IDUG

Db2 meets watsonx.data – A DBA Guide

George Baklarz, IBM

IBM

@IDUGDb2
#IDUG_NA24

Session Code: AIML2 | Platform: Db2 LUW

Agenda

- What exactly is watsonx.data and why do I care?
  - How did we get here?
  - Why open source is key
- Contrast and Compare watsonx.data to Db2 technology
  - Learn new acronyms to impress your friends
  - Looks like a database, queries like a database, but is it really a database?
- A visual tour of what watsonx.data
  - The UI always impresses
  - Db2 integration with watsonx.data
- Workloads that are suitable for watsonx.data
  - It isn't a one-sized solution that solves all problems!
  - DBAs and Developers need to determine the best platform based on their requirements

It's all about the Data*

* Seen in at least 27 previous IDUG presentations

There’s more data

Exploding data growth

The aggregate volume of data stored is set to grow over 250% in the next 5 years.

In more locations

Multiple locations, clouds, applications and silos

82% of enterprises are inhibited by data silos.

In more formats

Documents, images, video

80% of time is spent on data cleaning, integration and preparation.

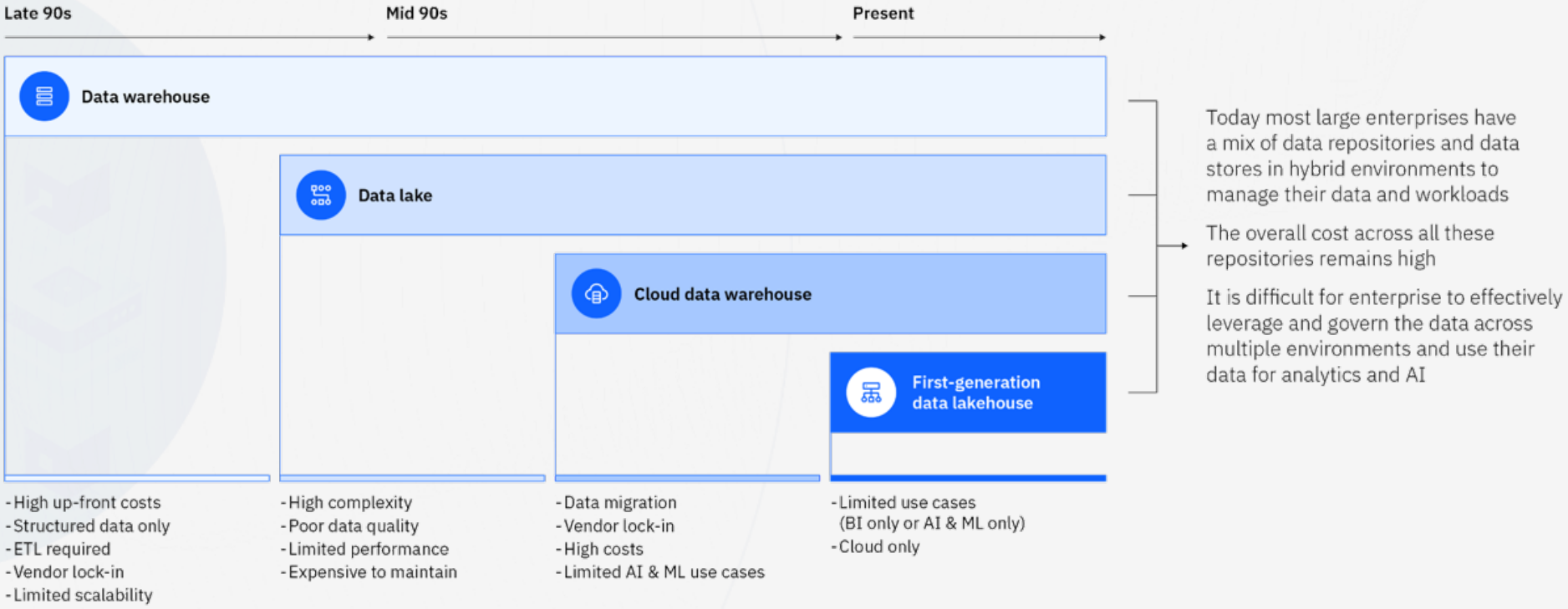
With less quality

Stale and inconsistent

82% of enterprises say data quality is a barrier on their data integration projects.

... Leading to more cost and complexities with governing data used for AI

The Data Journey



watsonx.data

Scale AI workloads, for all your data, anywhere

A fit-for-purpose data store, based on an open lakehouse architecture, supported by querying, governance and open data formats to access and share data

Access all your data through a single point of entry across all clouds and on-premises environments.

Get started in minutes with built-in governance, security and automation.

Reduce the cost of a data warehouse by up to 50%* through workload optimization across multiple query engines and storage tiers.

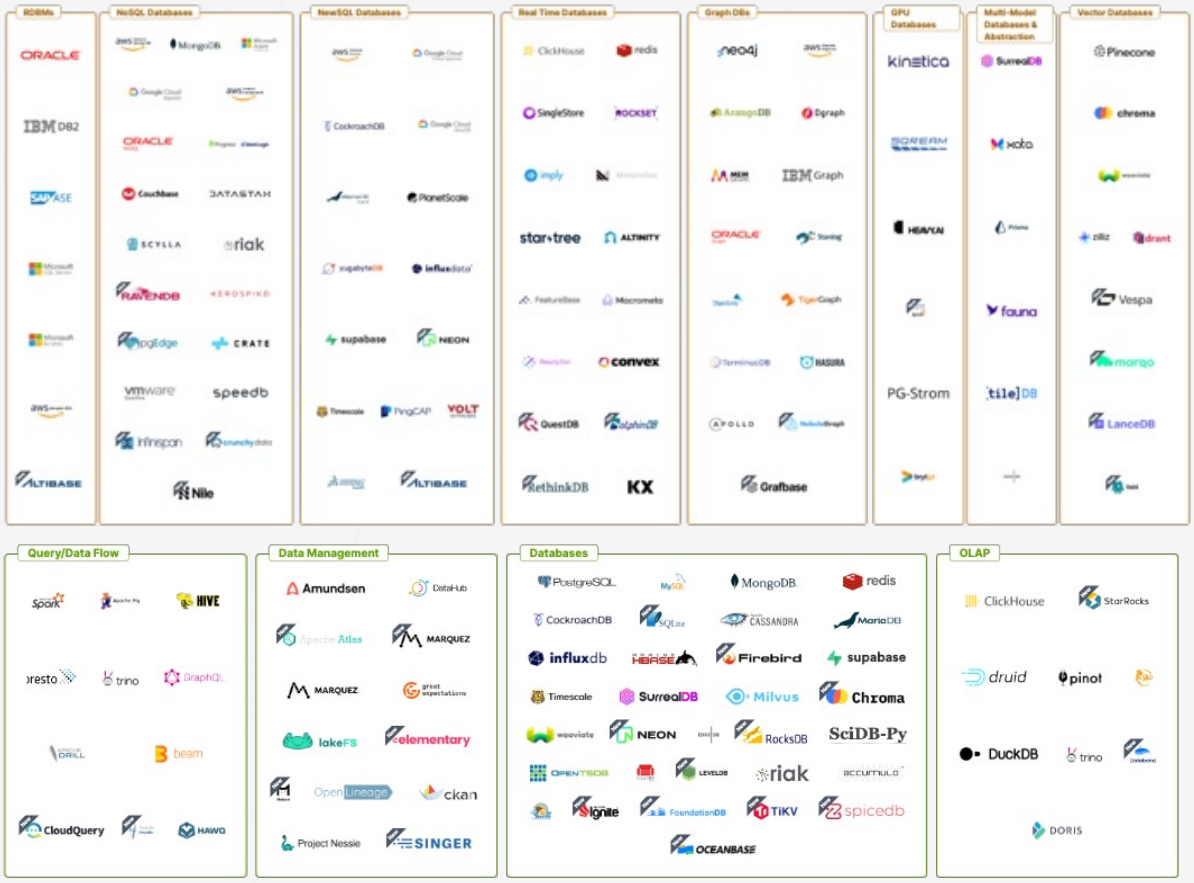
*When comparing published 2023 list prices normalized for VPC hours of IBM watsonx.data to several major cloud data warehouse vendors. Savings may vary depending on configurations, workloads and vendors.

