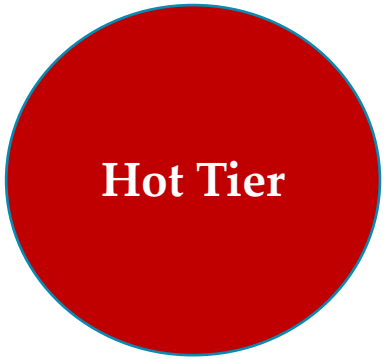
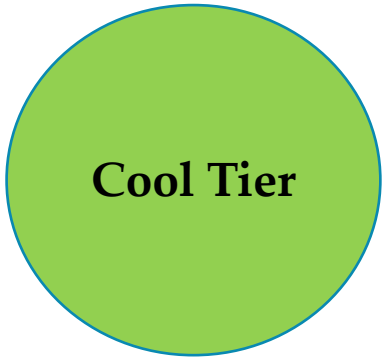
Managing Data Lifecycle

Blob Access Tiers



Highest storage cost
Lowest data access cost



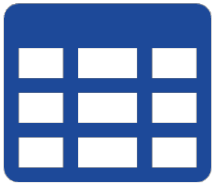
Lowest storage cost
Higher data access cost

Lowest storage cost
Highest data retrieval cost
Data is offline

Data Types

Four types of Data

Types of Data



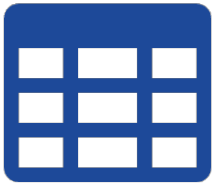
Structure Data

Semi-structured Data

Unstructured Data

Streaming Data

Types of Data



Structure Data

- Data That is organized. It has a strict defined schema which defines field names, data types, and the relationship between tables.
- Example – Database, Data Warehouse, ERP, CRM
- Schema-on-write
  - Highly precise schema that is defined on Write
- Difficult to make changes to the schema to accept new data changes
- Extract Transform Load (ETL)

Types of Data

Semi-structured Data

- Data That is NOT organized and does not conform to a formal structure of tables. But it does have structures such as tags or metadata associated with it. This allows records and fields within the data.
- Example – CSV, XML, JSON
- Easy to make changes in Schema
    - Schema is not strictly enforced
- Schema-on-read

Types of Data

Unstructured Data

- Data that does not have a pre-defined data model, and it is not organized in any particular manner that allows traditional analysis.
- Example – Videos, images, social media post, emails, music
- 90% of all new data is unstructured
- Does not have a schema or attributes within the data
- Highly flexible to accept new changes to the data
- Vast assortment of data types and growing everyday

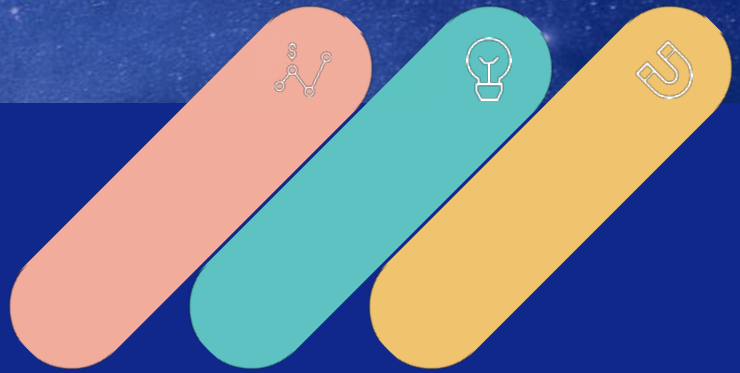
Types of Data

Streaming Data

- Data not at rest. Data that is continuous flow from one place to another place. This flow of the data provides an opportunity for immediate analysis or consumption.
- Example – Media, Satellite, IOT
- Streaming Data Analysis
  - Batch – After the stream is stored the data is analyzed to look for patterns and relationships
  - Real-time – The data is analyzed during gathering to make an immediate reaction to a trigger

