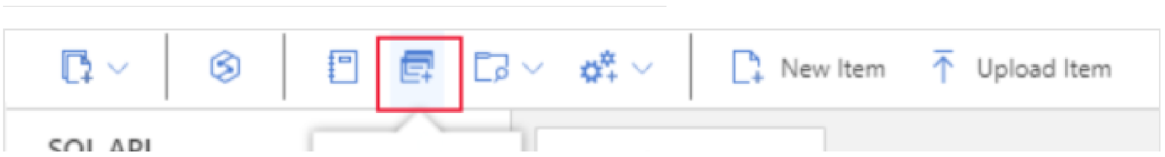
Understand aggregate functions

You can use aggregate functions to summarize data in SELECT queries; you place aggregate functions in the SELECT clause. The SQL API query language supports the following aggregate functions:

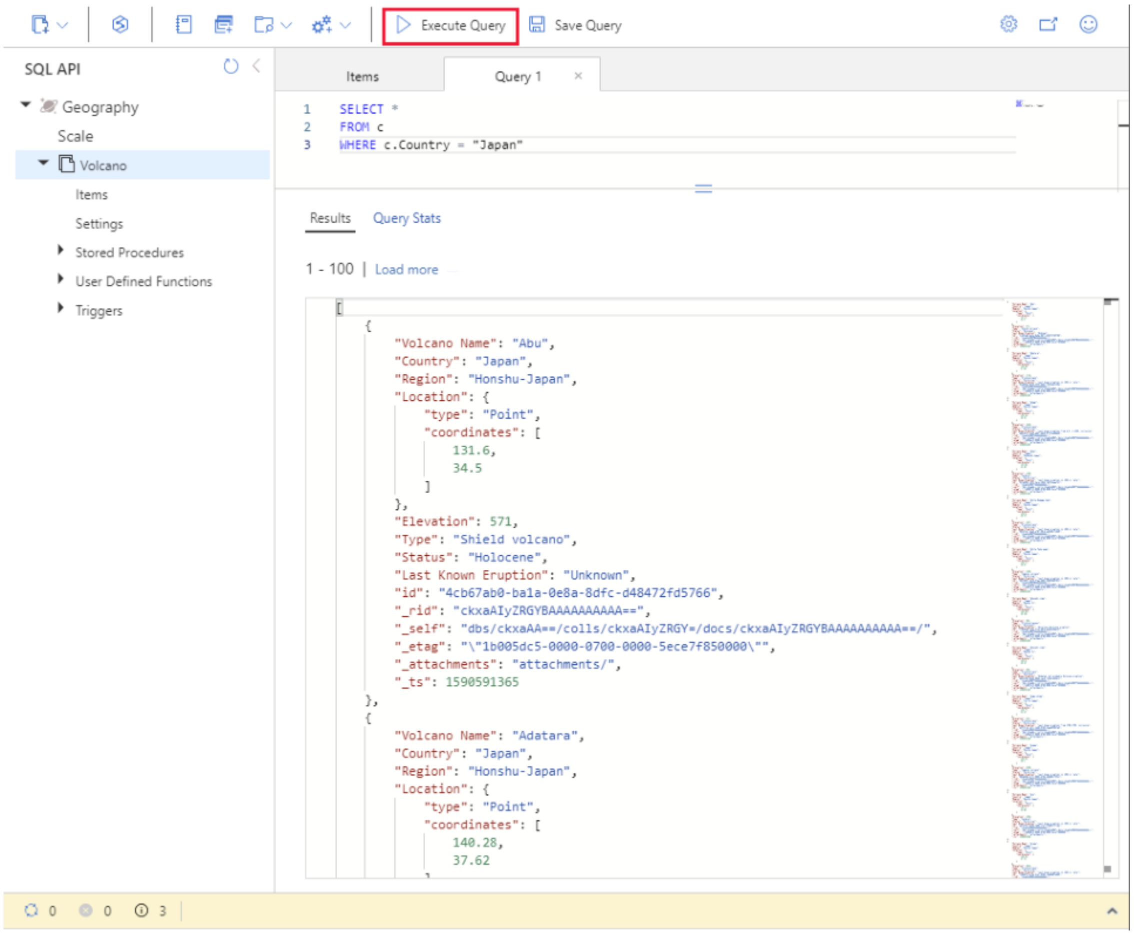
- COUNT(p). This function returns a count of the number of instances of field p in the result set. To count all the items in the result set, set p to a scalar value, such as 1.
- SUM(p). This function returns the sum of all the instances of field p in the result set. The values of p must be numeric.
- AVG(p). This function returns the mathematical mean of all the instances of field p in the result set. The values of p must be numeric.
- MAX(p). This function returns the maximum value of field p in the result set.
- MIN(p). This function returns the minimum value of field p in the result set.

Although the syntax of aggregate functions is similar to ANSI SQL, unlike ANSI SQL the SQL API query language doesn't support the GROUP BY clause; you can't generate subtotals for different values of the same field in a single query. You're able to include more than one aggregate function in the SELECT clause of your queries.

Query documents with the SQL API using Data Explorer
You can use Data Explorer in the Azure portal to create and run queries against a Cosmos DB container. The Items page for a container provides the New SQL Query command in the toolbar:



In the query pane that appears, you can enter a SQL query. Select Execute Query to run it. The results will be displayed as a list of JSON documents



```
SELECT *
FROM c
WHERE c.Country = "Japan"
```

```
[
    {
        "Volcano Name": "Abu",
        "Country": "Japan",
        "Region": "Honshu-Japan",
        "Location": {
            "type": "Point",
            "coordinates": [
                131.6,
                34.5
            ]
        },
        "Elevation": 571,
        "Type": "Shield volcano",
        "Status": "Holocene",
        "Last Known Eruption": "Unknown",
        "id": "4cb67ab0-ba1a-0e8a-8dfc-d48472fd5766",
        "_rid": "ckxaAIyZRGYBAAAAAAAAAA==",
        "_self": "dbs/ckxaAA==/colls/ckxaAIyZRGY=/docs/ckxaAIyZRGYBAAAAAAAAAA==/",
        "_etag": "\"1b005dc5-0000-0700-0000-5ece7f850000\"",
        "_attachments": "attachments/",
        "_ts": 1590591365
    },
    {
        "Volcano Name": "Adatara",
        "Country": "Japan",
        "Region": "Honshu-Japan",
        "Location": {
            "type": "Point",
            "coordinates": [
                140.28,
                37.62
```

You can save the query text if you need to repeat it in the future. The query is saved in a separate container. You can retrieve it later using the Open Query command in the toolbar.

Note: The Items page also lets you modify and delete documents. Select a document from the list to display it in the main pane. You can modify any of the fields, and select Update to save the changes. Select Delete to remove the document from the collection. The New Item command enables you to manually add a new document to the collection. You can use the Upload Item to create new documents from a file containing JSON data.

III. Manage Azure Blob storage

Azure currently supports three different types of blobs; Block blobs, Page blobs, and Append blobs. You typically use page blobs to implement virtual disk storage for Azure virtual machines; they're optimized to support random read and write operations. Append blobs are suitable for storing data that grows in chunks, such as logs or other archive data. Block blobs are best for static data, and are the most appropriate type of storage for holding the image data held by Contoso.

Create an Azure Storage container

In an Azure storage account, you store blobs in containers. A container provides a convenient way of grouping related blobs together, and you can organize blobs in a hierarchy of folders inside a container, similar to files in a file system on disk.

You create a container in an Azure Storage account. You can do this using the Azure portal, or using the Azure CLI or Azure PowerShell from the command line.

Use the Azure portal

In the Azure portal, go to the Overview page for your Azure Storage account, and select Containers.

On the Containers page, select + Container, and provide a name for the new container. You can also specify the public access level. For a container that will be used to hold blobs, the most appropriate access level is Blob. This setting supports anonymous read-only access for blobs. However, unauthenticated clients can't list the blobs in the container. This means they can only download a blob if they know its name and location within the container.

Use the Azure CLI

If you prefer to use the Azure CLI, the az storage container create command creates a new container. This command takes a number of optional parameters, and you can find the full details on the az storage container create page on the Microsoft website. The example below creates a container named images for storing blobs. The container is created in a storage account named contosodata. The container provides anonymous blob access.

```
az storage container create \
  --name images \
  --account-name contosodata \
  --resource-group contoso-group \
  --public-access blob
```

Use Azure PowerShell

You can use the New-AzStorageContainer PowerShell cmdlet to create a new storage container. The details are available on the New-AzStorageContainer page on the Microsoft website. You must first obtain a reference to the storage account using the Get-AzStorageAccount command. The code below shows an example:

```
Get-AzStorageAccount `
  -ResourceGroupName "contoso-group" `
  -Name "contosodata" | New-AzStorageContainer `
    -Name "images" `
    -Permission Blob
```

Upload a blob to Azure Storage

After you've created a container, you can upload blobs. Depending on how you want to organize your blobs, you can also create folders in the container.

