

# Delta Lake

Μάθημα: Επιχειρηματική Ευφυΐα και Ανάλυση Μεγάλων Δεδομένων

Αλεξάνδρα Σταθοπούλου

Σταύρος Σιώπης

Μαρία Σχοινάκη

Σοφία Παπαϊωάννου



DELTA LAKE

# Εισαγωγή στο Delta Lake



## Τι είναι;

Open - source storage framework

Βασίζεται στο Apache Spark 

Δημιουργήθηκε για να ενισχύσει τις δυνατότητες των data lakes και να καλύψει τις ελλείψεις τους.

**Ιδρύθηκε: 2017**

**Συνιδρυτές:** Ali Ghodsi και Matei Zaharia

**Εταιρεία:**



# Εταιρείες που χρησιμοποιούν την Delta Lake

---

Uber



NETFLIX

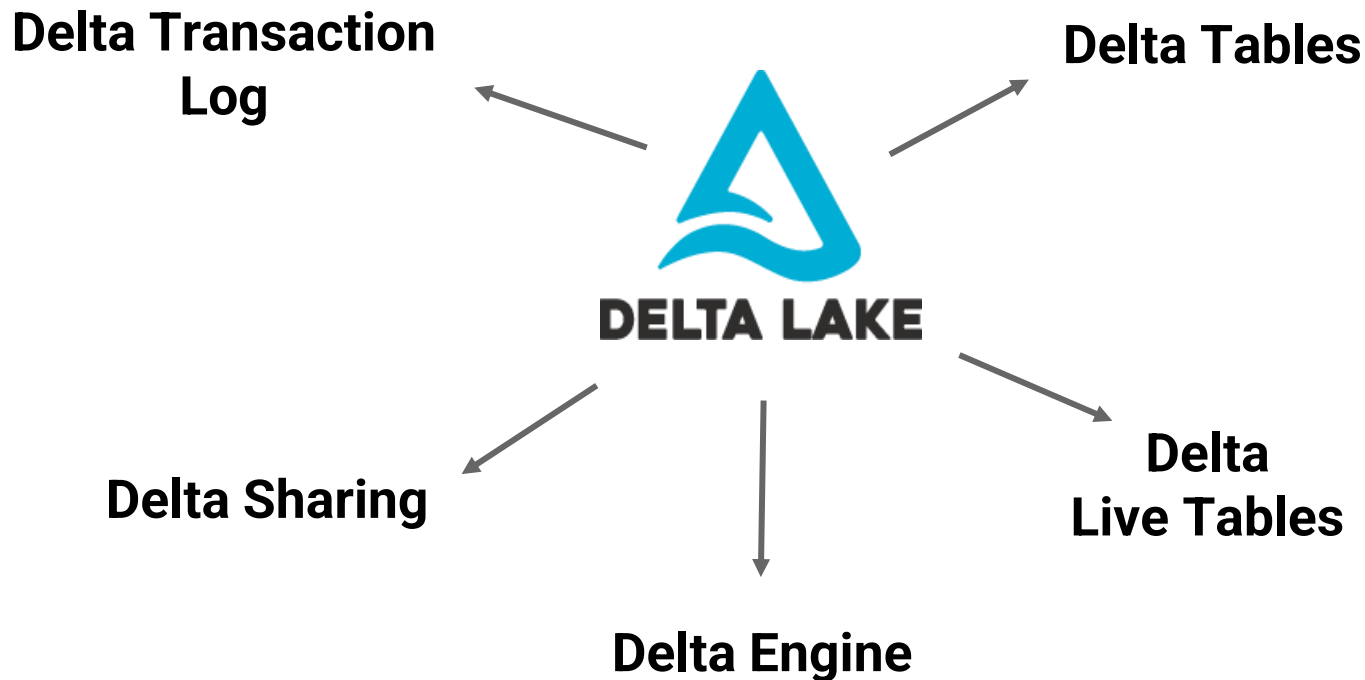


T Mobile



# Βασικά χαρακτηριστικά Delta Lake

---



# Γιατί υπάρχει ανάγκη; (1/2)

Ο μεγάλος όγκος ηλεκτρονικών δεδομένων που παράγονται καθημερινά στον σύγχρονο κυβερνοχώρο από ποικίλες πηγές και σε διαφορετικές μορφές δημιουργεί για τις επιχειρήσεις ανάγκες που οι συμβατικοί data lakes δεν μπορούν να καλύψουν.

