Db2 Warehouse Datalake Tables

Data Server Day 2024
Stockholm, Sweden

Kelly Schlamb
WW Technology Sales Enablement
kschlamb@ca.ibm.com

Notices and disclaimers

This document is distributed “as is” without any warranty, either express or implied. In no event shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.

Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM.

Not all offerings are available in every country in which IBM operates.

Any statements regarding IBM’s future direction, intent or product plans are subject to change or withdrawal without notice.

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at: www.ibm.com/legal/copytrade.shtml.

Certain comments made in this presentation may be characterized as forward looking under the Private Securities Litigation Reform Act of 1995.

Forward-looking statements are based on the company’s current assumptions regarding future business and financial performance. Those statements by their nature address matters that are uncertain to different degrees and involve a number of factors that could cause actual results to differ materially. Additional information concerning these factors is contained in the Company’s filings with the SEC.
Copies are available from the SEC, from the IBM website, or from IBM Investor Relations.

Any forward-looking statement made during this presentation speaks only as of the date on which it is made. The company assumes no obligation to update or revise any forward-looking statements except as required by law; these charts and the associated remarks and comments are integrally related and are intended to be presented and understood together.

Agenda

01 Data management architectures

02 Open-source file & table formats

03 Db2 WH datalake tables

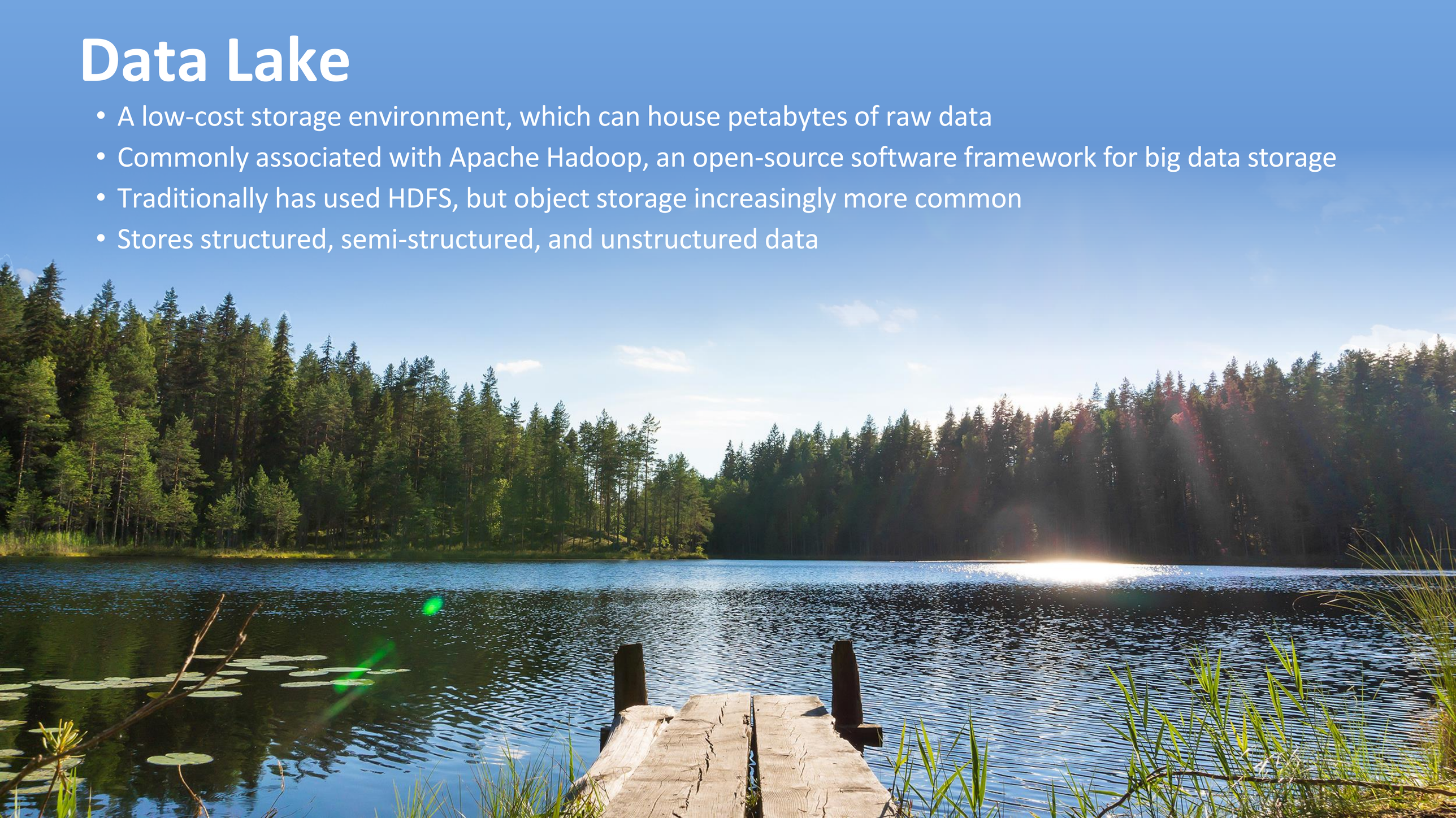
04 Creating and working with datalake tables

Data Warehouse

- Highly performant data management platform
- Data from multiple sources organized into a centralized, highly-structured relational database
- Primarily supports data analytics and business intelligence applications
- Data stored in proprietary formats on fast, expensive block-based storage devices