Database Landscape

You are here →

OMG!

Another database to add to the landscape?



© 2024 Matt Turck (@mattturck), Aman Kabeer (@AmanKabeer11) & FirstMark (@firstmarkcap)

The Components

The IBM approach to a data lakehouse architecture combines the best of IBM with the best of open source

Object Storage

MINIO

Amazon S3

IBM Cloud Object Storage

ceph

File Formats

Parquet

CSV

Apache orc

{JSON}

AVRO

Table Management

ICEBERG

DELTA LAKE

Apache hudi

Metastore

HIVE

AWS Glue Data Catalog

Microsoft Purview Data Catalog

Databricks Unity Catalog

Engines

presto

APACHE Spark

Vector Store

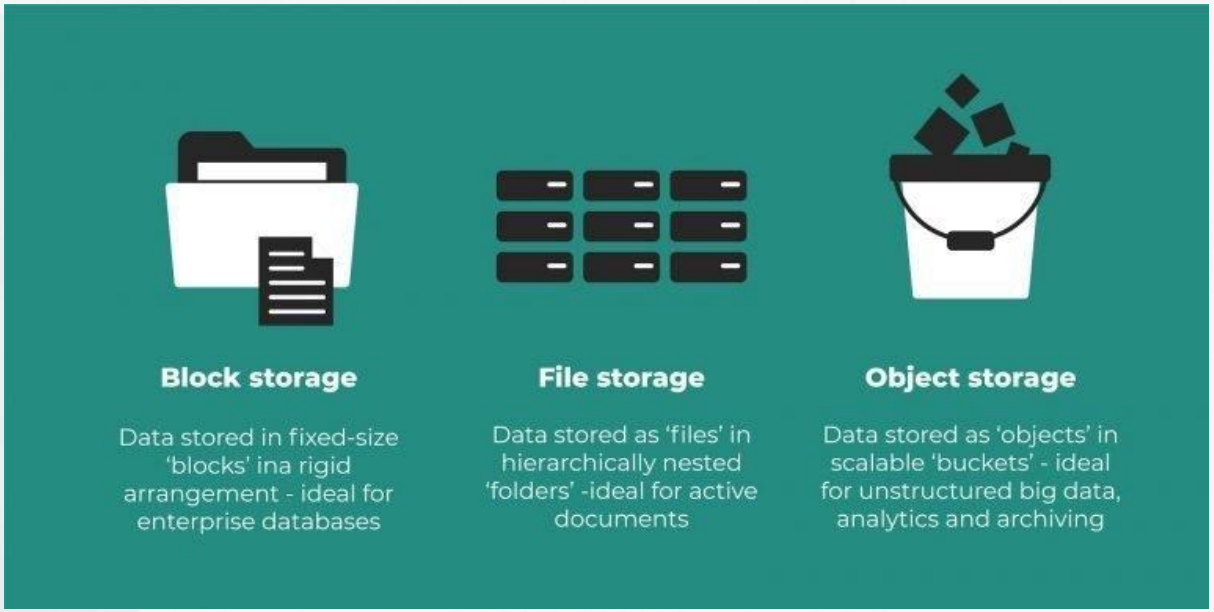
milvus

Object Storage

Why is Storage Relevant?

- Lakehouse is an emerging data management architecture that combines **open**, **flexible** and **low-cost** storage of data lakes with the transactional qualities and performance of a data warehouse
- This enables all your data (structured, semi-structured and unstructured) to reside in **commodity storage**, bringing together the best of data lakes and warehouses to enable best-in-class machine learning, business intelligence and artificial intelligence in one solution **without vendor lock-in**
- Desired storage properties for Lakehouse? (CHEAP, FAST, DURABLE, SCALABLE, EASY and anything and everything happening at once!)
  - Elastic, durable and efficient **hybrid cloud object storage** to store persistent data
  - High performance storage for working data sets
  - Data sharing across multiple engines
  - Data catalog for data management

What is Object Storage?



Object storage:

- Low cost
- Near unlimited scalability
- Extreme durability & reliability (99.999999999%)
- High throughput
- High latency (but can be compensated for)
- Basic units are objects, which are organized in buckets

- Most notable provider for object storage is Amazon S3 (Simple Storage Service)
- Other vendors offer S3-compatible object storage

The Rise of Cloud Object Storage

Cloud object storage technology is displacing HDFS as de facto storage technology for data lakes

	COS	HDFS	COS vs. HDFS
Elasticity	Yes (decoupled)	No	S3 is more elastic
Cost/TB/Month	$23	$206	10X
Performance	20MB/s/core	90MB/s/core	2x better price/perf
Availability	99.99%	99.9% (estimated)	10X
Durability	99.999999999%	99.9999% (estimated)	10X+
Transactional writes	Most technologies now provide strong consistency	Yes	Comparable

File Formats

Common data file formats

Computer systems and applications store data in files

Data can be stored in binary or text format

File formats can be open or closed (proprietary/lock-in)

Open formats (Parquet, ORC, and Avro) are commonly used in data lakes and lakehouses

CSV

- Human-readable text
- Each row corresponds to a single data record
- Each record consists of one or more fields, delimited by commas

{JSON}

- Human-readable text
- Open file and data interchange format
- Consists of attribute-value pairs and arrays
- JSON = JavaScript Object Notation

Parquet

- Open-source
- Binary columnar storage
- Designed for efficient data storage and fast retrieval
- Highly compressible
- Self-describing

Apache orc

- Open-source
- Binary columnar storage
- Designed and optimized for Hive data
- Self-describing
- Similar in concept to Parquet

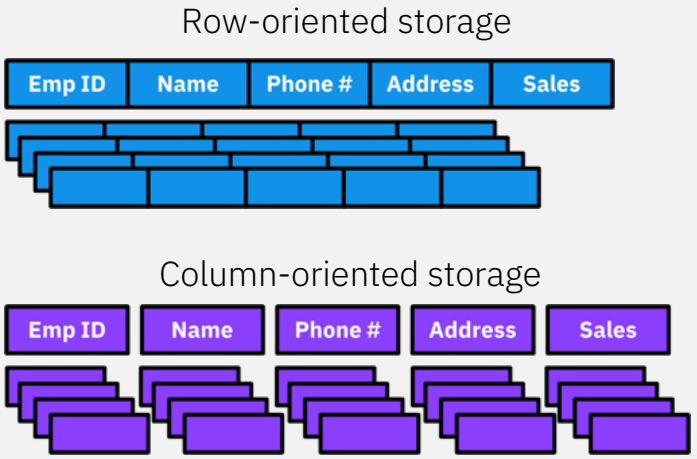
AVRO

- Open-source
- Row-oriented data format and serialization framework
- Robust support for schema evolution
- Mix of text/binary

Apache Parquet

Parquet is designed to support fast data processing for complex data

- Open-source
- Columnar storage
- Highly compressible with configurable compression options and extendable encoding schemas by data type
- Self-describing: schema and structure metadata is included
- Schema evolution with support for automatic schema merging

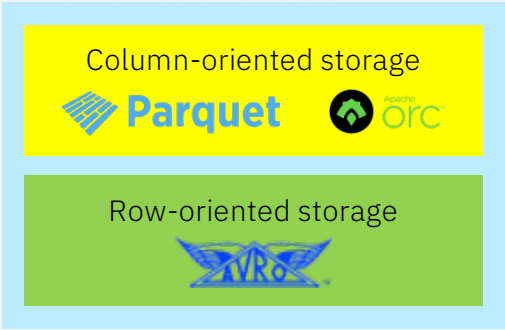


Why do these things matter in a lakehouse?

