
Wyprawa po Świecie Big Data:
Od Chaosu do Wartości
Przewodnik po nowoczesnych architekturach,
technologiach i strategiach przetwarzania danych.

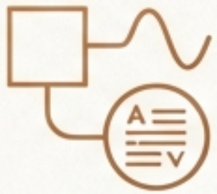
Krajobraz Big Data:

Krajobraz Big Data: Więcej Niż "Dużo Danych"

> „Big Data to ekosystem technologii (takich jak Spark, Hadoop, Kafka, Delta Lake), metod (w tym ETL, przetwarzanie strumieniowe i uczenie maszynowe) oraz podejść architektonicznych, które przekształcają chaotyczne strumienie informacji w praktyczne wnioski.”

Volume (Objętość)

Velocity (Szybkość)



Variety (Różnorodność)

Veracity (Wiarygodność)

Value (Wartość)

Kluczowe charakterystyki, które definiują wyzwania i możliwości w świecie danych.

5V w Praktyce: Od Koncepcji do Technologii

Volume (Objętość)

Ogromne ilości danych – od terabajtów do eksabajtów.
Przykład: Logi aplikacji generowane przez miliony użytkowników mobilnych.
Technologie: Przechowywane w systemach rozproszonych jak **HDFS**, **Amazon S3**, **Azure Data Lake Storage**, często zarządzane przez **Delta Lake** lub **Apache Iceberg**.

Velocity (Szybkość)

Analiza tysięcy transakcji na sekundę w systemach wykrywania oszustw.
Przykład: Systemy wykrywania oszustw analizujące tysiące transakcji na sekundę.
Technologie: **Apache Kafka**, **Flink**, **Spark Structured Streaming**.

Variety (Różnorodność)

Łączenie danych transakcyjnych (SQL), zdarzeń clickstream (JSON) i recenzji (tekst).
Przykład: Platforma e-commerce łączy dane transakcyjne (SQL), zdarzenia clickstream (JSON) i recenzje użytkowników (tekst).
Technologie: **Apache Spark**, **Parquet**, **ORC**, **Delta**.

Veracity (Wiarygodność)

Konieczność filtrowania dezinformacji i zapewnienia jakości danych.
Przykład: Analityka mediów społecznościowych musi filtrować dezinformację.
Technologie: Frameworki do walidacji jak **Great Expectations**, oczyszczanie w potokach **ETL/ELT**.

Value (Wartość)

Przekształcanie danych w decyzje biznesowe, np. optymalizacja cen.
Przykład: Optymalizacja cen w czasie rzeczywistym na platformach jak Uber.
Technologie: **Data Science**, **Machine Learning**, **analityka predykcyjna**.

Dwie Ścieżki Przetwarzania: Batch vs. Stream

Batch Processing

Przetwarzanie dużych wolumenów danych zebranych w określonym czasie, w dyskretnych, zaplanowanych cyklach (np. co godzinę, co noc).

Cechy kluczowe: Wysoka przepustowość, wyższa latencja, dane kompletne w momencie przetwarzania.

Przypadki użycia: Codzienne raporty biznesowe, miesięczne fakturowanie, transformacje w hurtowniach danych (dbt).