## Ανάγκες:

### 1. Αξιοπιστία

Συνέπεια  
Δεδομένων



Ενημερώσεις



Ποιότητα



### 2. Ενοποίηση Batch και Streaming Δεδομένων

**Πρόβλημα:** Data lakes > μόνο batch επεξεργασία

**Λύση:** Delta lake > batch και streaming

### 3. Αποτελεσματική Διαχείριση Μεταδεδομένων

**Πρόβλημα:** Data lakes > αργή και μη επεκτάσιμη διαχείριση

**Λύση:** Delta lake > Logs για καταγραφή και διαχείριση αλλαγών

# Γιατί υπάρχει ανάγκη; (2/2)

Ο μεγάλος όγκος ηλεκτρονικών δεδομένων που παράγονται καθημερινά στον σύγχρονο κυβερνοχώρο από ποικίλες πηγές και σε διαφορετικές μορφές δημιουργεί για τις επιχειρήσεις ανάγκες που οι συμβατικοί data lakes δεν μπορούν να καλύψουν.

## 4. Ιστορικότητα και Time Travel

**Πρόβλημα:** Data lakes > ~~data versioning/ debugging~~

**Λύση:** Delta lake > Time Travel και Audit Trails

## 5. Απόδοση και Συμπίεση δεδομένων

**Πρόβλημα:** Data lakes > χρήση μη βελτιστοποιημένων αρχείων

**Λύση:** Delta lake > συγχώνευση αρχείων και Z-ordering

## 6. Συμβατότητα με Υπάρχουσες Τεχνολογίες



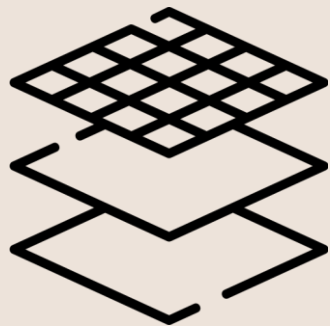
Apache spark Presto Apache Hive Azure Google Cloud

## 7. Εφαρμογές σε Πραγματικό Χρόνο

**Πρόβλημα:** Ανάγκη για real-time analytics



**Λύση:** Delta lake > διαχείρισης δεδομένων μέσω streaming



# Αρχιτεκτονική

# Διάγραμμα Δομής

## Delta Table Structure

- Parquet Files: Κύρια δεδομένα
- Delta Logs: Καταγραφή συναλλαγών (JSON)
- Metadata: Πληροφορίες σχήματος και αλλαγών



# Αρχιτεκτονική Επιπέδου Αποθήκευσης

Τα δεδομένα στο **Delta Lake** αποθηκεύονται σε αρχεία **Parquet**.

Οργάνωση της δομής αρχείων:

- Τα **Parquet** αρχεία περιέχουν τα κύρια δεδομένα.
- Τα **Delta Logs** αποθηκεύονται σε ειδικό φάκελο **\_delta\_log**.
- Τα **metadata** διατηρούν πληροφορίες σχήματος και ιστορικές εκδόσεις.

Παράδειγμα δομής φακέλων:

**/path/to/delta\_table/**

├── **part-0000-01.parquet**    **# Τα κύρια δεδομένα**

├── **part-0000-02.parquet**

├── **\_delta\_log/**                    **# Transaction Logs**

|    ├── **00000000000000000001.json**

|    ├── **00000000000000000002.json**

|    └── **...**

└── **metadata**                    **# Schema και πληροφορίες εκδόσεων**

# Συμβατότητα με Parquet

Το **Delta Lake** χρησιμοποιεί αρχεία **Parquet** ως το κύριο format αποθήκευσης δεδομένων. Το **Parquet** είναι ένα **columnar format** που προσφέρει:

**Συμπίεση δεδομένων:** Βελτιστοποίηση χώρου αποθήκευσης.

**Γρήγορη ανάγνωση και αναζήτηση δεδομένων:** Η columnar αρχιτεκτονική επιτρέπει την ταχύτερη ανάκτηση συγκεκριμένων στηλών δεδομένων.

**Υποστήριξη για μεγάλα δεδομένα:** Είναι ιδανικό για αναλύσεις σε **Big Data** περιβάλλοντα.

Παράδειγμα δομής αρχείων:

**/path/to/delta\_table/**

└── **part-0000-01.parquet**

└── **part-0000-02.parquet**

└── **...**

# Delta Logs

Τα **Delta Logs** είναι το βασικό χαρακτηριστικό που προσθέτει αξιοπιστία και συνέπεια στα δεδομένα.



Τι είναι τα **Delta Logs**:

- Πρόκειται για **JSON** αρχεία που παρακολουθούν όλες τις συναλλαγές που γίνονται στο **Delta Table**.
- Διατηρούν πληροφορίες για **Insert, Update, Delete** συναλλαγές και **Metadata** για σχήμα, partitions και ιστορικό εκδόσεων.

Λειτουργίες των **Delta Logs**:

- Υποστήριξη **ACID** συναλλαγών για εγγύηση συνέπειας των δεδομένων.
- Παρέχουν ιστορικό αλλαγών-εκδόσεων για το **Time Travel**.
- Διασφαλίζουν **data lineage**.

A - Atomicity  
C - Consistency  
I - Isolation  
D - Durability



Παράδειγμα δομής Delta Logs:

**/path/to/delta\_table/\_delta\_log/**

—— 00000000000000000001.json # Πρώτη συναλλαγή

—— 00000000000000000002.json # Δεύτερη συναλλαγή

—— ...