Use the Azure portal

If you're using the Azure portal, go to the page for your storage account and select Containers under Blob service. On the Containers page, select the container you want to use.

On the page for the container, in the toolbar, select Upload. In the Upload blob dialog box, browse to the file container the data to upload. The Advanced drop-down section provides options you can modify the default options. For example, you can specify the name of a folder in the container (the folder will be created if it doesn't exist), the type of blob, and the access tier. The blob that is created is named after the file you uploaded.

Use the Azure CLI

Use the az storage blob upload command to upload a file to a blob in a container. The details describing the parameters for this command are available on the az storage blob upload page on the Microsoft website. The following example uploads a local file named racer_green_large.gif in the data folder to a blob called racer_green in the *bikes folder in the images container in the contosodata storage account.

```
az storage blob upload \
  --container-name images \
  --account-name contosodata \
  --file "\data\racer_green_large.gif" \
  --name "bikes\racer_green"
```

If you need to upload several files, use the az storage blob upload-batch command. This command takes the name of a local folder rather than a file name, and uploads the files in that folder to separate blobs. The example below uploads all gif files in the data folder to the bikes folder in the images container.

```
az storage blob upload-batch \
  --account-name contosodata \
  --source "\data" \
  --pattern "*.gif" \
  --destination "images\bikes"
```

Use Azure PowerShell

Azure PowerShell provides the Set-AzStorageBlobContent cmdlet to upload blob data to Azure storage, as follows:

```
Get-AzStorageAccount `
  -ResourceGroupName "contoso-group" `
  -Name "contosodata" | Set-AzStorageBlobContent `
    -Container "images" `
    -File "\data\racer_green_large.gif" `
    -Blob "bikes\racer_green"
```

Azure PowerShell doesn't currently include a batch blob upload command. If you need to upload multiple files, you can write your own PowerShell script (use the Get-ChildItem cmdlet) to iterate through the files and upload each one individually.

List the blobs in a container

If you've been granted the appropriate access rights, you can view the blobs in a container.

Use the Azure portal

If you're using the Azure portal, go to the page for your storage account and select Containers under Blob service. On the Containers page, select the container holding your blobs. If the container has a folder structure, move to the folder containing the blobs you want to see. The blobs in that folder should be displayed.

Use the Azure CLI

In the Azure CLI, you can use the az storage blob list command to view the blobs in a container. This command iterates recursively through any folders in the container. The example below lists the blobs previously uploaded to the images container:

```
az storage blob list \
  --account-name contosodata \
  --container-name "images"
```

Use Azure PowerShell
From Azure PowerShell, run the Get-AzStorageBlob cmdlet, as illustrated in the following example:

```
Get-AzStorageAccount `
  -ResourceGroupName "contoso-group" `
  -Name "contosodata" | Get-AzStorageBlob `
    -Container "images"
```

Download a blob from a container
You can retrieve a blob from Azure Storage and save it in a local file on your computer.

Use the Azure portal
If you're using the Azure portal, go to the page for your storage account and select Containers under Blob service. On the Containers page, select the container holding your blobs. If the container has a folder structure, move to the folder containing the blobs you want to download. Select the blob to view its details. On the details page, select Download.

Use the Azure CLI
The Azure CLI provides the az storage blob download and az storage blob download-batch commands. These commands are analogous to those available for uploading blobs. The example below retrieves the racer_green" blob from the bikes folder in the images container.

```
az storage blob download \
  --container-name images \
  --account-name contosodata \
  --file "racer_green_large.gif" \
  --name "bikes\racer_green"
```

Use Azure PowerShell
In Azure PowerShell, use the Get-AzStorageBlobContent cmdlet.

```
Get-AzStorageAccount `
  -ResourceGroupName "contoso-group" `
  -Name "contosodata" | Get-AzStorageBlobContent `
    -Container "images" `
    -Blob "bikes\racer_green_large.gif" `
    -Destination "racer_green_large.gif"
```

Delete a blob from a container
Deleting a blob can reclaim the resources used in the storage container. However, if you've enabled the soft delete option for the storage account, the blob is hidden rather than removed, and you can restore it later. You can enable or disable soft delete in the Azure portal, and specify the time for which the blob is retained. Select the Data protection page under Blob service. If the blob isn't restored by the end of the retention period, it will be removed from storage.

Use the Azure portal
If you're using the Azure portal, go to the page for your storage account and select Containers under Blob service. On the Containers page, select the container holding your blobs. If the container has a folder structure, move to the folder containing the blobs you want to download. Select the blob to view its details. On the details page, select Delete. You'll be prompted to confirm the operation.

If you've enabled soft delete for the storage account, the blobs page listing the blobs in a container includes the option Show deleted blobs. If you select this option, you can view and undelete a deleted blob.

Use the Azure CLI
You can delete a single blob with the az storage blob delete command, or a set of blobs with the az storage blob delete-batch command. The command below removes the racer-green blob from the bikes folder in the images container:

```
az storage blob delete ^
  --account-name contosodata ^
  --container-name "images" ^
  --name "bikes\racer_green"
```

Use Azure PowerShell
Use the Remove-AzStorageBlob cmdlet to delete a storage blob from Azure PowerShell. By default, deletion runs without prompts. You can add the -Confirm flag to prompt the user to confirm that they really want to delete the blob:

```
Get-AzStorageAccount `
  -ResourceGroupName "contoso-group" `
  -Name "contosodata" | Remove-AzStorageBlob `
    -Container "images" `
    -Blob "bikes\racer_green" `
    -Confirm
```

Delete an Azure Storage container
Removing a container automatically deletes all blobs held in that container. If you aren't careful, you can lose a great deal of data.

Use the Azure portal
In the Azure portal, select Containers under Blob service, select the container to delete, and then select Delete in the toolbar.

Use the Azure CLI
In the Azure CLI, use the az storage container delete command. The following example deletes the images container referenced in previous examples.

```
az storage container delete \
  --account-name contosodata \
  --name "images"
```

Use Azure PowerShell
The Remove-AzStorageContainer cmdlet deletes a storage container. The -Confirm flag prompts the user to confirm the delete operation. The code below shows an example:

```
Get-AzStorageAccount `
  -ResourceGroupName "contoso-group" `
  -Name "contosodata" | Remove-AzStorageContainer `
    -Name "images" `
    -Confirm
```

IV. Manage Azure File storage

Create a file share
Microsoft provides two graphical tools you can use to create and manage file shares in Azure Storage: the Azure portal, and Azure Storage Explorer.

Use the Azure portal
Select File shares in the main pane of the Overview page for an Azure Storage account (this is also available in the File service section of the command bar):

On the File shares page, select + File share. Give the file share a name, and optionally specify a quota. Azure allows you to store up to 5 PiB of files across all files shares in a storage account. A quota enables you to limit the amount of space an individual file share consumes, to prevent it from starving other file shares of file storage. If you have only one file share, you can leave the quota empty.

After you've created a share, you can use the Azure portal to add directories to the share, upload files to the share, and delete the share. The Connect command generates a PowerShell script that you can run to attach to the share from your local computer. You can then use the share as though it was a local disk drive.

Use Azure Storage Explorer

Azure Storage Explorer is a utility that enables you to manage Azure Storage accounts from your desktop computer. You can download it from the Azure Storage Explorer page on the Microsoft website. You can use Storage Explorer to create blob containers and file shares, as well as upload and download files.

A version of this utility is also available in the Azure portal, on the Overview page for an Azure Storage account.

To create a new file share, right-click File Shares, and then select Create file share. In the Azure portal, Storage Explorer displays the same dialog box that you saw earlier. In the desktop version, you simply enter a name for the new file share; you don't get the option to set a quota at this point.

As with the Azure portal, once you have created a new share, you can use Storage Explorer to create folders, and upload and download files.

Upload and download files

You can upload and download individual files to and from Azure File storage manually, by using Storage Explorer, the Azure portal, or by connecting the file share to your desktop computer and dragging and dropping files in File Explorer.

However, if you need to transfer a significant number of files in and out of Azure File storage, you should use the AzCopy utility. AzCopy is a command-line utility optimized for transferring large files (and blobs) between your local computer and Azure File storage. It can detect transfer failures, and restart a failed transfer at the point an error occurred - you don't have to repeat the entire operation.

Generate an SAS token

Before you can use AzCopy, you generate a Shared access signature (SAS) token. A SAS token provides controlled, time-limited, anonymous access to services and resources in a storage account; users don't have to provide any additional credentials. SAS tokens are useful in situations where you don't know in advance which users will require access to your resources.