Stream Processing

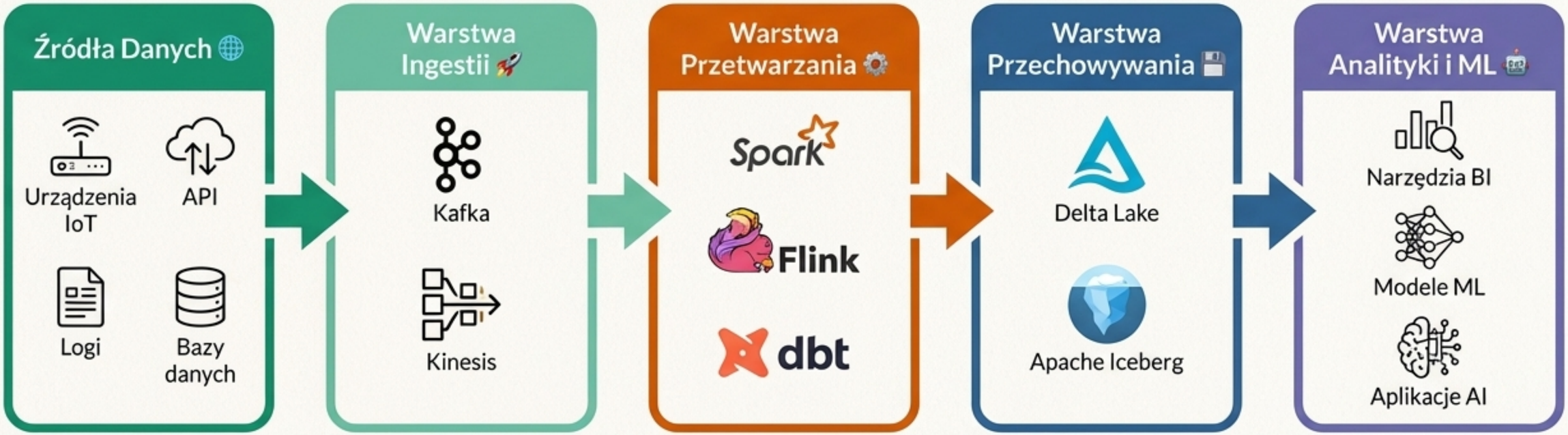
Obsługa ciągłych przepływów danych – zdarzenia są przetwarzane natychmiast po ich nadejściu, co umożliwia analizę w czasie rzeczywistym.

Cechy kluczowe: Niska latencja (milisekundy-sekundy), przetwarzanie rekord po rekordzie, wymaga zarządzania stanem.

Przypadki użycia: Wykrywanie oszustw, monitorowanie sensorów IoT, personalizacja w czasie rzeczywistym.

Aspekt	Batch Processing	Stream Processing
Typ danych	Historyczne, statyczne	Ciągłe, w czasie rzeczywistym
Latencja	Minuty do godzin	Milisekundy do sekund
Model	Okresowe zadania (jobs)	Ciągły przepływ zdarzeń
Złożoność	Łatwiejszy do wdrożenia	Trudniejszy (zarządzanie stanem)
Narzędzia	Spark, dbt, Hive, Airflow	Kafka, Flink, Spark Streaming

Anatomia Nowoczesnego Potoku Danych



Ten przepływ end-to-end ilustruje, jak nowoczesne systemy danych obsługują pozyskiwanie, transformację i analizę na dużą skalę, tworząc fundament dla podejmowania decyzji opartych na danych.

Ekosystem Big Data: Narzędzia i Ich Role

Warstwa Ingestii (Data Ingestion)

Responsibility: Zbieranie i przesyłanie strumieniowe danych z różnych źródeł.
Tools: **Apache Kafka**, Amazon Kinesis, Fivetran / Airbyte.

Warstwa Przechowywania (Data Storage)

Responsibility: Niezawodne i tanie przechowywanie ogromnych zbiorów danych.
Tools: **Amazon S3 / ADLS**, **Delta Lake**, Apache Iceberg / Hudi.

Warstwa Przetwarzania (Data Processing)

Responsibility: Obliczeniowy kręgosłup – transformacja, czyszczenie i analiza danych.
Tools: **Apache Spark**, **Apache Flink**, dbt.

Warstwa Zapytań (Data Query and Access)

Responsibility: Narzędzia do interaktywnych zapytań i eksploracji danych.
Tools: **Presto / Trino**, Databricks SQL, Snowflake.

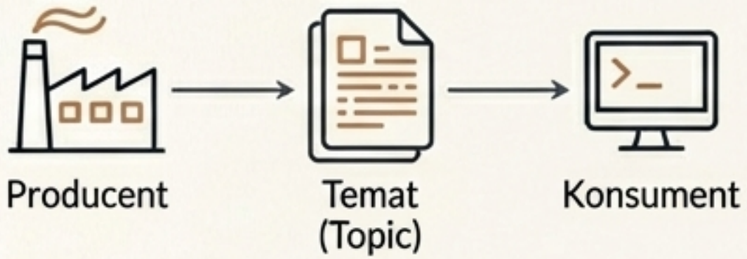
Warstwa ML i Analityki (Machine Learning & Analytics)

Responsib: Łączy inżynierię danych z AI, umożliwiaąc modelowanie predykcyjne.
Tools: **MLflow**, TensorFlow / PyTorch, Power BI / Tableau.