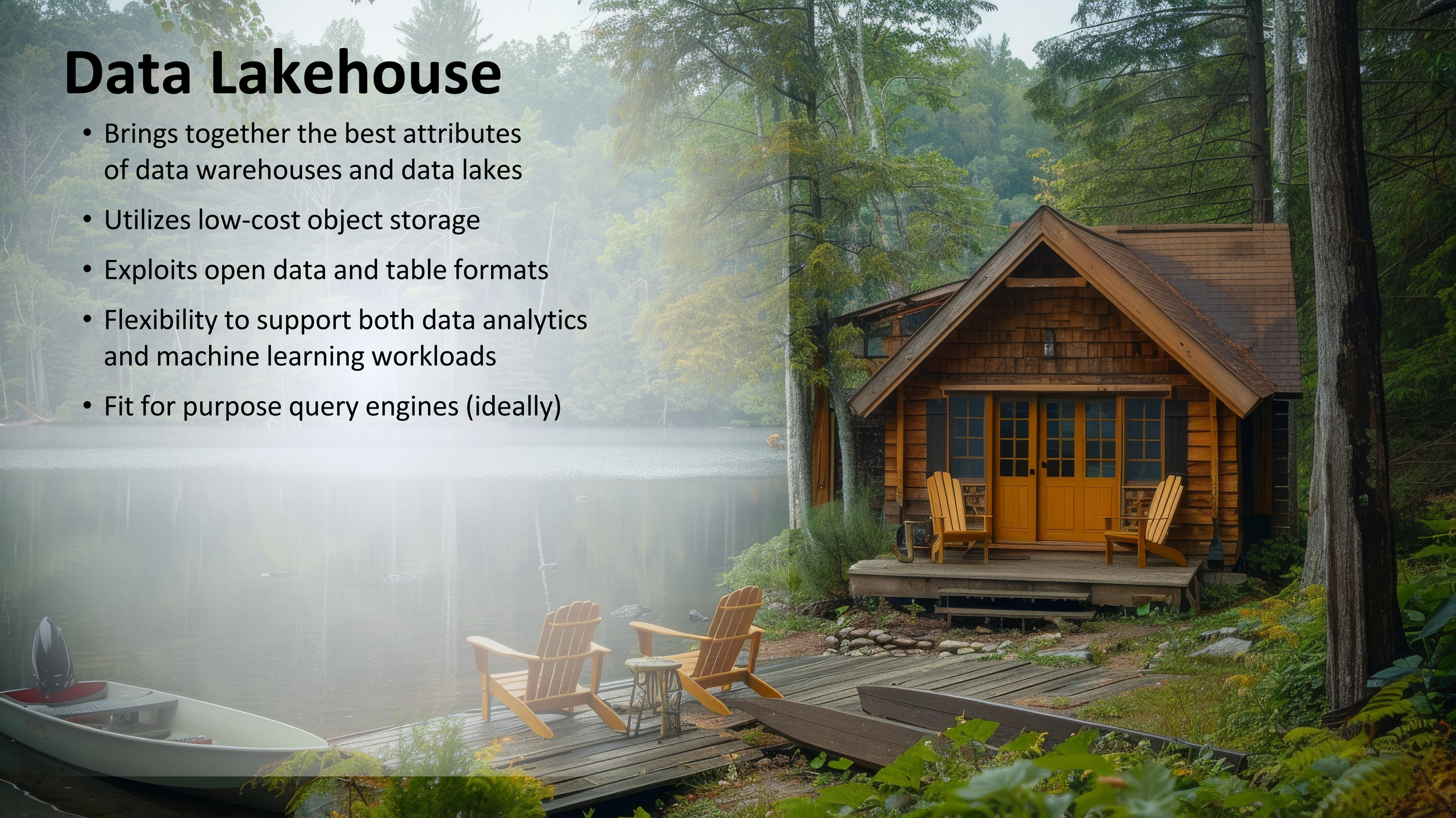
Data Lake

- A low-cost storage environment, which can house petabytes of raw data
- Commonly associated with Apache Hadoop, an open-source software framework for big data storage
- Traditionally has used HDFS, but object storage increasingly more common
- Stores structured, semi-structured, and unstructured data

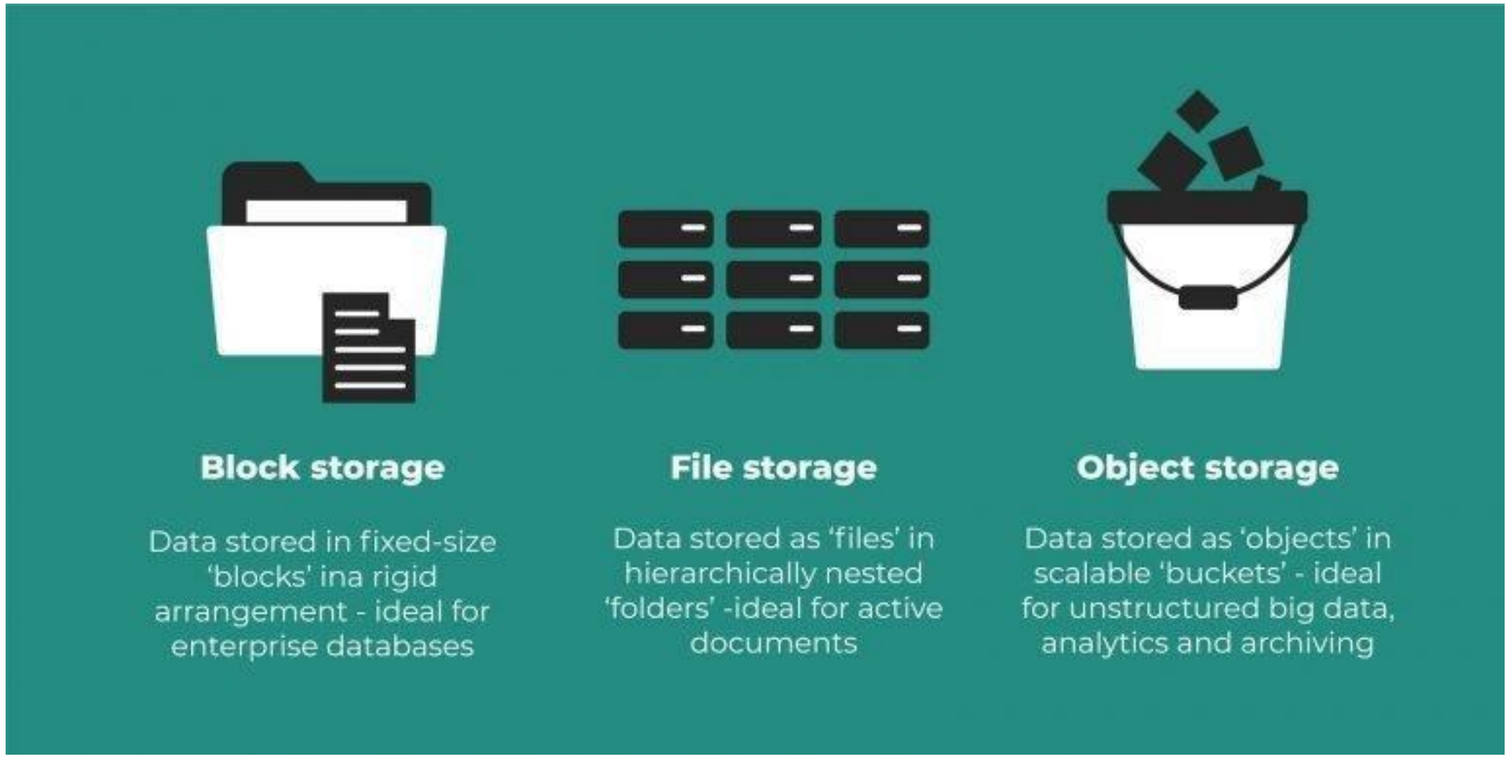


Data Lakehouse

- Brings together the best attributes of data warehouses and data lakes
- Utilizes low-cost object storage
- Exploits open data and table formats
- Flexibility to support both data analytics and machine learning workloads
- Fit for purpose query engines (ideally)



What is object storage?



