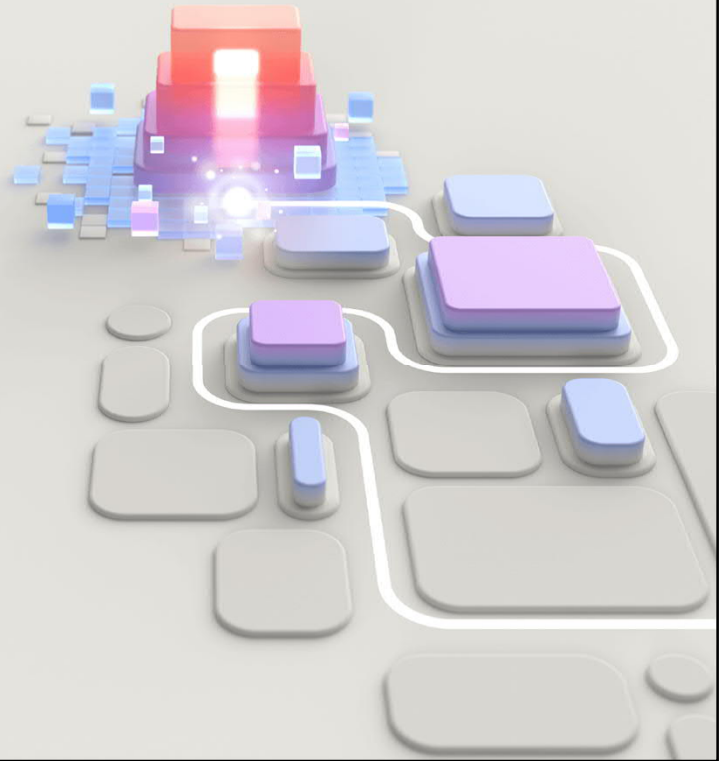


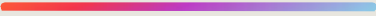


# Explore fundamentals of data



© Copyright Microsoft Corporation. All rights reserved.

# Agenda



- Core data concepts
- Data roles and services

© Copyright Microsoft Corporation. All rights reserved.

# Learning objectives

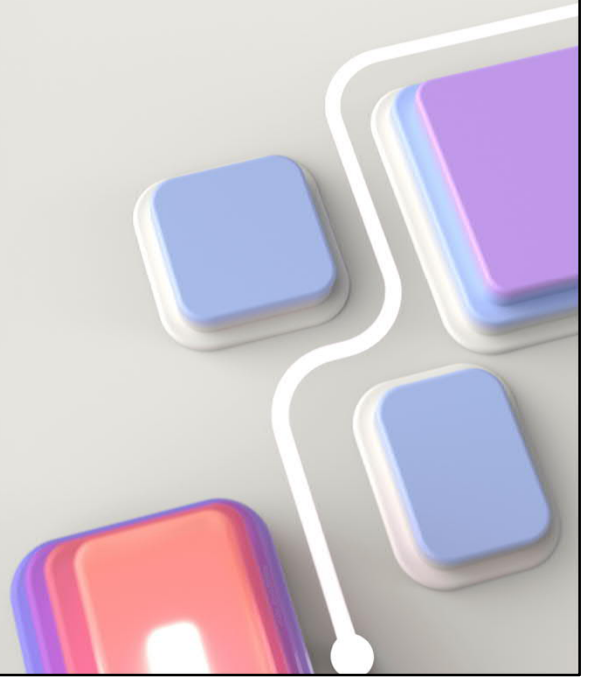
After completing this module, you will be able to:

- 1 Identify common data formats.
- 2 Describe options for storing data in files and databases.
- 3 Describe characteristics of transactional and analytical data processing solutions.
- 4 Identify common data professional roles.
- 5 Identify common cloud services used by data professionals.

© Copyright Microsoft Corporation. All rights reserved.

# 1: Core data concepts

© Copyright Microsoft Corporation. All rights reserved.



# What is data?

Values used to record information – Often representing *entities* that have one or more *attributes*

## Structured

Customer				
ID	FirstName	LastName	Email	Address
1	Joe	Jones	joe@litware.com	1 Main St.
2	Samir	Nadoy	samir@northwind.com	123 Elm Pl.