Κάθε νέο αρχείο JSON καταγράφει μια σειρά αλλαγών που προστέθηκαν ή αφαιρέθηκαν από τα αρχεία Parquet.

# Batch και Streaming Data

Το **Delta Lake** ενοποιεί **batch** και **streaming** δεδομένα στην ίδια αρχιτεκτονική:

## Batch Επεξεργασία:

- Ιδανική για μεγάλα ιστορικά δεδομένα.
- Παράδειγμα: Reporting, ανάλυση δεδομένων.

## Streaming Επεξεργασία:

- Real-time εισαγωγή και ανάλυση δεδομένων.
- Παράδειγμα: IoT, ανίχνευση συναλλαγών.

Παράδειγμα κώδικα:

### # Batch Processing

```
df = spark.read.format("delta").load("/path/to/table")
```

### # Streaming Processing

```
streaming_df =  
spark.readStream.format("delta").load("/path/to/table")
```

# Αποθήκευση Πληροφορίας Έκδοσης

Το **Delta Lake** διατηρεί ιστορικές εκδόσεις μέσω των **Delta Logs**:

## Versioning των Δεδομένων:

- Κάθε συναλλαγή δημιουργεί ένα νέο JSON Delta Log.
- Προηγούμενες εκδόσεις παραμένουν διαθέσιμες.

## Time Travel:

- Δυνατότητα πρόσβασης σε προηγούμενες εκδόσεις των δεδομένων.

Παράδειγμα Κώδικα (Time Travel):

```
SELECT * FROM delta.`/path/to/table` VERSION AS OF 5;
```



# Διαδικασίες Διαχείρισης Δεδομένων

# Εισαγωγή & Ενημέρωση Δεδομένων (UPSERT)

Υλοποίηση Insert, Update, Delete

Το **Delta Lake** υποστηρίζει **UPSERT** (Insert, Update, Delete) λειτουργίες μέσω **MERGE INTO SQL** και **Spark API**.

## 1. Insert (Εισαγωγή Δεδομένων)

- Προσθήκη νέων δεδομένων στον πίνακα.
- Παράδειγμα.

```
new_data.write.format("delta").mode("append").save("/path/to/delta_table")
```

## 1. Update (Ενημέρωση Δεδομένων)

- Χρήση **MERGE INTO** για ενημέρωση συγκεκριμένων δεδομένων.
- Παράδειγμα σε SQL.

```
MERGE INTO delta_table AS target
```

```
USING updates AS source
```

```
ON target.id = source.id
```

```
WHEN MATCHED THEN UPDATE SET target.value = source.value
```

```
WHEN NOT MATCHED THEN INSERT (id, value) VALUES (source.id, source.value);
```

## 1. Delete (Διαγραφή Δεδομένων)

- Διαγραφή εγγραφών με βάση συγκεκριμένα κριτήρια.
- Παράδειγμα.

```
deltaTable.delete("date < '2024-12-19'")
```

# Δυνατότητες Συμπίεσης Δεδομένων

Πώς χρησιμοποιούνται Μέθοδοι  
Συμπίεσης στο Delta Lake

Το **Delta Lake** χρησιμοποιεί συμπίεση δεδομένων για αποδοτική αποθήκευση και βελτίωση απόδοσης:

## 1. Columnar Compression (Parquet):

- Τα δεδομένα αποθηκεύονται σε columnar format, επιτρέποντας συμπίεση κατά στήλη.
- Συμπίεση σε μορφές όπως Snappy, GZIP, ή LZ4.

## 1. Διαδικασία Συμπίεσης:

- Συγχώνευση μικρών αρχείων σε μεγαλύτερα αρχεία για βελτίωση απόδοσης.

Παράδειγμα:

**OPTIMIZE delta.`/path/to/table` ZORDER BY  
(column\_name);**

## 1. Οφέλη Συμπίεσης:

- Μείωση του χώρου αποθήκευσης.
- Ταχύτερη ανάγνωση και αναζήτηση δεδομένων.



# Δυνατότητες Διαγραφής Δεδομένων

Πώς γίνεται η Διαγραφή  
Δεδομένων και Ενημερώσεων

Η διαγραφή δεδομένων στο **Delta Lake** γίνεται με τη χρήση **DELETE SQL** ή **Spark API**.

## 1. Delete Statement

- Διαγραφή συγκεκριμένων εγγραφών με βάση κριτήρια.

Παράδειγμα:

**DELETE FROM delta\_table WHERE condition = 'value';**

## 1. Διαδικασία Εφαρμογής:

- Το **Delta Lake** δεν διαγράφει φυσικά τα αρχεία, αλλά καταγράφει τις αλλαγές στα **Delta Logs**.
- Η πραγματική διαγραφή ολοκληρώνεται κατά τη διαδικασία **VACUUM**.