Source: https://www.openpr.com/news/2367430/global-object-storage-market-market-revenue-market-growth

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

amazon S3
IBM Spectrum Fusion
Storage Ceph
DELL EMC ISILON
RedHat OpenShift
MINIO
Cloud Object Storage
NetApp

The rise of cloud object storage for data lakes and lakehouses

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

	Object Storage	HDFS	Object Storage 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

Source: https://www.databricks.com/blog/2017/05/31/top-5-reasons-for-choosing-s3-over-hdfs.html

Common open 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

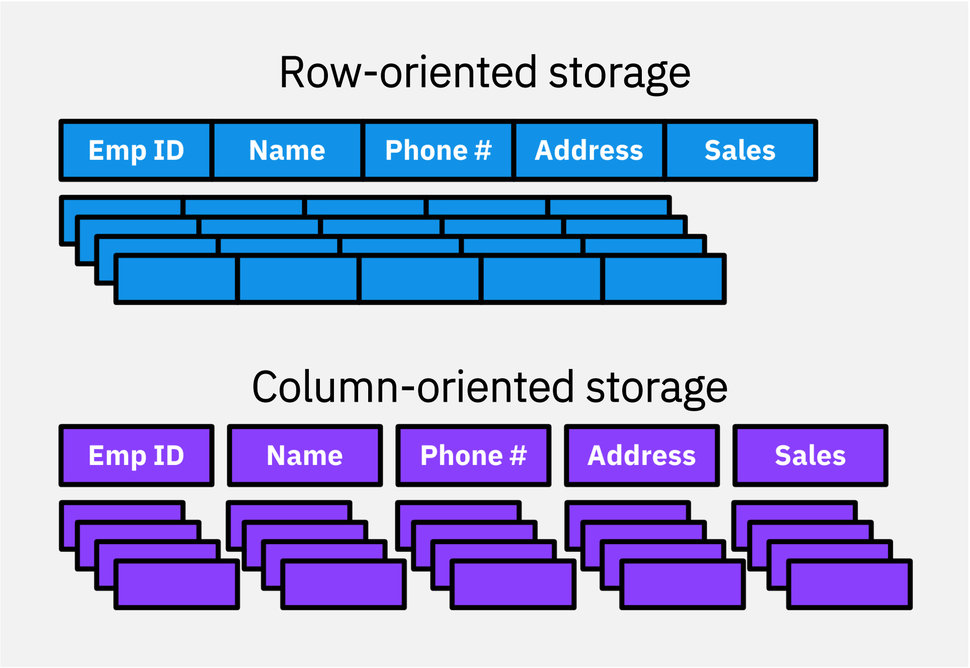
AVRO™

- 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

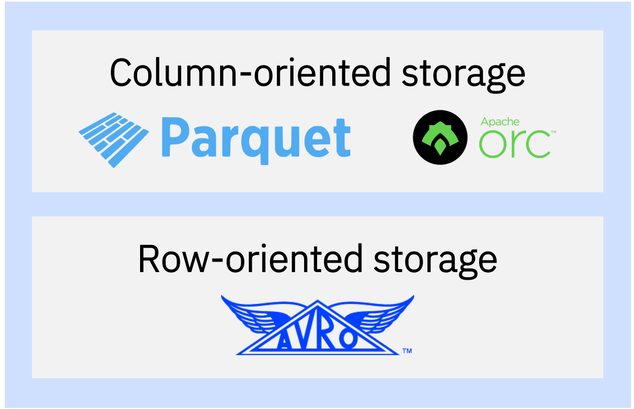


Why do these things matter in a lakehouse?

- Performance of queries directly impacted by size and amount 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)

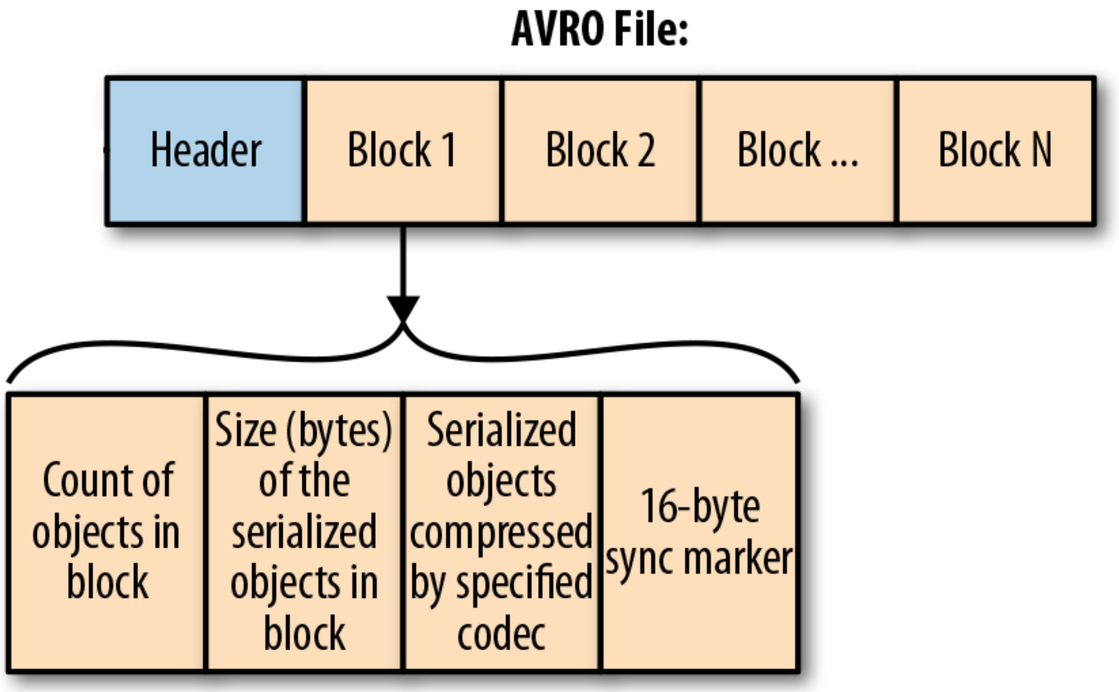
Apache ORC

- 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

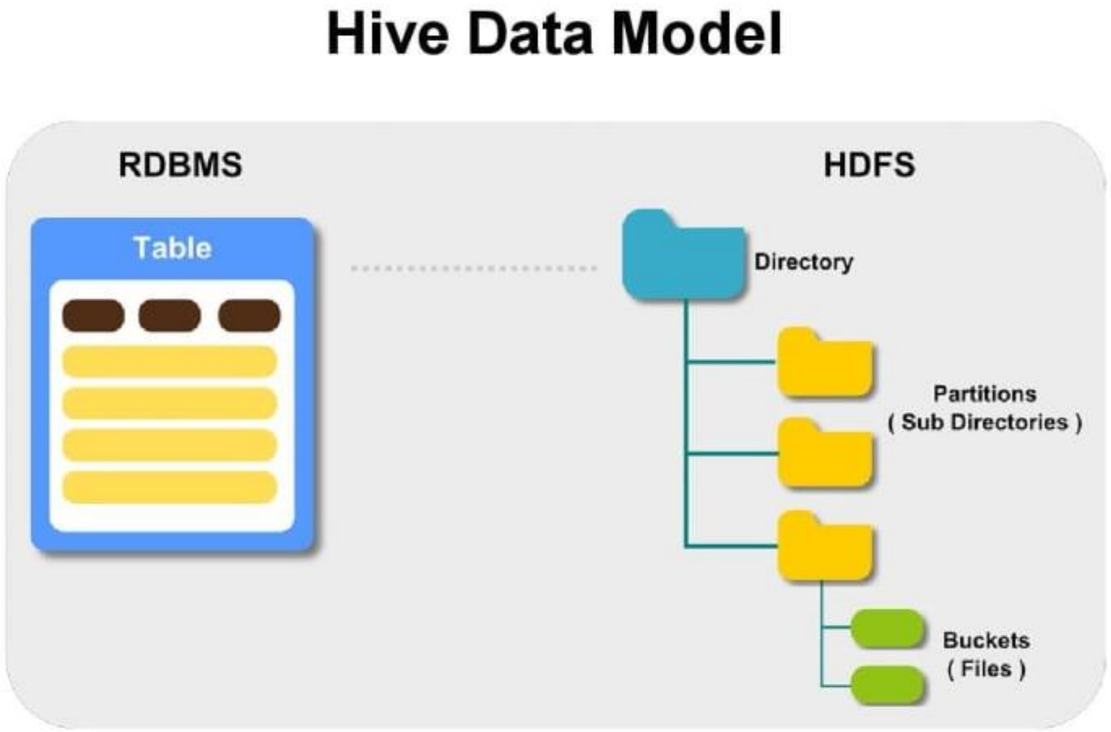
- Open-source, **row-based** storage and serialization format
  - Can be used for file storage or message passing
- Beneficial for write-intensive workloads
- Format contains a mix of text and binary
  - Data definition: Text-based JSON
  - Data blocks: Binary
- Robust support for schema evolution
 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