Apache Kafka: Centralny Układ Nerwowy Przesyłania Strumieniowego

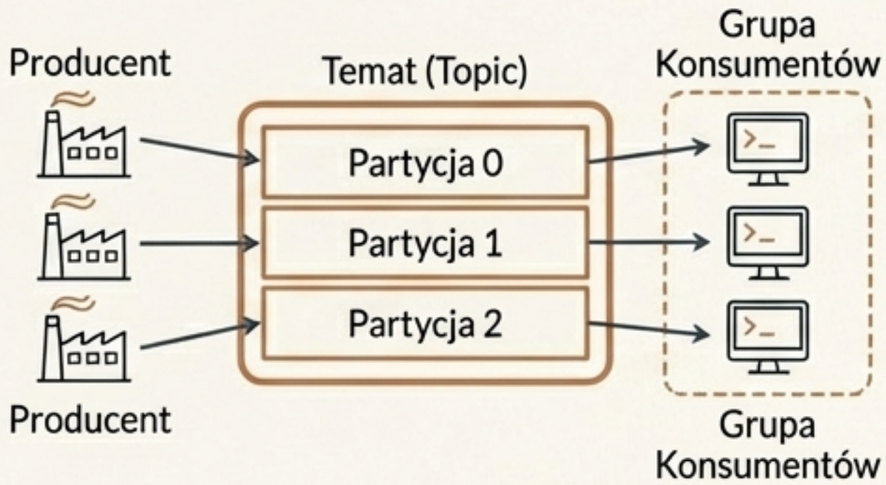
Rozproszona platforma streamingowa, która działa jako bufor dla zdarzeń w czasie rzeczywistym, oddzielając producentów od konsumentów.

Model Publish-Subscribe



Producenci wysyłają wiadomości (zdarzenia) do tematów. Konsumenci subskrybują tematy, aby je odczytać.

Partycje i Skalowalność



Partycje i Skalowalność
Tematy są dzielone na partycje, co pozwala na równoległe przetwarzanie i horyzontalne skalowanie.

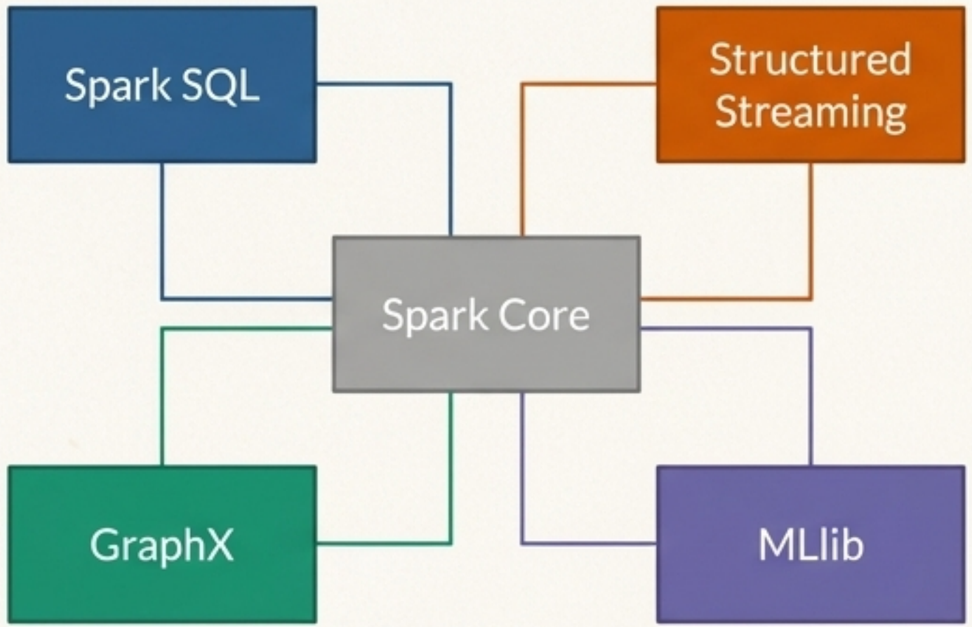
Offset
Unikalny identyfikator wiadomości w partycji.

Replikacja
Zapewnia odporność na awarie.

