# Classify your data

## Structured data

Structured data, sometimes referred to as relational data, is data that adheres to a strict schema, so all of the data has the same fields or properties. The shared schema allows this type of data to be easily searched with query languages such as SQL (Structured Query Language). This capability makes this data style perfect for applications such as CRM systems, reservations, and inventory management.

Structured data is often stored in database tables with rows and columns with key columns to indicate how one row in a table relates to data in another row of another table.

## Semi-structured data

Semi-structured data is less organized than structured data, and is not stored in a relational format, as the fields do not neatly fit into tables, rows, and columns. Semi-structured data contains tags that make the organization and hierarchy of the data apparent - for example, key/value pairs. Semi-structured data is also referred to as non-relational or NoSQL data.

***XML***

XML expresses the shape of the data using tags. These tags come in two forms: elements such as <FirstName> and \_attributes that can be expressed in text like Age="23". Elements can have child elements to express relationships - such as the <Hobbies> tag above which is expressing a collection of <Hobby> elements. It tends to be more verbose making it larger to store, process, or pass over a network.

***JSON***

JSON has a lightweight specification and relies on curly braces to indicate data structure. Compared to XML, it is less verbose and easier to read by humans. JSON is frequently used by web services to return data.

***YAML***

YAML is a relatively new data language that’s growing quickly in popularity in part due to its human-friendliness. The data structure is defined by line separation and indentation, and reduces the dependency on structural characters like parentheses, commas and brackets. It’s often used for configuration files.

## Unstructured data

The organization of unstructured data is ambiguous. Unstructured data is often delivered in files, such as photos or videos. The video file itself may have an overall structure and come with semi-structured metadata, but the data that comprises the video itself is unstructured. Therefore, photos, videos, and other similar files are classified as unstructured data.

# Transactions

A transaction is a logical group of database operations that execute together.

Here's the question to ask yourself regarding whether you need to use transactions in your application: Will a change to one piece of data in your dataset impact another? If the answer is yes, then you'll need support for transactions in your database service.

Transactions are often defined by a set of four requirements, referred to as **ACID** guarantees. ACID stands for **Atomicity**, **Consistency**, **Isolation**, and **Durability**:

* **Atomicity** means a transaction must execute exactly once and must be atomic; either all of the work is done, or none of it is. Operations within a transaction usually share a common intent and are interdependent.
* **Consistency** ensures that the data is consistent both before and after the transaction.
* **Isolation** ensures that one transaction is not impacted by another transaction.
* **Durability** means that the changes made due to the transaction are permanently saved in the system. Committed data is saved by the system so that even in the event of a failure and system restart, the data is available in its correct state.

## OLTP vs OLAP

Transactional databases are often called **OLTP** (Online Transaction Processing) systems. OLTP systems commonly support lots of users, have quick response times, and handle large volumes of data. They are also highly available (meaning they have very minimal downtime), and typically handle small or relatively simple transactions.

On the contrary, **OLAP** (Online Analytical Processing) systems commonly support fewer users, have longer response times, can be less available, and typically handle large and complex transactions.

# Choose a storage solution on Azure

Choosing the correct storage solution can lead to better performance, cost savings, and improved manageability.

## Azure Cosmos DB

Azure Cosmos DB supports **semi-structured** **data**, or NoSQL data, by design. So, supporting new fields. It supports SQL for queries and every property is indexed by default. You can create queries. It is also ACID-compliant, so you can be assured that your transactions are completed according to those strict requirements.

## Azure Blob storage

Azure Blob storage supports storing files (**unstructured data**) such as photos and videos. It also works with Azure Content Delivery Network (CDN) by caching the most frequently used content and storing it on edge servers. Azure CDN reduces latency in serving up those images to your users.

By using Azure Blob storage, you can also move images from the hot storage tier to the cool or archive storage tier, to reduce costs and focus throughput on the most frequently viewed images and videos.

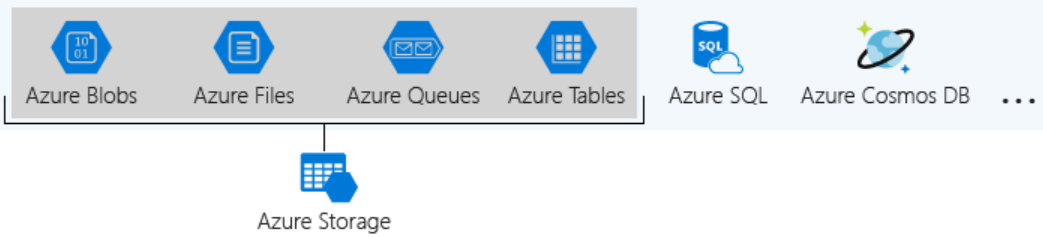
## Azure SQL Database

**Structured** data will most likely be queried by business analysts, who are more likely to know SQL than any other query language.