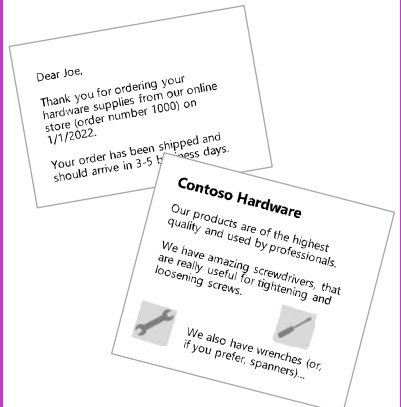
Product		
ID	Name	Price
123	Hammer	2.99
162	Screwdriver	3.49
201	Wrench	4.25

## Semi-structured

```
{
  "firstName": "Joe",
  "lastName": "Jones",
  "address": {
    "streetAddress": "1 Main
St.",
    "city": "New York",
    "state": "NY",
    "postal Code": "10099"
  },
  "contact": {
    {
      "type": "home",
      "number": "555 123-
4567"
    },
    {
      "type": "email",
      "address":
"joe@litware.com"
    }
  }
}
```

```
{
  "firstName": "Samir",
  "lastName": "Nadoy",
  "address": {
    "streetAddress": "123 Elm
Pl.",
    "unit": "500",
    "city": "Seattle",
    "state": "WA",
    "postal Code": "98999"
  },
  "contact": {
    {
      "type": "email",
      "address":
"samir@northwind.com"
    }
  }
}
```

## Unstructured



© Copyright Microsoft Corporation. All rights reserved.

Data is a collection of facts such as numbers, descriptions, and observations used to record information. Data structures in which this data is organized often represents *entities* that are important to an organization (such as customers, products, sales orders, and so on). Each entity typically has one or more *attributes*, or characteristics (for example, a customer might have a name, an address, a phone number, and so on).

You can classify data as *structured*, *semi-structured*, or *unstructured*.

- **Structured data** is data that adheres to a fixed schema, so all of the data has the same fields or properties. Structured data is often stored in database tables with rows and columns, and multiple tables can reference one another by using key values in a relational model.
- **Semi-structured data** is information that has some structure, but which allows for some variation between entity instances. For example, while most customers may have an email address, might have multiple email addresses, and some might have none at all.
- **Unstructured data** is data that does not have a fixed schema. Examples include documents, free-form text, images, videos, audio streams, etc.

# How is data stored?

## Files

### Delimited Text

```
FirstName,LastName,Email
Joe,Jones,joe@litware.com
Samir,Nadoy,samir@northwind.com
```

### JavaScript Object Notation (JSON)

```
{
  "customers":
  [
    { "firstName": "Joe", "lastName": "Jones" },
    { "firstName": "Samir", "lastName": "Nadoy" }
  ]
}
```

### Extensible Markup Language (XML)

```
<Customer firstName="Joe" lastName="Jones"/>
```

### Binary Large Object (BLOB)

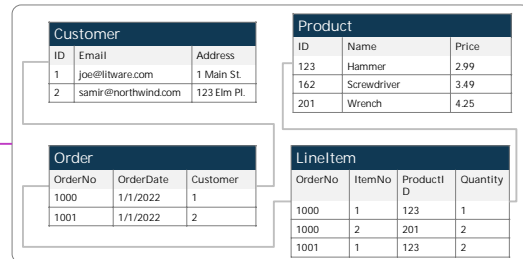
```
10110101101010110010...
```

### Optimized formats:

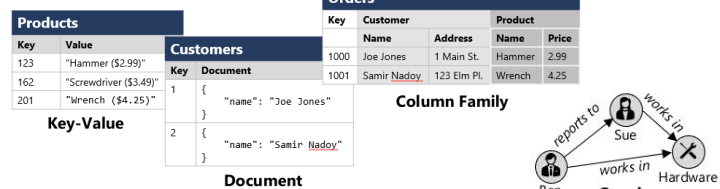
- Avro, ORC, Parquet

## Databases

### Relational



### Non-Relational



© Copyright Microsoft Corporation. All rights reserved.