Source: https://www.oreilly.com/library/view/operationalizing-the-data/9781492049517/ch04.html

What are Hive tables?

- **Apache Hive** was introduced in 2010 to provide a data warehouse-like structure on top of **Hadoop**
- Supports the distributed analysis of large datasets in Hadoop's **HDFS**, as well as S3-compatible object storage
- SQL-like **HiveQL** queries are converted to **MapReduce** jobs
- **"Schema on read"** enforces structure at query time
- Tables are just **"data files in directories"** – supporting plain text, ORC, RCFile, Parquet, and other formats
- **Metadata store** (HMS) component tracks metadata such as schema and location
- No concurrency control, inefficient updates/deletes, and schema changes require rewriting entire dataset



Source: https://dev.to/aws-builders/introduction-to-hivea-sql-layer-above-hadoop-kk1

Table management and formats

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

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

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

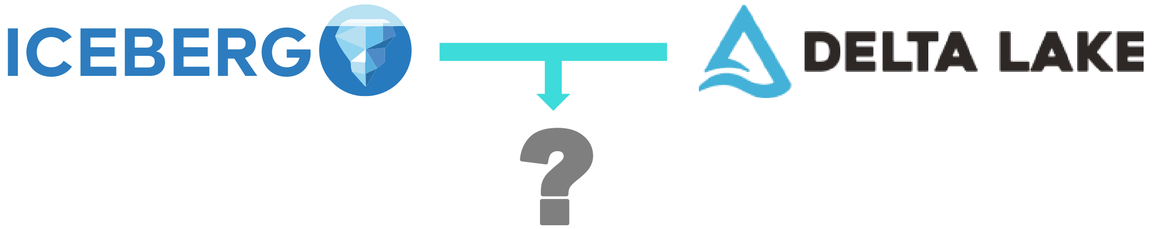
Table management and formats

Sits “above” the data file layer

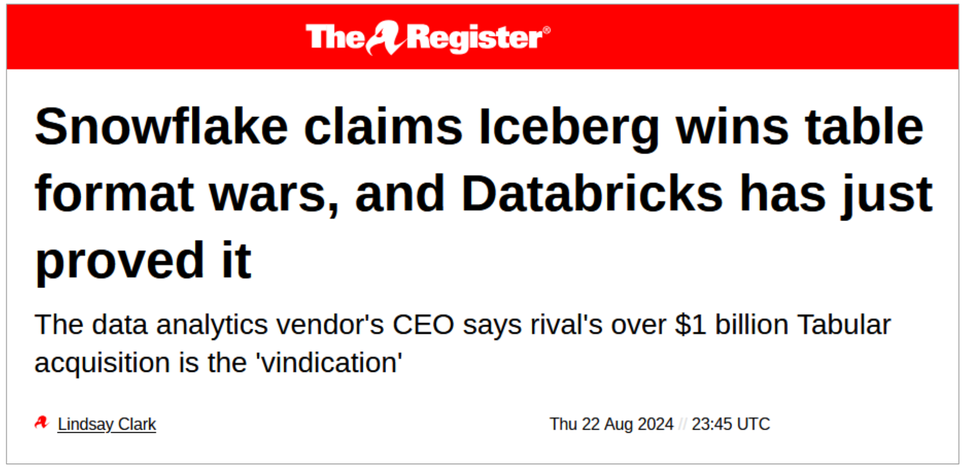
Organizes and manages table metadata and data

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



Tabular founded by original creators of Iceberg (they now join Databricks)



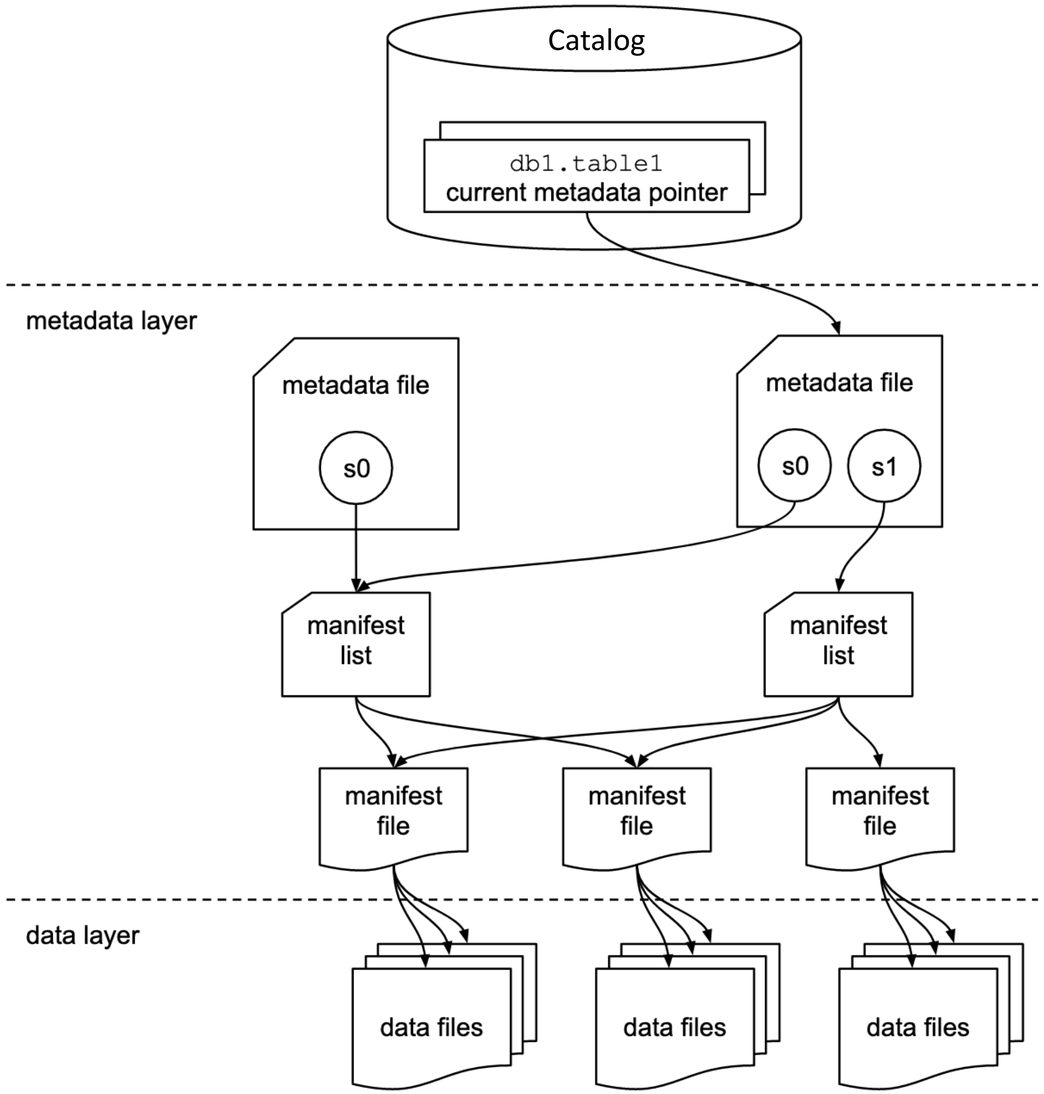
It appears the intention is to make Iceberg and Delta Lake more compatible over time, with enhanced interoperability of analytics workloads.