- Performance of queries directly impacted by size and number of file(s) being read
- Ability to read/write data to an open format from multiple runtime engines enables collaboration
- Size of data stored, amount of data scanned, and amount of data transported affect the charges incurred in using a lakehouse (depending on the pricing model)

Apache ORC

- Open-source, columnar storage format
  - Similar in concept to Parquet, but different design
  - Parquet considered to be more widely used than ORC
- Highly compressible, with multiple compression options
  - Considered to have higher compression rates than Parquet
- Self-describing and type-aware
- Support for schema evolution
- Built-in indexes to enable skipping of data not relevant to a query
- Excellent performance for read-heavy workloads
  - ORC generally better for workloads involving frequent updates or appends
  - Parquet generally better for write-once, read-many analytics



Apache Avro

- Open-source, row-based storage and serialization format
  - Can be used for file storage or message passing
- Beneficial for write-intensive workloads
- Format contains a mix of text and binary
  - Data definition: Text-based JSON
  - Data blocks: Binary
- Robust support for schema evolution
  - Handles missing/added/changed fields
- Language-neutral data serialization
  - APIs included for Java, Python, Ruby, C, C++, and more

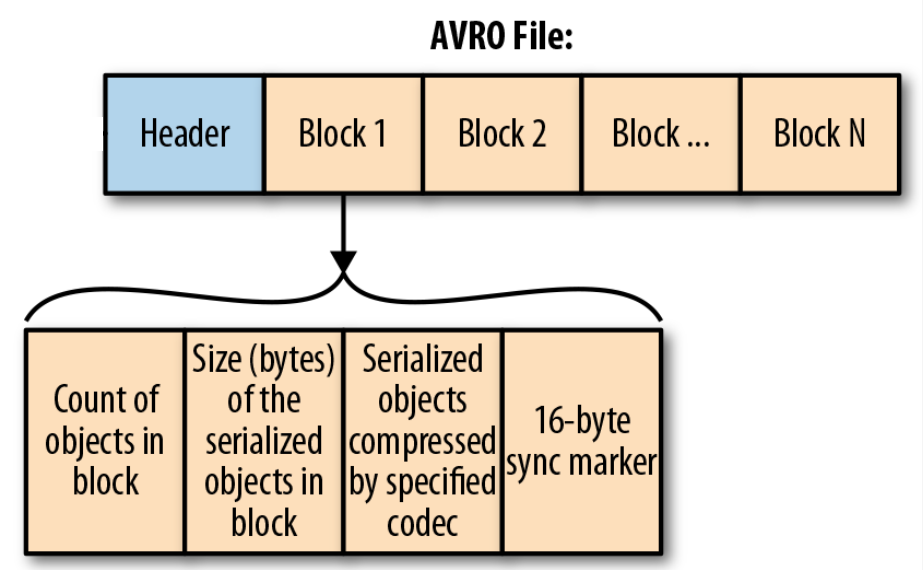


Table Management

Table management

Sits “above” the data file layer

Organizes and manages table metadata and data

Typically supports multiple underlying disk file formats (Parquet, Avro, ORC, etc.)

May offer transactional concurrency, I/U/D, indexing, time-based queries, and other capabilities

ICEBERG

- Open-source
- Designed for large, petabyte (PB)-scale tables
- ACID-compliant transaction support
- Capabilities not traditionally available with other table formats, including schema evolution, partition evolution, and table version rollback – all without re-writing data
- Advanced data filtering
- Time-travel queries let you see data at points in the past

Apache hudi

- Open-source
- Manages the storage of large datasets on HDFS and cloud object storage
- Includes support for tables, ACID transactions, upserts/ deletes, advanced indexes, streaming ingestion services, concurrency, data clustering, and asynchronous compaction
- Multiple query options: snapshot, incremental, and read-optimized

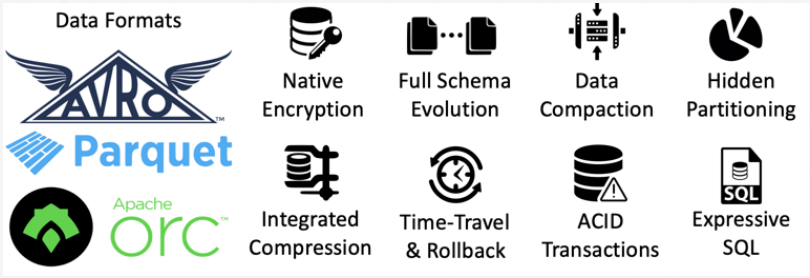
DELTA LAKE

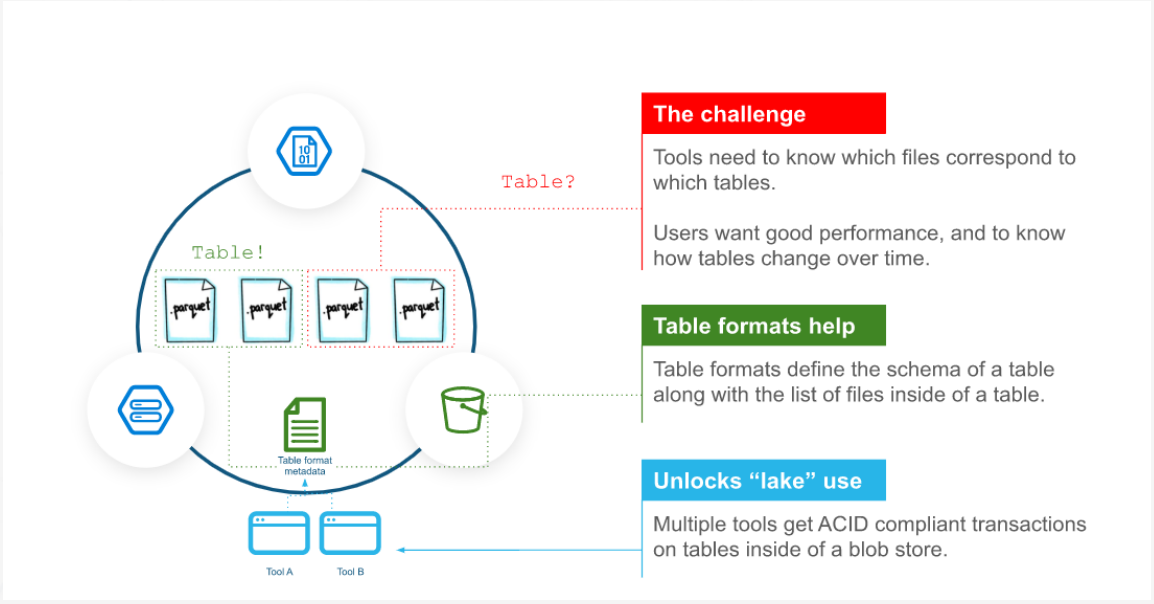
- Open-source, but Databricks is primary contributor and user, and controls all commits to the project – so “closed”
- Foundation for storing data in the Databricks Lakehouse Platform
- Extends Parquet data files with a file-based transaction log for ACID transactions and scalable metadata handling
- Capabilities include indexing, data skipping, compression, caching, and time-travel queries
- Designed to handle batch as well as streaming data

What is Apache Iceberg?