## Files

### > animated slide, click to proceed

File formats for data include:

- **Delimited text:** Data is stored in plain text format with specific field delimiters and row terminators. The most common format for delimited data is *comma-separated values (CSV)* in which fields are separated by commas, and rows are terminated by a carriage return / new line. Optionally, the first line may include the field names. Other common formats include *tab-separated values (TSV)* and *space-delimited* (in which tabs or spaces are used to separate fields), and *fixed-width* data in which each field is allocated a fixed number of characters. Delimited text is a good choice for structured data.
- **JavaScript Object Notation (JSON):** A hierarchical document schema is used to define data entities (*objects*) that have multiple *attributes*. Each attribute might be an object (or a collection of objects); making JSON a very flexible format that's good for both structured and semi-structured data.
- **Extensible Markup Language (XML):** XML is a text-based format that defines data entities and their attributes using markup *tags*. XML was a commonly used format in the early 2000's, but the increasing popularity of JSON has reduced its prevalence. For example:  

```
<CustomerEmail FirstName="Joe" LastName="Jones">joe@litware.com</Customer>
```
- **Binary Large Object (BLOB):** BLOB is the term used to describe binary data. Technically, all files are BLOBs (as ultimately, all files are stored as bits); but plain text formats like CSV and JSON, store binary values that map to specific text characters based on a set of ASCII or UNICODE codes; and can be opened and read by humans. Other files such as Word documents, PDFs, images, audio or video streams, and so on use a binary format that can only be interpreted by compatible software applications. In Azure, unstructured data is usually stored as a *block blob* file – a format that supports basic read and write operations.
- **Optimized formats:** As the volume of data that organizations need to work with has grown, a number of data formats that include features to enable metadata, compression, indexing, and other optimization techniques for specific types of workload have been created and are in common use. These include:
  - **Avro:** 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.

- **Optimized Row Columnar (ORC):** ORC, 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:** Parquet is a columnar format created by Cloudera and Twitter. 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.

## Databases

A database is used to define a central system in which data can be stored and queried. In a simplistic sense, the file system on which files are stored is a kind of database; but when we use the term in a professional data context, we usually mean a dedicated system for managing data *records* rather than files.

- **Relational Databases** are commonly used to store and query structured data. The data is stored in tables that represent *entities*, such as customers, products, or sales orders. Each instance of an entity is assigned a *primary key* that uniquely identifies it; and these keys are used to reference the entity instance in other tables. For example, a customer's primary key can be referenced in a sales order record to indicate which customer placed the order. This use of keys to reference data entities enables a relational database to be *normalized*; in other words, it eliminates duplication of data values – the details of an individual customer are stored only once; not for each sales order the customer places. The tables are managed and queried using Structured Query Language (SQL) – which is based on an ANSI standard, so it's similar across multiple database systems
- **Non-Relational databases** are data management systems that do not apply a relational schema to the data. Common types of non-relational database include *key-value* stores, in which each record consists of a unique key and an associated value, which can be in any format; *document* databases, which are a specific form of key-value database in which the value is a JSON document (which the system is optimized to query), *Column family databases*, which store 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, and *graph* databases, which store entities as *nodes* with links to define relationships between them. Non-relational databases are often referred to as *NoSQL* database, even though some support a variant of the SQL language.

# Operational data workloads



Data is stored in a database that is optimized for *online transactional processing* (OLTP) operations that support applications

A mix of *read* and *write* activity

For example:

- Read the *Product* table to display a catalog
- Write to the *Order* table to record a purchase

Data is stored using *transactions*

Transactions are "ACID" based:

- Atomicity – Each transaction is treated as a single unit of work, which succeeds completely or fails completely
- Consistency – Transactions can only take the data in the database from one valid state to another
- Isolation – Concurrent transactions cannot interfere with one another
- Durability – When a transaction has succeeded, the data changes are persisted in the database

© Copyright Microsoft Corporation. All rights reserved.

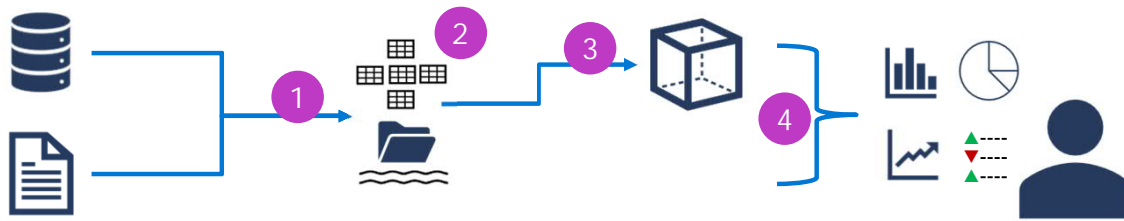
OLTP is typically a *live* system in which data storage is optimized for both *read* and *write* operations in order to support *transactional* workloads

Updates are made transactionally, for example, an order is placed. Each individual order is stored in the OLTP system.

