**Wybór technologii składowania danych w Azure Databricks**

**Porównanie: Delta Lake vs Apache Iceberg**

Delta Lake i Apache Iceberg to nowoczesne, otwarte formaty tabel zoptymalizowane do pracy na dużych zbiorach danych w środowiskach lakehouse. Obie technologie wspierają transakcje ACID, wersjonowanie danych (Time Travel), ewolucję schematów oraz obsługę dużych wolumenów danych, ale różnią się podejściem, integracją i elastycznością.

| **Cecha** | **Delta Lake** | **Apache Iceberg** |
| --- | --- | --- |
| **Organizacja** | Linux Foundation (twórca: Databricks) | Apache Software Foundation (open source) |
| **Zastosowanie w Databricks** | Domyślny, w pełni zintegrowany format | Wspierany, ale mniej zintegrowany |
| **Silniki obliczeniowe** | Spark (głównie Databricks) | Spark, Flink, Trino, Hive, Presto |
| **Transakcje ACID** | Tak (transakcje dziennikowane w logu Delta) | Tak (snapshoty i metadane transakcyjne) |
| **Time Travel** | Tak (dostęp do wersji danych z przeszłości) | Tak (snapshot-based Time Travel) |
| **Ewolucja schematu** | Częściowa (wymaga przekształceń dla niektórych zmian) | Pełna (dynamiczne dodawanie/usuwanie kolumn, zmiana typu) |
| **Partycjonowanie** | Statyczne, zdefiniowane ręcznie | Dynamiczne i ukryte partycjonowanie |
| **Merge-on-read** | Nie (merge-on-write) | Tak (efektywne przy dużej ilości aktualizacji/usunięć) |
| **Zgodność z różnymi narzędziami** | Ograniczona głównie do Databricks i Spark | Szeroka – współpraca z AWS, GCP, Snowflake, Trino, Flink itd. |
| **Optymalizacja zapytań** | Z-Ordering, automatyczne kompaktowanie | Brak natywnego Z-Ordering, ale możliwe indeksowanie przez integracje |
| **Typowy format plików** | Parquet | Parquet, ORC, Avro |

**Kiedy wybrać Delta Lake:**

* Korzystasz głównie z Databricks i Apache Spark.
* Chcesz mieć pełną integrację z ekosystemem Azure (np. Unity Catalog, notebooks, MLflow).
* Potrzebujesz prostego i wydajnego strumieniowania danych (Structured Streaming).
* Zależy Ci na łatwym zarządzaniu jakością danych w architekturze Medallion.
* Twoja organizacja już korzysta z Databricks – Delta Lake jest domyślny i dobrze wspierany.

**Kiedy wybrać Apache Iceberg:**

* Działasz w środowisku wielochmurowym lub hybrydowym.
* Używasz narzędzi innych niż Spark (np. Trino, Flink, Presto).
* Zależy Ci na silnym zarządzaniu ewolucją schematu i partycjami.
* Potrzebujesz niezależności od jednego dostawcy (vendor lock-in).
* Planujesz tworzenie otwartych data products udostępnianych wielu konsumentom.

**Krytyka architektury medalionu:**

1. **Złożoność operacyjna**  
   Trzy warstwy wymagają zarządzania większą liczbą tabel, pipeline’ów i harmonogramów.
2. **Wydłużony czas dostępu do danych**  
   Dane użytkowe dostępne są dopiero w Gold – opóźnienia mogą być nieakceptowalne w systemach bliskich real-time.
3. **Duplikacja danych**  
   Te same dane fizycznie istnieją w różnych wersjach w każdej warstwie – zwiększa koszty i zużycie storage.
4. **Trudniejsze śledzenie pochodzenia (data lineage)**  
   Przy wielu transformacjach trudniej utrzymać przejrzysty łańcuch pochodzenia danych.
5. **Zbyt sztywna struktura**  
   Nie każdy przypadek biznesowy pasuje do modelu Bronze → Silver → Gold – wymusza zbędne etapy.
6. **Zwiększone koszty przechowywania i obliczeń**  
   Transformacje i przechowywanie danych w każdej warstwie generują dodatkowe koszty.
7. **Przerost formy nad treścią dla małych projektów**  
   W małych organizacjach lub projektach architektura może być zbyt ciężka i nieopłacalna.
8. **Problemy z aktualizacjami danych historycznych**  
   Gdy dane w Bronze się zmieniają, potrzebna jest pełna rekalkulacja wyższych warstw – nie zawsze jest to proste.
9. **Trudność w testowaniu pipeline’ów**  
   Wiele warstw oznacza więcej testów jednostkowych, integracyjnych i regresyjnych.
10. **Zbyt długa droga od surowych do analitycznych danych**  
    Może powodować frustrację u użytkowników oczekujących szybkiego dostępu.
11. **Sztuczne dzielenie procesów biznesowych**  
    Niektóre transformacje logicznie pasują do jednej warstwy, ale wymuszane są w innej.
12. **Potrzeba zaawansowanego orkiestratora**  
    Budowanie i utrzymanie sekwencji zależności między warstwami wymaga narzędzi typu Airflow, ADF, dbt.
13. **Trudności w rollbackach i poprawkach danych**  
    Przy błędzie w Silver lub Gold cofnięcie i poprawne przeliczenie wymaga dobrej wersjonowalności i koordynacji.
14. **Brak elastyczności dla danych strumieniowych**  
    Architektura z natury skupia się na batchowych przekształceniach.
15. **Możliwość przetwarzania tych samych danych wielokrotnie**  
    W Gold może dojść do sytuacji, gdzie dane są agregowane i filtrowane ponownie mimo wcześniejszej obróbki w Silver.
16. **Trudne zarządzanie wersjami modeli analitycznych**  
    Gdy modele są w warstwie Gold, aktualizacje logiki biznesowej mogą rozbić downstream.
17. **Nieintuicyjne dla nietechnicznych użytkowników**  
    Potrzeba zrozumienia, która warstwa zawiera jakie dane – bariera wejścia dla analityków.
18. **Potencjalna redundancja logiczna**  
    Często te same transformacje są implementowane w wielu miejscach (Silver i Gold) dla różnych potrzeb.
19. **Brak standaryzacji nazw i konwencji może prowadzić do chaosu**  
    Bez narzuconych standardów każda warstwa może być zarządzana inaczej przez różne zespoły.
20. **Nie każda organizacja ma wystarczające zasoby kadrowe**  
    Utrzymanie architektury medalionu wymaga zespołu DevOps/DataOps, czego często brakuje.