ICEBERG

- High-performance format for huge analytic tables
- Brings the simplicity of SQL to big data and data lakes
- Fully open source and accessible
- Rapidly becoming the industry standard



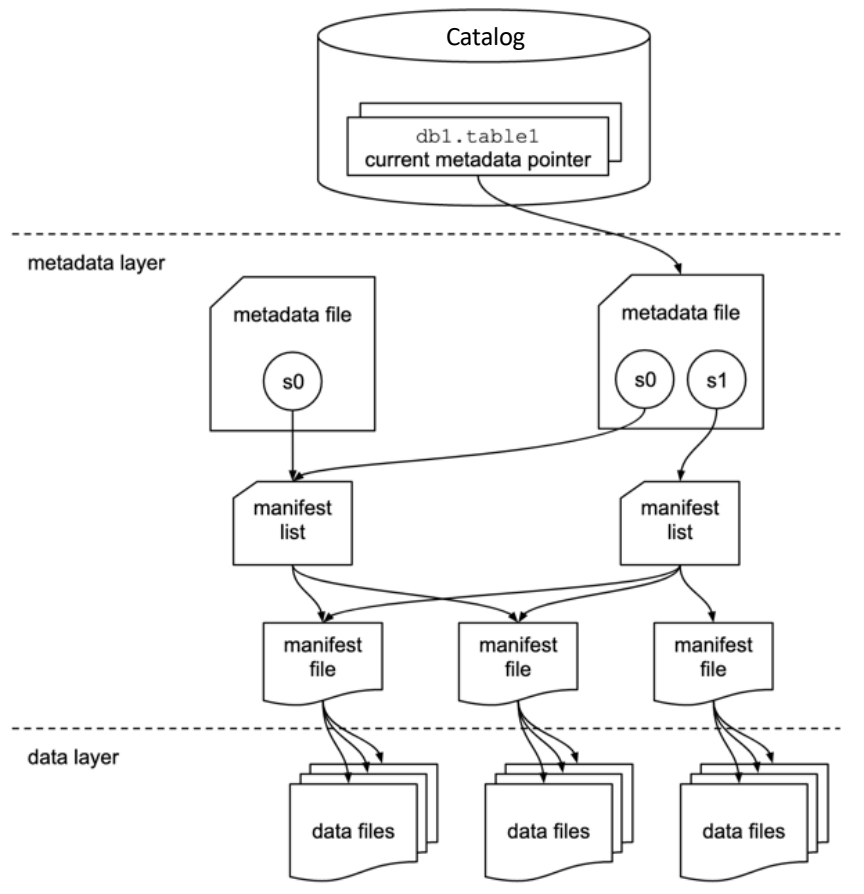


Why Apache Iceberg for data lakehouses?

Open-source data table format that helps simplify data processing on large dataset stored in data lakes

Developers love it because it has:

- SQL — Use it to build the data lake and perform most operations without learning a new language
- Data Consistency — ACID compliance (not just append data operations to tables)
- Schema Evolution — Add/remove columns without distributing underlying table structure
- Data Versioning — Time travel support that lets you analyze data changes between update and deletes
- Cross Platform Support — Supports variety of storage systems and query engines (Spark, Presto, Hive, +++)



Metastore

What is a metastore?

- Manages metadata for the tables in the lakehouse, including:
  - Schema information (column names, types)
  - Location and type of data files
- Similar in principle to the system catalogs of a relational database
- Shared metastore ensures query engines see schema and data consistently
- May be a built-in component of a larger integration/governance solution

HMS used by watsonx.data

- Hive metastore (HMS) is a component of Hive, but can run standalone
- Open-source
- Manage tables on HDFS and cloud object storage
- Pervasive use in industry

AWS Glue Data Catalog

- Component of AWS Glue integration service
- Inventories data assets of AWS data sources
- Includes location, schema, and runtime metrics

Microsoft Purview Data Catalog

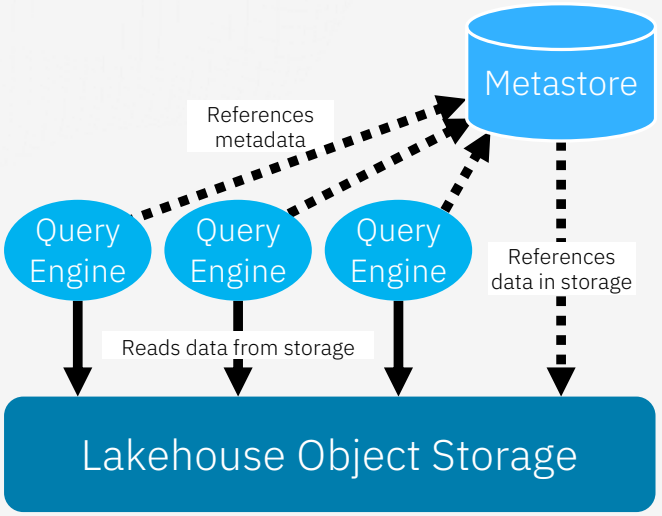
- Component of Microsoft Purview data governance solution
- Helps manage on-premises, multicloud, and SaaS data
- Offers discovery, classification, and lineage

Databricks Unity Catalog

- Provides centralized access control, auditing, lineage, and data discovery across a Databricks lakehouse
- Contains data and AI assets including files, tables, machine learning models, and dashboards

Hive Metastore (HMS)

- Open-source **Apache Hive** was built to provide an SQL-like query interface for data stored in Hadoop
- **Hive Metastore (HMS)** is a component of Hive that stores metadata for tables, including schema and location
- HMS can be deployed standalone, without the rest of Hive (often needed for lakehouses, like watsonx.data)
- Query engines use the metadata in HMS to optimize query execution plans
- The metadata is stored in a traditional relational database (PostgreSQL in the case of watsonx.data)
- In watsonx.data, IBM Knowledge Catalog integrates with HMS to provide policy-based access and governance



Engines

Presto

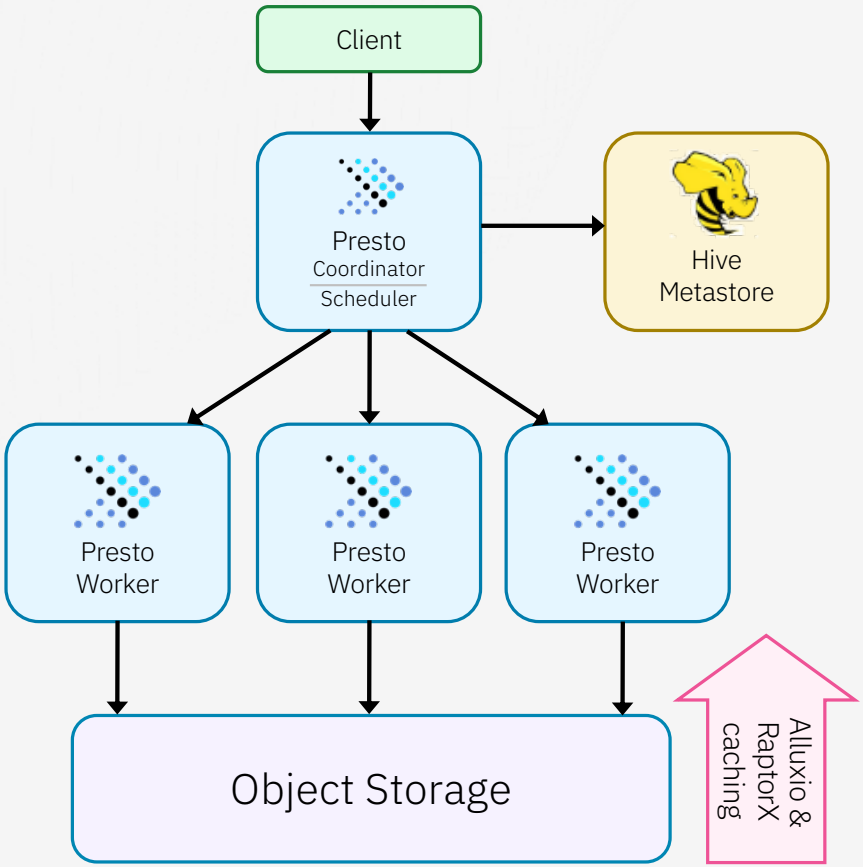
- Presto is an open-source distributed SQL engine suitable for querying large amounts of data
- Supports both relational and non-relational sources
- Easy to use with data analytics and business intelligence tools
- Supports both interactive and batch workloads
- In watsonx.data, spin up one or more Presto compute engines of various sizes – cost effective, in that engines are ephemeral and can be spun up and shut down as needed

- Presto connectors allow access to data in-place, allowing for **no-copy data access and federated querying**
- Consumers are abstracted from the physical location of data
- A wide variety of data sources are supported, including:

Presto Architecture

The structure of Presto is similar to that of classical MPP database management systems.