Apache Iceberg open data table format

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

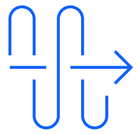
People love it because it has:

- SQL access — 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, +++)



Source: https://iceberg.apache.org/spec/

ACID transactions



ACID refers to a set of properties of database transactions intended to **guarantee data validity** despite errors, power failures, and other mishaps

Atomicity

Guarantees that each transaction is a single event that either succeeds or fails completely; there is no half-way state.

Consistency

Ensures that data is in a consistent state when a transaction starts and when it ends, guaranteeing that data is accurate and reliable.

Isolation

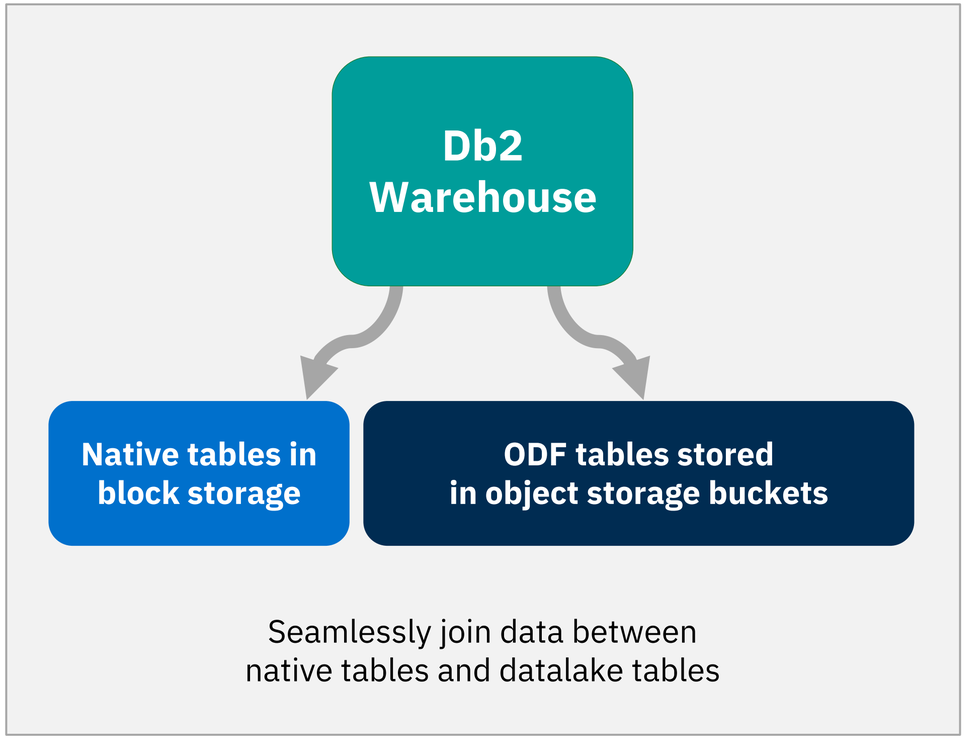
Allows multiple transactions to occur at the same time without interfering with each other, ensuring that each transaction executes independently.

Durability

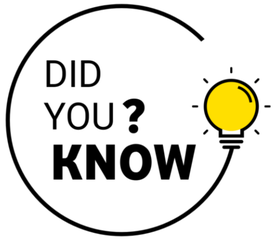
Means that data is not lost or corrupted once a transaction is submitted. Data can be recovered in the event of a system failure, such as a power outage.

Db2 Warehouse DATALAKE tables

- Work with Db2 data in open data & table formats (e.g. Parquet, Iceberg) hosted on low-cost object storage
- Optimize resources by segmenting workloads across the warehouse and other datalake/lakehouse engines
- Seamlessly combine warehouse data with enterprise lakehouse data
- Export Db2 warehouse data to object storage (e.g. CTAS), while retaining the ability to query that data
- Use a datalake engine (e.g. Spark) to cleanse and transform data; then bring that curated data into Db2



Some interesting facts about Db2 datalake tables



Two types of datalake tables:
Hive ("normal") & Iceberg

Data not owned by Db2 - stored externally outside of the database

New CREATE/ ALTER/RENAME/ DROP DATALAKE statements

Insert data using INSERT, SELECT INTO, or CREATE TABLE AS SELECT (CTAS)

Operations (DDL, DML) are outside of Db2 transactional control (either succeed or fail)

Supports Parquet, ORC, Avro, text file, and JSON data file formats (depending on table type)

Collect statistics using the ANALYZE TABLE statement

Supported in Db2 WH 11.5.9 (OpenShift/K8s) & Db2 Warehouse on Cloud (Gen 3)

Based on technology from IBM Db2 Big SQL

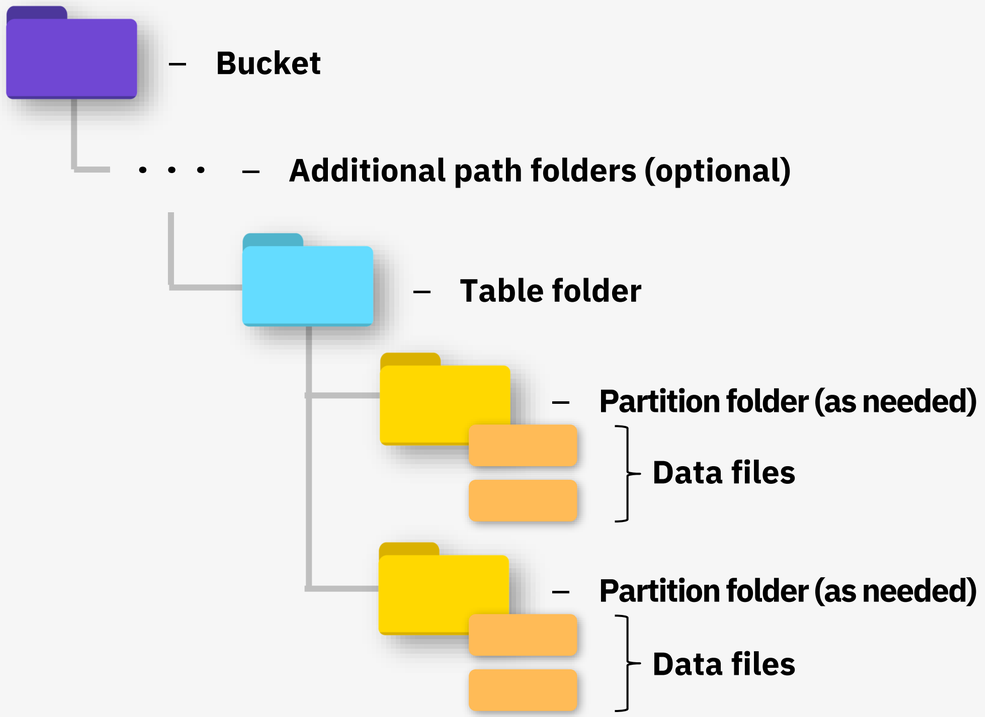
Uses a built-in Hive Metastore (HMS)