Παράδειγμα Vacuum:

**VACUUM delta.`/path/to/delta\_table` RETAIN 7 HOURS;**

## 1. Πλεονεκτήματα

- Ασφάλεια μέσω αναστρέψιμων διαγραφών (Time Travel).
- Συμβατότητα με μεγάλους όγκους δεδομένων.

# Συμπίεση και Αποδοτικότητα Αποθήκευσης

Διαδικασίες Καθαρισμού και  
Βελτιστοποίησης

Το **Delta Lake** βελτιστοποιεί την αποθήκευση δεδομένων μέσω διαδικασιών καθαρισμού και βελτιστοποίησης.

## 1. Διαδικασία Καθαρισμού (VACUUM)

- Αφαιρεί αρχεία που δεν χρησιμοποιούνται για διατήρηση χώρου αποθήκευσης.
- **Προσοχή:** **Time Travel** επηρεάζεται μετά το **VACUUM**.

## 1. Βελτιστοποίηση Αρχείων (OPTIMIZE)

- Συγχώνευση μικρών αρχείων σε μεγαλύτερα για καλύτερη απόδοση αναζήτησης.
- Παράδειγμα.

**OPTIMIZE delta.`/path/to/delta\_table`;**

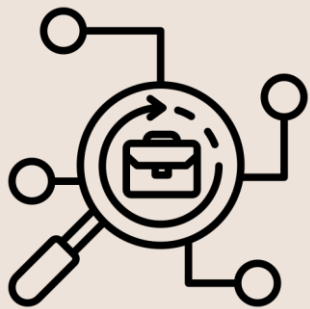
## 1. Z-Ordering

- Τα δεδομένα αναδιατάσσονται ώστε να βελτιστοποιούνται οι αναζητήσεις σε συγκεκριμένες στήλες.
- Παράδειγμα.

**OPTIMIZE delta.`/path/to/delta\_table` ZORDER BY (column\_name);**

## 1. Οφέλη Βελτιστοποίησης

- Μείωση κόστους αποθήκευσης.
- Ταχύτερη απόδοση ερωτημάτων και αναλύσεων.
- Συνεπής εμπειρία σε μεγάλα datasets.



# Παράδειγμα χρήσης

# Παράδειγμα σε Python: Δημιουργία Πίνακα στο Delta Lake (1/3)

Η δημιουργία ενός Delta Table από ένα DataFrame με χρήση PySpark

SparkSession δημιουργήθηκε επιτυχώς με υποστήριξη Delta Lake!

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Test Spark") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .config("spark.jars.packages", "io.delta:delta-core_2.12:2.4.0") \
    .getOrCreate()

data = [(1, "Maria", 3210191), (2, "Sophia", 3210157), (3, "Alexandra", 8190156)]
columns = ["id", "name", "student_id"]

df = spark.createDataFrame(data, columns)
df.show()

# Path
delta_path = "/content/delta-table"

# Save as Delta Table
df.write.format("delta").mode("overwrite").save(delta_path)
```

```
+---+-----+-----+
| id|   name|student_id|
+---+-----+-----+
|  1|  Maria|   3210191|
|  2| Sophia|   3210157|
|  3|Alexandra|  8190156|
+---+-----+-----+
```

## Βήματα:

1. Δημιουργία **SparkSession** με υποστήριξη **Delta Lake**.
1. Δημιουργία ενός **DataFrame** με τα δεδομένα.
1. Αποθήκευση ως Delta Table.

# Παράδειγμα σε Python: Δημιουργία Πίνακα στο Delta Lake (2/3)

Εισαγωγή νέων δεδομένων σε έναν υπάρχοντα Delta Table μέσω append mode

```
# New Data
new_data = [(4, "Stavros", 8190320)]
new_df = spark.createDataFrame(new_data, columns)

# Show the DataFrame
new_df.show()

# Append the New Row
new_df.write.format("delta").mode("append").save(delta_path)
```

```
+---+-----+-----+
| id|  name|student_id|
+---+-----+-----+
|  4|Stavros|   8190320|
+---+-----+-----+
```

## Βήματα:

1. Δημιουργία νέων δεδομένων σε **DataFrame**.
1. Εισαγωγή δεδομένων στον υπάρχοντα **Delta Table**.

# Παράδειγμα σε Python: Δημιουργία Πίνακα στο Delta Lake (3/3)

Ανάγνωση δεδομένων από έναν Delta Table και εκτέλεση ερωτημάτων

```
# Path to the Delta Table
delta_path = "/content/delta-table"

# Reading the Delta Table
df_read = spark.read.format("delta").load(delta_path)

# Displaying the Delta Table contents
df_read.show()

# Query
df_read.filter("student_id == 8190320").show()

print("Data were correct read and filtered.")
```



```
↔ +---+-----+-----+
  | id|    name|student_id|
+---+-----+-----+
  | 1|   Maria|   3210191|
  | 2|  Sophia|   3210157|
  | 3| Alexandra|  8190156|
  | 4|  Stavros|  8190320|
+---+-----+-----+