- **Client:** Issues user query and receives final result.
- **Coordinator:** Parses statement, plans query execution, and manages worker nodes. Gets results from workers and returns final result to client.
- **Workers:** Execute tasks and process data.
- **Connectors:** Integrate Presto with external data sources like object stores, relational databases, or Hive.
- **Caching:** Accelerated query execution through metadata and data caching (provided by Alluxio and RaptorX).



Powered by presto

Adobe

Digital advertising platform

Over 2000 daily reports and 100s of pipelines on a 7 PB data lake with over 400 billion records

Bolt

Ride-hailing, micromobility rentals, and food delivery in Europe and Africa

Up to 100,000 daily queries (over 1.5 million queries per month) with over 2000 active internal users on 2 PB data lake

Meta

Social media

30,000 queries per day with 1000 daily active users on a 300 PB data lake

Uber

Ride-hailing, food delivery

Over 100 million queries per day with 7000 weekly active users on a 50 PB data lake

ByteDance

Internet technology

Over 2 million queries per day for business intelligence and one-off use cases

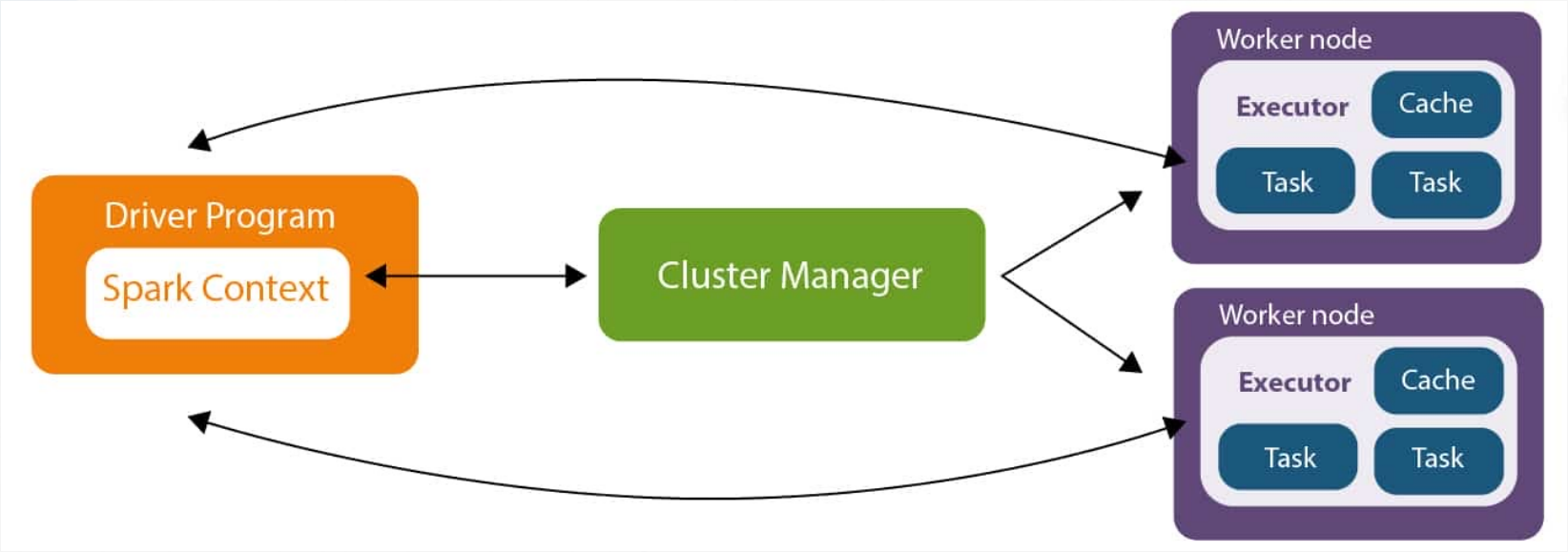
twilio

Communications API technology

Over 2700 active internal users running 1 million queries scanning 40 PB of data per month

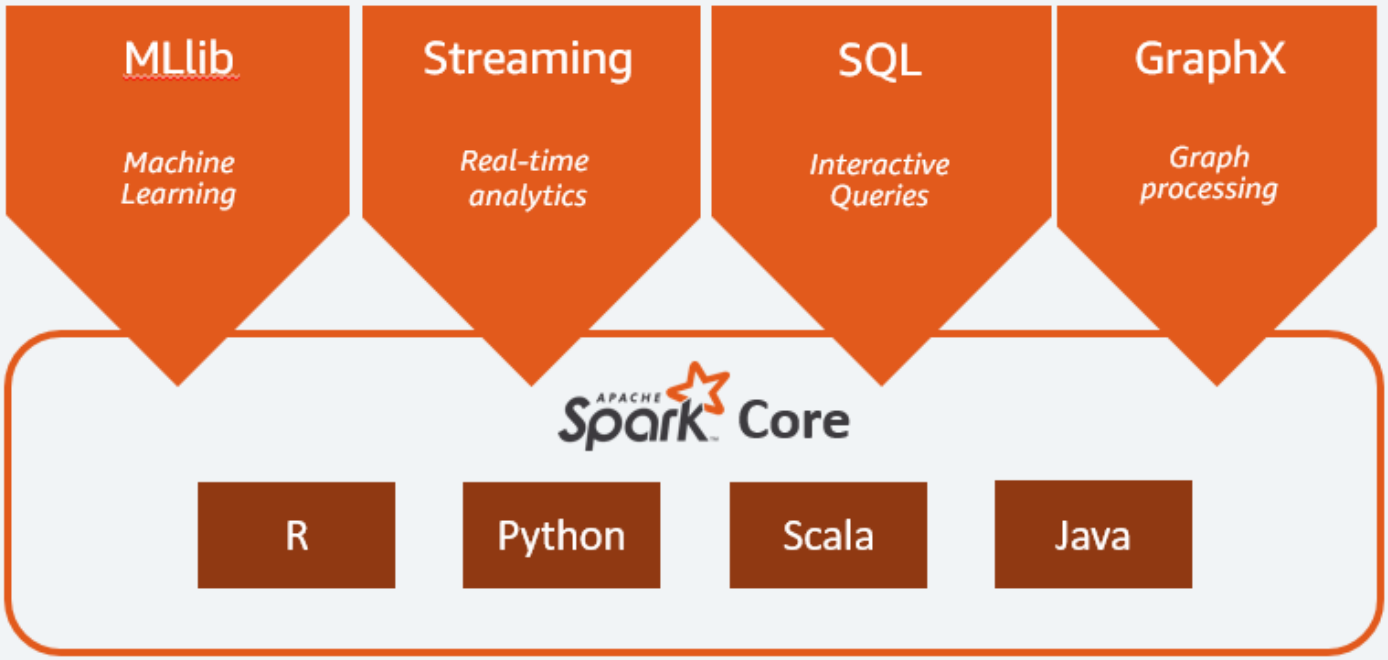
Apache Spark

- Apache Spark is an open-source data-processing engine for large data sets
  - It is designed to deliver the computational speed, scalability, and programmability required for *big data*, specifically for streaming data, graph data, ML, and AI applications.
- The basic Apache Spark architecture diagram:



Apache Spark and Machine Learning

- Spark has libraries that extend the capabilities to ML, AI, and stream processing.
- Apache Spark MLlib
- Spark Streaming
- Spark SQL
- Spark GraphX



SQL Support

SQL Support

SQL ISO IEC 9075

Presto follows standard SQL (ISO/IEC 9075). It doesn't mean that every feature has been implemented!