Transactions support ACID semantics:

- Atomicity – each transaction is treated as a single unit, which success completely or fails completely. For example, a transaction that involved debiting funds from one account and crediting the same amount to another account must complete both actions. If either action can't be completed, then the other action must fail.
- Consistency – transactions can only take the data in the database from one valid state to another. To continue the debit and credit example above, the completed state of the transaction must reflect the transfer of funds from one account to the other.
- Isolation – concurrent transactions cannot interfere with one another and must result in a consistent database state. For example, while the transaction to transfer funds from one account to another is in-process, another transaction that checks the balance of these accounts must return consistent results - the balance-checking transaction can't retrieve a value for one account that reflects the balance *before* the transfer, and a value for the other account that reflects the balance *after* the transfer.
- Durability – when a transaction has been committed, it will remain committed. After the account transfer transaction has completed, the revised account balances are persisted so that even if the database system were to be switched off, the committed transaction would be reflected when it is switched on again.

# Analytical data workloads



- 1 Operational data is extracted, transformed, and loaded (ETL) into a *data lake* for analysis
- 2 Data is loaded into a schema of tables - typically in a Spark-based *data lakehouse* with tabular abstractions over files in the data lake, or a data warehouse with a fully relational SQL engine
- 3 Data in tables may be aggregated and loaded into an online analytical processing (OLAP) model, or cube
- 4 The files in the data lake, relational tables, and analytical model can be queried to produce *reports* and *dashboards*

© Copyright Microsoft Corporation. All rights reserved.

*Data lakes* are common in large-scale analytics scenarios, where a large volume of data in multiple formats must be collected and analyzed.

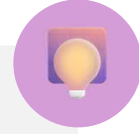
*Data warehouses* are an established way to store data in a relational schema that is optimized for read operations – primarily queries to support reporting and data visualization. *Data Lakehouses* are a more recent innovation that combine the flexible and scalable storage of a data lake with the relational querying semantics of a data warehouse. The table schema may require some *denormalization* of data in an OLTP data source (introducing some duplication to make queries perform faster)

Online Analytical Processing (OLAP) is an aggregated type of data storage that is optimized for analytical workloads. Data is imported aggregated so that aggregations of numeric *facts* (for example, sales revenue, or number of items sold) can be pre-calculated across *dimensions* (for example, product, date, or geographic location). The aggregation will typically occur at different levels allowing you to drill down or up, for example to find total sales by region, by city, or by an individual address. Because OLAP data is aggregated periodically, once the aggregation has been performed, queries which need the summaries that it contains are very fast.

Note: Different types of user might perform data analytical work at different stages of the overall architecture. For example:

- Data scientists might work directly with data files in a data lake to explore and model data.
- Data Analysts might query tables directly in the data warehouse to produce complex reports and visualizations.
- Business users might consume pre-aggregated data in an analytical model (cube) in the form of reports or dashboards.

# 1: Knowledge check

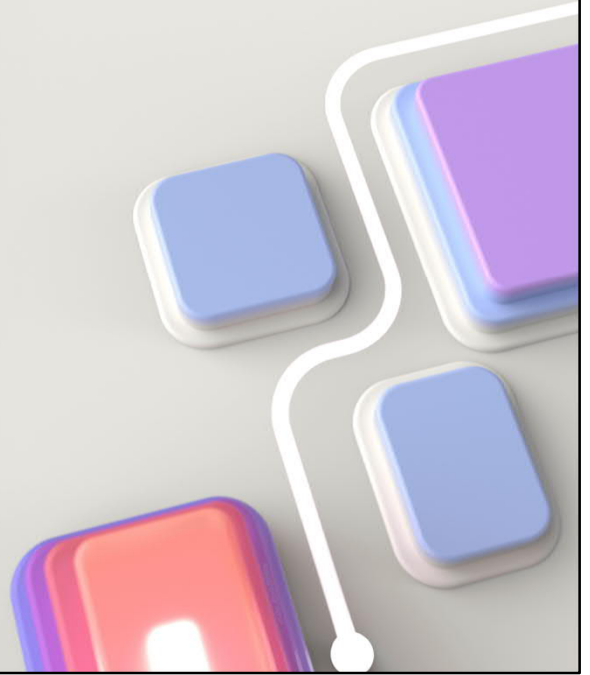


- 1 How is data in a relational table organized?
  - ☒ Rows and Columns
  - ☐ Header and Footer
  - ☐ Pages and Paragraphs
  