You can create an SAS token for connecting to Azure File storage using the Azure portal. On the page for your storage account, under Settings, select Shared access signature. On the Shared access signature page, under Allowed services, select File. Under Allowed resource types, select Container and Object. Under Permissions, select the privileges that you want to grant to users. Set the start and end time for the SAS token, and specify the IP address range of the computers your users will be using. Select Generate SAS and connection string to create the SAS token. Copy the value in the SAS token field somewhere safe.

Upload files

To transfer a single file into File Storage using AzCopy, use the form of the command shown in the following example. Run this command from the command line. In this example, replace <storage-account-name> with the name of the storage account, replace <file-share> with the name of a file share in this account, and replace <SAS-token> with the token you created using the Azure portal. You must include the quotes where shown.

```
azcopy copy "myfile.txt" https://<storage-account-name>.file.core.windows.net/<file-share-name>/myfile.txt<SAS-token>
```

You can transfer the entire contents of a local folder to Azure File storage using a similar command. You replace the file name ("myfile.txt") with the name of the folder. If the folder contains subfolders that you want to copy, add the --recursive flag.

```
azcopy copy "myfolder" "https://<storage-account-name>.file.core.windows.net/<file-share-name>/myfolder<SAS-token>" –recursive
```

As the process runs, AzCopy displays a progress report:

```
INFO: Scanning...

INFO: Any empty folders will be processed, because source and destination both support folders

Job b86eeb8b-1f24-614e-6302-de066908d4a2  has started

Log file is located at: C:\Users\User\.azcopy\b86eeb8b-1f24-614e-6302-de066908d4a2.log

11.5 %, 126 Done, 0 Failed, 48 Pending, 0 Skipped, 174 Total, 2-sec Throughput (Mb/s): 8.2553
```

When the transfer is complete, you'll see a summary of the work performed.

```
Job b86eeb8b-1f24-614e-6302-de066908d4a2  summary

Elapsed Time (Minutes): 0.6002

Number of File Transfers: 161

Number of Folder Property Transfers: 13

Total Number of Transfers: 174

Number of Transfers Completed: 174

Number of Transfers Failed: 0

Number of Transfers Skipped: 0

TotalBytesTransferred: 43686370

Final Job Status: Completed
```

Download files

You can also use the AzCopy copy command to transfer files and folders from Azure File Storage to your local computer. The command is similar to that for uploading files, except that you switch the order of the arguments; specify the files and folders in the file share first, and the local files and folders second. For example, to download the files from a folder named myfolder in a file share named myshare to a local folder called localfolder, use the following command:

```
azcopy copy "https://<storage-account-name>.file.core.windows.net/myshare/myfolder<SAS-token>" "localfolder" --recursive
```

Azure Data Fundamentals: Explore modern data warehouse analytics in Azure

https://docs.microsoft.com/en-us/learn/paths/azure-data-fundamentals-explore-data-warehouse-analytics/

Azure Data Fundamentals: Examine Components of a modern data warehouse

I. Describe modern data warehousing

What is modern data warehousing?
A modern data warehouse might contain a mixture of relational and non-relational data, including files, social media streams, and Internet of Things (IoT) sensor data. Azure provides a collection of services you can use to build a data warehouse solution, including Azure Data Factory, Azure Data Lake Storage, Azure Databricks, Azure Synapse Analytics, and Azure Analysis Services. You can use tools such as Power BI to analyze and visualize the data, generating reports, charts, and dashboards.

Combine batch and stream processing
A typical large-scale business requires a combination of up-to-the-second data, and historical information. The up-to-the-second data might be used to help monitor real-time, critical manufacturing processes, where an instant decision is required. Other examples include streams of stock market data, where the current prices are required to make informed split-second buy or sell decisions.

Historical data is equally important, to give a business a more stabilized view of trends in performance. A manufacturing organization will require information such as the volumes of sales by products across a month, a quarter, or a year, to determine whether to continue producing various items, or whether to increase or decrease production according to seasonal fluctuations. This historical data can be generated by batch processes at regular intervals, based on the live sales data that might be captured continually.

Any modern data warehouse solution must be able to provide access to the streams of raw data, and the cooked business information derived from this data.

II. Explore Azure data services for modern data warehousing

What is Azure Data Factory?

Azure Data Factory is described as a data integration service. The purpose of Azure Data Factory is to retrieve data from one or more data sources, and convert it into a format that you process. The data sources might present data in different ways, and contain noise that you need to filter out. Azure Data Factory enables you to extract the interesting data, and discard the rest. The interesting data might not be in a suitable format for processing by the other services in your warehouse solution, so you can transform it. For example, your data might contain dates and times formatted in different ways in different data sources. You can use Azure Data Factory to transform these items into a single uniform structure. Azure Data Factory can then write the ingested data to a data store for subsequent processing.

You define the work performed by Azure Data Factory as a pipeline of operations. A pipeline can run continuously, as data is received from the various data sources. You can create pipelines using the graphical user interface provided by Microsoft, or by writing your ow n code. The image below shows the pipeline editor in Azure Data Factory.

What is Azure Data Lake Storage?

A data lake is a repository for large quantities of raw data. Because the data is raw and unprocessed, it's very fast to load and update, but the data hasn't been put into a structure suitable for efficient analysis. You can think of a data lake as a staging point for your ingested data, before it's massaged and converted into a format suitable for performing analytics.

Azure Data Lake Storage combines the hierarchical directory structure and file system semantics of a traditional file system with security and scalability provided by Azure. Azure Data Lake Storage is essentially an extension of Azure Blob storage, organized as a near-infinite file system. It has the following characteristics:

- Data Lake Storage organizes your files into directories and subdirectories for improved file organization. Blob storage can only mimic a directory structure.
- Data Lake Storage supports the Portable Operating System Interface (POSIX) file and directory permissions to enable granular Role-Based Access Control (RBAC) on your data.
- Azure Data Lake Storage is compatible with the Hadoop Distributed File System (HDFS). Hadoop is highly flexible and programmable analysis service, used by many organizations to examine large quantities of data. All Apache Hadoop environments can access data in Azure Data Lake Storage Gen2.

In an Azure Data Services data warehouse solution, data is typically loaded into Azure Data Lake Storage before being processed into a structure that enables efficient analysis in Azure Synapse Analytics. You can use a service such as Azure Data Factory (described above) to ingest and load the data from a variety of sources into Azure Data Lake Storage.

What is Azure Databricks?

Azure Databricks is an Apache Spark environment running on Azure to provide big data processing, streaming, and machine learning. Apache Spark is a highly efficient data processing engine that can consume and process large amounts of data very quickly. There are a significant number of Spark libraries you can use to perform tasks such as SQL processing, aggregations, and to build and train machine learning models using your data.

Azure Databricks provides a graphical user interface where you can define and test your processing step by step, before submitting it as a set of batch tasks. You can create Databricks scripts and query data using languages such as R, Python, and Scala. You write your Spark code using notebooks. A notebook contains cells, each of which contains a separate block of code. When you run a notebook, the code in each cell is passed to Spark in turn for execution. The image below shows a cell in a workbook that runs a query and generates a graph.

Azure Databricks also supports structured stream processing. In this model, Databricks performs your computations incrementally, and continuously updates the result as streaming data arrives.

What is Azure Synapse Analytics?

Azure Synapse Analytics is an analytics engine. It's designed to process large amounts of data very quickly.

Using Synapse Analytics, you can ingest data from external sources, such as flat files, Azure Data Lake, or other database management systems, and then transform and aggregate this data into a format suitable for analytics processing. You can perform complex queries over this data and generate reports, graphs, and charts.

Reading and transforming data from an external source can consume considerable resources. Azure Synapse Analytics enables you to store the data you have read in and processed locally, within the service (this is described later). This approach enables you to repeatedly query the same data without the overhead of fetching and converting it each time. You can also use this data as input to further analytical processing, using Azure Analysis Services.

Azure Synapse Analytics leverages a massively parallel processing (MPP) architecture. This architecture includes a control node and a pool of compute nodes.

The Control node is the brain of the architecture. It's the front end that interacts with all applications. The MPP engine runs on the Control node to optimize and coordinate parallel queries. When you submit a processing request, the Control node transforms it into smaller requests that run against distinct subsets of the data in parallel.

The Compute nodes provide the computational power. The data to be processed is distributed evenly across the nodes. Users and applications send processing requests to the control node. The control node sends the queries to compute nodes, which run the queries over the portion of the data that they each hold. When each node has finished its processing, the results are sent back to the control node where they're combined into an overall result.

Azure Synapse Analytics supports two computational models: SQL pools and Spark pools.

In a SQL pool, each compute node uses an Azure SQL Database and Azure Storage to handle a portion of the data.