+---+-----+-----+
  | id|    name|student_id|
+---+-----+-----+
  | 4|  Stavros|  8190320|
+---+-----+-----+

Data were correct read and filtered.
```

Βήματα:

1. Ανάγνωση του **Delta Table** με `read.format("delta")`.
1. Εκτέλεση ερωτημάτων και εμφάνιση αποτελεσμάτων.



# Τεχνολογίες

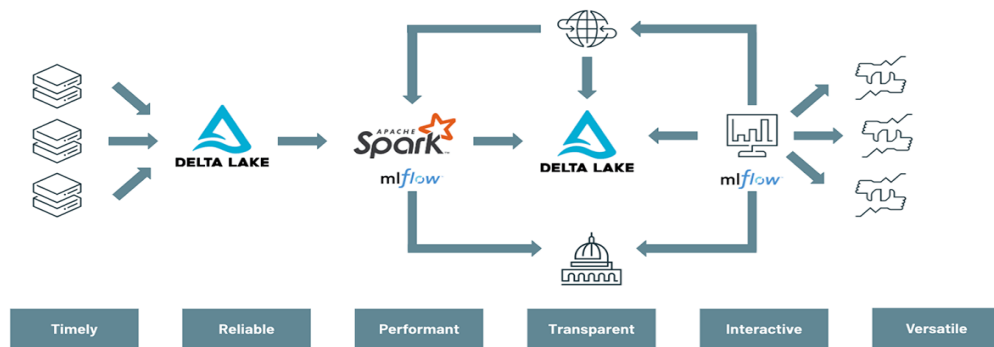
# Apache Spark και Delta Lake

## Αλληλεπίδραση:

Το Delta Lake λειτουργεί ως storage layer που κάθεται πάνω στο Apache Spark. Χρησιμοποιεί τις δυνατότητες του Spark για την επεξεργασία δεδομένων, όπως structured streaming και batch processing.

## Πλεονεκτήματα:

- ACID Συναλλαγές: Διασφάλιση συνέπειας και ακεραιότητας δεδομένων.
- Ενοποίηση Streaming & Batch: Διαχείριση ροής (streaming) και παρτίδας (batch) δεδομένων σε ενιαία pipeline.
- Βελτιστοποίηση Ερωτημάτων: Τεχνικές όπως *Z-order indexing* και *data skipping* βελτιώνουν την ταχύτητα αναζητήσεων.





# Delta Lake σε συνδυασμό με Εργαλεία Μηχανικής Μάθησης

## Αλληλεπίδραση

Το Delta Lake παρέχει **αξιόπιστα datasets** με λειτουργίες όπως το **time travel**, διευκολύνοντας τα πειράματα σε Machine Learning και Data Science.

## Πλεονεκτήματα:

- Αξιοπιστία Δεδομένων: Ακριβή και συνεπή δεδομένα για την εκπαίδευση μοντέλων.
- Χρονική Αναδρομή (Time Travel): Πρόσβαση σε προηγούμενες εκδόσεις δεδομένων για αναπαραγωγές και διορθώσεις.
- Ενσωμάτωση με Εργαλεία: Υποστηρίζει πλατφόρμες όπως MLlib, TensorFlow, και MLflow.

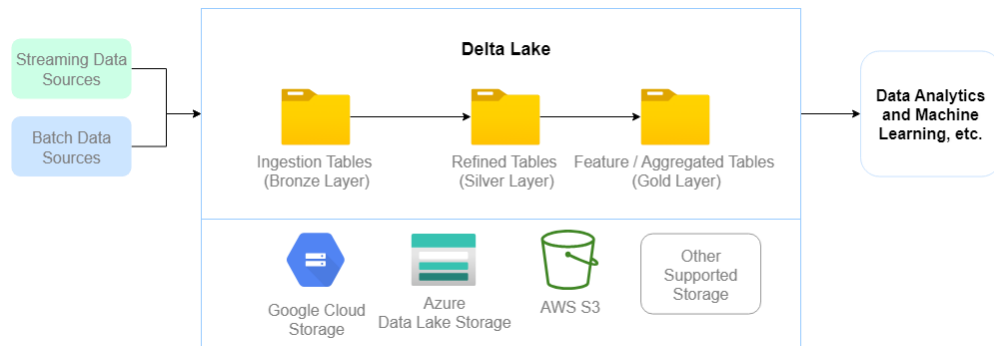
# Υποστήριξη Cloud-Native

## Αλληλεπίδραση:

Το Delta Lake υποστηρίζει **cloud platforms** όπως AWS, Azure και Google Cloud χρησιμοποιώντας αποθηκευτικά συστήματα όπως S3, ADLS, και GCS.

## Πλεονεκτήματα:

- Απεριόριστη Κλιμάκωση: Το cloud εξασφαλίζει απεριόριστη αποθήκευση και επεξεργασία.
- Απλοποίηση Υποδομών: Η ενσωμάτωση με cloud services μειώνει το κόστος ανάπτυξης υποδομών.
- Συνεργατικότητα: Δυνατότητες όπως το Delta Sharing διευκολύνουν τη διαμοίραση δεδομένων μεταξύ οργανισμών



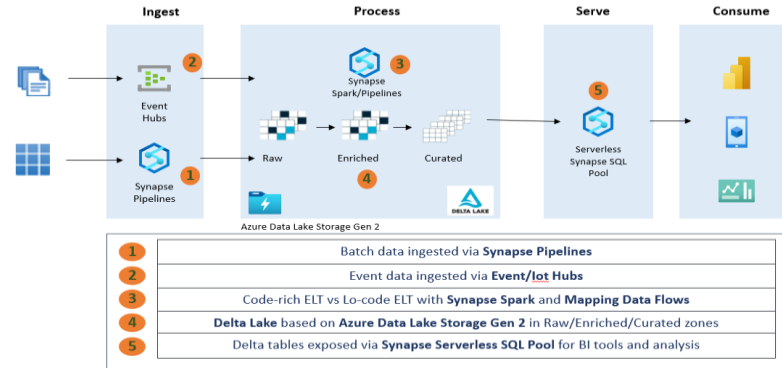
# Υποστήριξη SQL

## Αλληλεπίδραση:

Το Delta Lake παρέχει SQL συμβατότητα μέσω εργαλείων όπως Presto, Trino, και Databricks SQL, διευκολύνοντας την πρόσβαση δεδομένων από αναλυτές.

## Πλεονεκτήματα:

- Ευκολία Χρήσης: Οι αναλυτές μπορούν να χρησιμοποιήσουν τις υπάρχουσες γνώσεις SQL χωρίς να απαιτούνται νέες δεξιότητες.
- Ενσωμάτωση BI Εργαλείων: Σύνδεση με Tableau και Power BI για δημιουργία οπτικοποιήσεων.
- Πολύπλοκες Αναλύσεις: Υποστήριξη για σύνθετες λειτουργίες, όπως joins και aggregations



Delta Lake with Azure Synapse Analytics