- 2 Which of the following is an example of unstructured data?
  - ☐ A comma-delimited text file with *EmployeeID*, *EmployeeName*, and *EmployeeDesignation* fields
  - ☒ Audio and Video files
  - ☐ A table within relational database
  
- 3 What is a data warehouse?
  - ☐ A non-relational database optimized for read and write operations
  - ☒ A relational database optimized for read operations
  - ☐ A storage location for unstructured data files

© Copyright Microsoft Corporation. All rights reserved.

## 2: Data roles and services

© Copyright Microsoft Corporation. All rights reserved.



# Data professional roles



## Database Administrator

- Database provisioning, configuration and management
- Database security and user access
- Database backups and resiliency
- Database performance monitoring and optimization



## Data Engineer

- Data integration pipelines and ETL processes
- Data cleansing and transformation
- Analytical data store schemas and data loads



## Data Analyst

- Analytical modeling
- Data reporting and summarization
- Data visualization

© Copyright Microsoft Corporation. All rights reserved.

Consider that in real-world scenarios, actual job roles might be a combination of these roles, or a subset of a single role, based on the size of the organization.

Note also that there are additional data-related roles not mentioned here, such as data scientist and data architect; and that there are other technical professionals that work with data, including application developers and software engineers. We're focusing on these three roles because they represent the core data-related operations in most organizations and reflect common job titles for data professionals.

# Microsoft cloud services for data

## Operational data workloads



### Azure SQL

- Family of SQL Server based relational database services



### Azure Database for open-source

- Maria DB, MySQL, PostgreSQL



### Azure Cosmos DB

- Highly scalable non-relational database system



### Azure Storage

- File, blob, and table storage
- Hierarchical namespace for data lake storage

## Analytical data workloads

### Software-as-a-Service (SaaS)



### Microsoft Fabric

Unified, SaaS based analytics platform based on open and governed lakehouse:

- Data ingestion and ETL
- Data Lakehouse
- Data Warehouse
- Data Science and ML
- Realtime Analytics
- Data visualization
- Data governance and management



### Microsoft Purview

Solution for enterprise-wide data governance and discoverability:

- Create a map of your data and track data lineage across multiple data sources.
- Enforce data governance across the enterprise and ensure the integrity of data.

### Platform-as-a-Service (PaaS)



### Azure Databricks

- Apache Spark analytics and data processing

others...

© Copyright Microsoft Corporation. All rights reserved.

*The slide shows some of the most commonly used service for working with data. The list is not exhaustive.*

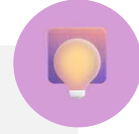
*We'll explore many of these later in the course.*

Azure SQL covers a family of relational database solutions based on the Microsoft SQL Server database engine. Options include:

- Azure SQL Database – a fully managed platform-as-a-service (PaaS) database hosted in Azure
- Azure SQL Managed Instance – a hosted instance of SQL Server, which allows more flexible configuration than Azure SQL DB but with more administrative responsibility for the owner.
- Azure SQL VM – a virtual machine with an installation of SQL Server, allowing maximum configurability with full management responsibility.

Note that some services are not easily categorized – for example, Microsoft Fabric includes some of the data pipeline processing capabilities of Azure Data Factory, a SQL Server based relational database engine that is optimized for data warehousing, and a Spark processing engine that offers similar functionality to Azure Databricks (Spark is an Apache open source technology for processing large volumes of data in parallel using programming languages like Scala and Python).

## 2: Knowledge check



- 1 Which one of the following tasks is the responsibility of a database administrator?
  - ☒ Backing up and restoring databases
  - ☐ Creating dashboards and reports
  - ☐ Creating pipelines to process data in a data lake
  
- 2 Which role is most likely to use Azure Data Factory to define a data pipeline for an ETL process?
  - ☐ Database Administrator
  - ☒ Data Engineer
  - ☐ Data Analyst
  
- 3 Which services would you use as a SaaS solution for data analytics?
  - ☐ Azure SQL Database
  - ☒ Microsoft Fabric
  - ☐ Azure Databricks

© Copyright Microsoft Corporation. All rights reserved.



© Copyright Microsoft Corporation. All rights reserved.