



Studium Licencjackie

Kierunek Metody Ilościowe w Ekonomii i Systemy Informacyjne
Specjalność Informatyka Gospodarcza

Imię i nazwisko autora Maciej Pojedyński
Nr albumu 122138

Porównanie architektur Lambda i Kappa w kontekście analityki czasu rzeczywistego

Praca licencjacka
pod kierunkiem naukowym
Prof. Dr. Hab. Andrzej Sobczak
Instytut Informatyki i Gospodarki Cyfrowej

Warszawa 2025

Spis treści

WSTĘP	1
ROZDZIAŁ 1. ARCHITEKTURA DANYCH	3
1.1 Architektura korporacyjna jako kontekst dla architektury danych	3
1.2 Definicja i znaczenie architektury danych.....	5
1.3 Problemy w architekturze danych	8
1.4 Składowanie danych	9
1.4 Pozyskiwanie danych	11
1.5 Wzrost ilości danych	15
1.6 Jakość i spójność danych	15
1.7 Integracja danych	16
1.9 Nowoczesne podejście do architektury danych	19
1.9.1 Automatyzacja i sztuczna inteligencja w zarządzaniu danymi	21
1.9.2 Wykorzystanie chmury obliczeniowej	22
1.10 Podsumowanie	24
ROZDZIAŁ 2. ARCHITEKTURA LAMBDA I KAPPA – STRUKTURA, ZALETY I PORÓWNANIE	25
2.1 CHARAKTERYSTYKA ARCHITEKTUR LAMBDA I KAPPA.....	25
2.1.1 Podstawowe założenia architektury lambda	25
2.1.2 Warstwa wsadowa	26
2.1.3 Warstwa strumieniowa	27
2.1.4 Warstwa serwująca	28
2.2 Integracja warstw w architekturze Lambda.....	29
2.3 Zalety i wady architektury Lambda	31
2.4 Podstawowe założenia architektury Kappa	33
2.4.1 Warstwy w architekturze Kappa	33
2.5 Integracja warstw w architekturze Kappa	34
2.6 Zalety i wady architektury Kappa	35
2.7 Porównanie Lambda vs Kappa	38
2.8 Identyfikacja metadanych	40
2.8.1 Czym są metadane?	40
2.8.2 Zastosowanie metadanych w klasyfikacji architektur	41
2.8.3 Propozycje metadanych	41
2.9 Podsumowanie.....	42
ROZDZIAŁ 3. BUDOWA MODELU UCZENIA MASZYNOWEGO	44
3.1 CEL I ZAŁOŻENIA MODELU	44
3.1.1 Drzewo decyzyjne – definicja i uzasadnienie wyboru	44
3.2 DANE WEJŚCIOWE.....	45
3.2.1 Charakterystyka zbioru danych	45
3.2.2 Opis zmiennych wejściowych	46
3.2.3 Zmienna docelowa	47
3.2.4 Przygotowanie danych i jakość	47
3.2.5 Podsumowanie	47

3.3	BUDOWA MODELU KLASYFIKACYJNEGO	48
3.3.1	Wybór algorytmu	48
3.3.2	Struktura pipeline'u	48
3.3.3	Podział pracy	49
3.3.4	Parametry modelu.....	49
		2
3.3.5	Podsumowanie	49
3.4	WIZUALIZACJA STRUKTURY DRZEWY DECYZYJNEGO	49
3.4.1	Opis elementów drzewa decyzyjnego	50
3.4.2	Przykład interpretacji fragmentu drzewa.....	51
3.4.3	Podsumowanie	51
3.5	EWALUACJA MODELU KLASYFIKACYJNEGO	51
3.5.1	Metody oceny modelu	52
3.5.2	Wyniki modelu	52
3.5.3	Analiza macierzy pomyłek	52
3.5.4	Wnioski z ewaluacji	53
3.6	WNIOSKI KOŃCOWE Z CZĘŚCI PRAKTYCZNEJ	53
ROZDZIAŁ 4. PODSUMOWANIE		55
BIBLIOGRAFIA		57
STRESZCZENIE		61



Wstęp

W czasach, gdzie olbrzymie ilości danych są generowane w zatrważającym tempie zarówno przez użytkowników oraz urządzenia IoT, architektury przetwarzania danych są poddawane wyzwaniom wiążącym się z szybkością analiz danych oraz przetwarzaniu danych w czasie rzeczywistym. Rozwiązaniem tego problemu są dwie najbardziej popularne architektury przetwarzania danych: Lambda oraz Kappa, które pozwalają na efektywne zarządzanie danymi zarówno w trybie wsadowym, jak i strumieniowym. Wybranie odpowiedniej architektury ma istotny wpływ pod względem analiz decyzji w czasie rzeczywistym, co staje się coraz częściej używane w środowiskach informatycznych.

Celem niniejsze pracy jest przygotowanie analizy porównawczej architektur przetwarzania danych: Lambda i Kappa, pod względem ich funkcjonowania w przetwarzaniu danych w czasie rzeczywistym, wsadowym oraz ich integracji w systemach informatycznych. Architektury zostały przedstawione zarówno pod względem teoretycznym ze szczegółowym opisaniem funkcjonowania oraz praktycznymi zastosowaniami ich implementacji. Ważnym elementem

badania jest przygotowanie algorytmu klasyfikacyjnego, który wspomogę wykwalifikowanym specjalistom w doborze odpowiedniej architektury do środowiska.

W pierwszej części pracy omówione zostały podstawowe założenia, które mają istotny wpływ na pracę z danymi oraz zrozumienie działania architektury przetwarzania danych. Następnie przedstawiono charakterystyki obu architektur wraz z zastosowaniem praktycznym. W kolejnym etapie porównane zostały zalety oraz wady obu architektur na przykładach scenariuszy przetwarzania danych, które umożliwiły przygotowanie metadanych, których założeniem jest uczenie modelu uczenia maszynowego. Po omówieniu filarów architektury danych, przedstawiony został proces tworzenia modelu uczenia maszynowego klasyfikującego opartego na drzewie decyzyjnym, uwzględniając wybór algorytmu, przygotowanie danych pod względem czyszczenia, podzielenia na zbiór treningowy, testowy oraz optymalizację algorytmu. Kończącym procesem było przedstawienie wyników klasyfikacji i interpretacja rezultatów w kontekście praktycznych zastosowań.