You submit queries in the form of Transact-SQL statements, and Azure Synapse Analytics runs them. However, unlike an ordinary SQL Server database engine, Azure Synapse Analytics can receive data from a wide variety of sources. To do this, Azure Synapse Analytics uses a technology named PolyBase. PolyBase enables you to retrieve data from relational and non-relational sources, such as delimited text files, Azure Blob Storage, and Azure Data Lake Storage. You can save the data read in as SQL tables within the Synapse Analytics service.

You specify the number of nodes when you create a SQL pool. You can scale the SQL pool manually to add or remove compute nodes as necessary.

In a Spark pool, the nodes are replaced with a Spark cluster. You run Spark jobs comprising code written in Notebooks, in the same way as Azure Databricks. You can write the code for notebook in C#, Python, Scala, or Spark SQL (a different dialect of SQL from Transact-SQL). As with a SQL pool, the Spark cluster splits the work out into a series of parallel tasks that can be performed concurrently. You can save data generated by your notebooks in Azure Storage or Data Lake Storage.

You specify the number of nodes when you create the Spark cluster. Spark pools can have autoscaling enabled, so that pools scale by adding or removing nodes as needed. Autoscaling can occur while processing is active.

What is Azure Analysis Services?

Azure Analysis Services enables you to build tabular models to support online analytical processing (OLAP) queries. You can combine data from multiple sources, including Azure SQL Database, Azure Synapse Analytics, Azure Data Lake store, Azure Cosmos DB, and many others. You use these data sources to build models that incorporate your business knowledge. A model is essentially a set of queries and expressions that retrieve data from the various data sources and generate results. The results can be cached in-memory for later use, or they can be calculated dynamically, directly from the underlying data sources.

Analysis Services includes a graphical designer to help you connect data sources together and define queries that combine, filter, and aggregate data. You can explore this data from within Analysis Services, or you can use a tool such as Microsoft Power BI to visualize the data presented by these models.

Compare Analysis Services with Synapse Analytics
Azure Analysis Services has significant functional overlap with Azure Synapse Analytics, but it's more suited for processing on a smaller scale.

Use Azure Synapse Analytics for:

- Very high volumes of data (multi-terabyte to petabyte sized datasets).
- Very complex queries and aggregations.

- Data mining, and data exploration.
- Complex ETL operations. ETL stands for Extract, Transform, and Load, and refers to the way in which you can retrieve raw data from multiple sources, convert this data into a standard format, and store it.
- Low to mid concurrency (128 users or fewer).

Use Azure Analysis Services for:

- Smaller volumes of data (a few terabytes).
- Multiple sources that can be correlated.
- High read concurrency (thousands of users).
- Detailed analysis, and drilling into data, using functions in Power BI.
- Rapid dashboard development from tabular data.

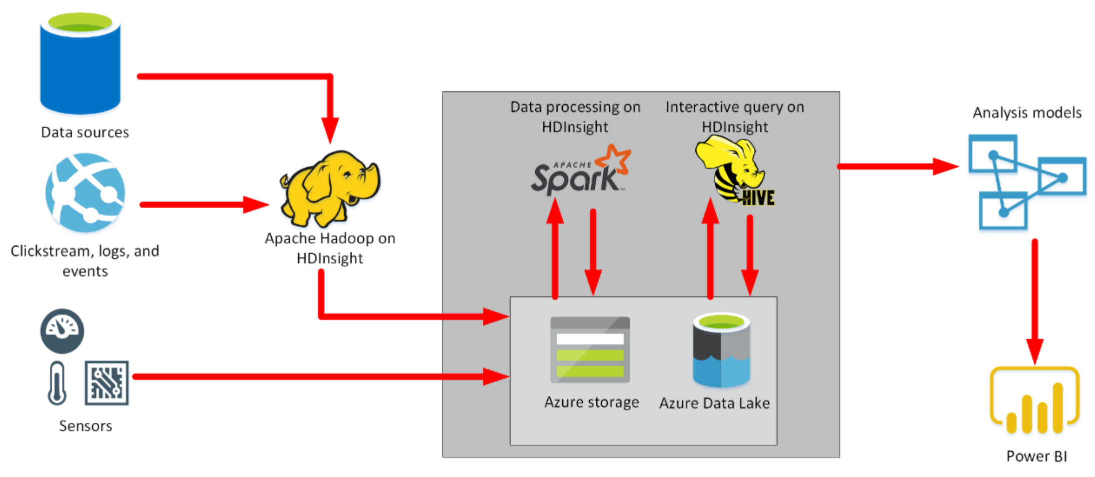
Combine Analysis Services with Synapse Analytics

Many scenarios can benefit from using Synapse Analytics and Analysis Services together. If you have large amounts of ingested data that require preprocessing, you can use Synapse Analytics to read this data and manipulate it into a model that contains business information rather than a large amount of raw data. The scalability of Synapse Analytics gives it the ability to process and reduce many terabytes of data down into a smaller, succinct dataset that summarizes and aggregates much of this data. You can then use Analysis Services to perform detailed interrogation of this information, and visualize the results of these inquiries with Power BI.

What is Azure HDInsight?

Azure HDInsight is a big data processing service, that provides the platform for technologies such as Spark in an Azure environment. HDInsight implements a clustered model that distributes processing across a set of computers. This model is similar to that used by Synapse Analytics, except that the nodes are running the Spark processing engine rather than Azure SQL Database.

You can use Azure HDInsight in conjunction with, or instead of, Azure Synapse Analytics. As well as Spark, HDInsight supports streaming technologies such as Apache Kafka, and the Apache Hadoop processing model. The image below shows where you might use the components of HDInsight in a data warehousing solution.


Data sources
Clickstream, logs, and events
Sensors
Apache Hadoop on HDInsight
Data processing on HDInsight
Interactive query on HDInsight
Azure storage
Azure Data Lake
Analysis models
Power BI

Azure Data Fundamentals: Explore data ingestion in Azure

I. Describe common practices for data loading

Ingest data using Azure Data Factory

Azure Data Factory is a data ingestion and transformation service that allows you to load raw data from many different sources, both on-premises and in the cloud. As it ingests the data, Data Factory can clean, transform, and restructure the data, before loading it into a repository such as a data warehouse. Once the data is in the data warehouse, you can analyze it.

Data Factory contains a series of interconnected systems that provide a complete end-to-end platform for data engineers. You can load static data, but you can also ingest streaming data. Loading data from a stream offers a real-time solution for data that arrives quickly or that changes rapidly. Using streaming, you can use Azure Data Factory to continually update the information in a data warehouse with the latest data.

Data Factory provides an orchestration engine. Orchestration is the process of directing and controlling other services, and connecting them together, to allow data to flow between them. Data Factory uses orchestration to combine and automate sequences of tasks that use different services to perform complex operations.

Understand linked services

Data Factory moves data from a data source to a destination. A linked service provides the information needed for Data Factory to connect to a source or destination. For example, you can use an Azure Blob Storage linked service to connect a storage account to Data Factory, or the Azure SQL Database linked service to connect to a SQL database.

The information a linked service contains varies according to the resource. For example, to create a linked service for Azure Blob Storage, you provide information such as the name of the Azure subscription that owns the storage account, the name of the storage account, and the information necessary to authenticate against the storage account. To create a linked service to a different resource, such as Azure SQL Database, you specify the database server name, the database name, and the appropriate credentials.

Understand datasets

A dataset in Azure Data Factory represents the data that you want to ingest (input) or store (output). If your data has a structure, a dataset specifies how the data is structured. Not all datasets are structured. Blobs held in Azure Blob storage are an example of unstructured data.

A dataset connects to an input or an output using a linked service. For example, if you're reading and processing data from Azure Blob storage, you'd create an input dataset that uses a Blob Storage linked service to specify the details of the storage account. The dataset would specify which blob to ingest, and the format of the information in the blob (binary data, JSON, delimited text, and so on). If you're using Azure Data Factory to store data in a table in a SQL database, you would define an output dataset that uses a SQL Database linked service to connect to the database, and specifies which table to use in that database.

Understand pipelines

A pipeline is a logical grouping of activities that together perform a task. The activities in a pipeline define actions to perform on your data. For example, you might use a copy activity to transform data from a source dataset to a destination dataset. You could include activities that transform the data as it is transferred, or you might combine data from multiple sources together. Other activities enable you to incorporate processing elements from other services. For example, you might use an Azure Function activity to run an Azure Function to modify and filter data, or an Azure Databricks Notebook activity to run a notebook that performs more advanced processing.

Pipelines don't have to be linear. You can include logic activities that repeatedly perform a series of tasks while some condition is true using a ForEach activity, or follow different processing paths depending on the outcome of previous processing using an If Condition activity.

Sometimes when ingesting data, the data you're bringing in can have different column names and data types to those required by the output. In these cases, you can use a mapping to transform your data from the input format to the output format.