No RI Support

No referential integrity, triggers, generated values or features that are generally associated with transactional systems.

Data Types

Additional types:

- TINYINT
- ARRAY, MAP, ROW
- JSON
- UUID
- Others

Missing:

- NUMERIC
- BLOB: VARBINARY
- CLOB: VARCHAR

Functions

Large library of built-in functions are available.

Can create your own functions using SQL.

Watsonx.data UI

Use Cases and Interoperability

Use cases

Deploy AI/ML at scale

Build, train, tune, deploy, and monitor trusted AI and ML models for mission-critical workloads with data in IBM watsonx.data; strengthen compliance with lineage and reproducibility of data used for AI.

Apply real-time analytics and BI

Combine data from existing sources with new data in watsonx.data to unlock new, faster insights without the cost and complexity of duplicating and moving data across different environments.

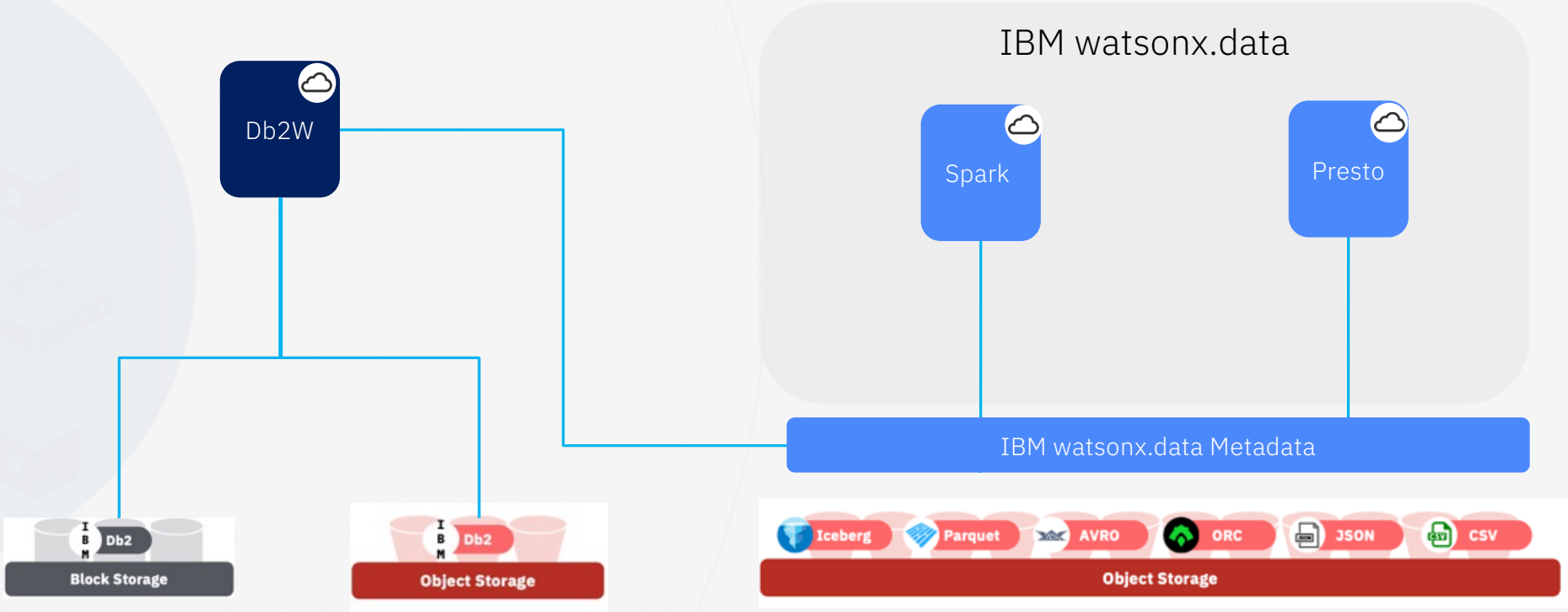
Streamline data engineering

Reduce data pipelines, simplify data transformation, and enrich data for consumption using SQL, Python or an AI-infused conversational interface.

Share data responsibly

Enable self-service access for more users to more data while you strengthen security and compliance with centralized governance and local automated policy enforcement.

Db2 LUW Interoperability with watsonx.data



Create a Db2 Table Accessible by watsonx.data

```
CREATE DATALAKE TABLE hiveschema.db2exported
  (
  id int,
  name varchar(32)
  )
STORED AS PARQUET
LOCATION 'DB2REMOTE://hive-bucket//hiveschema/db2exported'
TBLPROPERTIES('bigsql.external.catalog'='watsonxdata')

CREATE DATALAKE TABLE iceberg.db2exported(id INT, name VARCHAR(32))
 (
  id int,
  name varchar(32)
  )
STORED AS PARQUET
STORED BY ICEBERG
LOCATION 'DB2REMOTE://iceberg-bucket//iceberg/db2exported'
TBLPROPERTIES('iceberg.catalog'='watsonxdata')
```

- The table is created in both the Db2 & watsonx.data catalog and data is shared
- The value of the property is the name used to register the metastore when setting up the connection.

Provide Db2 access to watsonx.data Tables

```
CALL EXTERNAL_CATALOG_SYNC(
  'metastore-name',
  'schema-name',
  'table-name',
  'exist-action',
  'error-action',
  ‘options')
```

- Brings the table definitions into the Db2 catalog
- The data is shared between the 2 systems
- Need to re-sync the schema if the table changes
- Multiple tables & schemas can be specified using regular expressions
- The metastore-name is the name used to register the metastore when setting up the connection.

IBM Data Gate for watsonx.data

First-class integration of IBM Z data into watsonx.data

Modernize access to mainframe data for analytics and AI by making IBM Z data readily available to watsonx.data

Provides easy and efficient access to data for the lakehouse, with lower overhead, latency and higher throughput compared to most other replication options

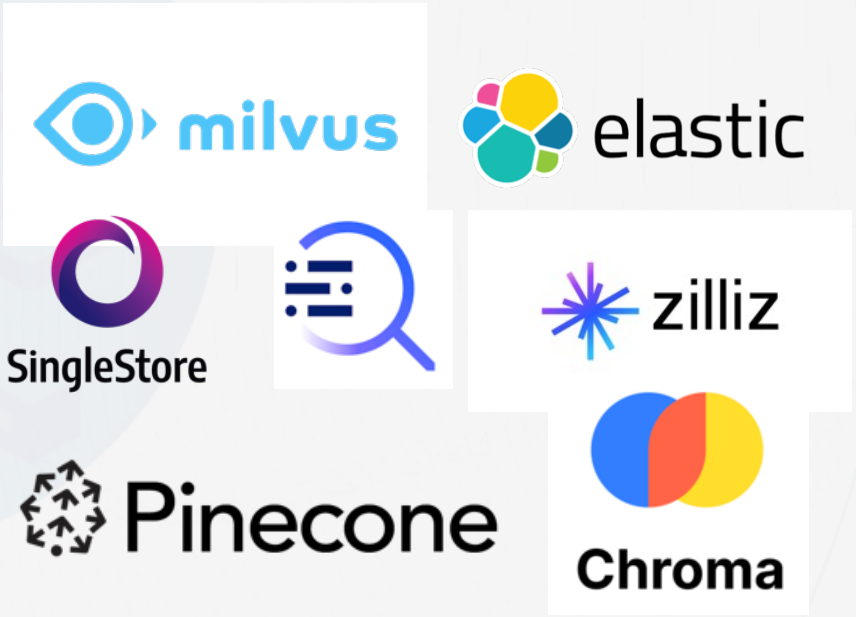
- Seamlessly synchronize Db2 for z/OS, IMS DB, and VSAM to Iceberg in watsonx.data
- Use the most up-to-data transactional data for analytics and AI
- Reduce the time it takes to access, analyze and score IBM Z data

Vector Database

What are Vector databases?