W niniejszej pracy zastosowany został model klasyfikacji drzewa decyzyjnego, co umożliwiło na otrzymanie wysokiej wartości skuteczności predykcji. Wynik modelu został sprawdzony przy pomocy miary *accuracy* oraz macierzy pomyłek. Podejście to pozwala na automatyzację wyboru odpowiedniej architektury na podstawie metadanych przygotowanych dzięki analizie porównawczej zalet i wad obu architektur, co może stanowić wspomaganie

wykwalfikowanych specjalistów podczas wyboru odpowiedniego środowiska przetwarzania dużych wolumenów danych.

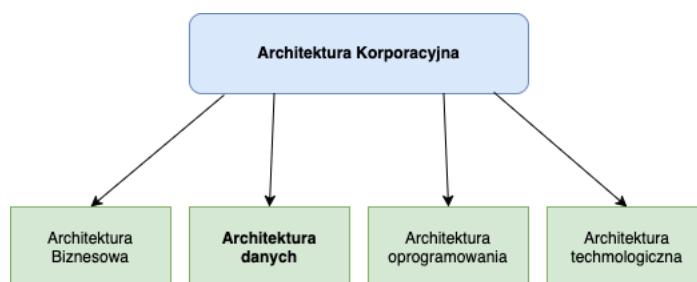
Praca kończy się wnioskami z przeprowadzonego badania oraz propozycjami na dalszy rozwój modelu klasyfikacyjnego, szczególnie z uwzględnieniem wykorzystania innych algorytmów klasyfikujących oraz rozbudowy metadanych.

Rozdział 1. Architektura Danych

1.1 Architektura korporacyjna jako kontekst dla architektury danych

Architektura danych stanowi kluczowy element pracy osoby na stanowisku Data Scientist. Określa sposób, w jaki dane są wprowadzane do systemu komputerowego, co wpływa na ich poprawne przetwarzanie oraz usprawnia codzienne operacje. Architektura danych jest podzbiorem architektury korporacyjnej, która składa się z czterech komponentów, tak jak jest to przedstawione na rysunku 1: Architektury biznesowej, Architektury technicznej, Architektury aplikacji oraz Architektury danych.

Zanim przedstawiona zostanie definicja sformułowania „Architektura korporacyjna”, zaprezentowane zostaną definicje przez trzy wiodące ośrodki, które pomogą nakreślić czym jest architektura korporacyjna, są nimi: TOGAF, Gartner i EABOK.



Rysunek 1. Podział architektury korporacyjnej

Źródło: opracowanie własne na podstawie: Sobczak, A. (2011), *Architektura korporacyjna – główne koncepcje*, dostępne online: https://andrzejsozczak.net/sites/default/files/architektura_korporacyjna_glowne_koncepcje.pdf

TOGAF, czyli The Open Group Architecture Framework, to standard opracowany przez stowarzyszenie The Open Group. Jest to jedno z najczęściej stosowanych podejść do architektury korporacyjnej. Ich definicja przedstawia się następująco, „Terminu „korporacja” w kontekście „architektury korporacyjnej” można używać do określenia zarówno całego przedsiębiorstwa – obejmującego wszystkie jego usługi informacyjne i technologiczne, procesy i infrastrukturę – jak i określoną domenę w obrębie przedsiębiorstwa. W obu przypadkach architektura obejmuje wiele systemów i wiele grup funkcjonalnych w obrębie przedsiębiorstwa.” (The Open Group, 2011)

Gartner to renomowana międzynarodowa firma zajmująca się analizą i doradztwem. Główne obszary jej działalności koncentrują się na wspieraniu organizacji w podejmowaniu kluczowych decyzji strategicznych związanych z technologiami IT.

Definicja architektury korporacyjnej, według tej firmy, przedstawia się następująco: „Architektura korporacyjna (EA) to dyscyplina proaktywnego i całościowego kierowania reakcjami przedsiębiorstw na siły zakłócające poprzez identyfikację i analizę realizacji zmian w kierunku pożądanej wizji biznesowej i wyników. EA zapewnia wartość, przedstawiając liderom biznesowym i IT gotowe do podpisu rekomendacje dotyczące dostosowywania polityk i projektów w celu osiągnięcia zamierzonych wyników biznesowych, które wykorzystują istotne zakłócenia biznesowe.” (Gartner)

Jest to wszechstronne i niezawodne źródło informacji, pełniące rolę przewodnika dla organizacji dążących do osiągnięcia Enterprise Agility. W EABOK znajdują się liczne informacje dotyczące różnych modeli, ram oraz podejść, które wspierają wspólny rozwój i zdolność adaptacji w obliczu nieustannych zmian. Ich definicja jest następująca: „Architektura korporacyjna (EA) to model organizacyjny; abstrakcyjna reprezentacja przedsiębiorstwa, która łączy strategię, operacje i technologię, aby stworzyć plan działania prowadzący do sukcesu. Istnieje wiele definicji słowa przedsiębiorstwo: w EA przedsiębiorstwo to złożona organizacja, która próbuje przejść zmiany.” (EABOK, 2020)

Z wyżej wymienionych definicji architektury korporacyjnej można wyodrębnić trzy wspólne, istotne terminy: złożoność, zarządzanie oraz skupienie. Złożoność przedsiębiorstwa, zarówno jako całość, jak i w kontekście jego poszczególnych części, które tworzą architekturę, wymaga zaplanowania i analizy zmian w odniesieniu do strategii, technologii i operacji. Zarządzanie zmianą i strategią ma na celu dostosowanie polityk oraz projektów, aby osiągnąć zamierzony wynik biznesowy. Skupienie na dążeniu do pożądanych rezultatów jest kolejnym wspólnym czynnikiem, który pojawia się w każdej z definicji. Wszystkie wymienione fragmenty pokazują ten sam cel architektury korporacyjnej, którym jest osiągnięcie zaplanowanych wyników poprzez integrację strategii, technologii i operacji.

1.2 Definicja i znaczenie architektury danych

Architektura danych odgrywa kluczową rolę w przechowywaniu, przetwarzaniu oraz zarządzaniu danymi. W dobie Big Data dane stały się najcenniejszym aktywem organizacji.

Współczesne przedsiębiorstwa opierają swoją działalność na danych, które napędzają innowacje, a także umożliwiają podejmowanie bardziej trafnych decyzji biznesowych. Prawidłowe zarządzanie danymi pozwala zapewnić ich bezpieczeństwo oraz dostępność w całej organizacji, co skutkuje oszczędnością czasu przedsiębiorstwa na konwertowaniu, sprawdzaniu i zabezpieczaniu danych.