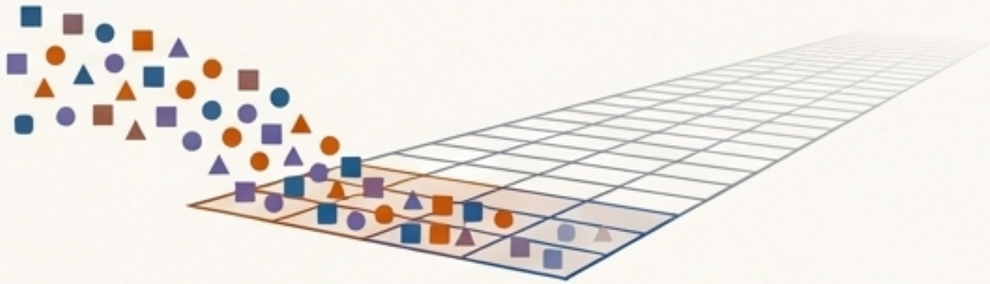
Retencja
Dane są przechowywane nawet po odczytaniu.

Apache Spark: Zunifikowany Silnik do Przetwarzania Danych

Spark to ujednolicony silnik analityczny, który obsługuje zarówno przetwarzanie wsadowe (batch), jak i strumieniowe (streaming) przy użyciu tego samego API.



Koncepcja: Nieskończona Tabela



Spark Structured Streaming traktuje strumień danych jako nieskończoną tabelę. Każde nowe zdarzenie to kolejny wiersz dodawany do tej tabeli.

Model Przetwarzania

Model: Micro-batching (przetwarzanie w małych, dyskretnych paczkach).

Ten sam kod może działać na danych historycznych (batch) i danych czasu rzeczywistego (stream), co znacznie upraszcza rozwój i utrzymanie.

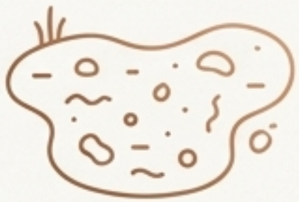
Ewolucja Przechowywania: Od Hurtowni Danych do Architektury Lakehouse

Data Warehouse

Dane: Strukturalne, po transformacji (Schema-on-write).

Zastosowanie: Business Intelligence, raportowanie.

Przykłady: Snowflake, Redshift, BigQuery.



Data Lake

Dane: Wszystkie typy – surowe, nieustrukturyzowane (Schema-on-read).

Zastosowanie: Eksploracja danych, trenowanie modeli ML.

Przykłady: Amazon S3, ADLS z Hadoop.

Lakehouse

Dane: Ujednolicone – wszystkie formaty w jednym miejscu.

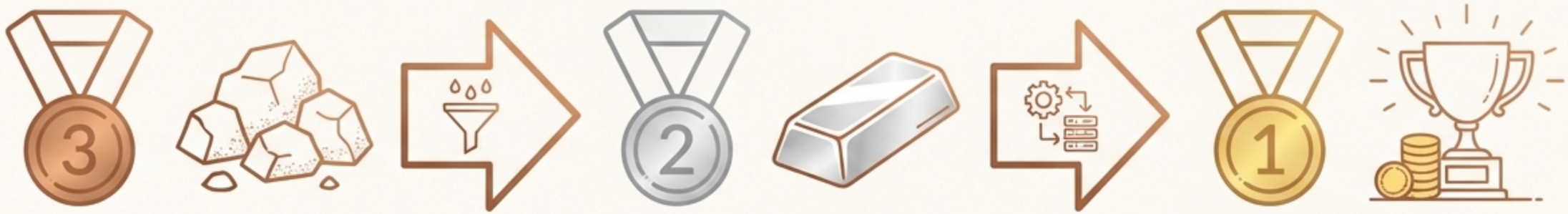
Zastosowanie: Zunifikowana analityka i AI na tych samych danych.

Przykłady: **Databricks Delta Lake, Apache Iceberg, Apache Hudi.**

Lakehouse łączy elastyczność Data Lake z funkcjami zarządzania i wydajnością Data Warehouse, wprowadzając **transakcje ACID**, zarządzanie schematem i szybkie zapytania SQL bezpośrednio na tanim storage'u obiektowym.

Utrzymanie Jakości i Porządku: Architektura Medallion

Architektura Medallion (opracowana przez Databricks) organizuje dane w **warstwy logiczne** (Bronze, Silver, **Gold**), które stopniowo poprawiają **jakość, użyteczność i ład danych.**



Warstwa BRONZE (Surowa)

- **Cel:** Surowe dane pobrane ze źródeł, bez transformacji.
- **Przykład:** Surowe zdarzenia JSON z Kafki zapisane do tabeli Delta.