You can run a pipeline manually, or you can arrange for it to be run later using a trigger. A trigger enables you to schedule a pipeline to occur according to a planned schedule (every Saturday evening, for example), or at repeated intervals (every few minutes or hours), or when an event occurs such as the arrival of a file in Azure Data Lake Storage, or the deletion of a blob in Azure Blob Storage.

Ingest data using PolyBase

PolyBase is a feature of SQL Server and Azure Synapse Analytics that enables you to run Transact-SQL queries that read data from external data sources. PolyBase makes these external data sources appear like tables in a SQL database. Using PolyBase, you can read data managed by Hadoop, Spark, and Azure Blob Storage, as well as other database management systems such as Cosmos DB, Oracle, Teradata, and MongoDB.

PolyBase enables you to transfer data from an external data source into a table, as well as copy data from an external data source in Azure SYnapse Analytics or SQL Server. You can also run queries that join tables in a SQL database with external data, enabling you to perform analytics that span multiple data stores.

Azure Data Factory provides PolyBase support for loading data. For instance, Data Factory can directly invoke PolyBase on your behalf if your data is in a PolyBase-compatible data store.

Ingest data using SQL Server Integration Services

SQL Server Integration Services (SSIS) is a platform for building enterprise-level data integration and data transformations solutions. You can use SSIS to solve complex business problems by copying or downloading files, loading data warehouses, cleaning and mining data, and managing SQL database objects and data. SSIS is part of Microsoft SQL Server.

SSIS can extract and transform data from a wide variety of sources such as XML data files, flat files, and relational data sources, and then load the data into one or more destinations.

SSIS includes a rich set of built-in tasks and transformations, graphical tools for building packages, and the Integration Services Catalog database, where you store, run, and manage packages. A package is an organized collection of connections, control flow elements, data flow elements, event handlers, variables, parameters, and configurations, that you assemble using either the graphical design tools that SQL Server Integration Services provides, or build programmatically. You then save the completed package to SQL Server, the Integration Services Package Store, or the file system.

You can use the graphical SSIS tools to create solutions without writing a single line of code. You can also program the extensive Integration Services object model to create packages programmatically and code custom tasks and other package objects.

SSIS is an on-premises utility. However, Azure Data factory allows you to run your existing SSIS packages as part of a pipeline in the cloud. This allows you to get started quickly without having to rewrite your existing transformation logic.

The SSIS Feature Pack for Azure is an extension that provides components that connect to Azure services, transfer data between Azure and on-premises data sources, and process data stored in Azure. The components in the feature pack support transfer to or from Azure storage, Azure

Data Lake, and Azure HDInsight. Using these components, you can perform large-scale processing of ingested data.

Ingest data using Azure Databricks
Azure Databricks is an analytics platform optimized for the Microsoft Azure cloud services platform. Databricks is based on Spark, and is integrated with Azure to streamline workflows. It provides an interactive workspace that enables collaboration between data scientists, data engineers, and business analysts.

Databricks can process data held in many different types of storage, including Azure Blob storage, Azure Data Lake Store, Hadoop storage, flat files, SQL databases, and data warehouses, and Azure services such as Cosmos DB. Databricks can also process streaming data. For example, you could capture data being streamed from sensors and other devices.

You write and run Spark code using notebooks. A notebook is like a program that contains a series of steps (called cells). A notebook can contain cells that read data from one or more data sources, process the data, and write the results out to a data store. The scalability of Azure Databricks makes it an ideal platform for performing complex data ingestion and analytics tasks.

Azure Data Factory can incorporate Azure Databricks notebooks into a pipeline. A pipeline can pass parameters to a notebook. These parameters can specify which data to read and analyze. The notebook can save the results, which the Azure Data Factory pipeline can use in subsequent activities.

II. Load data in Azure Synapse Analytics
See Video on Website

Azure Data Fundamentals: Explore storage and processing in Azure

I. Describe data storage and processing in Azure

What is Azure Synapse Analytics?
Azure Synapse Analytics is a generalized analytics service. You can use it to read data from many sources, process this data, generate various analyses and models, and save the results.

You can select between two technologies to process data:

Transact-SQL. This is the same dialect of SQL used by Azure SQL Database, with some extensions for reading data from external sources, such as databases, files, and Azure Data Lake storage. You can use these extensions to load data quickly, generate aggregations and other analytics, create tables and views, and store information using these tables and views. You can use the results for later reporting and processing.

Spark. This is the same open-source technology used to power Azure Databricks. You write your analytical code using notebooks in a programming language such as C#, Scala, Python, or SQL. The Spark libraries provided with Azure Synapse Analytics enable you to read data from external sources, and also write out data in a variety of different formats if you need to save your results for further analysis.

Azure Synapse Analytics uses a clustered architecture. Each cluster has a control node that is used as the entry point to the system. When you run Transact-SQL statements or start Spark jobs from a notebook, the request is sent to the control node. The control node runs a parallel processing engine that splits the operation into a set of tasks that can be run concurrently. Each task performs part of the workload over a subset of the source data. Each task is sent to a compute node to actually do the processing. The control node gathers the results from the compute nodes and combines them into an overall result.

What is Azure Databricks?

Azure Databricks is an analytics platform optimized for the Microsoft Azure cloud services platform. Designed with the founders of Apache Spark, Databricks is integrated with Azure to provide one-click setup, streamlined workflows, and an interactive workspace that enables collaboration between data scientists, data engineers, and business analysts.

Databricks can process data held in many different types of storage, including Azure Blob storage, Azure Data Lake Store, Hadoop storage, flat files, databases, and data warehouses. Databricks can also process streaming data. Databricks uses an extensible architecture based on drivers.

The processing engine is provided by Apache Spark. Spark is a parallel-processing engine that supports large-scale analytics. You write application code that consumes data from one or more sources, and merge, reformat, filter, and remodel this data, and then store the results. Spark distributes the work across a cluster of computers. Each computer can process its data in parallel with the other computers. The strategy helps to reduce the time required to perform the work. Spark is designed to handle massive quantities of data.

You can write the Spark application code using several languages, such as Python, R, Scala, Java, and SQL. Spark has a number of libraries for these languages, providing complex analytical routines that have been optimized for the clustered environment. These libraries include modules for machine learning, statistical analysis, linear and non-linear modeling, predictive analytics, and graphics.

You write Databricks applications using a Notebook. A notebook contains a series of steps (cells), each of which contains a block of code. For example, one cell might contain the code that connects to a data source, the next cell reads the data from that source and converts it into a model in-memory, the next cell plots a graph, and a final cell saves the data from the in-memory model to a repository.

What is Azure HDInsight?

Azure HDInsight is a managed analytics service in the cloud. It's based on Apache Hadoop, a collection of open-source tools and utilities that enable you to run processing tasks over large amounts of data. HDInsight uses a clustered model, similar to that of Synapse Analytics. HDInsight stores data using Azure Data Lake storage. You can use HDInsight to analyze data using frameworks such as Hadoop Map/Reduce, Apache Spark, Apache Hive, Apache Kafka, Apache Storm, R, and more.

Hadoop Map/Reduce uses a simple framework to split a task over a large dataset into a series of smaller tasks over subsets of the data that can be run in parallel, and the results then combined. You write your Map/Reduce code in a language such as Java, and then submit this code as a job to the Hadoop cluster. Hadoop Map/Reduce has largely been replaced by Spark, which offers a more advanced set of operations and a simpler interface.

Like Map/Reduce jobs, Spark jobs are parallelized into a series of subtasks tasks that run on the cluster. You can write Spark jobs as part of an application, or you can use interactive notebooks. These notebooks are the same as those that you can run from Azure Databricks. Spark includes libraries that you can use to read and write data in a wide variety of data stores (not just HDFS). For example, you can connect to relational databases such as Azure SQL Database, and other services such as Azure Cosmos DB.

Apache Hive provides interactive SQL-like facilities for querying, aggregating, and summarizing data. The data can come from many different sources. Queries are converted into tasks, and parallelized. Each task can run on a separate node in the HDInsight cluster, and the results are combined before being returned to the user.

Apache Kafka is a clustered streaming service that can ingest data in real time. It's a highly scalable solution that offers publish and subscribe features.

Apache Storm is a scalable, fault tolerant platform for running real-time data processing applications. Storm can process high volumes of streaming data using comparatively modest computational requirements. Storm is designed for reliability, so that events shouldn't be lost. Storm solutions can also provide guaranteed processing of data, with the ability to replay data that wasn't successfully processed the first time. Storm can interoperate with a variety of event sources, including Azure Event Hubs, Azure IoT Hub, Apache Kafka, and RabbitMQ (a message queuing service). Storm can also write to data stores such as HDFS, Hive, HBase, Redis, and SQL databases. You write a Storm application using the APIs provided by Apache.