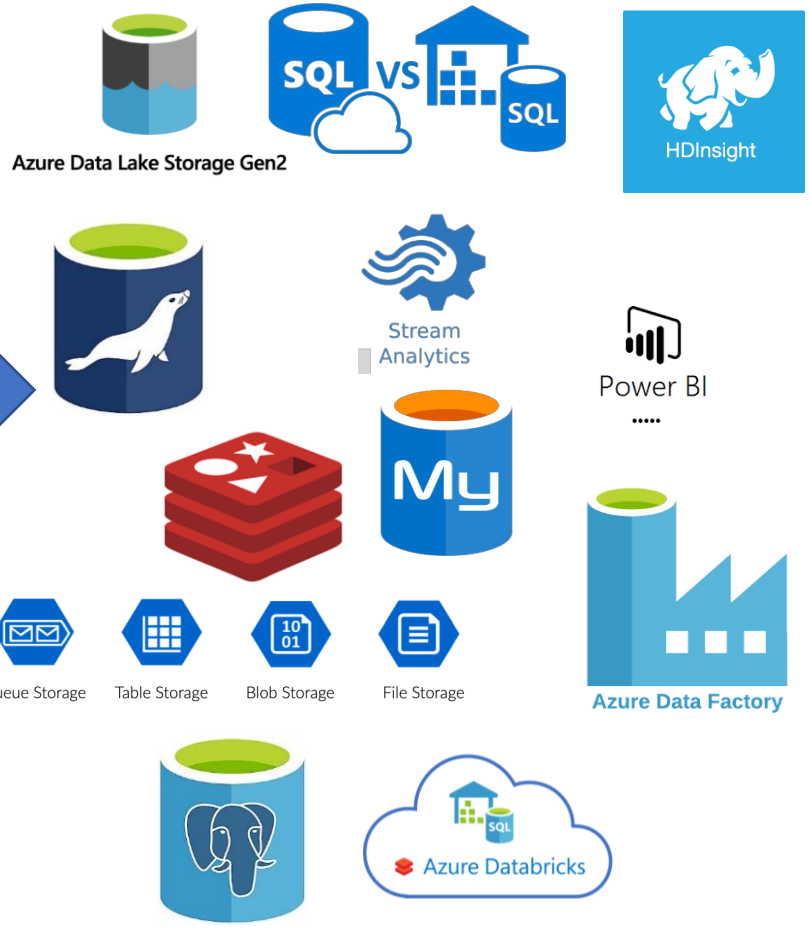
Data store

Understand data store models

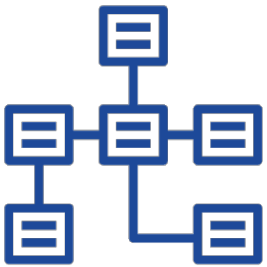


Why we need different data store?

- Store different types of data in different data stores
- Categorized by
  - Structure of data
  - Types of Operations on data.



Relational database management systems



Data Structure

- Organize data as a series of two-dimensional tables with rows and columns
- Schema-on-write
- Normalized
- Relationships are enforced using database constraints

Operation

- Structured Query Language (SQL)
- ACID (Atomic, **Consistent**, Isolated, Durable)

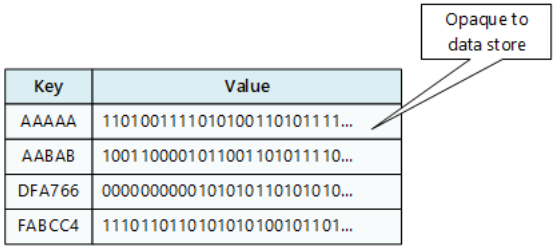
Examples

- Inventory management
- Order management
- Reporting database
- Accounting

Azure services

- Azure SQL Database
- Azure Database for MySQL
- Azure Database for PostgreSQL
- Azure Database for MariaDB

Key/value stores



Data Structure

- Each data value associated with a unique key
- Scalable
- No relationships between entities.

Operation

- Support simple insert/delete operation
- Highly optimized for applications performing simple lookups

Examples

- Data caching
- Session management
- User preference and profile management
- Product recommendation and ad serving

Azure services

- Azure Cosmos DB
- Azure Cache for Redis

Document databases

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" }

Data Structure

- Stores a collection of documents
- Document contains the data for single entity, such as a customer or an order.
- Documents are retrieves by unique keys
- Document data is semi-structured, meaning that data types of each field are not strictly defined.

Operation

- Individual documents are retrieved and written as a single block.
- Data requires index on multiple fields.