It is a database designed to store, manage, and search vector data, allowing for highly efficient and fast similarity searches. For example, in image or audio search, data with similar characteristics to a vector can be retrieved quickly.

Vector Databases on the Market



What's the difference between a Vector DB vs Regular DB?

A vector database is a collection of data stored as mathematical representations. Vector databases make it easier for machine learning models to remember previous inputs, allowing machine learning to be used to power search, recommendations, and text generation use-cases.

Vector Database Use Case Examples

You can use a vector database for Generative AI capabilities for use cases such as similarity search.

Main Use Cases	Description
Image/Video similarity search	• Prevention of copyright violations when sending outside the company • Search for similar images of diagrams/photos in external documents • Efficient collection of similar images for creating AI models
Personalized service and proposal/recommendation systems	• Recommends information and products based on the user's behavior, preferences, profile, and needs. • Recommendations can be generated by finding vectors that most closely match the user's interests.
Question and answer system (natural language processing)	Vector search is essential to the use of Natural Language Processing and is used in multiple applications, including chatbots. It allows users to interact directly with the system by asking questions that are vectorized and matched to the closest match. By representing data points as vectors, vector search provides a better semantic understanding of the query and more accurate results for each user.

Vector Embedding Concepts

A vector embedding is the internal representation of input data in a deep learning model. They are a way to convert words and sentences and other data into numbers that capture their meaning and relationships.

Dimensions

The dimensionality of a vector embedding is equivalent to the length of a vector.

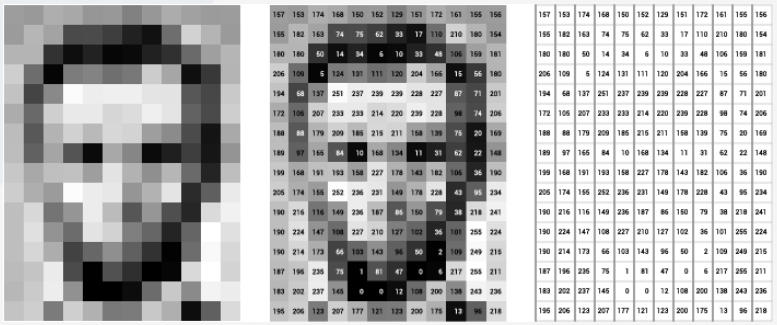
Segment Size

To easily process data, Milvus segments the data into smaller segmented data files.
The segment size is determined by the index file size of the embeddings.

Index Parameters

Setting key parameters with Milvus for operations (e.g., creating a table).
The number of parameters can affect search performance.

Types of Embeddings
- **Images**
- **Text**
- **Audios**
- **Videos**
- **Multimodal data**

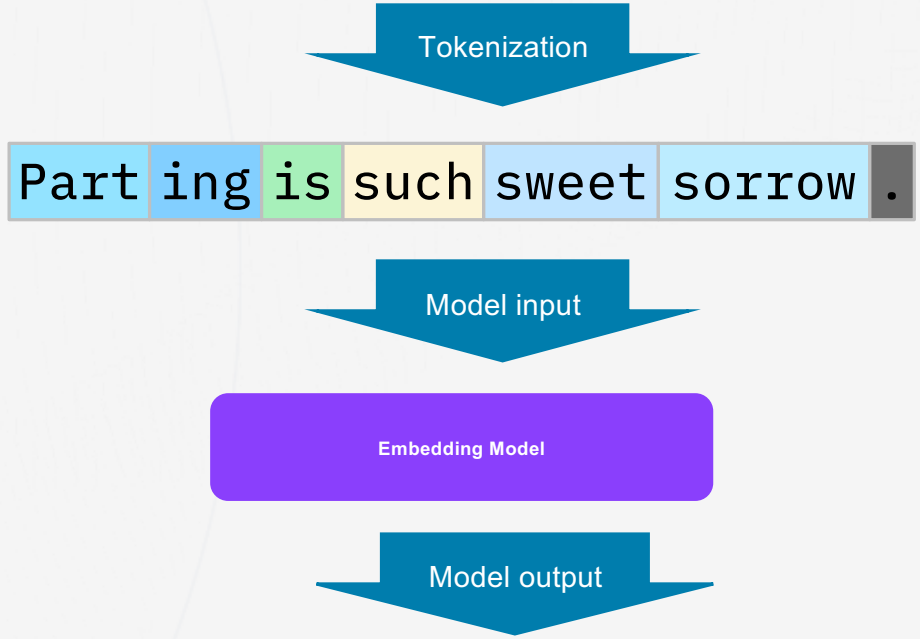


[157, 153, 174, ⋯, 161, 155, 156]

Dimensions

Vector embeddings

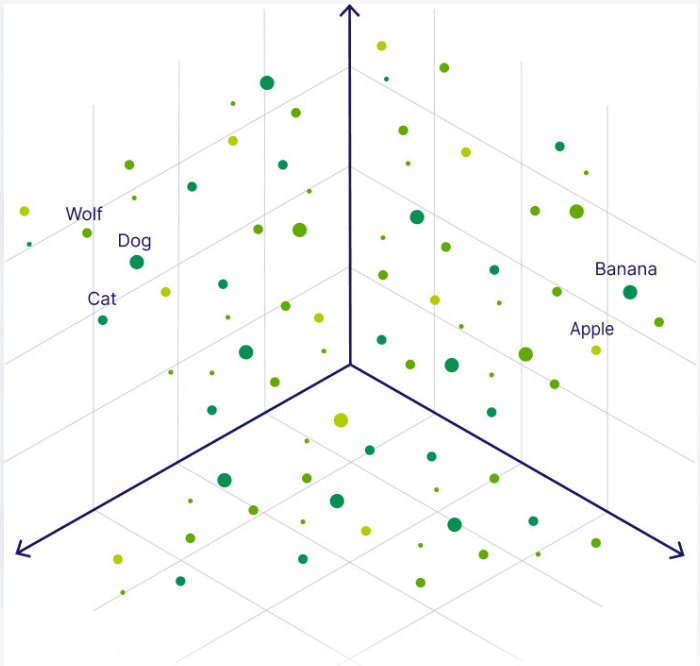
Input text: "Parting is such sweet sorrow."



Vector embedding: `[-0.028, 0.559, 0.142, -0.398, ... , 0.401]`

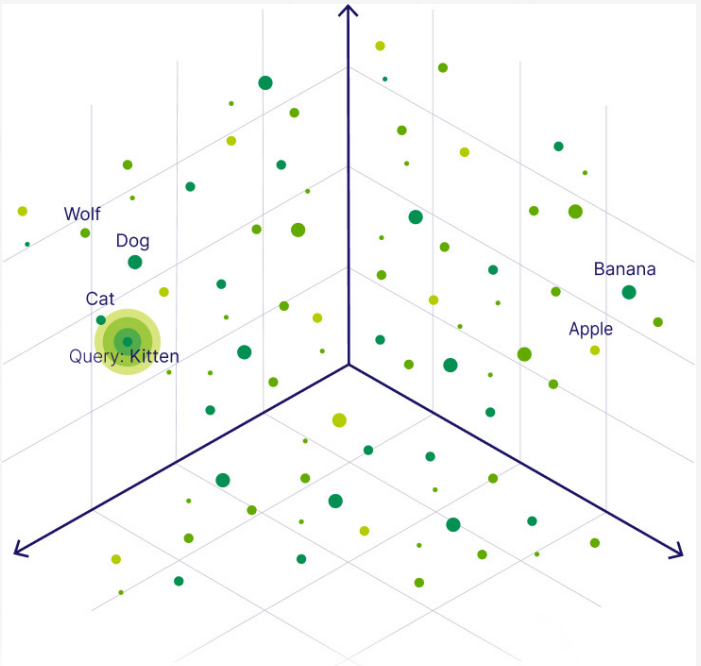
Similarity search

Example Word Vectors



What is "Kitten" similar to?