What is Azure Data Factory?
Azure Data Factory is a service that can ingest large amounts of raw, unorganized data from relational and non-relational systems, and convert this data into meaningful information. Data Factory provides a scalable and programmable ingestion engine that you can use to implement complex hybrid extract-transform-load (ETL), extract-load-transform (ELT), and data integration projects.

For example, imagine a gaming company that collects petabytes of game logs that are produced by games in the cloud. The company wants to analyze these logs to gain insights into customer preferences, demographics, and usage behavior. It also wants to identify up-sell and cross-sell opportunities, develop compelling new features, drive business growth, and provide a better experience to its customers.

To analyze these logs, the company needs to use reference data such as customer information, game information, and marketing campaign information that is in an on-premises data store. The company wants to utilize this data from the on-premises data store, combining it with additional log data that it has in a cloud data store.

To extract insights, the company wants to process the joined data by using a Spark cluster in the cloud (using Azure HDInsight), and publish the transformed data into a cloud data warehouse such as Azure Synapse Analytics. The company can use the information in the data warehouse to generate and publish reports. They want to automate this workflow, and monitor and manage it on a daily schedule. They also want to execute it when files land in a blob store container.

Using Azure Data Factory, you can create and schedule data-driven workflows (called pipelines) that can ingest data from the disparate data stores used by the gaming company. You can build complex ETL processes that transform data visually with data flows or by using compute services such as Azure HDInsight, Azure Databricks, and Azure SQL Database. You can then publish the transformed data to Azure Synapse Analytics for business intelligence applications to consume.

A pipeline is a logical grouping of activities that performs a unit of work. Together, the activities in a pipeline perform a task. For example, a pipeline might contain a series of activities that ingests raw data from Azure Blob storage, and then runs a Hive query on an HDInsight cluster to partition the data and store the results in a Cosmos DB database.

What is Azure Data Lake?
Azure Data Lake is a collection of analytics and storage services that you can combine to implement a big data solution. It comprises three main elements:

- Data Lake Store
- Data Lake Analytics
- HDInsight

What is Data Lake Store?
Data Lake Store provides a file system that can store near limitless quantities of data. It uses a hierarchical organization (like the Windows and Linux file systems), but you can hold massive amounts of raw data (blobs) and structured data. It is optimized for analytics workloads.

Azure Data Lake Store is compatible with the Hadoop Distributed File System (HDFS). You can run Hadoop jobs using Azure HDInsight (see below) that can read and write data in Data Lake Store efficiently.

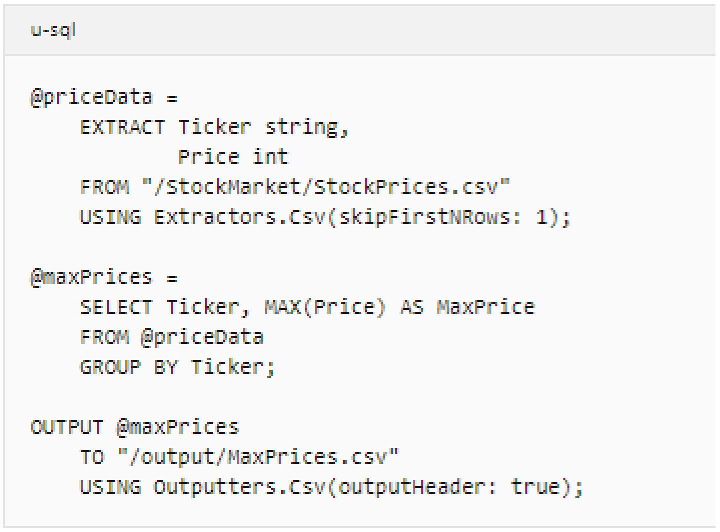
Azure Data Lake Store provides granular security over data, using Access Control Lists. An Access Control List specifies which accounts can access which files and folders in the store. If you are more familiar with Linux, you can use POSIX-style permissions to grant read, write, and search access based on file ownership and group membership of users.

Services such as Azure Data Factory, Azure Databricks, Azure HDInsight, Azure Data Lake Analytics, and Azure Stream Analytics can read and write Data Lake Store directly.

What is Data Lake Analytics?
Azure Data Lake Analytics is an on-demand analytics job service that you can use to process big data. It provides a framework and set of tools that you use to analyze data held in Microsoft Azure Data Lake Store, and other repositories. You write jobs that contain queries to transform data and extract insights.

You define a job using a language called U-SQL. This is a hybrid language that takes features from both SQL and C#, and provides declarative and procedural capabilities that you can use to process data.

The example U-SQL block below reads data from a file named StockPrices.csv, which is held in a folder named StockMarket in Data Lake Storage. This is a text file that contains stock market information (tickers, and prices, and possibly other data), held in comma-separated format. The EXTRACT statement reads the file line by line and pulls out the data in the Ticker, and Price fields (it skips the first line, where a CSV file typically holds field name information rather than data). The SELECT statement calculates that maximum price for each ticker. The OUTPUT statement stores the results to another file in Data Lake Storage.



```
@priceData =
    EXTRACT Ticker string,
            Price int
    FROM "/StockMarket/StockPrices.csv"
    USING Extractors.Csv(skipFirstNRows: 1);

@maxPrices =
    SELECT Ticker, MAX(Price) AS MaxPrice
    FROM @priceData
    GROUP BY Ticker;

OUTPUT @maxPrices
    TO "/output/MaxPrices.csv"
    USING Outputters.Csv(outputHeader: true);
```

It's important to understand that the U-SQL code only provides a description of the work to be performed. Azure Data Lake Analytics determines how best to actually carry out this work. Data Lake Analytics takes the U-SQL description of a job, parses it to make sure it is syntactically correct, and then compiles it into an internal representation. Data Lake Analytics then breaks down this internal representation into stages of execution. Each stage performs a task, such as extracting the data from a specified source, dividing the data into partitions, processing the data in each partition, aggregating the results in a partition, and then combining the results from across all partitions. Partitioning is used to improve parallelization, and the processing for different partitions is performed concurrently on different processing nodes. The data for each partition is determined by the U-SQL compiler, according to the way in which the job retrieves and processes the data.

A U-SQL job can output results to a single CSV file, partition the results across multiple files, or can write to other destinations. For example, Data Lake Analytics enables you to create custom outputters if you want to save data in a particular format (such as XML or HTML). You can also write data to the Data Lake Catalog. The catalog provides a SQL-like interface to Data Lake Storage, enabling you to create tables, and views, and run INSERT, UPDATE, and DELETE statements against these tables and views.

II. Explore Azure Synapse Analytics

What are the components of Azure Synapse Analytics?

Azure Synapse Analytics is an integrated analytics service that allows organizations to gain insights quickly from all their data at any hyperscale, from both data warehouses and big data analytics systems.

Azure Synapse is composed of the following elements:

- Synapse SQL pool: This is a collection of servers running Transact-SQL. Transact-SQL is the dialect of SQL used by Azure SQL Database, and Microsoft SQL Server. You write your data processing logic using Transact-SQL.
- Synapse Spark pool: This is a cluster of servers running Apache Spark to process data. You write your data processing logic using one of the four supported languages: Python, Scala, SQL, and C# (via .NET for Apache Spark). Spark pools support Azure Machine Learning through integration with the SparkML and AzureML packages.
- Synapse Pipelines: A Synapse pipeline is a logical grouping of activities that together perform a task. The activities in a pipeline define actions to perform on your data. For example, you might use a copy activity to transform data from a source dataset to a destination dataset. You could include activities that transform the data as it is transferred, or you might combine data from multiple sources together.
- Synapse Link: This component allows you to connect to Cosmos DB. You can use it to perform near real-time analytics over the operational data stored in a Cosmos DB database.
- Synapse Studio: This is a web user interface that enables data engineers to access all the Synapse Analytics tools. You can use Synapse Studio to create SQL and Spark pools, define and run pipelines, and configure links to external data sources.

What are SQL pools?

When you use Synapse SQL, your analytics workload runs using a SQL pool. In a SQL pool, the Control and Compute nodes in the cluster run a version of Azure SQL Database that supports distributed queries. You define your logic using Transact-SQL statements. You send your Transact-SQL statements to the control node, which splits up the work into queries that operate over a subset of the data, and then sends these smaller queries to the compute nodes. The data is split into chunks called distributions. A distribution is the basic unit of storage and processing for parallel queries that run on distributed data. Each of the smaller queries runs on one of the data distributions.

The control and compute nodes use the Data Movement Service (DMS) to move data across the nodes as necessary to run queries in parallel and return accurate results.

Synapse Analytics uses a technology called PolyBase to make external data look like SQL tables. You can run queries against these tables directly, or you can transfer the data into a series of SQL tables managed by Synapse Analytics for querying later. Synapse uses Azure Storage to manage your data while it's being processed.