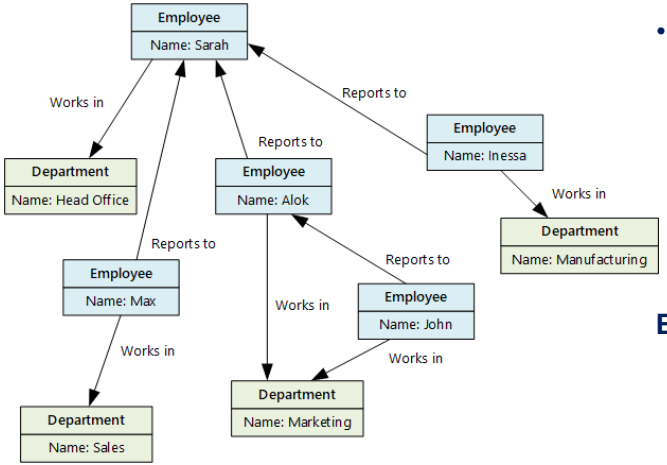
Examples

- Product catalog
- Content management
- Inventory management

Azure services

- Azure Cosmos DB

Graph databases



Data Structure

- Stores two types of information, nodes and edges.
- Nodes are similar to table rows or JSON documents
- Complex relationships between data items

Operation

- Efficiently perform queries across the network of nodes and edges and analyze the relationships between entities.
- Data requires index on multiple fields.

Examples

- Organization charts
- Social graphs
- Fraud detection
- Recommendation engines

Azure services

- Azure Cosmos DB Gremlin API
- SQL Server

Column-family databases

CustomerID	Column Family: Identity
001	**First name:** Mu Bae **Last name:** Min
002	**First name:** Francisco **Last name:** Vila Nova **Suffix:** Jr.
003	**First name:** Lena **Last name:** Adamcyz **Title:** Dr.

CustomerID	Column Family: Contact Info
001	**Phone number:** 555-0100 **Email:** someone@example.com
002	**Email:** vilanova@contoso.com
003	**Phone number:** 555-0120

Data Structure

- Organizes data into rows and columns
- Denormalized approach to structuring sparse data
- Each column family holds a set of columns that are logically related together and are typically retrieved or manipulated as a unit.

Examples

- Recommendations
- Personalization
- Sensor data
- Telemetry
- Messaging
- Social media analytics
- Web analytics

Operation

- Read and write operations for a row are usually atomic with a single column-family
- Update and delete operations are rare.

Azure services

- Azure Cosmos DB Cassandra API
- HBase in HDInsight

Data analytics

Data Structure

- Provide massively parallel solutions for ingesting, storing, and analyzing data.
- Data is distributed across multiple servers to maximize scalability.
- Usually denormalized in a "star" or "snowflake" schema
- Consisting of fact and dimension tables.

Operation

- Data analytics
- Enterprise BI

Examples

- Enterprise data warehouse

Azure services

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

Object storage

Data Structure

- Optimized for storing and retrieving large binary objects
- Stores can manage extremely large amounts of unstructured data.

Operation

- Identified by key.

Examples

- Images, videos, office documents, PDFs
- Static HTML, JSON, CSS
- Log and audit files
- Database backups

Azure services

- Azure Blob Storage
- Azure Data Lake Storage Gen2

Shared files

Data Structure

- Using file shares enables files to be accessed across a network.
- Requires SMB interface.
- Cross platform - Mount your Azure Files from Windows, Linux, or macOS.

Operation

- Accessible with standard I/O libraries.

Examples

- Legacy files
- Shared content accessible among a number of VMs or app instances

Azure services

- Azure Files

Time series databases

Data Structure

- Azure Time Series Insights is built to store, visualize, and query large amounts of time series data.

Examples

- Monitoring and event telemetry.
- Sensor or other IoT data.

Operation

- Records are generally appended sequentially in time order.
- Updates are rare.
- Deletes occur in bulk
- Data is read sequentially in either ascending or descending time order

Azure services

- Azure Time Series Insights

Search Engine Databases

Data Structure

- Data indexes from multiple sources and services.

Examples

- Product catalogs
- Site search
- Logging

Operation

- Searching can be exact or fuzzy.