Hive vs. Iceberg datalake tables

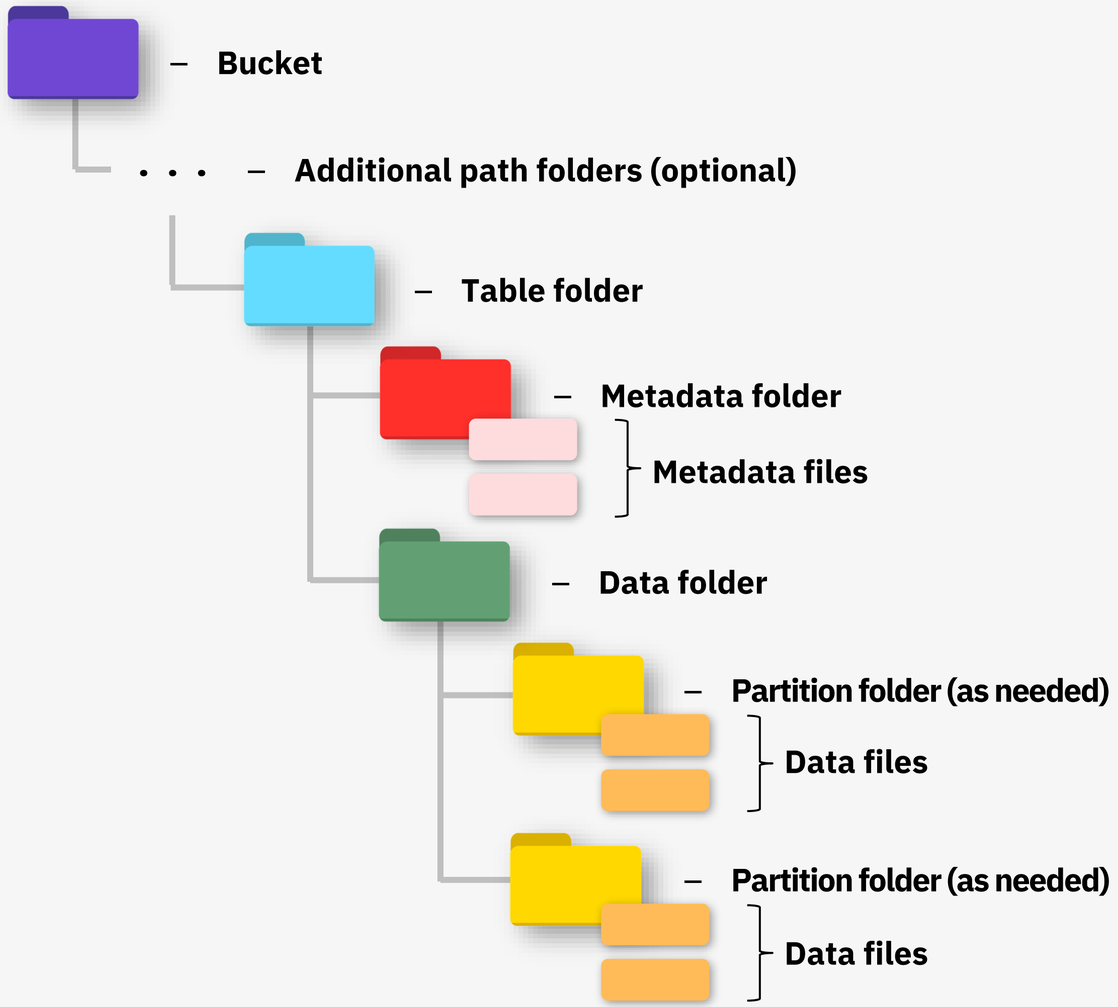
Capability/Behavior	Hive Datalake Tables	Iceberg Datalake Tables
ACID transaction support (of underlying table format type)	No	Yes
Suitability	Read-only or append-only tables (only INSERT supported)	Transactional workloads * (INSERT today; UPDATE and DELETE not *yet* supported)
Schema evolution	Requires rewriting dataset	Schema evolution supported
Supported data file formats	Parquet, ORC, Avro, text file, JSON	Parquet, ORC, Avro
Can create datalake table on top of existing data?	Yes	No (but can sync with external HMS)
CREATE DATALAKE TABLE syntax	Lack of STORED BY clause implies Hive	STORED BY ICEBERG

* While the Iceberg table format itself supports I/U/D operations, it is NOT suitable to handle the fast, high-frequency OLTP transactions that Db2 is built to handle.

Object storage organization for tables



Hive DatalakeTables



Iceberg Datalake Tables

Object storage credentials

Required by Db2:

- Bucket name
- Endpoint
- Access key
- Secret access key

Region Name	Region	Endpoint	Protocol	Signature Version(s) Support
US East (N. Virginia)	us-east-1	s3.us-east-1.amazonaws.com	HTTP and HTTPS	2 & 4

Access key	Secret access key
AKIAXFPVIVHXPJZUGKWY	DEylbP2rsB3zcsMyNUd36H0ap9KYMe36cT4AOEmm Hide

Storage access aliases

- Allows Db2 to locate and access object storage buckets
- Call to **SYSIBMADM.STORAGE_ACCESS_ALIAS.CATALOG( )** includes:
  - Name of storage access alias
  - Storage vendor/type (S3)
  - Storage endpoint
  - Access key
  - Secret access key
  - Bucket name
  - High-level folder in which to organize and store data
  - Who is authorized to use the alias

```
CALL SYSIBMADM.STORAGE_ACCESS_ALIAS.CATALOG ('s3_alias', 'S3',
 'ibm-lh-lakehouse-minio-svc.cpd.svc.cluster.local:9000',
 'fbsj2i5Cee4bww1BacBfo0v2', 'jv7oBC8jeBefq3fakEEFijmz',
 'db2hivebkt', '', 'I', '');
```