Warstwa SILVER (Oczyszczona)

- **Cel:** Dane oczyszczone, zdeduplikowane, połączone z danymi referencyjnymi.
- **Przykład:** Usunięcie nulli, wzbogacenie ID użytkownika o dane z profilu.

Warstwa GOLD (Biznesowa)

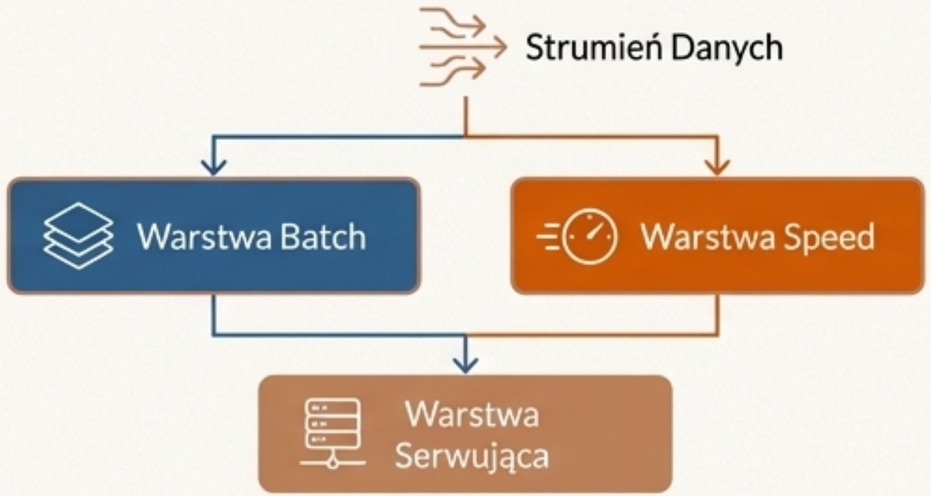
- **Cel:** Zagregowane zbiory danych gotowe do analizy biznesowej i ML.
- **Przykład:** Tabele zasilające dashboardy KPI, cechy dla modeli uczenia maszynowego.

Dwie Strategie Architektoniczne: Lambda vs. Kappa

Lambda Architecture

Idea

Łączy przetwarzanie wsadowe (batch) dla dokładności z przetwarzaniem strumieniowym (stream) dla wyników w czasie rzeczywistym.



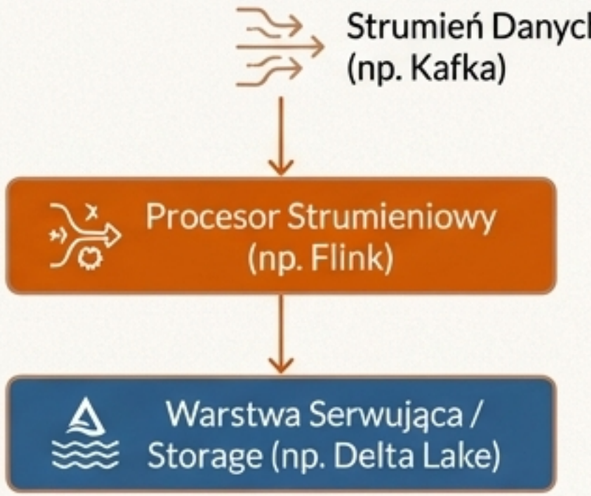
Wady

Duplikacja logiki w obu warstwach, trudność w utrzymaniu.

Kappa Architecture

Idea

Upraszcza model Lambda, eliminując warstwę batch. Wszystko jest przetwarzane w jednym potoku strumieniowym.



Zalety

Prostsza, spójna logika, prawdziwie architektura czasu rzeczywistego.

Lambda = Batch + Stream.
Kappa = Tylko Stream (z możliwością ponownego przetwarzania historii).