Dzięki temu organizacja może skupić się na osiąganiu wyników biznesowych. Architektura danych jest kluczowym elementem nowoczesnej infrastruktury IT, umożliwiającym płynne zarządzanie danymi w środowiskach złożonych technologicznie.

Aby lepiej zrozumieć architekturę danych, warto przyjrzeć się kluczowym elementom, które wyłaniają się z analizowanych definicji. IBM, TOGAF oraz Reis i Housley przedstawiają tę koncepcję w nieco odmienny sposób:

„Architektura danych opisuje sposób zarządzania danymi, od ich gromadzenia po transformację, dystrybucję i wykorzystanie.” (IBM, 1998)

” Opis struktury i interakcji głównych typów i źródeł danych przedsiębiorstwa, logicznych zasobów danych, fizycznych zasobów danych i zasobów zarządzania.” (The Open Group, 2011)

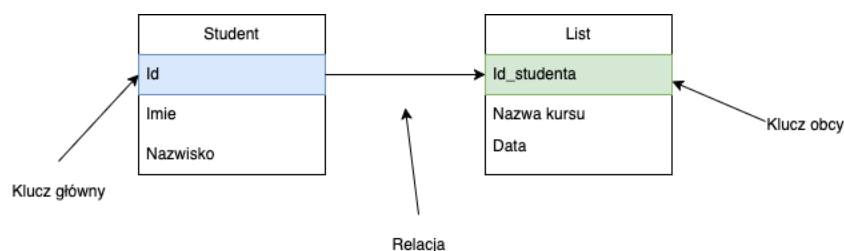
” Architektura danych to projekt systemów wspierających zmieniające się potrzeby przedsiębiorstwa w zakresie danych, opracowanych w wyniku podejmowania elastycznych i odwracalnych decyzji poprzez staranną ocenę kompromisów. „(Reis, Housley, 2023, 79-80str)

Powyższe definicje wskazują na wysoką wartość architektury danych w przedsiębiorstwie oraz w procesach zarządzania nimi. Architektura danych nie jest tylko technicznym narzędziem, ale stanowi fundament dla całej strategii organizacji, umożliwiając elastyczne dostosowanie do zmieniających się potrzeb biznesowych. Architektura danych to zbiór zasad, modeli i standardów, które określają sposób gromadzenia, przechowywania oraz wykorzystywania danych w organizacji.

Głównymi elementami architektury danych są: modele danych, metadane, standardy integracji danych oraz zarządzanie danymi.

W Architekturze danych występuje abstrakcyjna struktura zwana modelami danych. Cechuje się ona sposobem organizacji oraz przechowywaniem danych w systemie.

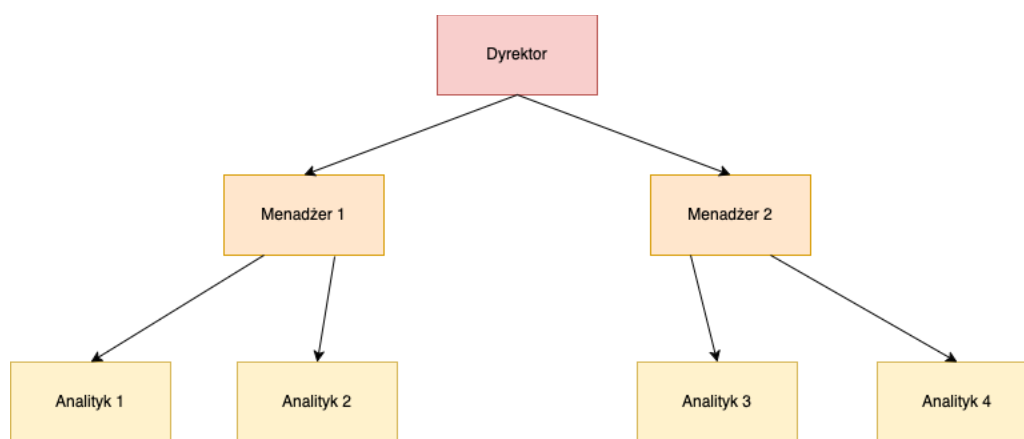
Najbardziej popularne modele, takie jak modelowanie ER, który tworzy dane na wzór tych występujących w Structured Query Language (SQL). W środowisku znajdują się tabele, które są połączone ze sobą przy pomocy relacji. W każdej tabeli występuje klucz główny oraz klucz obcy, które przedstawiają wartości występujące w obu tablicach i odpowiedzialne za połączenie. (Ballard, Herreman, Schau, Bell, Kim, Valencic, 1998)



Rysunek 2. Reprezentacja relacji w języku SQL

Źródło: opracowanie własne.

Występuje również hierarchiczny model, który jest reprezentowany przez oś. Każdy punkt danych jest połączony ze sobą tylko z jednym członkiem wymiaru. Hierarchie mają wiele poziomów hierarchii. Każdy z wymiarów, posiada wielu uczestników. Uczestnik to unikatowe imię lub identyfikator używany do określenia struktury hierarchii.



Rysunek 3. Przykład struktury hierarchicznej w kontekście modelowania danych

Źródło: opracowanie własne

Modele danych pojawiają się także w strukturach dokumentowych. W środowisku informatycznym posiadają nazwę NoSQL, ponieważ nie stosuje tradycyjnej, relacyjnej struktury danych. Dane mogą mieć różne pola oraz struktury, co zapewnia im większą elastyczność.

Metadane identyfikują dane jako główny składnik przy wykorzystaniu potencjału danych. Mogą być uznane za inwestycję, ponieważ ulepszają możliwości wykrycia danych oraz kwalifikacje w organizacji, czego skutkiem jest ułatwienie przyszłościowych korekt w projekcie oraz w rozwoju architektury. (Reis, Housley, 2023)

Zarządzanie danymi przynależy do obszarów inżynierii danych oraz uczenia maszynowego. Z każdym rokiem, obsługa danych staje się prostsza, złożoność pracy jest eliminowana, a świadomość organizacji na temat dbania i utrzymania danych staje się coraz bardziej jawna.