By default, an on-demand SQL pool is created in each Azure Synapse Analytics workspace. You can then provision additional pools, either on-demand or provisioned.

Azure Synapse Analytics is designed to run queries over massive datasets. You can manually scale the SQL pool up to 60 nodes. You can also pause a SQL pool if you don't require it for a while. Pausing releases the resources associated with the pool. You aren't charged for these resources until you manually resume the pool. However, you can't run any queries until the pool is resumed. Resuming a pool can take several minutes.

Use SQL pools in Synapse Analytics for the following scenarios:

- Complex reporting. You can use the full power of Transact-SQL to run complex SQL statements that summarize and aggregate data.
- Data ingestion. PolyBase enables you to retrieve data from many external sources and convert it into a tabular format. You can reformat this data and save it as tables and materialized views in Azure Synapse.

What are Spark pools?

Synapse Spark runs clusters based on Apache Spark rather than Azure SQL Database. You write your analytics jobs as notebooks, using code written in Python, Scala, C#, or Spark SQL (this is a different dialect from Transact-SQL). You can combine code written in multiple languages in the same notebook.

Notebooks also allow you to visualize data through graphs, and transform data as it's loaded. The data can then be used by Spark Machine Learning (SparkML) and Azure Machine Learning (AzureML) to train machine learning models that support artificial intelligence.

Spark pools enable you to process data held in many formats, such as csv, json, xml, parquet, orc, and avro. Spark can be extended to support many more formats with external data sources.

Spark pools provide the basic building blocks for performing in-memory cluster computing. A Spark job can load and cache data into memory and query it repeatedly. In-memory computing is much faster than disk-based applications. Spark pools in Azure Synapse are compatible with Azure Storage and Azure Data Lake Storage, so you can use Spark pools to process your data stored in Azure.

Spark pools can have autoscaling enabled, so that pools scale by adding or removing nodes as needed. Also, Spark pools can be shut down with no loss of data since all the data is stored in Azure Storage or Data Lake Storage.

Spark pools in Synapse Analytics are especially suitable for the following scenarios:

- Data Engineering/Data Preparation. Apache Spark includes many language features to support preparation and processing of large volumes of data so that it can be made more valuable and then consumed by other services within Synapse Analytics. This is enabled through the Spark libraries that support processing and connectivity.
- Machine Learning. Apache Spark comes with MLlib, a machine learning library built on top of Spark that you can use from a Spark pool in Synapse Analytics. Spark pools in Synapse Analytics also include Anaconda, a Python distribution with a variety of packages for data science including machine learning. When combined with built-in support for notebooks, you have an environment for creating machine learning applications.

What are Synapse pipelines?

A pipeline is a logical grouping of activities that together perform a task. For example, a pipeline could contain a set of activities that ingest and clean log data, and then kick off a mapping data flow to analyze the log data. The pipeline allows you to manage the activities as a set instead of each one individually. You deploy and schedule the pipeline instead of the activities independently.

The activities in a pipeline define actions to perform on your data. For example, you may use a copy activity to copy data from Azure Blob Storage into Azure Synapse using a SQL pool. Then, use a data flow activity or a notebook activity using a Spark pool to process and generate a machine learning model.

Synapse pipelines use the same Data Integration engine used by Azure Data Factory. This gives you the power in Synapse Studio to create pipelines that can connect to over 90 sources from flat files, databases, or online services. You can create codeless data flows that let you do complex mappings and transformations on data as it flows into your analytic solutions.

What is Synapse Link?

Azure Synapse Link for Azure Cosmos DB is a cloud-native hybrid transactional and analytical processing (HTAP) capability that enables you to run near real-time analytics over operational data stored in Azure Cosmos DB.

Synapse link uses a feature of Cosmos DB named Cosmos DB Analytical Store. Cosmos DB Analytical Store contains a copy of the data in a Cosmos DB container, but organized as a column store. Column stores group data by column rather than by row. Column stores are a more optimal format for running analytical workloads that need to aggregate data down a column rather than across a row, such as generating sum totals, averages, maximum or minimum values for a column. Cosmos DB automatically keeps the data in its containers synchronized with the copies in the column store.

Azure Synapse Link enables you to run workloads that retrieve data directly from Cosmos DB and run analytics workloads using Azure Synapse Analytics. The data doesn't have to go through an ETL (extract, transform, and load) process because the data isn't copied into Synapse Analytics; it remains in the Cosmos DB analytical store.

Business analysts, data engineers, and data scientists can now use Synapse Spark pools or Synapse SQL pools to run near real-time business intelligence, analytics, and machine learning pipelines. You can achieve this without impacting the performance of your transactional workloads on Azure Cosmos DB.

Synapse link has a wide range of uses, including:

- Supply chain analytics and forecasting. You can query operational data directly and use it to build machine learning models. You can use the results generated by these models back into Cosmos DB for near-real-time scoring. You can use these assessments to successively refine the models and generate more accurate forecasts.
- Operational reporting. You can use Synapse Analytics to query operational data using Transact-SQL running in a SQL pool. You can publish the results to dashboards using the support provided to familiar tools such as Microsoft Power BI.

- Batch data integration and orchestration. With supply chains getting more complex, supply chain data platforms need to integrate with a variety of data sources and formats. The Azure Synapse data integration engine allows data engineers to create rich data pipelines without requiring a separate orchestration engine.
- Real-time personalization. You can build engaging ecommerce solutions that allow retailers to generate personalized recommendations and special offers for customers in real time.
- IoT maintenance. Industrial IoT innovations have drastically reduced downtimes of machinery and increased overall efficiency across all fields of industry. One such innovation is predictive maintenance analytics for machinery at the edge of the cloud. The historical operational data from IoT device sensors could be used to train predictive models such as anomaly detectors. These anomaly detectors are then deployed back to the edge for real-time monitoring. Looping back allows for continuous retraining of the predictive models.

What is Synapse Studio?

Synapse Studio is a web interface that enables you to create pools and pipelines interactively. With Synapse Studio you can develop, test, and debug Spark notebooks and Transact-SQL jobs. You can monitor the performance of operations that are currently running, and you can manage the serverless or provisioned resources. All of these capabilities are accessed via the web-native Synapse Studio that allows for model management, monitoring, coding, and security.

Azure Data Fundamentals: Get started building with Power BI

I. Introduction

Microsoft Power BI is a collection of software services, apps, and connectors that work together to turn your unrelated sources of data into coherent, visually immersive, and interactive insights.

Power BI can be simple and fast, capable of creating quick insights from an Excel workbook or a local database. But Power BI is also robust and enterprise-grade, ready not only for extensive modeling and real-time analytics, but also for custom development. Therefore, it can be your personal report and visualization tool, but can also serve as the analytics and decision engine behind group projects, divisions, or entire corporations.

The parts of Power BI
Power BI consists of a Microsoft Windows desktop application called Power BI Desktop, an online SaaS (Software as a Service) service called the Power BI service, and mobile Power BI apps that are available on any device, with native mobile BI apps for Windows, iOS, and Android.

These three elements—Desktop, the service, and Mobile apps—are designed to let people create, share, and consume business insights in the way that serves them, or their role, most effectively.

How Power BI matches your role

How you use Power BI might depend on your role on a project or a team. And other people, in other roles, might use Power BI differently, which is just fine.

For example, you might view reports and dashboards in the Power BI service, and that might be all you do with Power BI. But your number-crunching, business-report-creating coworker might make extensive use of Power BI Desktop (and publish Power BI Desktop reports to the Power BI service, which you then use to view them). And another coworker, in sales, might mainly use her Power BI phone app to monitor progress on her sales quotas and drill into new sales lead details.

You also might use each element of Power BI at different times, depending on what you're trying to achieve, or what your role is for a given project or effort.

Perhaps you view inventory and manufacturing progress in a real-time dashboard in the service, and also use Power BI Desktop to create reports for your own team about customer engagement statistics. How you use Power BI can depend on which feature or service of Power BI is the best tool for your situation. But each part of Power BI is available to you, which is why it's so flexible and compelling.

We discuss these three elements—Desktop, the service, and Mobile apps—in more detail later. In upcoming units and modules, we'll also create reports in Power BI Desktop, share them in the service, and eventually drill into them on our mobile device.

The flow of work in Power BI

A common flow of work in Power BI begins in Power BI Desktop, where a report is created. That report is then published to the Power BI service and finally shared, so that users of Power BI Mobile apps can consume the information.

It doesn't always happen that way, and that's okay. But we'll use that flow to help you learn the different parts of Power BI and how they complement each other.

Okay, now that we have an overview of this module, what Power BI is, and its three main elements, let's take a look at what it's like to use Power BI.