Finding Similarities

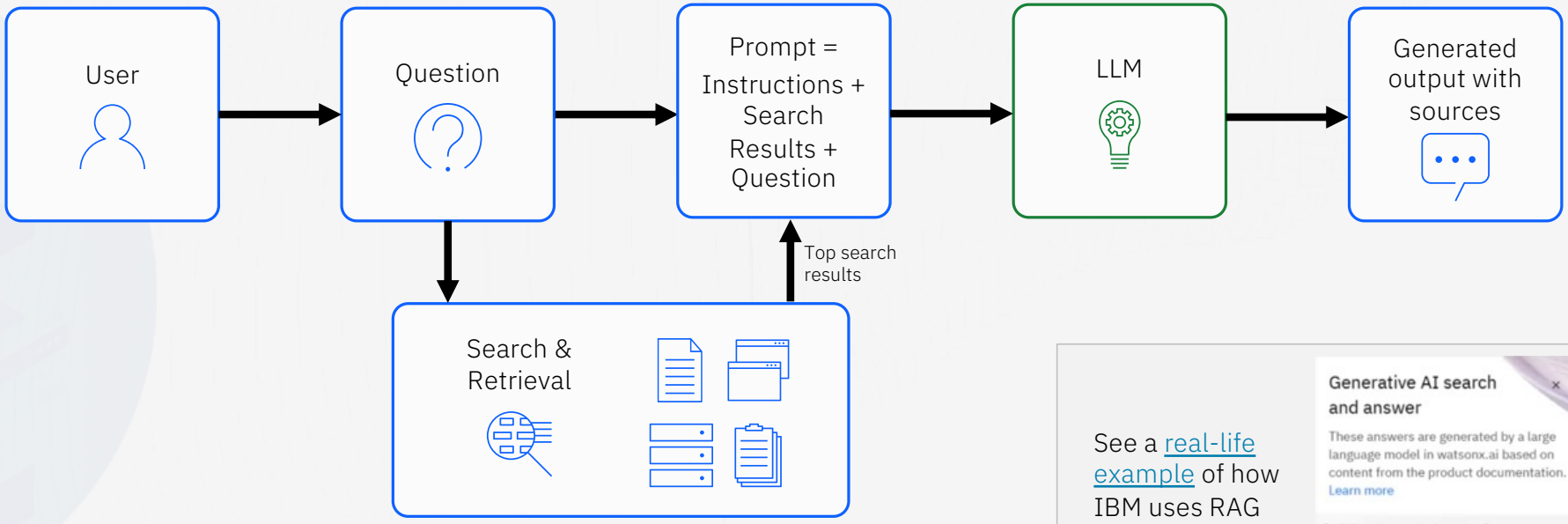


Common similarity measures: Cosine similarity, Euclidean distance, dot product

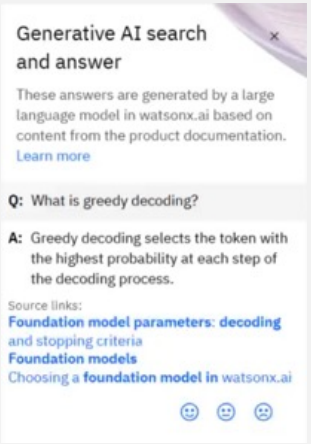
Source: https://weaviate.io/blog/what-is-a-vector-database

What is Retrieval Augmented Generation (RAG)?

RAG allows you to augment the capabilities of your LLM with your own data across many use cases like recommended search, creating question and answer based chatbots and more



See a real-life example of how IBM uses RAG to answer user questions about watsonx.ai in the product documentation



Advantages of Retrieval Augmented Generation (RAG)

RAG is a form of architecture that uses search results to enhance the generative model. It can be used as reference data for generating external data to generate reliable answers.

RAG configurations can improve reliability:

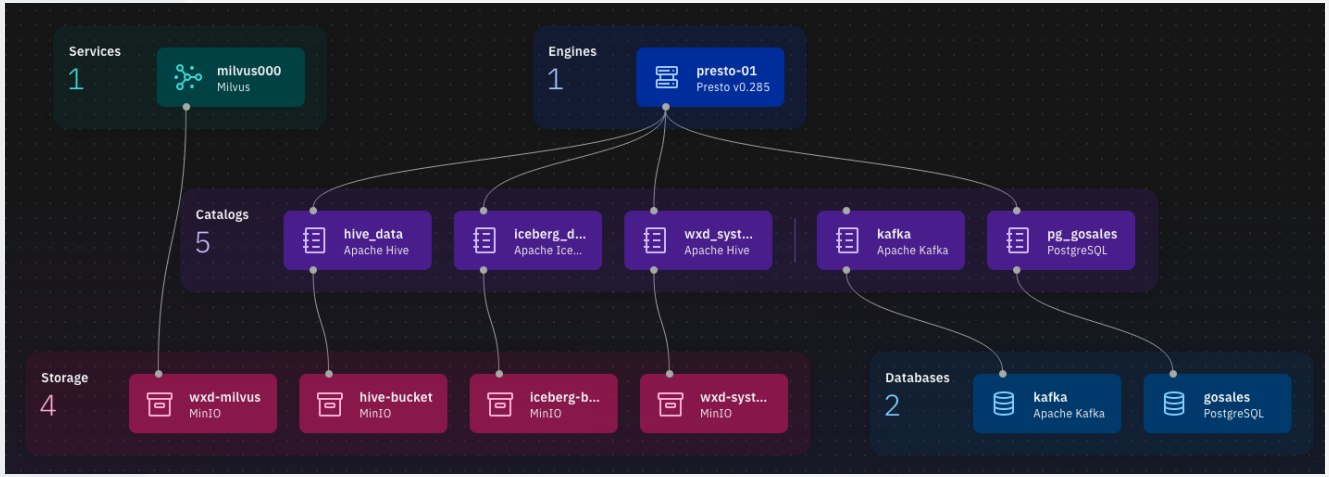
- Where did LLM get its answers?
- Is the answer based on the latest documentation?

Comparing a RAG to a "human interaction" would mean providing a person with up-to-date information and asking it to answer questions based on that information.

Summary

watsonx.data

Scale AI workloads, for all your data, anywhere



A fit-for-purpose data store based on an open lakehouse architecture

- Access all your data through a single point of entry across all clouds and on-premises environments
- Get started in minutes with built-in governance, security, and automation
- Reduce the cost of your data warehouse by up to 50% through workload optimization across multiple query engines and storage tiers[1]

IDUG

2024 NA Db2 Tech Conference

Db2 meets watsonx.data — A DBA Guide

George Baklarz

baklarz@ca.ibm.com

AIML2

Please fill out your session evaluation!

@IDUGDb2

#IDUG_NA24

Legal Disclaimer

Copyright © IBM Corporation 2024 All rights reserved.
U.S. Government Users Restricted Rights - Use, duplication, or disclosure restricted by GSA ADP Schedule Contract with IBM Corporation

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON CURRENT THINKING REGARDING TRENDS AND DIRECTIONS, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. FUNCTION DESCRIBED HEREIN MY NEVER BE DELIVERED BY IBM. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

IBM, the IBM logo, ibm.com and Db2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml