



#37 PARMA 2023

Databricks Lakehouse

Strategie vincenti dalla teoria alla pratica





Speaker

Andrea Bergonzi

Data Scientist @ Dataskills srl



andrea.bergonzi@dataskills.it



<https://www.linkedin.com/in/andrea-bergonzi-5a1390103/>

Dataskills

Specializzati nella creazione di soluzioni innovative nelle quattro aree principali della Data Science.

BUSINESS INTELLIGENCE

- Trasformare dati e informazioni in conoscenza

PREDICTIVE ANALYTICS

- Utilizzare i dati per offrire previsioni sul futuro

BIG DATA

- Gestire, immagazzinare e analizzare immense moli di dati

IOT ANALYTICS

- Estrarre e sfruttare i dati provenienti da device interconnessi

25

Anni di
esperienza

90+

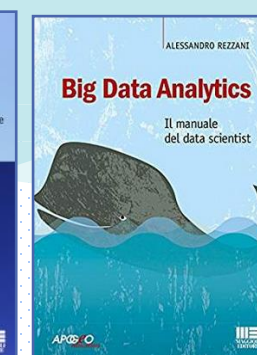
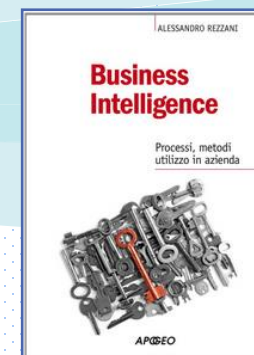
Progetti
realizzati

3

Libri di Data
Science
pubblicati

2

Professori
all'Università
Bocconi



Agenda

Teoria del Data LakeHouse

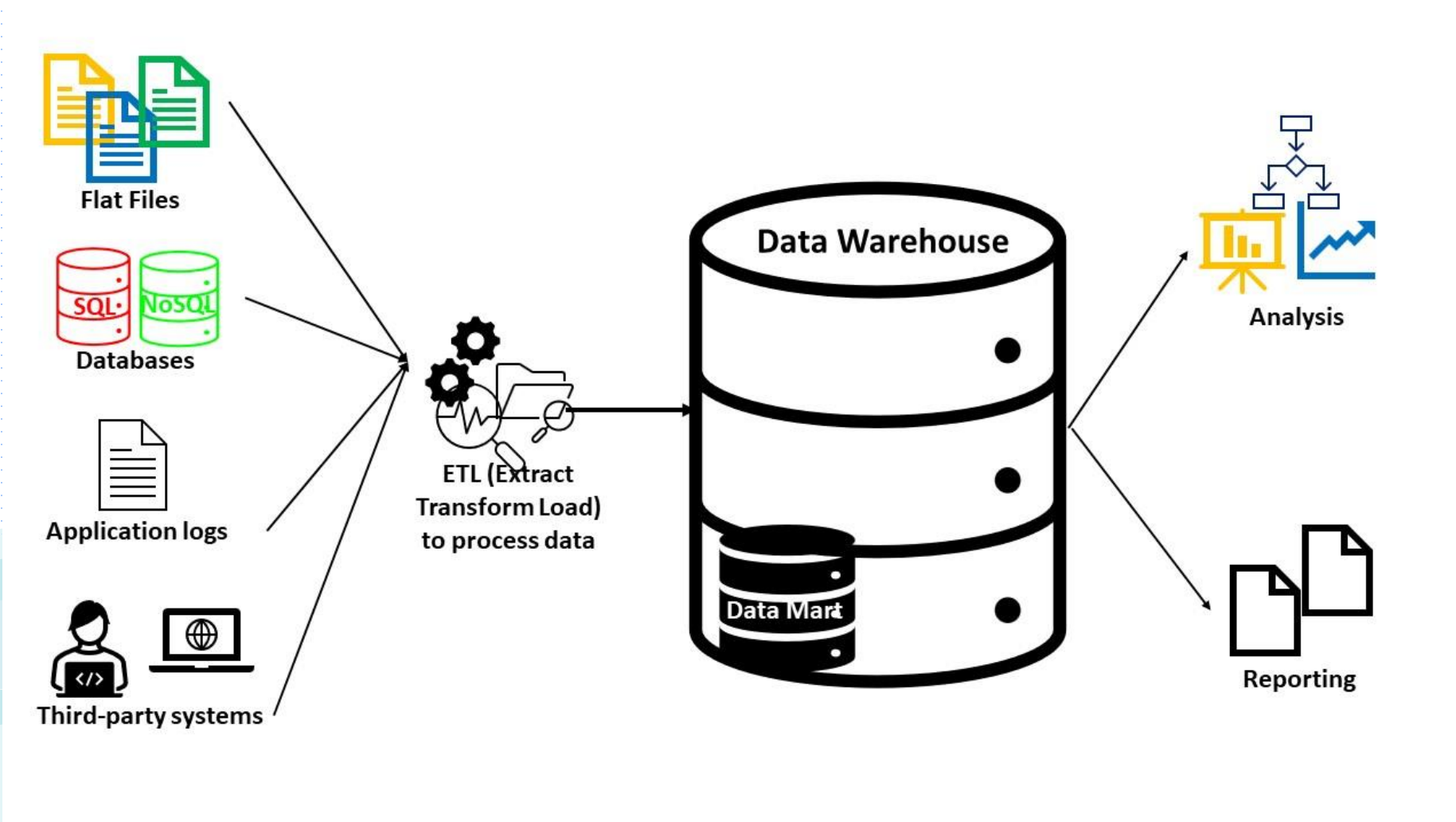
1. Introduzione al Paradigma LakeHouse
2. Medallion Architecture
3. Tecnologie per il Lakehouse: Databricks, Delta Lake e Unity Catalog

Nella Pratica: Implementazione di un Lakehouse

5. Ingestion tramite chiamate API
6. Costruzione della Medallion Architecture
7. Organizzazione e gestione dei notebook
8. Creazione dei workflow e orchestrazione delle pipeline
9. Tips & Tricks per un'implementazione efficace

II Paradigma LakeHouse

Concetti: Il Data Warehouse



Concetti: Il Data Warehouse

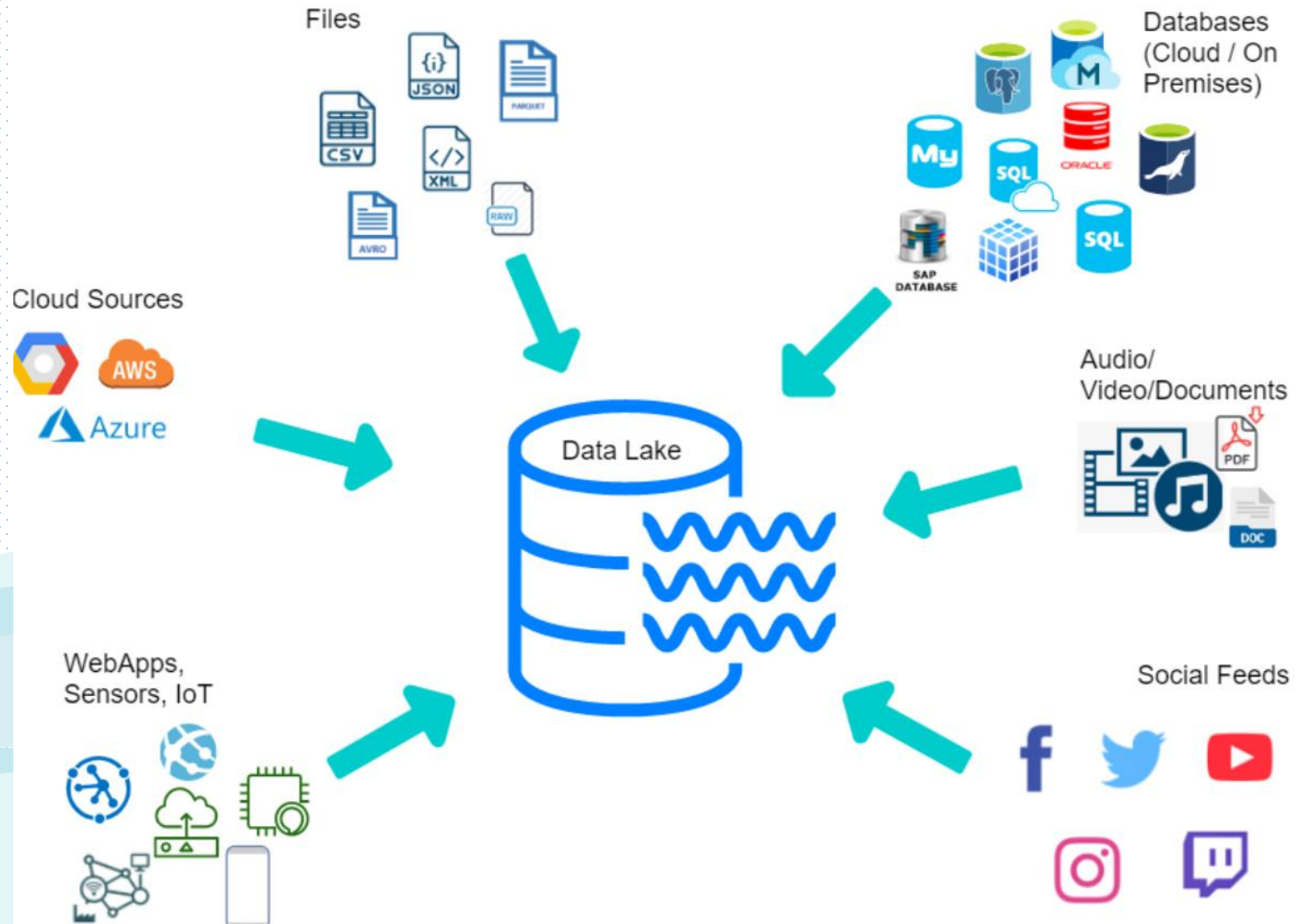
Punti di forza:

- High data quality
- Built for reporting

Punti di debolezza:

- Poor support for unstructured data, AI and streaming
- Closed, proprietary formats
- Expensive to scale

Concetti: Il Data Lake



Concetti: Il Data Lake

Punti di forza:

- Support for any kind of data
- Low cost for storage

Punti di debolezza:

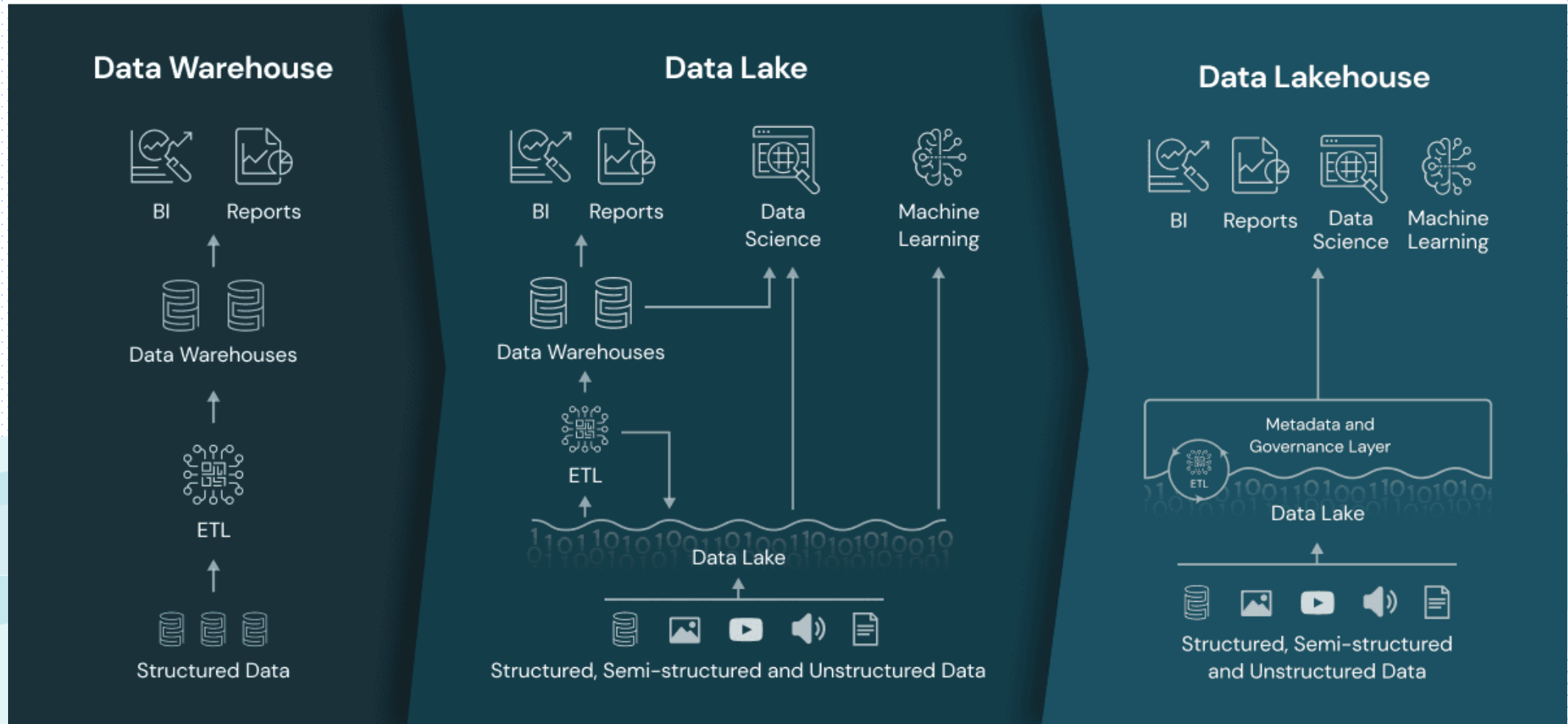
- Complex to set up
- Poor BI performance
- Difficulties in governance ("*data swamp*")

Data Warehouse vs. Data Lake

	Data Warehouse	Data Lake
Tipo di dati	Strutturati, ordinati, puliti	Eterogenei: strutturati e non strutturati, "grezzi"
Progettazione	Definita ex-ante ("Schema-on-write")	Definita ex-post ("Schema-on-read")
Utenti finali	Analisti di business, management	Data scientist, sviluppatori, analisti di business
Più adatto per	Reporting, Business Intelligence, visualizzazione	Machine Learning, Predictive Analytics, Data Mining
Qualità dei dati	Molto elevata	Non necessariamente elevata
Costi di progettazione e manutenzione	Relativamente elevati	Relativamente bassi
Competenze richieste per l'utilizzo	Relativamente basse	Relativamente elevate
Capacità di coinvolgere più utenti finali	Relativamente elevata	Relativamente bassa

Paradigma Lakehouse

«Il meglio dei due mondi»



Paradigma Lakehouse

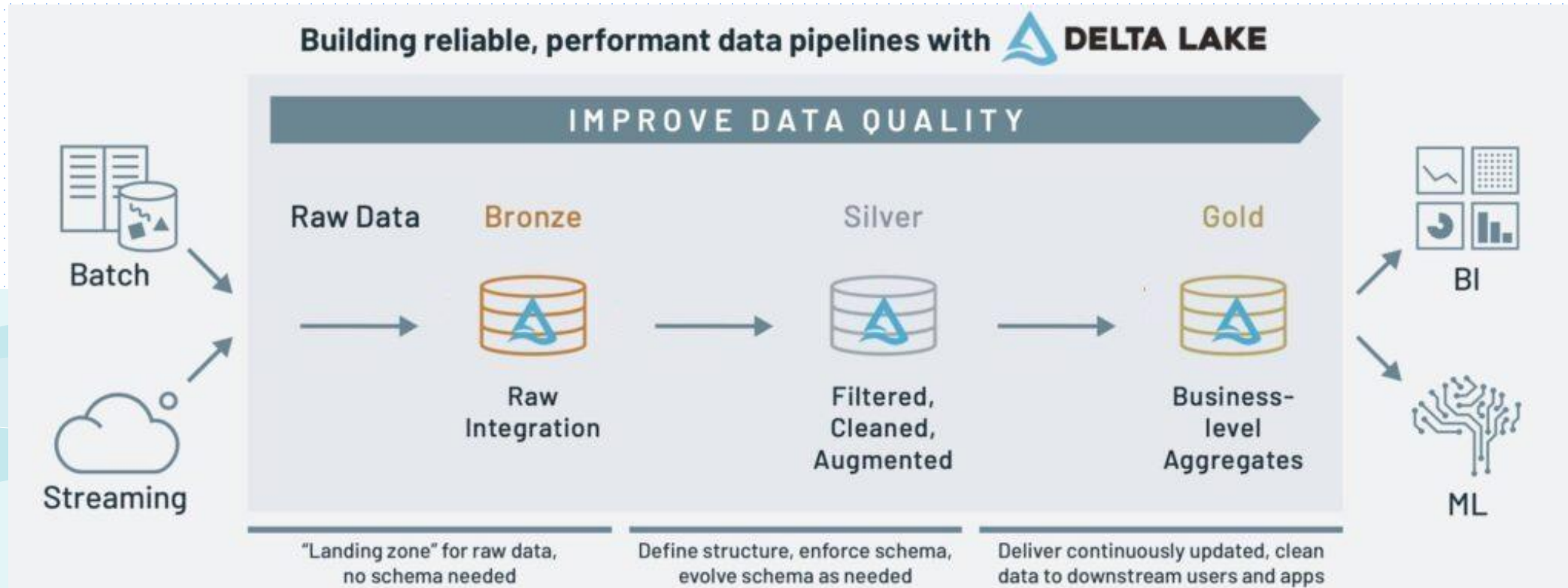
Vantaggi del Lakehouse

- **Flessibilità**
- **Scalabilità**
- **Openness**
- **Separazione tra storage e parte computazionale**
- **Costi ridotti**
- **Supporto per workload differenti**

Paradigma Lakehouse

Medallion Architecture

È il *data design pattern* utilizzato per organizzare logicamente i dati nel Lakehouse, con l'obiettivo di aumentare progressivamente la struttura e la qualità del dato tra un passaggio e l'altro negli strati dell'architettura.



Paradigma Lakehouse

Bronze Layer – Raw Data

- È dove “atterrano” i dati.
- Struttura delle tabelle è tendenzialmente “as-is” rispetto a quelle delle fonti.
- Posso aggiungere campi tecnici per catturare metadata
- Base per le rifiniture del dato negli stati successivi
- Archivio storico del dato grezzo (Cold data).

Paradigma Lakehouse

Silver Layer – Filtered, Cleaned, Augmented Data

I dati provenienti dal layer bronze sono elaborati, mergiati, ripuliti e conformati sufficientemente in modo da fornire un' *“Enterprise View”*.

Il Silver Layer riconcilia le diverse fonti e fornisce la possibilità di interrogazioni, reporting ad-hoc, advanced analytics e machine learning. I dati non sono però ancora business-ready ed è quindi raccomandato che questo layer sia utilizzato principalmente da utenti più avanzati come Data Scientist e Data Engineers.

Paradigma Lakehouse

Gold Layer – Business-level Aggregates

I dati nel Gold layer sono *consumption-ready* e possono essere organizzati in database “project-specific”. Il Gold Layer è quello che alimenta la reportistica e qui i dati devono essere de-normalizzati in ottica analitica.

Prima di entrare nel Gold Layer vengono applicate le varie regole di business ai dati in modo che siano puliti e certificati, e seguano rigide regole di **Data Quality** come in un Data Warehouse tradizionale.

Tecnologie per il LakeHouse

Databricks, Delta Lake e Unity Catalog



Che cos'è Databricks

Data Analytics Platform

- Databricks SQL
- Data Science & Engineering
- Machine Learning

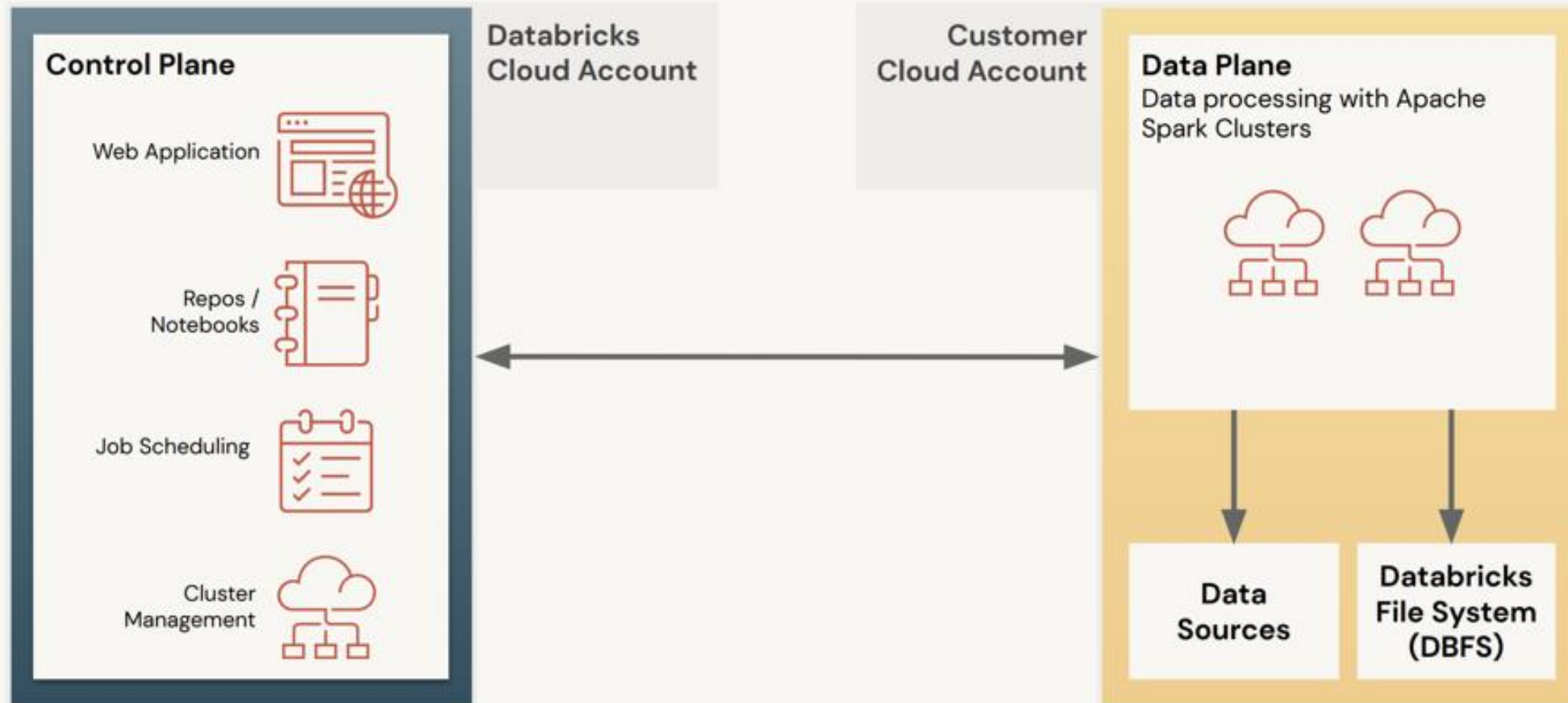
Che cos'è Databricks

Databricks è **Apache Spark**, ma con un'interfaccia utente semplice da utilizzare e tantissimi altri automatismi e servizi a disposizione

Un po' di storia...

- **2009:** Nasce Apache Spark nei laboratory AMPLab dell'Università di Berkley, California
- **2013:** Tre professori universitari: Ali Ghodsi, Ion Stoica, e Matei Zahara, già creatori di Apache Spark, decidono di fondare Databricks. Il progetto ottiene immediatamente finanziamenti importanti, come quello del fondo Andreessen Horowitz.
- **2015:** Databricks viene integrata come servizio all'interno del marketplace cloud di AWS.
- **2017:** Grazie alla partnership con Microsoft, I servizi di Databricks vengono introdotti all'interno della piattaforma Azure
- **2021:** Con la partnership con Google e l'inserimento all'interno di Google Cloud, Databricks è presente in tutti i maggiori cloud provider al mondo.
- **2022:** Oltre 5000 imprese si affidano a Databricks per gestire le proprie attività di Data Engineering e Data Science, tra cui metà delle società della Fortune 500.

Databricks Architecture



Delta Lake



Delta Lake in Databricks

Che cos'è Delta Lake?



- È un formato di dati **open source** che risiede sopra il **Lakehouse** per risolvere alcune delle sue limitazioni
- Il Delta Lake può essere pensato come un'estensione all'interno del Data lakehouse (come csv, parquet, avro...).
- Azure Databricks possiede nativamente un delta engine che facilita l'utilizzo del formato delta lake per le operazioni di data engineering.

Delta Lake in Databricks

Transaction log

Le transazioni all'interno del Transaction Log del Delta Lake (chiamate anche Delta Logs) sono record in ordine cronologico di tutte le transazioni avvenute all'interno della tabella in formato delta dal momento della sua creazione.

Questo permette di:

- Scrivere e leggere la stessa tabella anche **da più utenti contemporaneamente**.
- Garantire **l'atomicità** del Delta Lake.
- Effettuare operazioni di **roll-back** alla versione precedente della tabella

Unity Catalog



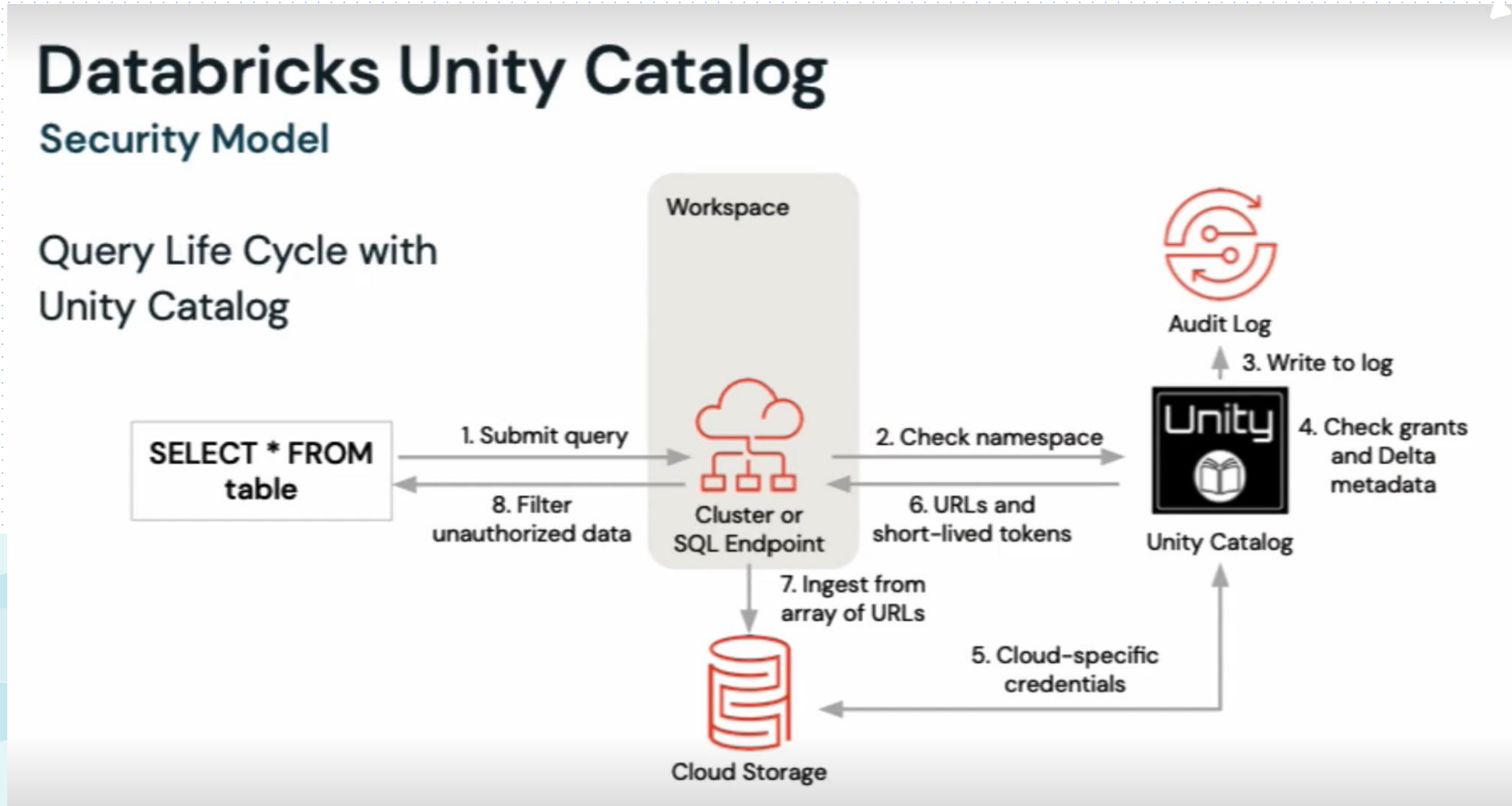
Che cos'è Unity Catalog?

"Soluzione di governance unificata per i dati e l'AI sul Lakehouse"

In particolare, garantisce:

- **Data Access Control**
- **Data Access Audit**
- **Data Lineage**
- **Data Discovery**

Come funziona Unity Catalog?



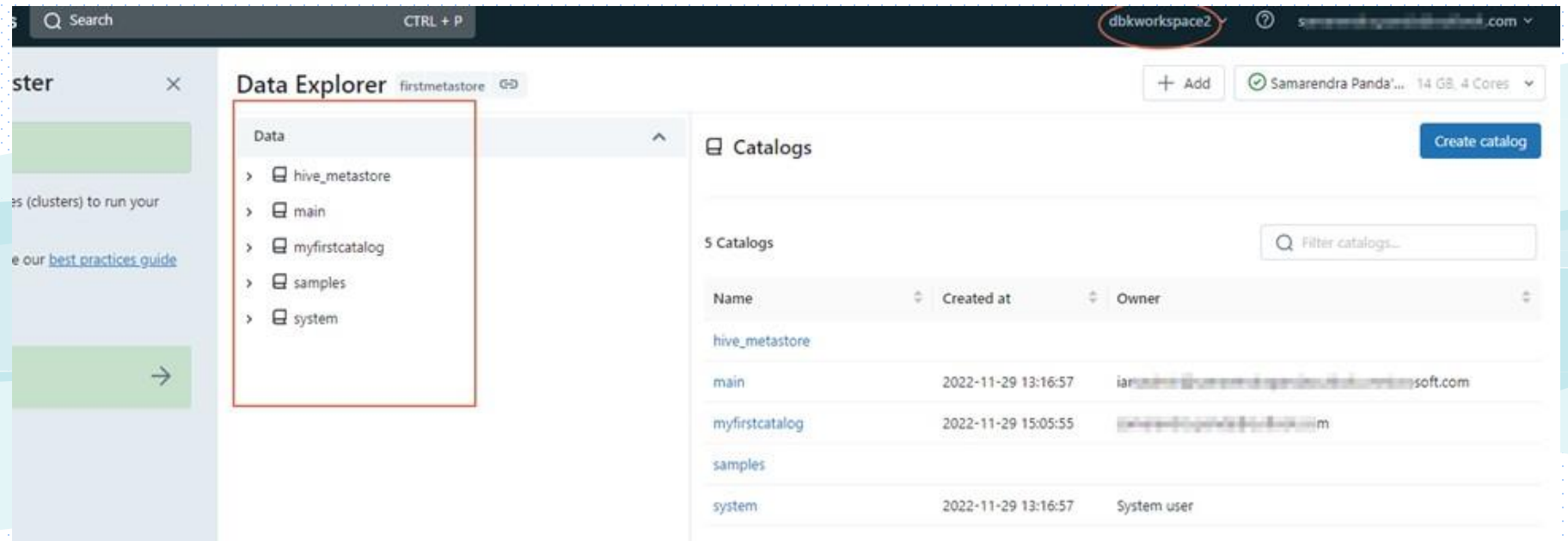
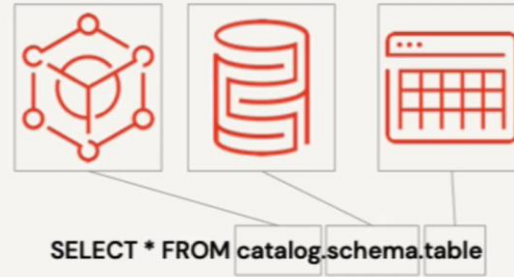
Databricks Unity Catalog

Three-layer namespace

Traditional two-layer namespace



Three-layer namespace with Unity Catalog



Screenshot of the Databricks Unity Catalog interface. The top navigation bar shows the workspace name `dbkworkspace2` and the user `Samarendra Panda` with a dropdown menu showing `14 GB, 4 Cores`. The left sidebar contains a 'Data Explorer' panel for the `firstmetastore` catalog, listing the following data sources:

- hive_metastore
- main
- myfirstcatalog
- samples
- system

The main panel displays a table of catalogs:


Name	Created at	Owner
hive_metastore		
main	2022-11-29 13:16:57	ian.panda@databricks.com
myfirstcatalog	2022-11-29 15:05:55	Samarendra Panda
samples		
system	2022-11-29 13:16:57	System user

Data LakeHouse

Strategie vincenti

Implementazione e Best Practice

Tre Asset principali:

- Notebooks
 - Schemas
 - Workflows
- 

Notebooks

1. Source To Bronze
 2. Bronze To Silver
 3. Bronze To Silver Special
 4. Silver To Gold
 5. Aggregate Views
- Utils













Workspace > Shared >

LakeHouse ☆



Share

Add ▾

Name ▾	Type	Owner	Created	
 Utils	Folder	Andrea Bergonzi	2/21/2023	
 05_AggregateViews	Folder	Andrea Bergonzi	2/21/2023	
 03_BronzeToSilverSpecial	Folder	Andrea Bergonzi	2/21/2023	
 02_BronzeToSilver	Folder	Andrea Bergonzi	2/21/2023	
 04_SilverToGold	Notebook	Andrea Bergonzi	2/21/2023	
 01_SourceToBronze	Notebook	Andrea Bergonzi	2/21/2023	

01_SourceToBronze

Python ▾

File Edit View Run Help [Last edit was 45 days ago](#) [Provide feedback](#)

▶ Run all

● Unknown ▾

📅 Schedule

Share

end_date

1900-01-01

start_date

1900-01-01



Cmd 1



Python



```
1 %run /Shared/LakeHouse/Utils/Functions
```

OK

```
<function __main__.extract_dates(xml, position)>
```

Command took 1.50 seconds -- by francesco.capuani@dataskills.it at 25/5/2023, 10:18:53 on Francesco Capuani's Cluster

Cmd 2

```
1 %sql
2 USE CATALOG `bigdata-catalog-prod`
```

OK

01_SourceToBronze

Python ▾

File Edit View Run **Help** Last edit was 45 days ago Provide feedback

▶ Run all

● Unknown ▾

📅 Schedule

Share

end_date

1900-01-01

start_date

1900-01-01



```
1  data_flows = {
2      "brand" : "dim"
3      , "campaign" : "dim_special"
4      , "category" : "dim"
5      , "company" : "dim"
6      , "deal" : "dim"
7      , "advertiser" : "dim"
8      , "lineitem" : "dim"
9      , "network" : "dim"
10     , "placement" : "dim"
11     , "zone" : "dim"
12     , "execution" : "fact"
13 }
14
15 environment = "prod" # is the env from which data are downloaded
16 FirstLoad = False
17 end_date = dbutils.widgets.get("end_date")
18 start_date = dbutils.widgets.get("start_date")
19 if start_date == "1900-01-01" or end_date == "1900-01-01":
20     yesterday = (datetime.today() - timedelta(days = 1)).strftime('%Y-%m-%d')
21     today = datetime.today().strftime('%Y-%m-%d')
22 else:
23     yesterday = start_date
24     today = end_date
```












Workspace > Shared > LakeHouse >

02_BronzeToSilver ☆



Share

Add ▾

Name ▾	Type	Owner	Created	
 zone	Notebook	Andrea Bergonzi	2/21/2023	⋮
 placement	Notebook	Andrea Bergonzi	2/21/2023	⋮
 network	Notebook	Andrea Bergonzi	2/21/2023	⋮
 lineitem	Notebook	Andrea Bergonzi	2/21/2023	⋮
 execution	Notebook	Andrea Bergonzi	2/21/2023	⋮
 deal	Notebook	Andrea Bergonzi	2/21/2023	⋮
 company	Notebook	Andrea Bergonzi	2/21/2023	⋮
 category	Notebook	Andrea Bergonzi	2/21/2023	⋮
 brand	Notebook	Andrea Bergonzi	2/21/2023	⋮
 advertiser	Notebook	Andrea Bergonzi	2/21/2023	⋮
 _BronzeToSilverOrchestration	Notebook	Andrea Bergonzi	2/21/2023	⋮

placement

SQL ▾



File Edit View Run Help

[Last edit was 236 days ago](#)[Provide feedback](#)

Run all

Unknown ▾

Schedule

Share

Cmd 3

SQL



```
1 CREATE OR REPLACE VIEW stage.vw_placement_silver AS
2 SELECT
3   ID as bk_placement
4   ,Name
5   ,IDOpenRTB
6   ,Type
7   ,DSPlatform
8   ,DSID
9   ,ADServerID
10  ,IDZone
11  ,IDPublisherCompany
12  ,IDVenueType
13  ,FloorPrice
14  ,Latitude
15  ,Longitude
16  ,Archived
17  ,IDPassback
18  ,VastFormat
19  ,OTS
20  ,Width
21  ,Height
22  ,MinDuration
23  ,MaxDuration
24  ,MaxBitrate
25  ,MaxSize
26  ,CreationDate
```

_BronzeToSilverOrchestration SQL ▾ ☆

File Edit View Run Help Last edit was 243 days ago Provide feedback

▶ Run all ● Unknown ▾ 📅 Schedule Share

Cmd 1

1 %run /Shared/LakeHouse/02_BronzeToSilver/advertiser

Out[5]: <function __main__.extract_dates(xml, position)>

OK

OK

Command took 8.58 seconds -- by andrea.bergonzi@dataskills.it at 28/2/2023, 16:15:36 on cluster prod bigdata download

Cmd 2

1 %run /Shared/LakeHouse/02_BronzeToSilver/brand

Out[14]: <function __main__.extract_dates(xml, position)>

OK

OK

Command took 10.32 seconds -- by andrea.bergonzi@dataskills.it at 28/2/2023, 16:15:36 on cluster prod bigdata download

Cmd 3

1 %run /Shared/LakeHouse/02_BronzeToSilver/category

Out[23]: <function __main__.extract_dates(xml, position)>

OK

04_SilverToGold

Python



File Edit View Run Help Last edit was 44 days ago Provide feedback

Run all

Unknown

Schedule

Cmd 1

```
1 %run /Shared/LakeHouse/Utils/Functions
```

```
  _sqldf: pyspark.sql.dataframe.DataFrame
```

OK

```
Out[20]: <function __main__.extract_dates(xml, position)>
```

Command took 1.40 seconds -- by andrea.bergonzi@dataskills.it at 22/3/2023, 10:55:29 on cluster prod bigdata

Cmd 2

```
1 data_flows = {"brand" : "dim"  
2 , "campaign" : "dim_special"  
3 , "category" : "dim"  
4 , "company" : "dim"  
5 , "deal" : "dim"  
6 , "advertiser" : "dim"  
7 , "lineitem" : "dim"  
8 , "network" : "dim"  
9 , "placement" : "dim"  
10 , "zone" : "dim"  
11 , "execution" : "fact"}
```

Command took 0.14 seconds -- by andrea.bergonzi@dataskills.it at 22/3/2023, 10:55:29 on cluster prod bigdata

Cmd 4

Dim tables

Cmd 5

```
1 for flow in data_flows.keys():  
2     if data_flows[flow] in ("dim", "dim_special"):  
3         if data_flows[flow] != "dim_special":  
4             silver_flow = f"vw_{flow}" # If not dim_special silver layer is a view, otherwise a table  
5         else:  
6             silver_flow = flow  
7         transform_and_save_dimension(  
8             f"stage.{silver_flow}_silver",  
9             f"dim_{flow}",  
10            f"sk_{flow}",  
11            [f"bk_{flow}"],  
12            spark,  
13            ignore_changes_cols=None,  
14            has_history_table=True,  
15            load_date=None,  
16            first_call=False)
```







▶ (40) Spark Jobs

Command took 1.18 minutes -- by andrea.bergonzi@dataskills.it at 27/2/2023, 17:28:02 on cluster prod bigdata download

Workspace > Shared > LakeHouse >

05_AggregateViews ☆





[Share](#)[Add](#) ▾

Name ▾	Type	Owner	Created	
 Execution_perDayAndHour	Notebook	Andrea Bergonzi	2/21/2023	
 Execution_perDay	Notebook	Andrea Bergonzi	2/21/2023	
 _AggregateViewsOrchestration	Notebook	Andrea Bergonzi	2/21/2023	

Workspace > Shared > LakeHouse >

Utils ☆

[Share](#)[Add](#) ▾

Name ▾	Type	Owner	Created	
 Setup	Notebook	Andrea Bergonzi	2/21/2023	
 Functions	Notebook	Andrea Bergonzi	2/21/2023	

Data Assets: Schemas, Tables, Models

▼ bigdata-catalog-prod

- > crm_temp
- > default
- > dwh
- > information_schema
- > sap_temp
- > stage

Catalog Explorer

[Provide Feedback](#)

+ Add

Starter Warehouse dev

2XS ▼

Catalog

Type to filter

▼ dwh

> Tables (157)

> Models (6)

Catalogs

Filter ...

6 catalogs

Na... C... O...