# Βιομηχανίες

# Ο Τομέας της Ενέργειας



**Πρόκληση:** Επεξεργασία δεδομένων από αισθητήρες και υποδομές.

**Λύση:** Χρήση Delta Lake για ανάλυση real-time και ιστορική ανάλυση με time travel. Χρήση IoT για Προγνωστική Συντήρηση.

**Αποτέλεσμα:** Αύξηση της αποδοτικότητας και της αξιοπιστίας των διαδικασιών.

## Παράδειγμα:

Η GE Aviation:

Επεξεργάζεται δεδομένα από αισθητήρες κινητήρων αεροσκαφών.

Χρησιμοποιεί ανάλυση real-time και time travel.

Ως αποτέλεσμα, προβλέπει σφάλματα και μειώνει τον χρόνο εκτός λειτουργίας.

# Χρηματοοικονομική Βιομηχανία



**Πρόκληση:** Αξιόπιστα δεδομένα για ανίχνευση απάτης, συμμόρφωση με κανονισμούς και εξατομικευμένες υπηρεσίες.

## 1. Ανίχνευση Απάτης σε Πραγματικό Χρόνο

**Λύση:** Χρήση Delta Lake για συγχώνευση ροών δεδομένων και ιστορικών δεδομένων.

**Αποτέλεσμα:** έγκαιρη ανίχνευση ανωμαλιών.

### Παράδειγμα:

Η PayPal χρησιμοποιεί το Delta Lake για real-time ανίχνευση απάτης σε δεδομένα petabytes.

## 2. Κανονιστική Συμμόρφωση και Αναφορές

**Λύση:** Καταγραφή ιστορικών δεδομένων και μετασχηματισμών

**Αποτέλεσμα:** Εύκολη δημιουργία ρυθμιστικών αναφορών.

### Παράδειγμα:

Ο οργανισμός Senses αναβάθμισε τα συστήματα EDW με Delta Lake.

## 3. Ανάλυση Κινδύνου και Επενδύσεων

**Λύση:** Επεξεργασία μεγάλων ποσοτικών δεδομένων real-time

**Αποτέλεσμα:** επιτάχυνση των επενδυτικών αποφάσεων, ενίσχυση της ανταγωνιστικότητας.



**Πρόκληση:** Διαχείριση δομημένων και μη δομημένων δεδομένων (ιατρικές εικόνες, σημειώσεις γιατρών, δεδομένα από αισθητήρες).

## 1. Κλινική Ανάλυση και Διάγνωση

**Λύση:** Delta Lake για συνδυασμό δεδομένων από EHRs(Ηλεκτρονικούς Φακέλους Ασθενών)) και εικόνες ασθενών.

**Παράδειγμα:** Ενσωμάτωση machine learning για ακριβείς διαγνώσεις.

## 2. Διαχείριση Ροών Εργασίας

**Λύση:** Οι δυνατότητες ACID του Delta Lake διασφαλίζουν τη συνοχή των δεδομένων,

**Αποτέλεσμα:** Βελτιστοποίηση της ροής εργασιών και της εμπειρίας των ασθενών με ανάλυση σε πραγματικό χρόνο.

# Βιομηχανία Λιανικής



**Πρόκληση:** Εξατομικευμένες προτάσεις και επεξεργασία τεράστιων όγκων clickstream δεδομένων.

## 1. Διαχείριση Εφοδιαστικής Αλυσίδας:

**Λύση:** Ανάλυση δεδομένων για παρακολούθηση αποθεμάτων, προγνωστική παραγγελία και βελτιστοποίηση διανομής.

## 2. Clickstream Analysis:

**Λύση:** Ανάλυση δεδομένων πλοήγησης χρηστών για βελτιστοποίηση προϊόντων και κατανόηση συμπεριφοράς.



# Βιομηχανία Τηλεπικοινωνιών



**Πρόκληση:** Ανάλυση ποιότητας υπηρεσιών και ανίχνευση κακόβουλης δραστηριότητας στα δίκτυα.

## 1. Παρακολούθηση SLA:

**Λύση:** Επεξεργασία δεδομένων real-time

**Αποτέλεσμα:** Διασφάλιση της συμμόρφωσης με συμφωνίες επιπέδου υπηρεσίας (SLA).

## 2. Ανίχνευση Ασφάλειας:

**Λύση:** Ανάλυση διαδικτυακών δεδομένων για

**Αποτέλεσμα:** Εντοπισμός κακόβουλης δραστηριότητας.

## 3. Βελτιστοποίηση Δικτύου:

**Λύση:** Χρήση predictive analytics για την προσαρμογή υποδομών δικτύου στις ανάγκες χρηστών.



# Ανταγωνιστές και Πλεονεκτήματα

# Ανταγωνισμός & Συγκρίσεις

Οι βασικοί ανταγωνιστές της **Delta Lake**:

## Έμμεσος Ανταγωνισμός:

**Data Lakes**, που συγκεντρώνουν αρχεία σε διάφορα format και τύπους δεδομένων χωρίς έλεγχο εκδόσεων ή τήρηση σχήματος, με αποτέλεσμα να γίνονται δύσχρηστοι.



## Άμεσος Ανταγωνισμός:

Open-source έργα που υποστηρίζουν αρχιτεκτονικές Lakehouse, συνδυάζοντας χαρακτηριστικά data lakes και data warehouses. Παραδείγματα: **Apache Iceberg**, **Apache Hudi** και **Alluxio**.





VS



- Ευέλικτη και χαμηλού κόστους αποθήκευση δεδομένων
- Δεδομένα σε πολλαπλά formats και πηγές
- Υποστήριξη διαφορετικών εργασιών (π.χ. Analytics, Data science κ.λπ.)
- Ευελιξία στις ερωτήσεις, χωρίς προκαθορισμένα ερωτήματα

- Αξιοπιστία, ταχύτητα, φιλικότητα προς προγραμματιστές
- Υποστήριξη αλλαγών & schema enforcement (αποφυγή αλλοίωσης πινάκων)
- Μεταδεδομένα αρχείων & αρχείο αλλαγών για ταχύτερες ερωτήσεις
- Εύκολες λειτουργίες δεδομένων (διαγραφή, μετονομασία, κατάργηση γραμμών κ.λπ.)

# ICEBERG



## VS



## DELTA LAKE

- Υποστήριξη πολλαπλών  
Spark, Flink κ.λπ.



- **Snapshots** για αρχείο αλλαγών και συναλλαγές
- **ACID** (Atomicity, Consistency, Isolation & Durability)
- Αποθήκευση (μόνο Parquet, ORC, Avro)



- Spark- - - centric
- **Transaction log** για διαχείριση μεταδεδομένων και ιστορικό αλλαγών
- **ACID** (Atomicity, Consistency, Isolation & Durability)
- Αποθήκευση και formats (μόνο Parquet)








VS




- Δεν αποθηκεύει (ιστορικά) δεδομένα

- **Multi-engine** (Σε   )  
TensorFlow)


- Χωρίς **ACID**

- **Αποθήκευση:** Υποστηρίζει    formats

- **Transaction log** για διαχείριση μεταβολών και ιστορικό αλλαγών

-  - centric, SQL-based queries

- **ACID** (Atomicity, Consistency, Isolation & Durability)

- Αποθήκευση και formats (μόνο Parquet) 

# Πλεονεκτήματα

## (1/4)

A - Atomicity  
C - Consistency  
I - Isolation  
D - Durability



## ACID

- ❑ **Ατομικότητα (Atomicity):** Κάθε λειτουργία εκτελείται είτε πλήρως είτε καθόλου.
- ❑ **Συνοχή (Consistency):** Τα δεδομένα και η βάση παραμένουν σε έγκυρη κατάσταση.
- ❑ **Απομόνωση (Isolation):** Οι συναλλαγές εκτελούνται ανεξάρτητα η μία από την άλλη.
- ❑ **Αντοχή (Durability):** Τα δεδομένα αποθηκεύονται μόνιμα, ακόμα και σε περίπτωση αποτυχίας του συστήματος.

# Πλεονεκτήματα

## (2/4)

### Transaction logs

- ❑ Αποθήκευση **δεδομένων, μεταδεδομένων και file paths** σε ξεχωριστά logs
- ❑ **Ανάκτηση** δεδομένων
- ❑ **Καταγραφή αλλαγών** (Audit Trail)
- ❑ **Time Travel και Versioning**
- ❑ **Εξέλιξη και ευελιξία** σχήματος
- ❑ **Διασφαλίζει ACID** συναλλαγές