Online transaction processing systems (OLTP) oraz Online analytical processing (OLAP) odgrywają kluczową rolę w zarządzaniu danymi. (Reis, Housley, 2023)

OLTP jest wykorzystywane do transakcji oraz procesowania zapytań przez bankierów, klientów oraz wykwalifikowanych pracowników informatycznych. Jest używany przez analityków danych, menadżerów oraz prezesów. (Reddy, Srinivasu, Poorna, Rikkula, 2010) System OLAP jest w stanie zarządzać oraz prezentować dane w różnych formatach, zależnie od potrzeby użytkownika. OLTP w odróżnieniu od OLAP zajmuje się aktualnymi danymi w bardziej szczegółowy sposób. (Reddy, Srinivasu, Poorna, Rikkula, 2010)

OLAP jest wykorzystywany na dużych ilościach danych historycznych, dostarczając podsumowania oraz agregowane funkcje. Dodatkowo, dane są łatwiejsze do użytku z powodu różnych poziomów ziarnistości. Odnosi się to do poziomu szczegółowości danych przechowywanych oraz wykorzystywanych w analizach. Może być niski lub wysoki a jego wydajność wpływa na elastyczność analiz.

Kategoria	System OLTP
Rodzaj przetwarzania	<i>Skupia się na operacjach transakcyjnych.</i>
Typ danych	<i>Operuje na danych bieżących i szczegółowych.</i>
Zorientowanie systemu	<i>Skoncentrowany na obsłudze procesów aplikacyjnych.</i>
Główna funkcja	<i>Wspiera codzienne działania operacyjne firmy.</i>
Transakcyjność	<i>Wykonywane są liczne i częste transakcje.</i>

Tabela 1. Charakterystyka systemu OLTP według wybranych kategorii

Źródło: opracowanie własne na podstawie: Reis, Housley, 2023, *Inżynieria danych w praktyce*, Helion

Kategoria	System OLAP
Rodzaj przetwarzania	<i>Służy do analiz i eksploracji danych.</i>
Typ danych	<i>Wykorzystuje dane zagregowane i podsumowane.</i>
Zorientowanie systemu	<i>Zorientowany na analizę tematyczną i wspieranie decyzji.</i>
Główna funkcja	<i>Wspomaga analizę historyczną i podejmowanie decyzji.</i>
Transakcyjność	<i>Transakcje są rzadkie i dotyczą dużych zbiorów danych.</i>

Tabela 2. Charakterystyka systemu OLAP według wybranych kategorii

Źródło: opracowanie własne na podstawie: Reis, Housley, 2023, *Inżynieria danych w praktyce*, Helion

Przykładem niskiej ziarnistości jest transakcja na poziomie pojedynczego zakupu, konkretnej daty oraz godziny. Umożliwia dokładne analizy, zwiększając rozmiar bazy i czas przetwarzania.

Wysoka ziarnistość w odróżnieniu od wcześniejszego poziomu, przechowuje dane zagregowane, czego przykładem jest podsumowanie sprzedaży z danego okresu.

Zapotrzebowanie na OLAP i OLTP w architekturze danych jest spowodowane odgrywanymi rolami na poziomie operacyjnym oraz analitycznym. Ich dobry poziom współdziałania wpływa na działanie całej architektury, pozwala na efektywne przetwarzanie, przechowywanie oraz analizowanie danych co jest niezbędne do podejmowania decyzji biznesowych oraz utrzymania systemów informatycznych. (Reddy, Srinivasu, Poorna, Rikkula, 2010)

Wraz ze zrozumieniem wagi danych na świecie, pojawiają się nowe problemy na drodze do idealnej architektury danych, które muszą być niwelowane oraz zapobieganie przed ponownym pojawieniem się.

1.3 Problemy w architekturze danych

Jednymi z wielu podczas budowy architektury danych, z którymi wykwalifikowani specjaliści próbują konkurować to złożoność systemu, wysokie koszty, problemy z integracją danych, czasochłonna transformacja danych, opóźnienie dotyczące danych, ograniczona elastyczność oraz kwestie bezpieczeństwa i prywatności.

Złożoność, czyli projektowanie, budowanie oraz utrzymanie architektury danych cechuje się czasochłonnością oraz dużą złożonością. W celu utrzymania mechanizmu, decydującym czynnikiem jest wysoko wykwalifikowana kadra, z specjalistycznymi umiejętnościami i zasobami, które skutkują zwiększającymi się kosztami utrzymania. (Serra, 2022)

Wysokie koszty, które rosną w zatrważającym tempie z powodu implementowania architektury danych oraz poważne inwestycje w środowisko informatyczne w którego skład wchodzi: sprzęty, oprogramowanie oraz personel. Wydatki zwiększają się wprost proporcjonalnie do ciągłego wzrostu aktualizacji i utrzymania. (Serra, 2022)

Integracja danych z różnych źródeł stanowi wyzwanie, ponieważ łączy zarządzanie różnorodnych formatów danych, struktur i wyzwań z kontrolą i utrzymaniem jakości. Skutkuje to redukcją czasu i zwiększonym nakładem pracy przy oczyszczaniu i przetwarzaniu danych. (Serra, 2022)

Transformacja danych staje się coraz poważniejszą wadą wraz ze wzrostem ilości przekształceń w celu doszlifowania ich do modelu danych. Proces ten, może nieść ze sobą błędy, które skutkują nieprecyzyjnymi analizami, ponieważ architektury danych są tworzone w celu obrabiania dużej ilości danych oraz zarządzania nimi, przetwarzanie może stawać się

wolniejsze. Może skutkować to, opóźnieniami związanymi z konwertowaniem danych. Architektury danych, korzystają z wielu rodzajów magazynów danych, od relacyjnych hurtowni danych po jeziora danych. (Serra, 2022)

Zależnie od wybranego rodzaju przetrzymywania danych, pojawiają się wyzwania. W przypadku relacyjnych hurtowni danych, problemem staje się generowanie analiz w tym samym momencie, w którym dane są przygotowywane, dlatego w czasie konserwacji, użytkownikowi zabiera się dostęp na ograniczony czas, w przeciwnym wypadku żaden z procesów nie jest generowany całkowicie i pojawiają się opóźnienia. (Serra, 2022)

Wyzwaniem podczas tworzenia architektury, jest dobranie odpowiednich magazynów, w których będą przetrzymywane określone formaty danych.

Ostatnim wyzwaniem jest bezpieczeństwo oraz prywatność, które zostaną omówione szczegółowo w kolejnym rozdziale.

Przechowywanie dużych ilości danych poufnych w głównym miejscu, powoduje zwiększenie ryzyka kradzieży oraz złamania praw prywatności. Wskutek tego istotne jest wykorzystywanie solidnych usług zabezpieczających.

Głównymi barierami w architekturze danych są złożoność, wysokie koszty, trudności z integracją, czasochłonna transformacja danych, opóźnienia, ograniczona elastyczność oraz kwestie bezpieczeństwa i prywatności. Zarządzanie systemami wymaga wysoko wykwalifikowanych specjalistów, czego skutkiem są koszty rosnące wprost proporcjonalnie z wzrostem infrastruktury i aktualizacji. Integracja formatów danych jest skomplikowana oraz czasochłonna tak jak w przypadku transformacji, w której mogą pojawiać się błędy podczas przetwarzania i niepoprawnych analiz. Duże ilości danych spowalniają przetwarzanie co przyczynia się do opóźnień.

Wybór odpowiednich magazynów danych, takich jak hurtownie oraz jeziora danych, analogicznie stanowią wyzwanie. Przechowywanie poufnych informacji w centralnym miejscu, utrudnia pracę specjalistów, którzy muszą wyróżnić się solidnymi umiejętnościami zabezpieczającymi.

1.4 Składowanie danych

W bazach danych, informacje są przechowywane przy pomocy tabel oraz indeksów. Tabela jest priorytetem mechanizmu składowania danych, a indeks skraca czas poszukiwania oraz umożliwia znalezienie danych pod wpływem uporządkowania. Przy wzroście liczby magazynów danych sytuacja opisana powyżej znacząco się komplikuje, przez złożoność procesu.

W dużych zbiorach wymagane są wyspecjalizowane struktury przyspieszające wyszukiwanie oraz pobieranie danych. Podczas wyboru magazynu danych istotne jest analizowanie wymogów pojemnościowych oraz operacyjnych wejścia-wyjścia, związane z przechowywaniem danych na dysku i w indeksach a także wejścia-wyjścia na poziomie pobierania. (Campbell, Majors, 2018)

Przechowywanie surowych danych w tradycyjnych systemach relacyjnych wyróżnia się składowaniem w blokach, które odpowiadają określonej wielkości bajtów na dysku. Bloki występują jako najniższa jednostka zapisu rekordów. Wymagają również zapisu metadanych – najczęściej w formie nagłówków oraz stopki. Informacje w nich zawarte, zawierają dane na temat adresu lokalizacji na dysku, obiekcie, do którego należy blok oraz wierszach i operacjach z danego bloku. Ze względu na wydajność, bloki danych są łączone w celu tworzenia ekstentów, które są reprezentowane jako kontenery pamięci oraz biorą udział w procesie alokacji pamięci w celu poszerzenia wielkości bloków lub ilości. (Campbell, Majors, 2018) W bazach, strukturą wykorzystywaną dla danych są drzewa binarne, potocznie nazywa Bdrzewami. Funkcja tej struktury to automatyczne sortowanie, grupowanie oraz równoważenie danych. B-drzewa są zoptymalizowane, dzięki czemu są łatwe do odczytu i zapisywania bloków danych. Struktura jest nazywana drzewem, ponieważ w trakcie przemieszczania się po nim możliwy jest wybór ścieżki spośród dwóch lub większej ilości stron podrzędnych, aby znaleźć się w poszukiwanej lokalizacji. B-drzewo jest reprezentowane jako odwrócone drzewo, gdzie korzenie są na górze. Korzeń jest początkiem indeksu zbudowanego na podstawie klucza.

Kluczem może być kolumna lub kolumny, zależnie od architektury. (Campbell, Majors, 2018) Zgodnie z bazami relacyjnymi, dane są przechowywane zgodnie z kluczem głównym, który pojawia się w pozostałych kolumnach jako klucz obcy w celu stworzenia relacji między tabelami, dzięki czemu można tworzyć zapytania, które otrzymują wyszukiwane wartości oraz informacje na temat danych.

Metadane posiadają wskaźniki do stron z niższych wierszy. Strona – Korzeń ma na niższym poziomie kolejne strony, które są nazywane dziećmi. Strona podrzędna może być liściem lub węzłem wewnętrznym. Węzeł wewnętrzny za pomocą indeksów przechowuje klucze dzielące oraz wskaźniki do określonych lokalizacji. Liście natomiast posiadają konkretne dane. W trakcie przesyłania danych do b-drzewa, odpowiednie liście są wyszukiwane przy pomocy indeksów. W momencie zapełnienia węzła, powstają nowe podziały. (Campbell, Majors, 2018) B-drzewa posiadają świetną wydajność zapytań, są zapisywane zgodnie z kolejnością sortowania, co pozwala na zoptymalizowane wyszukiwanie kluczy, minimalizują odczyty stron. Niestety dają niepożądane rezultaty w trakcie wyszukiwani pojedynczych wierszy.

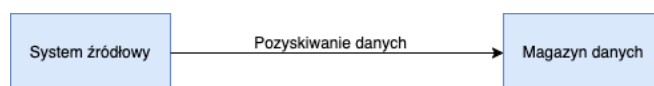
Kolejnym rodzajem przechowywania danych są pliki SST oraz drzewa LSM. (Campbell, Majors, 2018)

W plikach SST, pojawiają się liczne ilości plików, w których każdy posiada posortowaną relację klucz – wartość. W odróżnieniu od b-drzewa nie ma tu potrzeby tworzenia metadanych opisujących dane. Dzięki temu, że dane są posortowane, można je wykorzystywać jako indeks oparty na kluczu, który jest wyznacznikiem porządku sortowania.

W architekturze, która wykorzystuje jednocześnie pliki SST oraz drzewa LSM, dane są zapisywane zgodnie z procesem okresowego przenoszenia oraz składowania bloków danych w pamięci. Jednak dane po skończonym procesie przeniesienia stają się niemodyfikowalne. Jednakże, dowolny plik SST można delokalizować do pamięci w celu obróbki i ponownego załadowania, w przypadku, gdy cały plik nie mieści się w pamięci, możliwe jest wyszukiwanie operacji przy cenie niewielkiej liczby operacji wyszukiwania. (Campbell, Majors, 2018) W odróżnieniu od zapisywania na dysku, zbiory danych są bardzo proste, ponieważ ich proces polega na aktualizowaniu wskaźników. W momencie całkowitego przygotowania pod względem magazynów danych oraz wykorzystujących drzew oraz plików, można rozpocząć proces pozyskiwania danych z różnych źródeł, których liczba powiększa się z każdym rokiem.

1.4 Pozyskiwanie danych

Procesem odpowiedzialnym za przenoszenie danych z jednego miejsca do drugiego jest nazywane pozyskiwaniem danych. Oznacza to, przenoszenie z systemów źródłowych informacji do magazynów danych, a pozyskiwanie danych w tym procesie jest etapem pośrednim.