Nowoczesna Transformacja Danych: **Przejście z ETL na ELT**

Aspekt	ETL (Extract → Transform → Load)	ELT (Extract → Load → Transform)
Przepływ Procesu	Dane są ekstrahowane, transformowane na zewnętrznym silniku, a następnie ładowane do hurtowni.	Surowe dane są najpierw ładowane do data lake lub hurtowni, a następnie transformowane *na miejscu*.
Najlepsze dla	Tradycyjnych hurtowni danych z ograniczoną mocą obliczeniową.	Nowoczesnych architektur chmurowych (Snowflake, Databricks, BigQuery).
Narzędzia	Informatica, Talend, SSIS.	**dbt**, Spark SQL, Databricks workflows.
Zalety	Wczesna walidacja danych, ścisłe egzekwowanie schematu.	Lepsza skalowalność, tańszy storage, obsługa danych semi- i nieustrukturyzowanych.

Kluczowe Wnioski

Podejście ELT wykorzystuje moc obliczeniową nowoczesnych platform danych, co pozwala na większą **elastyczność** i i **skalowalność**. Dane są transformowane dopiero wtedy, gdy znany jest kontekst ich użycia.

Przykład Praktyczny

Przykład: W potoku **dbt + Snowflake** surowe dane lądują w schemacie 'raw', a dbt transformuje je w czyste modele analityczne bezpośrednio w hurtowni.

Radzenie Sobie z Rzeczywistością:
Obsługa Opóźnionych Danych

Problem

Definicja:
Opóźnione dane (late data) to zdarzenia, które docierają później niż oczekiwano na podstawie ich sygnatury czasowej (event time).

Konsekwencje:
Ignorowanie ich prowadzi do niekompletnych agregacji. Przechowywanie stanu w nieskończoność prowadzi do wycieków pamięci.

Rozwiązanie: Watermarking w Sparku

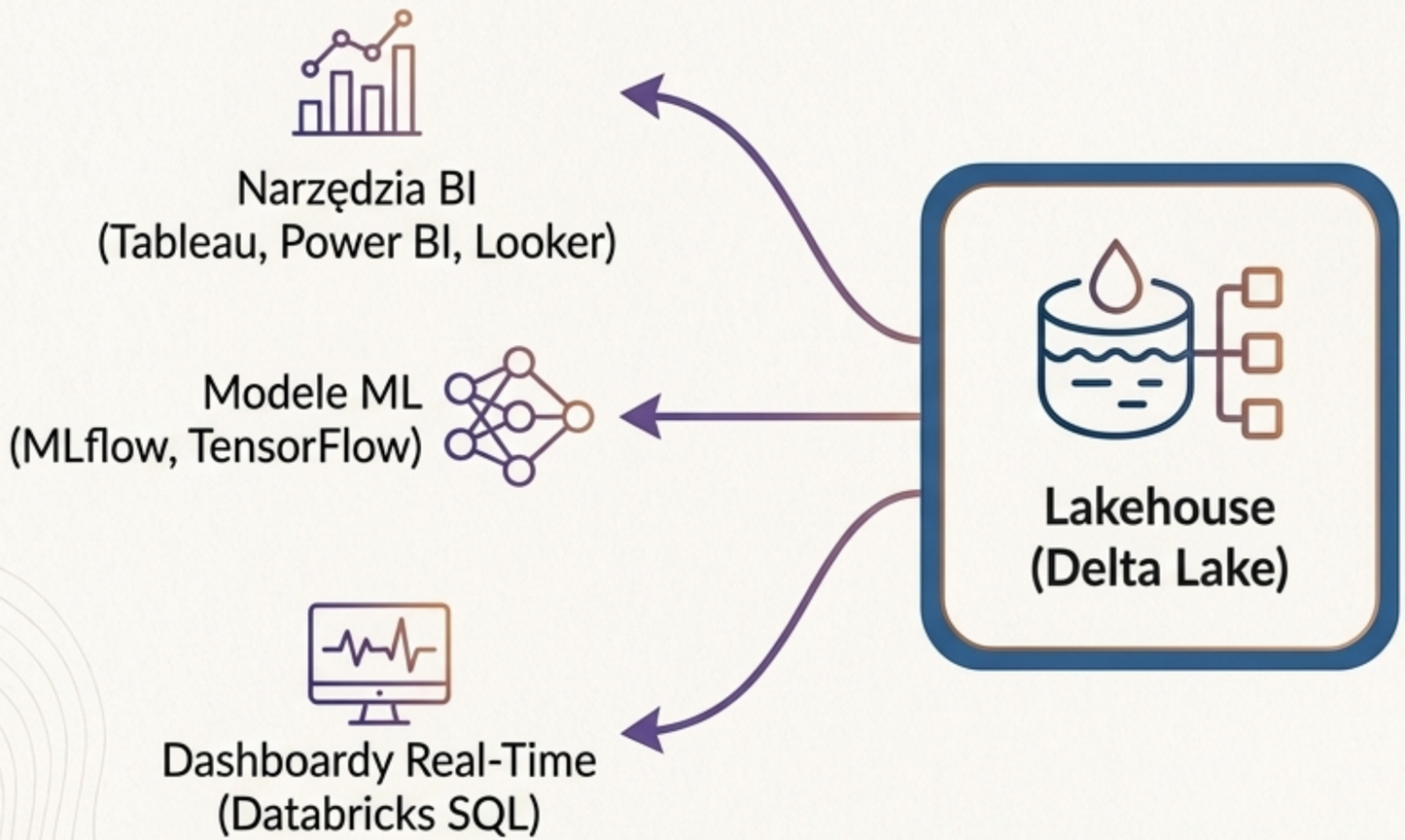
Definicja:
Mechanizm pozwalający na odrzucanie starych zdarzeń, które przybywają zbyt późno. Spark śledzi maksymalny zaobserwowany czas zdarzenia i ignoruje wszystko, co jest starsze niż max_event_time - próg_opóźnienia.


```
df.withWatermark("timestamp", "2 minutes")
```