SYSIBMADM.STORAGE_ACCESS_ALIAS.CATALOG Docs

Hive datalake table examples

```
CREATE DATALAKE TABLE HIVETAB1 (C1 DATE, C2 INT, C3 TIMESTAMP)
  STORED AS PARQUET
  LOCATION 'DB2REMOTE://myalias//db2tables/hivetab1';
```

```
CREATE DATALAKE TABLE HIVETAB2 (C1 CHAR(20), C2 FLOAT, C3 INT)
  STORED AS ORC
  LOCATION 'db2remote://stgalias//tables/myschema/hivetab2';
```

```
CREATE DATALAKE TABLE HIVETAB3 (C1 INT, C2 FLOAT, C3 DATE)
  STORED AS PARQUET
  LOCATION 'DB2REMOTE://s3alias//db2bucket/hivetab3'
  TBLPROPERTIES ('external.table.purge'='true');
```

```
CREATE DATALAKE TABLE HIVETAB4
  LOCATION 'db2remote://objstg//offloadedtables/hivetab4'
  AS SELECT * FROM MYDB2TABLE;
```

```
CREATE DATALAKE TABLE HIVETAB5 (ORDER_KEY INT NOT NULL, ORDER_NOTES VARCHAR(100) NOT NULL)
  PARTITIONED BY (ORDER_DATE VARCHAR(10))
  LOCATION 'DB2REMOTE://salesbktalias//orderdata/hivetab5';
```

CREATE DATALAKE TABLE Docs

Iceberg datalake table examples

```
CREATE DATALAKE TABLE ICEBERGTAB1 (C1 INT, C2 CHAR(100), C3 DATE, C4 TIMESTAMP, C5 FLOAT)
  STORED AS PARQUET
  STORED BY ICEBERG
  LOCATION 'DB2REMOTE://stgalias//icebergtab1'
  TBLPROPERTIES('external.table.purge'='false')
```

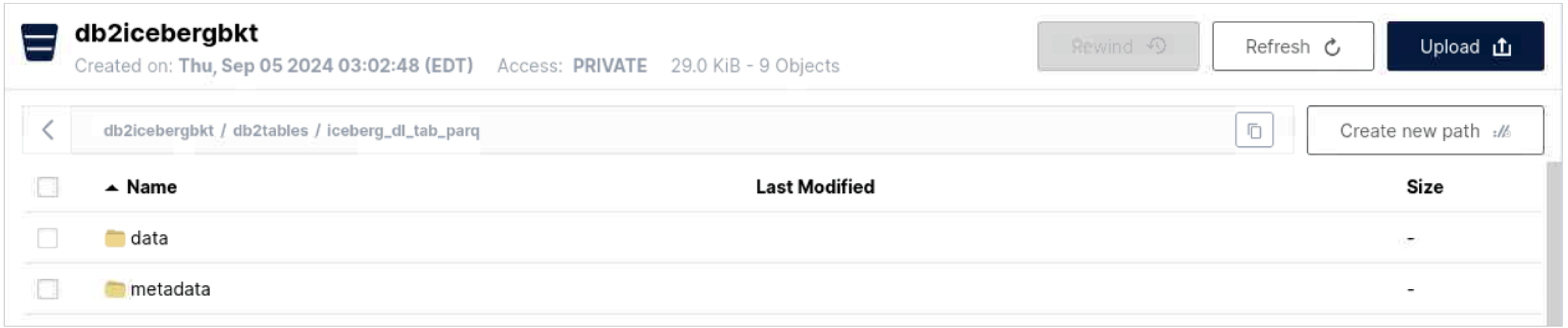
```
CREATE DATALAKE TABLE ICEBERGTAB2 (C1 CHAR(10), C2 FLOAT, C3 FLOAT)
  STORED AS PARQUET
  STORED BY ICEBERG
  LOCATION 'db2remote://myalias//mydb2tables/myschema/icebergtab2';
```

```
CREATE DATALAKE TABLE ICEBERGTAB3
  STORED AS PARQUET
  STORED BY ICEBERG
  LOCATION 'DB2REMOTE://s3bkt//db2data/icebergtab3'
  AS SELECT * FROM SALESDATA WHERE SALES_YEAR < 2021;
```

```
CREATE DATALAKE TABLE ICEBERGTAB4 (PROD_ID CHAR(30) NOT NULL, PROD_DESC CHAR(100) NOT NULL)
  PARTITIONED BY (TRUNCATE(10, PROD_ID))
  STORED BY ICEBERG
  LOCATION 'db2remote://salesbktalias//db2tables/icebergtab4';
```

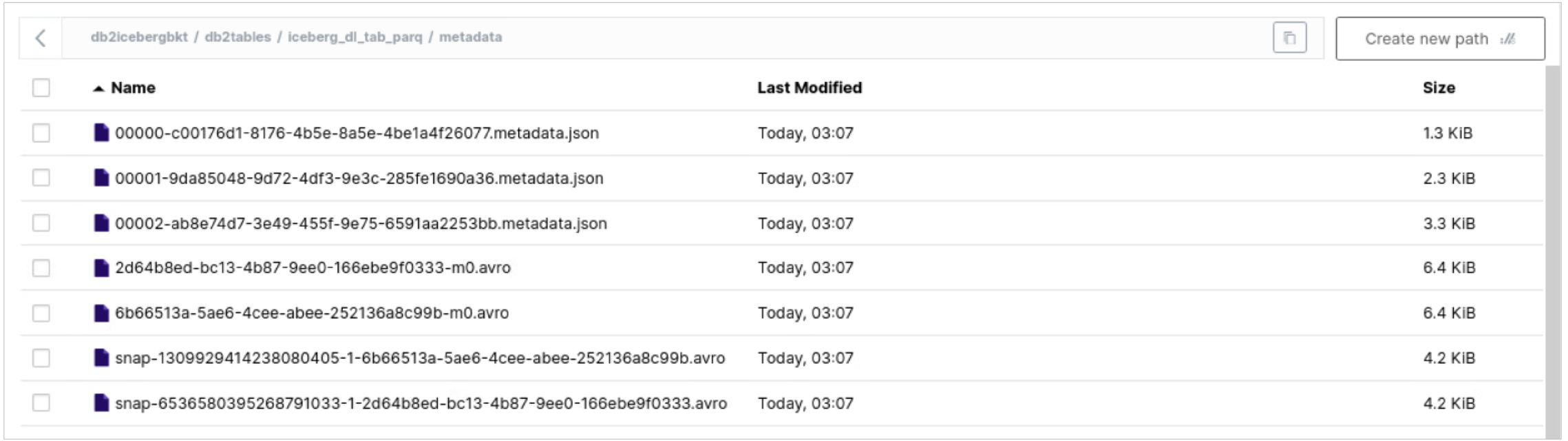
CREATE DATALAKE TABLE Docs

What does a Parquet-based Iceberg table look like in object storage?