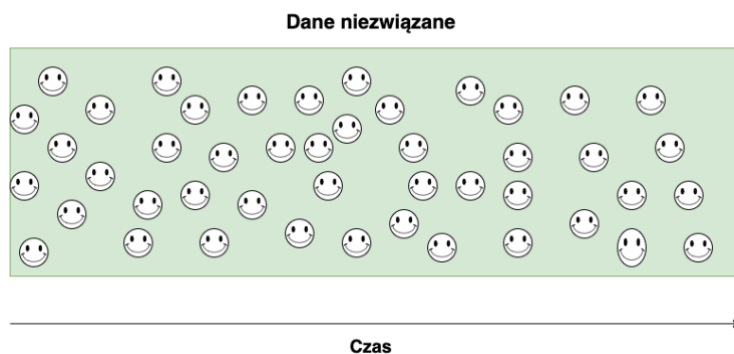


Rysunek 4. Proces pozyskiwania danych z systemu źródłowego do magazynu danych

Źródło: opracowanie własne.

Istotne jest zauważenie różnic między pozyskiwaniem a integracją danych, pierwszy proces to przenoszenie danych między obecną lokalizacją a zamierzoną. Integracja polega na połączeniu danych z różnych źródeł w nowy zestaw danych. (Reis, Housley, 2023)

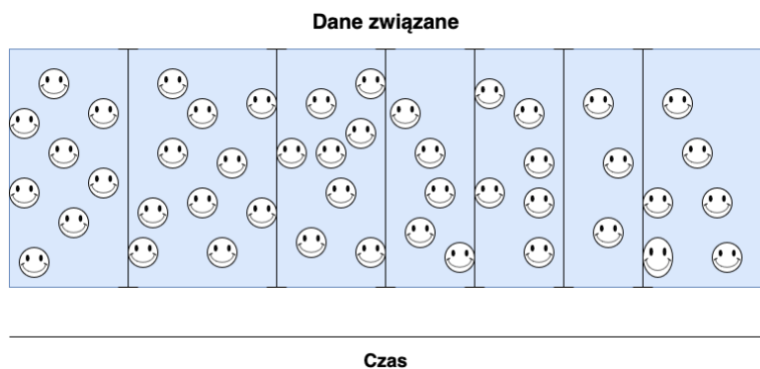
W świecie danych istnieją dwie formy prezentowania danych, związane oraz niezwiązane. Dane niezwiązane istnieją w czasie wystąpienia zdarzenia, cechują się ciągłością lub sporadycznością występowania.



Rysunek 5. Przykład danych niezwiązanych w relacyjnym modelu danych

Źródło: opracowanie własne na podstawie: Reis, Housley, 2023, *Inżynieria danych w praktyce*, Helion

Dane związane to zgrupowane dane niezwiązane, polega to na zakwalifikowaniu danych do odpowiednich grup, zależnie od sposobu rozmieszczenia np. przez czas.



Rysunek 6. Przykład danych związanych w relacyjnym modelu danych

Źródło: opracowanie własne na podstawie: Reis, Housley, 2023, *Inżynieria danych w praktyce*, Helion

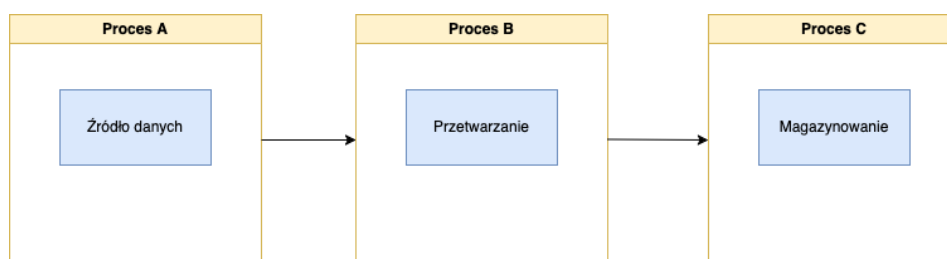
W książce *Inżynieria danych w praktyce*, Joe Reis, Matt Housley (2023), przedstawiona została synergia obu formatów danych ze sobą „wszystkie dane są niezwiązane, dopóki nie staną się związane.”. Oznacza to, że w momencie rozpisania listy zadań na papierze (dane związane), były one nie pogrupowanym zbiorem (danymi niezwiązanymi). (Reis, Housley, 2023)

Pozyskiwanie danych jest również definiowane przez częstość pobierania. Proces pozyskiwania danych może przetwarzać dane w trybie wsadowym, czyli w dużych ilościach, mikropartiami lub w czasie rzeczywistym. Zależnie od organizacji, częstość pobierania danych jest uzależniona od potrzeb.



Rysunek 7. Częstość pozyskiwania danych ze źródeł: przetwarzanie wsadowe, mikropartyjne i w czasie rzeczywistym
Źródło: opracowanie własne na podstawie: Reis, Housley, 2023, *Inżynieria danych w praktyce*, Helion

Częstość pozyskiwania danych źródłowych w czasie rzeczywistym, nie jest rzetelnym określeniem, ponieważ wiele systemów stara się osiągnąć taką częstość pozyskiwania, jednak każda baza danych, kolejka charakteryzuje się opóźnieniami, poprawnym jest „zbliżone do czasu rzeczywistego”, jednak umownie przyjęto sformułowanie „w czasie rzeczywistym”. (Reis, Housley, 2023)



Rysunek 8. Proces pozyskiwania synchronicznego

Źródło: opracowanie własne na podstawie: Reis, Housley, 2023, *Inżynieria danych w praktyce*, Helion

W przypadku pozyskiwania synchronicznego źródło, system pozyskiwania i miejsce docelowe są ze sobą ściśle powiązane. Jak widać na rysunku 8, na każdym etapie cyklu życia danych, procesy A, B i C, są zależne od siebie. W momencie, gdy proces A nie zostanie zrealizowany, pozostałe procesy nie będą mogły zostać uruchomione. (Reis, Housley, 2023)

Ten rodzaj pracy jest szerzej wykorzystywany w starszych systemach ETL, w którym najpierw dane muszą zostać przekształcone, następnie wyodrębnione dane z systemu źródłowego zostają załadowane do hurtowni danych. Proces występuje w strumieniach przetwarzania, oznacza to, że do momentu zrealizowania fazy pozyskiwania, pozostałe nie mogą się rozpocząć. (Reis, Housley, 2023)

Z drugiej strony pozyskiwanie asynchroniczne, może rozwiązywać problemy na poziomie pojedynczych zdarzeń. Główną ideą systemu jest fakt, że tak jak w przypadku synchronicznego, gdzie proces wsadowy jest uruchamiany wielokrotnie w systemie, dla każdego procesu jednokrotnie i rozpoczęcie kolejnych procesów jest uzależnione od wykonania poprzedzających, w asynchronicznym potoku, każdy etap może przetwarzać dane natychmiast po pojawieniu się dostępności w klastrze Apache beam. Jest uważana ona za środowisko, w którym przetwarzają się dane zarówno w trybie wsadowym i strumieniowym.