# Azure Storage

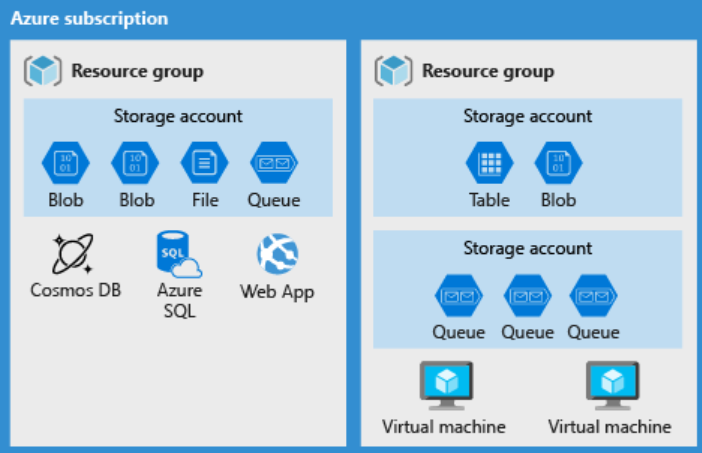
Azure provides many ways to store your data. There are multiple database options like Azure SQL Server, Azure Cosmos DB, and Azure Table Storage. Azure offers multiple ways to store and send messages, such as Azure Queues and Event Hubs. You can even store loose files using services like Azure Files and Azure Blobs.

Azure selected Azure Blobs, Azure Files, Azure Queues, and Azure Tables and placed them together under the name Azure Storage



## Storage account

A storage account is a container that groups a set of Azure Storage services together. Combining data services into a storage account lets you manage them as a group. The settings you specify when you create the account, or any that you change after creation, are applied to everything in the account.



## Storage account settings

A storage account defines a policy that applies to all the storage services in the account. For example, you could specify that all the contained services will be stored in the West US datacenter, accessible only over https, and billed to the sales department's subscription.

***Related to the data stored in the account***

* **Subscription**: The Azure subscription that will be billed for the services in the account.
* **Location**: The datacenter that will store the services in the account.
* **Performance**: Determines the data services you can have in your storage account and the type of hardware disks used to store the data. Standard allows you to have any data service (Blob, File, Queue, Table) and uses magnetic disk drives. Premium introduces additional services for storing data.
* **Replication**: Determines the strategy used to make copies of your data to protect against hardware failure or natural disaster. At a minimum, Azure will automatically maintain three copies of your data within the data center associated with the storage account. This is called locally-redundant storage (LRS), and guards against hardware failure but does not protect you from an event that incapacitates the entire datacenter. You can upgrade to one of the other options such as geo-redundant storage (GRS) to get replication at different datacenters across the world.
* **Access tier**: Controls how quickly you will be able to access the blobs in this storage account. Hot gives quicker access than Cool, but at increased cost. This applies only to blobs, and serves as the default value for new blobs.
* **Secure transfer required**: A security feature that determines the supported protocols for access. Enabled requires HTTPs, while disabled allows HTTP.
* **Virtual networks**: A security feature that allows inbound access requests only from the virtual network(s) you specify.

***Related to the account itself***

* **Name**: Each storage account has a name. The name must be globally unique within Azure, use only lowercase letters and digits and be between 3 and 24 characters.
* **Deployment model**: is the system Azure uses to organize your resources. The model defines the API that you use to create, configure, and manage those resources. Azure provides two deployment models:
  + *Resource Manager*: the current model that uses the Azure Resource Manager API. It adds the concept of a resource group, which is not available in the classic model, and lets you deploy and manage a collection of resources as a single unit. **Microsoft recommends that you use Resource Manager for all new resources.**
  + *Classic*: a legacy offering that uses the Azure Service Management API
* **Account kind**: Storage account kind is a set of policies that determine which data services you can include in the account and the pricing of those services. There are three kinds of storage accounts:
  + *StorageV2* (general purpose v2): the current offering that supports all storage types and all of the latest features. **Microsoft recommends that you use the General-purpose v2 option for new storage accounts.**
  + *Storage* (general purpose v1): a legacy kind that supports all storage types but may not support all features
  + *Blob storage*: a legacy kind that allows only block blobs and append blobs