# Πλεονεκτήματα (3/4)

## Απόδοση



### ❑ File Skipping:

- **Μερική Ανάγνωση:** Ανάγνωση τμημάτων των δεδομένων
- **Αναγνώριση Άσχετων Δεδομένων:** Παράκαμψη αρχείων που δεν πληρούν τα κριτήρια του ερωτήματος

### ❑ Z-Ordering:

- **Συγκατοίκηση Παρόμοιων Δεδομένων:** Ομαδοποίηση παρόμοιων δεδομένων κοντά μεταξύ τους για αποτελεσματική ανάκτηση και file skipping.

### ❑ Real-Time επεξεργασία δεδομένων:

- **Ενσωμάτωση Batch και Streaming:** Ανάλυση παλαιών και νέων δεδομένων
- **Αποδοτική Ενημέρωση Δεδομένων:** Πρόσθεση νέων δεδομένων χωρίς διατάραξη των παλιών

# Πλεονεκτήματα

(4/4)

## Data Operations



- ❑ **INSERT/UPDATE/DELETE/MERGE INTO:** Υποστηρίζει λειτουργίες αλλαγής και συγχώνευσης δεδομένων
- ❑ **Upserts & Overwrites:** Διευκολύνει τις ενημερώσεις και αντικαταστάσεις δεδομένων

## Κόστος



- ❑ Μειώνει την εξάρτηση με **ακριβά συστήματα**
- ❑ **Αποδοτική ανάλυση μεγάλου όγκου δεδομένων** με χαμηλό κόστος
- ❑ Αποφεύγει διαφορετικές υποδομές **batching & streaming**



# Μελλοντικά βήματα:

- **Multi-Engine** υποστήριξη
- Υποστήριξη πολλαπλών **Formats** αποθήκευσης
- Εξελιγμένα **Ευρετήρια** (Indexing) για καλύτερες αναζητήσεις
- Επέκταση υποδομών υποστήριξης για **multi-cloud περιβάλλοντα**
- Επέκταση API και **υποστήριξη γλωσσών** όπως Flink και SQL-based frameworks
- Υποστήριξη για διαχείριση **πολιτικών ασφάλειας και ελέγχου πρόσβασης**
- Δημιουργία **οπτικού περιβάλλοντος** για διαχείριση δεδομένων.
- Ενσωμάτωση **μηχανισμών μάθησης (AI)** για ανάλυση μεγάλων δεδομένων

# Ερωτήσεις?

Ευχαριστούμε πολύ!



# Πηγές: (1/2)

Εισαγωγή: <https://delta.io/>

Αρχιτεκτονική: <https://awadrahman.medium.com/making-sense-of-databricks-delta-components-930746867743>

Τεχνολογίες Διαχείρισης: <https://delta.io/learn/tutorials>

ACID Transactions στο Delta Lake: <https://www.sparkcodehub.com/spark-delta-lake/delta-lake-acid-transactions>

Transaction Log στο Delta Lake: <https://www.databricks.com/blog/2019/08/21/diving-into-delta-lake-unpacking-the-transaction-log.html>

Real-Time Data Processing: <https://community.nasscom.in/communities/big-data-analytics/understanding-delta-lake-acid-transactions-and-real-world-use-cases>

Versioning στο Delta Lake: <https://www.bolddata.org/blog/delta-lake-transaction-log-guide>

# Πηγές: (2/2)

Ανταγωνιστές: <https://www.trustradius.com/products/delta-lake/competitors>

Data lakes V Delta lake: <https://delta.io/blog/delta-lake-vs-data-lake/>

Apache Iceberg V Delta lake: <https://www.datacamp.com/blog/iceberg-vs-delta-lake>

Alluxio V Delta lake: <https://documentation.alluxio.io/os-en>

Πλεονεκτήματα: <https://www.qlik.com/us/data-lake/delta-lake>

<https://dennyglee.com/2024/01/03/a-peek-into-the-delta-lake-transaction-log/>