W Apache beam dane są odczytywane z różnych źródeł, niezależnie czy są one w chmurze bądź na serwerze. Następnie dane są procesowane zarówno w warstwie batch oraz streamingowej. Beam zapisuje rezultaty wynikające z procesowania danych do najbardziej popularnych miejsc docelowych przepływu danych. Szybkość zasobów jest zależna od dostępnych zasobów w organizacji. (Reis, Housley, 2023)

Przenoszenie danych między obecnym a docelowym obejmuje serializację oraz deserializację. Pierwszy termin określa kodowanie oraz obrabianie danych ze źródła do etapów transformacji oraz pośrednich etapów składowych. Podczas zdeserializacji, czyli proces polegający na odczytaniu wcześniej zapisanych danych, jest to odwrotny proces niż serializacja, trzeba zadbać o zalokowanie w odpowiednie miejsce dane.

Dbanie o poprawną przepustowość danych oraz skalowalność systemu stają się istotne wprost proporcjonalnie ze wzrostem ilości danych i korekcjami wymagań. Systemy muszą być projektowane w sposób umożliwiający skalowania umożliwiającego dopasowanie do oczekiwanej przepustowości. (Reis, Housley, 2023)

Niezawodność systemu, czyli umiejętność systemu do ciągłego czasu pracy bez opóźnień oraz możliwość przełączenia w tryb awaryjny. Trwałość systemu zapewnia, że dane nie zostaną utracone lub uszkodzone. W przypadku niektórych źródeł, dane nie są utrzymywane, jeśli podczas pozyskiwania zostaną one niepoprawnie zaimplementowane. W momencie zniknięcia danych nie ma możliwości odzyskania. W tym rozumieniu, trwałość oraz niezawodność są ze sobą współzależne. (Reis, Housley, 2023)

Pozyskiwanie danych to prawdopodobnie najbardziej wymagająca i czasochłonna część pracy, podczas przygotowywania architektury danych. Domyślnym elementem sukcesu tego procesu jest komunikacja oraz kontrola systemów, które umożliwiają spójne oraz bezpieczne przepływy danych do systemu docelowego. Istotnym wyzwaniem dla systemów jest ciągły wzrost ilości danych wymagających pozyskiwania, przesyłania oraz przetrzymywania w odpowiedniej lokalizacji.

1.5 Wzrost ilości danych

W erze cyfrowej, za sprawą szerokiego rozpowszechniania smartfonów i komputerów, dostęp do internetu stał się powszechny. Urządzenia te nieustannie są połączone z siecią, czego rezultatem jest generowanie ogromnych ilości danych. Aktywności użytkowników, które cechują się częstotliwością wykonywalności codzienną, takie jak korzystanie z aplikacji nawigacyjnych, przeglądanie przepisów kulinarnych, interakcje w mediach społecznościowych czy korzystanie z platform streamingowych, są przyczyną wzrostu ilości informacji przetwarzanych przez systemy cyfrowe.

Rozwój technologii Internetu Rzeczy (IoT) doprowadził do powstania nowej klasy urządzeń – inteligentnych zegarków, asystentów głosowych, ale również sprzętów gospodarstwa domowego, takich jak roboty kuchenne z wieloma funkcjami, które również ciągle zbierają oraz przesyłają dane, zwiększając skalę zjawiska Big Data (Minerva, Biru, Rotondi, 2015). Rezultatem ewolucji systemów informacyjnych było odejście od tradycyjnych form zapisu informacji, które zostały zastąpione przez cyfrowe repozytoria danych. Odpowiedzią na wzrastający popyt na przestrzeń magazynowania informacji, rozwinięto chmury obliczeniowe, która pozwala na przechowywanie i przetwarzanie danych w rozproszonych środowiskach serwerowych. Wiodące platformy takie jak Amazon Web Services, Microsoft Azure oraz Apple iCloud stanowią dziś fundament infrastruktury cyfrowej wielu przedsiębiorstw. (Armbrust, Fox, Griffith, Joseph, Katz, Konwinski, Lee, Patterson, Rabkin, Stoica, Zaharia)

Wzrost ilości danych generuje zapotrzebowanie na zwiększenie pamięci dyskowej, ale także wymaga zastosowania nowych metod analitycznych oraz sztucznej inteligencji, które umożliwią przetwarzanie surowych danych w użyteczne informacje. Jednocześnie, wraz z rosnącą ilością danych, ważnym aspektem staje się ich jakość, spójność oraz integralność, które również muszą być na wysokim poziomie, czego efektem jest efektywność oraz niezawodność systemów informatycznych (Batini, Scannapieco, 2016, 21-23)

1.6 Jakość i spójność danych

Problemy jakości danych pojawiają się na całym świecie. W momencie, gdy są niekontrolowane, ich jakość może się pogorszyć w szybkim tempie, czego skutkiem są olbrzymie koszty dla firmy.

W książce *Data Quality: The accuracy dimension*, omawiane są najważniejsze wyzwania z kontrolowaniem jakości danych. Według autorów, trzeba kontrolować jakość danych w różnych etapach. Dane mogą w początkowym przerabianiu być poprawne, jednak z napływem ilości ludzi oddziałujących na dane, może dojść do uszkodzenia danych. (Olson, Jack E, 2023)

Kontrola jakościowa nie może zostać przydzielona małej ilości ludzi, musi to być szeroki zakres specjalistów. Danych nie można zostawić po jednokrotnym poprawieniu i pozostawiona bez nadzoru. W przypadku, gdy zostaną bez opieki, mogą powrócić do słabej jakości, dlatego kontrola jakości musi być stała. Dane staną się coraz bardziej odporne na pogorszenie jakości w momencie, gdy specjalista będzie zwiększać swoje kompetencje.

Jakość oraz spójność danych, jako optymalizacja, która idzie w kierunku pożądanego stanu przez zlecniodawcę procesu. Jakość danych definiuje się trzema głównymi cechami:

dokładnością, kompletnością oraz terminowością. Pierwsza cecha odpowiada na pytanie, czy to co otrzymano, odpowiada temu co było zaplanowane oraz czy istnieją duplikaty. (Reis, Housley, 2023)

Kompletność przedstawia rekordy danych oraz ich zgodność z wymaganiami. Terminowość pilnuje, czy dane są dostępne w odpowiednim czasie.