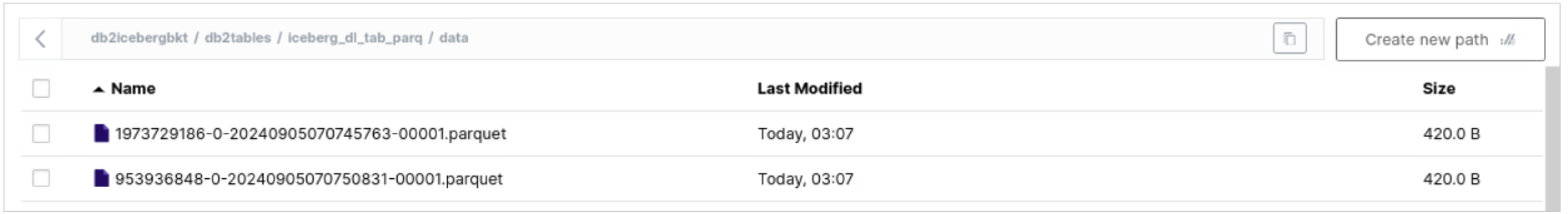


What does a Parquet-based Iceberg table look like in object storage? (cont.)

metadata

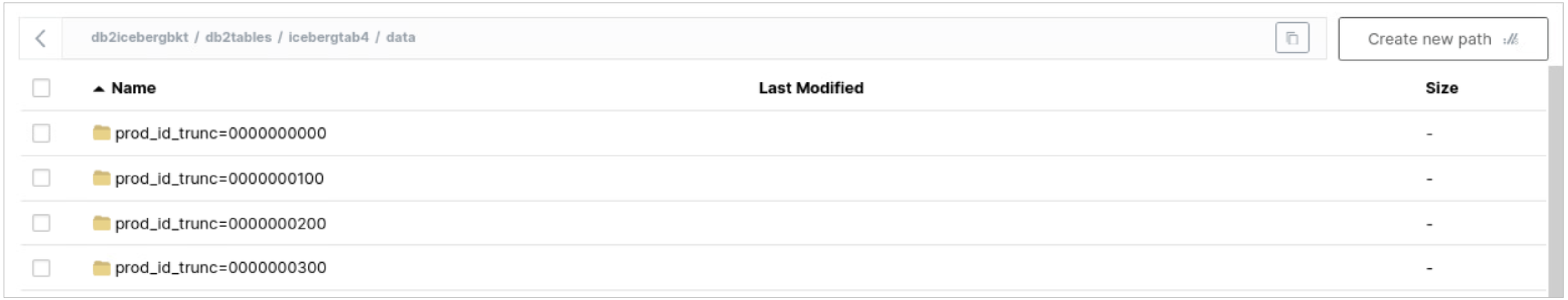


data

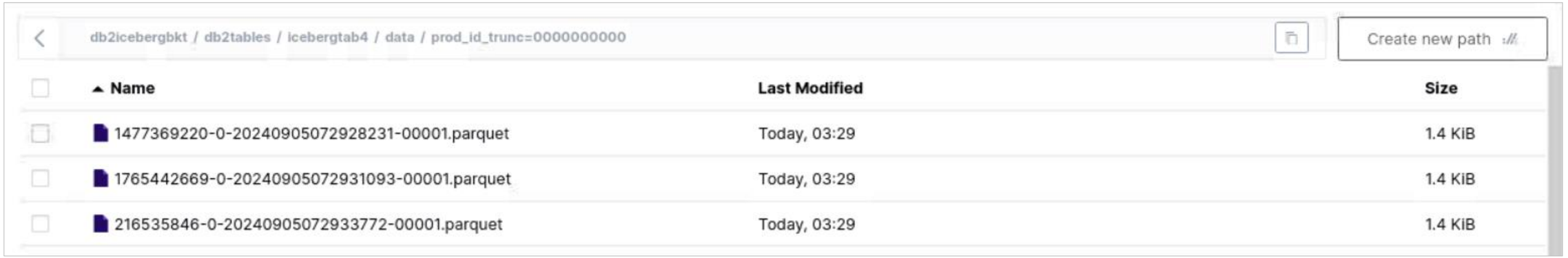


What does a Parquet-based Iceberg table look like in object storage? (cont.)

Folders for a partitioned table (under "data" folder)



Data files in one of these partitions



Which tables are datalake tables? Which are Iceberg?

```
SELECT TABSCHEMA,
       TABNAME,
       SUBSTR(PROPERTY,18,1) AS IS_ICEBERG,
       SUBSTR(PROPERTY,22,1) AS IS_DATALAKE_TABLE
  FROM SYSCAT.TABLES
  ORDER BY TABSCHEMA, TABNAME;
```

```
TABSCHEMA                       TABNAME                         IS_ICEBERG IS_DATALAKE_TABLE
------------------------------- ------------------------------- ---------- ----------------
DB2INST1                        HIVE_DL_TAB_ORC                            Y
DB2INST1                        HIVE_DL_TAB_PARQ                           Y
DB2INST1                        HIVE_DL_TAB_TXT                            Y
DB2INST1                        ICEBERG_DL_TAB_PARQ             Y          Y
DB2INST1                        POLICY
DB2INST1                        USER_TABLE

  6 record(s) selected.
```

Where are my datalake tables located?

```
SELECT TABSCHEMA, TABNAME, LOCATION
  FROM SYSHADOOP.HCAT_TABLES ORDER BY TABSCHEMA, TABNAME;
```

```
TABSCHEMA                       TABNAME                         LOCATION
------------------------------- ------------------------------- --------------------------------------------------
DB2INST1                        HIVE_DL_TAB_ORC                 s3a://db2hivebkt/db2tables/hive_dl_tab_orc
DB2INST1                        HIVE_DL_TAB_PARQ                s3a://db2hivebkt/db2tables/hive_dl_tab_parq
DB2INST1                        HIVE_DL_TAB_TXT                 s3a://db2hivebkt/db2tables/hive_dl_tab_txt
DB2INST1                        ICEBERG_DL_TAB_PARQ             s3a://db2icebergbkt/db2tables/iceberg_dl_tab_parq

  4 record(s) selected.
```

SYSHADOOP.HCAT_TABLES Docs

Dropping datalake tables

- By default, DROP DATALAKE TABLE does not remove the table's directories and files
  - Information is only removed from the Db2 catalogs
  - Data is managed outside of Db2, and depending on your environment there could be other external data users or future needs for that data
- Use DELETE DATA option to delete the data as well
  - Requires table's `external.table.purge` TBLPROPERTIES value be set to `true` (or for Iceberg tables, alternatively set `gc.enabled` to `true`)
- Deleted directories and files are moved to the storage's trash bin
  - System administrator enables and manages trash bin support, including regular cleaning
  - Deleted directories and files moved to .Trash folder in storage
- Alternatively, use DELETE DATA PURGE open to permanently delete the files

Restrictions and limitations

- Familiarize yourself with datalake table restrictions and limitations
  - Unsupported data types (e.g. BINARY, DECFLOAT, LOB, GRAPHIC, XML, UDT, ...)
  - Unsupported features (e.g. enforced constraints, generated columns, RCAC, LBAC, isolation levels, data capture changes, CGTT, ...)
  - Tables can't be range-partitioned, MDC, MQT, Temporal, or Typed
  - Can't specify IN TABLESPACE (data is external and doesn't live in a table space)
  - Can't create text indexes on datalake tables
- Schema names, table names, and column names must be unique, even when different case specified ('ABC' = 'abc')
- Iceberg tables:
  - UPDATE and DELETE not supported (*yet*)
  - Time travel queries not supported

Additional resources

My blogs:

- How to create datalake tables in Db2WH
- How to integrate Db2WH with watsonx.data

Relevant Db2 Warehouse documentation:

- Datalake table terminology
- Introduction to datalake tables
- Creating datalake tables
- Restrictions and limitations

Open-source documentation:

- Parquet data file format
- ORC data file format
- Avro data file format
- Iceberg table format

Going to TechXchange in Las Vegas in October?

- Attend my hands-on lab on Db2 datalake tables and watsonx.data integration (session #1848)