██████████_Lakehouse ☆

Runs



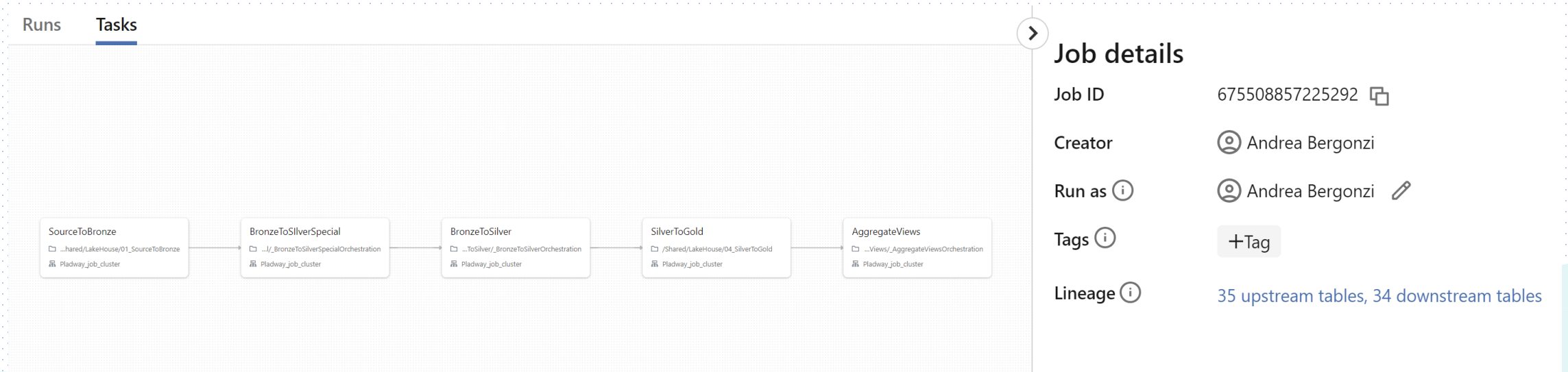
Job details

Git

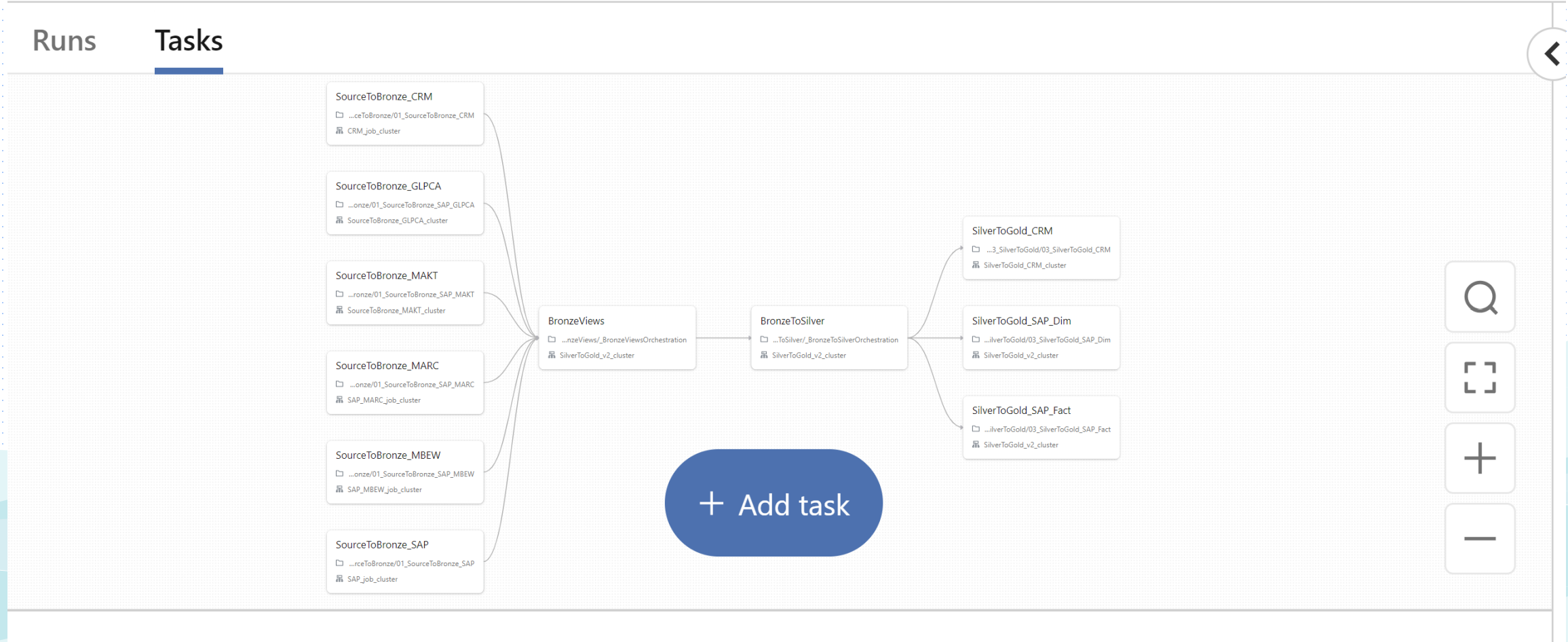
Schedules & Triggers

Compute


Workflow & Orchestration



Workflow & Orchestration



Tips & Tricks



Sulle tabelle

- **PARTITION BY**
- **OPTIMIZE** e **Z-ORDERING** dove possibile

OPTIMIZE ricompatta una serie file piccoli in file più grandi;

ZORDER colloca i dati in base a uno più campi nello stesso set di file

SQL

```
OPTIMIZE events
```

```
OPTIMIZE events WHERE date >= '2017-01-01'
```

```
OPTIMIZE events
```

```
WHERE date >= current_timestamp() - INTERVAL 1 day
```

```
ZORDER BY (eventType)
```

Sui DataFrame

- Evitare il **Partition Shuffling**

`df.cache()` dopo Wide Transformations

- Evitare il **Partition Skewing**

`df.coalesce()` o `df.repartition()` dopo Wide Transformations

- Evitare **API Throttling**

`df.mapPartitions()` per chiamate API

Sui Cluster

- Environment Variables
- Init Scripts

▼ Advanced options

Azure Data Lake Storage credential passthrough ?

☐ Enable credential passthrough for user-level data access

Spark Logging Init Scripts **JDBC/ODBC** SSH

Spark config ?

```
spark.databricks.cluster.profile singleNode
spark.master local[*, 4]
spark.databricks.delta.preview.enabled true
```

Environment variables ?

```
ENV_TYPE=dev
```

```
1  #!/bin/bash
2  sudo apt-get update
3  sudo mkdir /usr/local/sap
4  sudo apt-get install unzip
5  sudo cp /dbfs/FileStore/nwrfc750P_11_70002752.zip /usr/local/sap/
6  sudo unzip /usr/local/sap/nwrfc750P_11_70002752.zip -d /usr/local/sap/
7  echo SAPNWRFC_HOME='/usr/local/sap/nwrfcsdk' >> /etc/environment
8  cd /etc/ld.so.conf.d/
9  touch nwrfcsdk.conf
10 sudo cp /dbfs/FileStore/nwrfcsdk.conf /etc/ld.so.conf.d/nwrfcsdk.conf
11 sudo ldconfig && ldconfig -p | grep sap
```



#37 PARMA 2023

Grazie!!!

Data Saturday #37 Feedback Form