II. Use Power BI

The activities and analyses that you'll learn with Power BI generally follow a common flow. The common flow of activity looks like this:

1. Bring data into Power BI Desktop, and create a report.
2. Publish to the Power BI service, where you can create new visualizations or build dashboards.
3. Share dashboards with others, especially people who are on the go.
4. View and interact with shared dashboards and reports in Power BI Mobile apps.

As mentioned earlier, you might spend all your time in the Power BI service, viewing visuals and reports that have been created by others. And that's fine. Someone else on your team might spend their time in Power BI Desktop, which is fine too. To help you understand the full continuum of Power BI and what it can do, we'll show you all of it. Then you can decide h ow to use it to your best advantage.

So, let's jump in and step through the experience. Your first order of business is to learn the basic building blocks of Power BI, which will provide a solid basis for turning data into cool reports and visuals.

III. Building blocks of Power BI

Everything you do in Microsoft Power BI can be broken down into a few basic building blocks. After you understand these building blocks, you can expand on each of them and begin creating elaborate and complex reports. After all, even seemingly complex things are built from basic building blocks. For example, buildings are created with wood, steel, concrete and glass, and cars are made from metal, fabric, and rubber. Of course, buildings and cars can also be basic or elaborate, depending on how those basic building blocks are arranged.

Let's take a look at these basic building blocks, discuss some simple things that can be built with them, and then get a glimpse into how complex things can also be created.

Here are the basic building blocks in Power BI:

- Visualizations
- Datasets
- Reports
- Dashboards
- Tiles

Visualizations

A visualization (sometimes also referred to as a visual) is a visual representation of data, like a chart, a color-coded map, or other interesting things you can create to represent your data visually. Power BI has all sorts of visualization types, and more are coming all the time. The following image shows a collection of different visualizations that were created in the Power BI service.

Visualizations can be simple, like a single number that represents something significant, or they can be visually complex, like a gradient-colored map that shows voter sentiment about a certain social issue or concern. The goal of a visual is to present data in a way that provides context and insights, both of which would probably be difficult to discern from a raw table of numbers or text.

Datasets
A dataset is a collection of data that Power BI uses to create its visualizations.

You can have a simple dataset that's based on a single table from a Microsoft Excel workbook, similar to what's shown in the following image.

Datasets can also be a combination of many different sources, which you can filter and combine to provide a unique collection of data (a dataset) for use in Power BI.

For example, you can create a dataset from three database fields, one website table, an Excel table, and online results of an email marketing campaign. That unique combination is still considered a single dataset, even though it was pulled together from many different sources.

Filtering data before bringing it into Power BI lets you focus on the data that matters to you. For example, you can filter your contact database so that only customers who received emails from the marketing campaign are included in the dataset. You can then create visuals based on that subset (the filtered collection) of customers who were included in the campaign. Filtering helps you focus your data—and your efforts.

An important and enabling part of Power BI is the multitude of data connectors that are included. Whether the data you want is in Excel or a Microsoft SQL Server database, in Azure or Oracle, or in a service like Facebook, Salesforce, or MailChimp, Power BI has built-in data connectors that let you easily connect to that data, filter it if necessary, and bring it into your dataset.

After you have a dataset, you can begin creating visualizations that show different portions of it in different ways, and gain insights based on what you see. That's where reports come in.

Reports
In Power BI, a report is a collection of visualizations that appear together on one or more pages. Just like any other report you might create for a sales presentation or write for a school assignment, a report in Power BI is a collection of items that are related to each other. The following image shows a report in Power BI Desktop—in this case, it's the second page in a five-page report. You can also create reports in the Power BI service.

Reports let you create many visualizations, on multiple pages if necessary, and let you arrange those visualizations in whatever way best tells your story.

You might have a report about quarterly sales, product growth in a particular segment, or migration patterns of polar bears. Whatever your subject, reports let you gather and organize your visualizations onto one page (or more).

Dashboards
When you're ready to share a single page from a report, or a collection of visualizations, you create a dashboard. Much like the dashboard in a car, a Power BI dashboard is a collection of visuals from a single page that you can share with others. Often, it's a selected group of visuals that provide quick insight into the data or story you're trying to present.

A dashboard must fit on a single page, often called a canvas (the canvas is the blank backdrop in Power BI Desktop or the service, where you put visualizations). Think of it like the canvas that an artist or painter uses—a workspace where you create, combine, and rework interesting and compelling visuals. You can share dashboards with other users or groups, who can then interact with your dashboards when they're in the Power BI service or on their mobile device.

Tiles
In Power BI, a tile is a single visualization on a report or a dashboard. It's the rectangular box that holds an individual visual. In the following image, you see one tile, which is also surrounded by other tiles.

When you're creating a report or a dashboard in Power BI, you can move or arrange tiles however you want. You can make them bigger, change their height or width, and snuggle them up to other tiles.

When you're viewing, or consuming, a dashboard or report—which means you're not the creator or owner, but the report or dashboard has been shared with you—you can interact with it, but you can't change the size of the tiles or their arrangement.

All together now
Those are the basics of Power BI and its building blocks. Let's take a moment to review.

Power BI is a collection of services, apps, and connectors that lets you connect to your data, wherever it happens to reside, filter it if necessary, and then bring it into Power BI to create compelling visualizations that you can share with others.

Now that you've learned about the handful of basic building blocks of Power BI, it should be clear that you can create datasets that make sense to you and create visually compelling reports that tell your story. Stories told with Power BI don't have to be complex, or complicated, to be compelling.

For some people, using a single Excel table in a dataset and then sharing a dashboard with their team will be an incredibly valuable way to use Power BI.

For others, the value of Power BI will be in using real-time Azure SQL Data Warehouse tables that combine with other databases and real-time sources to build a moment-by-moment dataset.

For both groups, the process is the same: create datasets, build compelling visuals, and share them with others. And the result is also the same for both groups: harness your ever-expanding world of data, and turn it into actionable insights.

Whether your data insights require straightforward or complex datasets, Power BI helps you get started quickly and can expand with your needs to be as complex as your world of data requires. And because Power BI is a Microsoft product, you can count on it being robust, extensible, Microsoft Office–friendly, and enterprise-ready.

Now let's see how this works. We'll start by taking a quick look at the Power BI service.

IV. Tour and use the Power BI service

But because some people begin in the Power BI service, let's take a quick look at that first, and learn about an easy and popular way to quickly create visuals in Power BI: apps.

An app is a collection of preset, ready-made visuals and reports that are shared with an entire organization. Using an app is like microwaving a TV dinner or ordering a fast-food value meal: you just have to press a few buttons or make a few comments, and you're quickly served a collection of entrees designed to go together, all presented in a tidy, ready-to-consume package.

Create out-of-box dashboards with cloud services
With Power BI, connecting to data is easy. From the Power BI service, you can just select the Get Data button in the lower-left corner of the home page.

The canvas (the area in the center of the Power BI service) shows you the available sources of data in the Power BI service. In addition to common data sources like Microsoft Excel files, databases, or Microsoft Azure data, Power BI can just as easily connect to a whole assortment of software services (also called SaaS providers or cloud services): Salesforce, Facebook, Google Analytics, and more.

For these software services, the Power BI service provides a collection of ready-made visuals that are pre-arranged on dashboards and reports for your organization. This collection of visuals is called an app. Apps get you up and running quickly, with data and dashboards that your organization has created for you. For example, when you use the GitHub app, Power BI connects to your GitHub account (after you provide your credentials) and then populates a predefined collection of visuals and dashboards in Power BI.

There are apps for all sorts of online services. The following image shows a page of apps that are available for different online services, in alphabetical order. This page is shown when you select the Get button in the Services box (shown in the previous image). As you can see from the following image, there are many apps to choose from.

For our purposes, we'll choose GitHub. GitHub is an application for online source control. When you select the Get it now button in the box for the GitHub app, the Connect to GitHub dialog box appears. Note that Github does not support Internet Explorer, so make sure you are working in another browser.

After you enter the information and credentials for the GitHub app, installation of the app begins.

After the data is loaded, the predefined GitHub app dashboard appears.

In addition to the app dashboard, the report that was generated (as part of the GitHub app) and used to create the dashboard is available, as is the dataset (the collection of data pulled from GitHub) that was created during data import and used to create the GitHub report.

On the dashboard, you can select any of the visuals and interact with them. As you do so, all the other visuals on the page will respond. For example, when the May 2018 bar is selected in the Pull Requests (by month) visual, the other visuals on the page adjust to reflect that selection.

Update data in the Power BI service

You can also choose to update the dataset for an app, or other data that you use in Power BI. To set update settings, select the schedule update icon for the dataset to update, and then use the menu that appears. You can also select the update icon (the circle with an arrow) next to the schedule update icon to update the dataset immediately.

The Datasets tab is selected on the Settings page that appears. In the right pane, select the arrow next to Scheduled refresh to expand that section. The Settings dialog box appears on the canvas, letting you set the update settings that meet your needs.