Korzyści:

- Poprawna obsługa opóźnionych danych w ramach dozwolonego progu.
- Kontrola zużycia pamięci poprzez ograniczanie wzrostu stanu.

Ostatnia Mila: Od Danych do Decyzji i Predykcji



Wyzwania w Integracji

- **Niedopasowanie latencji:** Narzędzia BI oczekują zapytań wsadowych, a strumienie są ciągłe.
- **Świeżość danych:** Dashboardy muszą odświeżać się w sekundach bez przeciążania silnika zapytań.
- **Ewolucja schematu:** Struktury danych w czasie rzeczywistym często się zmieniają.

Nowoczesne Rozwiązanie: Platformy takie jak **Databricks** upraszczają tę integrację, oferując natywne konektory i silniki SQL zoptymalizowane do zapytań na danych strumieniowych zapisanych w **Delta Lake**.

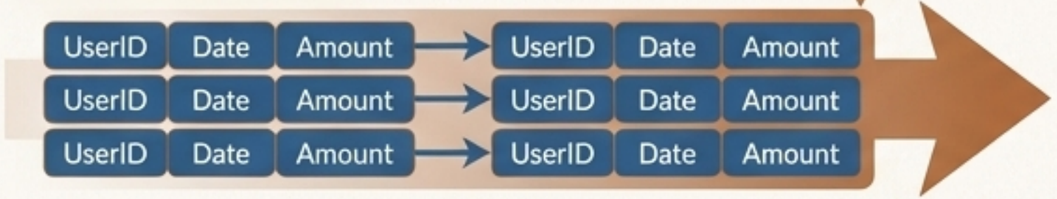
Fundament Wydajności: Kolumnowy Format Przechowywania **Parquet**

Parquet to format przechowywania kolumnowego zoptymalizowany pod kątem obciążeń analitycznych. Dane są przechowywane w kolumnach, a nie wierszach, co drastycznie redukuje ilość operacji I/O na dysku.

Przechowywanie Wierszowe

UserID	Date	Amount
1	12-12-12	150
2	12-13-12	450
3	22-12-13	100

SUM(Amount)



Przechowywanie Kolumnowe - Parquet

UserID	Date	Amount
1	12-12-12	150
2	12-13-12	420
3	22-12-13	100

SUM(Amount)

Predicate Pushdown

Zapytania (np. WHERE date > '...") odczytują tylko te bloki danych, które spełniają warunek.

Efektywna Kompresja

Kolumny tego samego typu danych kompresują się znacznie lepiej.

Ewolucja Schematu

Metadane schematu są osadzone w pliku, co pozwala na dodawanie nowych kolumn bez przepisywania starych danych.

Połączenie technologii takich jak **Spark**, **Delta Lake** i formatu **Parquet** tworzy rdzeń nowoczesnych, wydajnych i niezawodnych platform danych, które napędzają analitykę i AI na masową skalę.