Azure services

- Azure Search

General Consideration

Data Stores

Relational database

- Azure SQL Database
- Azure Database for MySQL
- Azure Database for PostgreSQL
- Azure Database for MariaDB

Data analytics

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

Key/value stores

- Azure Cosmos DB Table API
- Azure Cache for Redis

Document databases

- Azure Cosmos DB SQL API

Column-family databases

- Azure Cosmos DB Cassandra API
- HBase in HDInsight

Graph databases

- Azure Cosmos DB Gremlin API
- SQL Server

Shared files

- Azure Files

Object storage

- Azure Blob Storage
- Azure Data Lake Storage Gen2

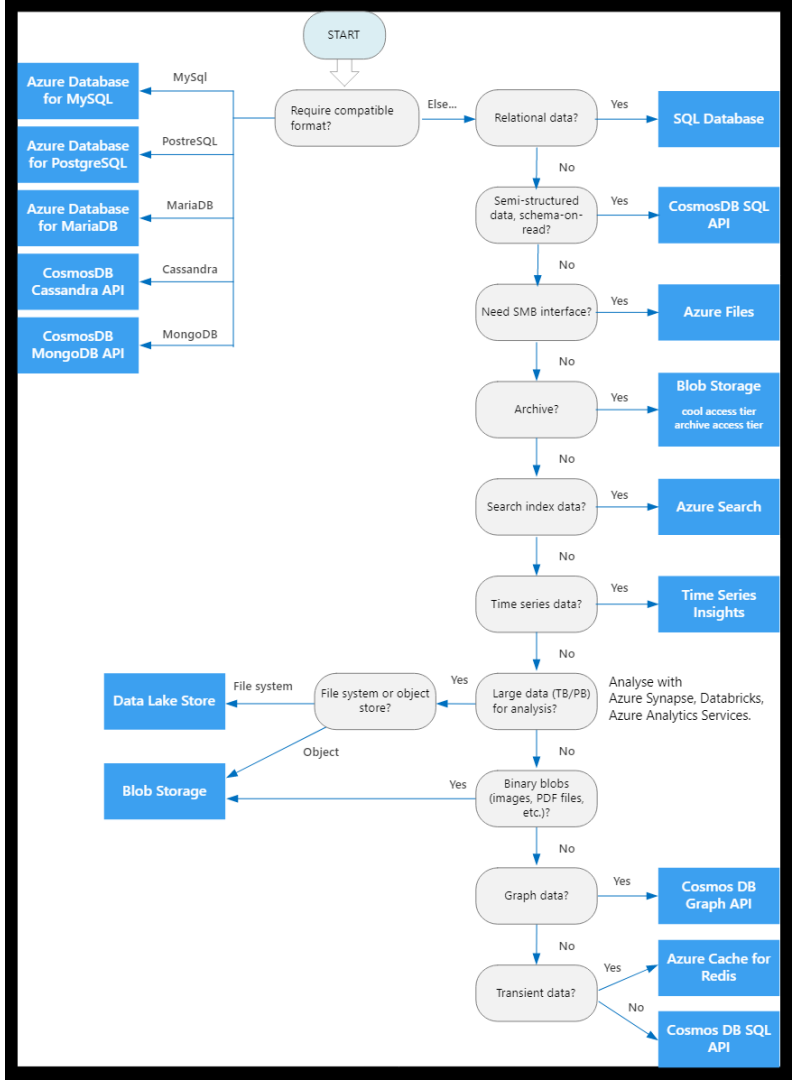
Search Engine Databases

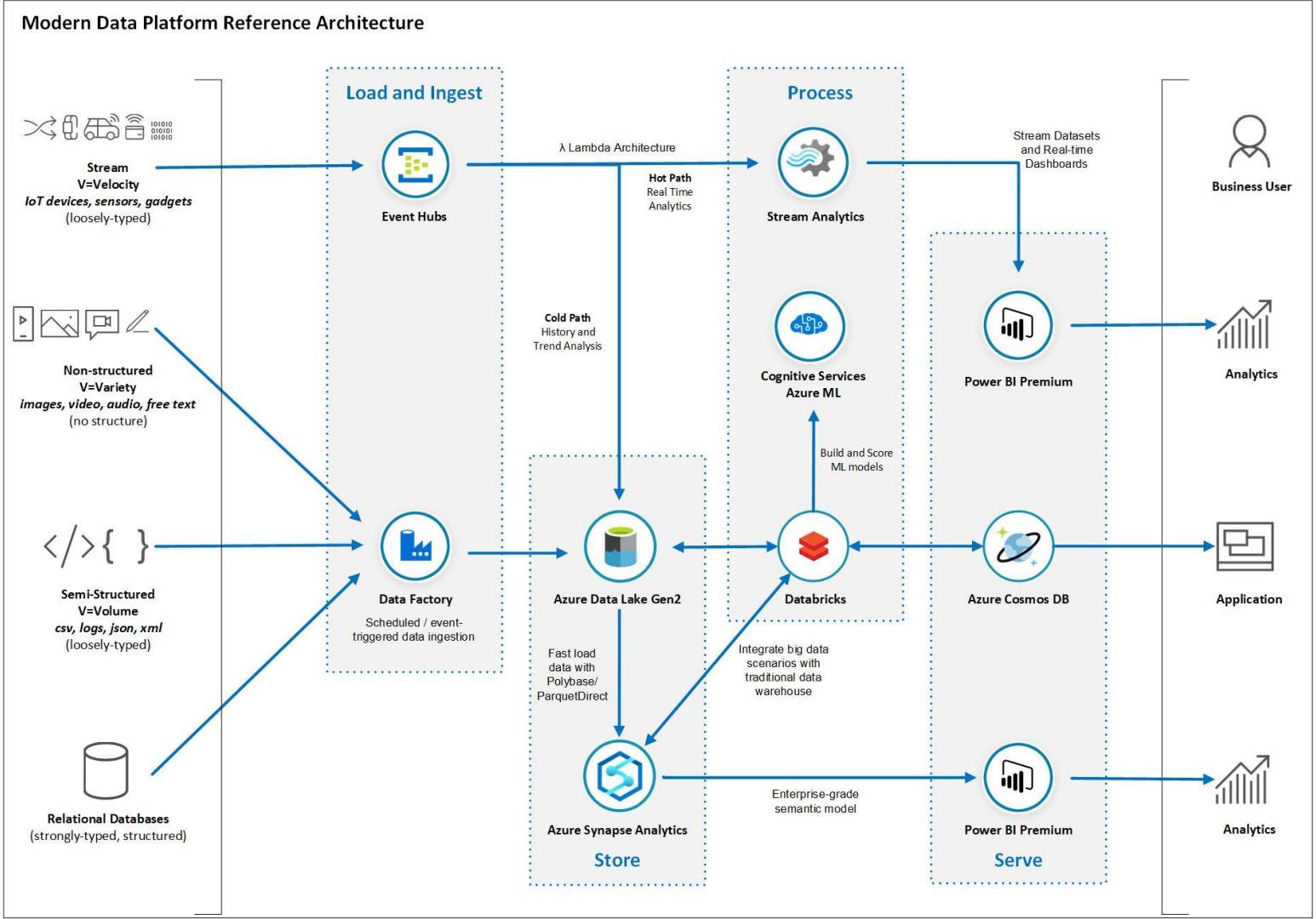
- Azure Search

Time series databases

- Azure Time Series Insights

HOW TO CHOOSE DATABASE




Modern Data Platform Reference Architecture
Load and Ingest
Process
Stream
V=Velocity
IoT devices, sensors, gadgets
(loosely-typed)
Event Hubs
λ Lambda Architecture
Hot Path
Real Time
Analytics
Stream Analytics
Stream Datasets
and Real-time
Dashboards
Business User
Cold Path
History and
Trend Analysis
Cognitive Services
Azure ML
Power BI Premium
Analytics
Non-structured
V=Variety
images, video, audio, free text
(no structure)
Build and Score
ML models
Semi-Structured
V=Volume
csv, logs, json, xml
(loosely-typed)
Data Factory
Scheduled / event-
triggered data ingestion
Azure Data Lake Gen2
Databricks
Azure Cosmos DB
Application
Fast load
data with
Polybase/
ParquetDirect
Integrate big data
scenarios with
traditional data
warehouse
Relational Databases
(strongly-typed, structured)
Azure Synapse Analytics
Enterprise-grade
semantic model
Power BI Premium
Analytics
Store
Serve

RTO and RPO

Recovery Time Objective (RTO) and Recovery Point Objective (RPO)

RTO and RPO

Recovery Time Objective (RTO)

The maximum acceptable time that an application can be unavailable after an incident.

If your RTO is 90 minutes, you must be able to restore the application to a running state within 90 minutes from the start of a disaster.

If you have a very low RTO, you might need a warm standby running to protect against a regional outage.

Recovery Point Objective (RPO)

The maximum duration of data loss that is acceptable during a disaster.

For example; one standalone database with hourly backups provides an RPO of 60 minutes.

If you require a lower RPO you'll need to design accordingly