Problemy jakościowe nie można rozwiązać przy pomocy technicznych środków. Trzeba przygotować zbiór zasad, które muszą być przestrzegane, w celu osiągnięcia zamierzonych efektów.

1.7 Integracja danych

Wraz z rozwojem technologii i upływem czasu, liczba źródeł, z których pozyskiwane są dane zwiększyła się w znaczący sposób. Współcześnie dane pochodzą z różnych kanałów: aplikacji internetowych, mediów społecznościowych, urządzeń mobilnych, systemów transakcyjnych oraz rozwiązań z zakresu Internetu Rzeczy (IoT). Reprezentowana wieloźródłowość danych tworzy wyzwania związane z ich spójnością, przetwarzaniem do odpowiedniego typu danych oraz integracją. (Reis, Housley, 2023)

Nowoczesna architektura danych skupia się zarówno na poprawnym przetwarzaniu dużych wolumenów informacji, ale również sprawnego zarządzania metadanymi, dzięki którym możliwe jest sprawdzenie pochodzenia, struktury oraz jakości danych. Jedną z odpowiedzi na problem dynamicznych i zmiennych źródeł danych jest używanie zaawansowanych technik optymalizacji, takich jak algorytm roju cząsteczek (PSO), wykorzystywanych w połączeniu z metadanymi w procesach ETL. To rozwiązanie umożliwia na elastyczne oraz efektywne zarządzanie integracją wśród danych. (Jun-Zhou, Yu, 2021)

W rezultacie integracja danych stała się nieodłącznym obszarem działalności specjalistów ds. danych oraz inżynierii systemów. Nieustannie zwiększająca się złożoność oraz różnorodność danych zarówno komplikują procesy ich przetwarzania jak również zwiększa złożoność budowy infrastruktury. Wraz ze wzrostem liczby źródeł oraz objętości przetwarzanych informacji, rosną oczekiwania względem bezpieczeństwa, poufności oraz zgodności z regulacjami prawnymi.

1.8 Bezpieczeństwo i prywatność danych

Bezpieczeństwo jest pierwszoplanowym składnikiem prywatności. Prywatność od dawna ma szczególną wartość do organizacji. Wykwalifikowani specjaliści przetwarzają dane związane z życiem prywatnym ludzi. W zakres tego obszaru wchodzi: informacje finansowe, dane

dotyczące prywatnej komunikacji, historie medyczne, dokumentacje edukacyjne oraz historie zatrudnieni.

Najsłabszym ogniwem bezpieczeństwa i prywatności są ludzie. Coraz łatwiej oszukać się ludzi, poprzez wysyłanie phishingowych wiadomości, chatboty wysyłające linki śledzące oraz nieuwaga ludzka, prowadzi do wycieku danych poufnych oraz informacji. Głównym zabezpieczeniem w sytuacji tego typu zagrożeń jest zwiększona czujność, pozwalająca oceniać zdarzenia obiektywnie, podwójne sprawdzanie informacji oraz stawianie bezpieczeństwa jako nawyk. Zwiększająca się ilość przepisów, zasad, regulaminów nie jest idealna i zawsze znajdują się luki, które mogą zostać wykorzystane w złym celu. Mimo, że regulacje stwarzają pozory bezpieczeństwa, nie można stać się obojętnym na zagrożenie. Procedury muszą się stać łatwe i proste, aby każdy pracownik organizacji był w stanie z nich korzystać i aplikować w codziennych zadaniach. Zwiększona ilość spotkań na temat bezpieczeństwa i prywatności powinna być wykorzystywana w całości, jednak, żeby tego rodzaju praktyki stały się owocne, elementarnym zadaniem jest stawianie bezpieczeństwa na pierwszym miejscu. Każdy jest odpowiedzialny i odgrywa kluczową rolę w organizacji.

W celu zwiększenia bezpieczeństwa oraz zmniejszenia prawdopodobieństwa wycieków danych, trzeba stosować zasadę najmniejszych uprawnień. Członkowie organizacji potrzebują tylko takiej ilości dostępów, jakie są im potrzebne do codziennej pracy, w przypadku aktywności nieregularnych dostęp musi być kontrolowany i zabrany w momencie zakończenia pracy lub przyznany na określony czas. (Reis, Housley, 2023)

Bezkrytyczne przyznanie uprawnień jest uznawane jako praktyka błędna, ponieważ może prowadzić do licznych zagrożeń. Oczywiście ta zasada ma również użyteczność w prywatności, ponieważ pracownicy będą ufać, że ich prywatne dane nie są dostępne dla każdej jednostki a jedynie wybranej grupie, której sprawdzanie jest poparte odpowiednimi kwalifikacjami oraz powodami.

W przypadku danych historycznych, uprawnienia do odczytu powinny być rozdawane dopiero po przejściu odpowiednich szkoleń, które gwarantują szybkie i optymalne wyszukiwanie. Mimo tych zabezpieczeń przed ludźmi, trzeba mieć na uwadze przygotowanie kopii zapasowych w przypadkach nieprzewidywalnych, jak ataki hakerskie bądź usunięcie bazy danych przez stażystę. (Reis, Housley, 2023)

Kolejnym filarem cyberbezpieczeństwa oraz prywatności są technologie wykorzystywane do zabezpieczania systemów i zasobów danych. Oprogramowanie tak samo jak wiedza, z biegiem czasu wymaga aktualizacji oraz wprowadzania poprawek i zapobiegania pojawieniu się ich w

przyszłości. Usługi zarządcze umożliwiają automatyczne aktualizowanie bez interwencji specjalisty, jednak monitorowanie ich jest kluczowe.

W przypadku prywatności szyfrowanie jest podstawowym sposobem zabezpieczenia. Szyfrowanie pojawia się na różnych etapach życia danych, od danych znajdujących się w magazynach danych po dane w trakcie przesyłania. Dane znajdujące się w spoczynku, czyli na urządzeniu pamięci masowej muszą zostać zaszyfrowane, w celu ochrony w przypadku kradzieży. Wykwalifikowani specjaliści powinni zaszyfrować serwery dla wszystkich przechowywanych danych na serwerach, plikach, bazach danych oraz pamięci masowej w chmurze. Szyfrowanie podczas przesyłania jest również szczególnie istotne. Mimo, że jest współczesnym trybem działania protokołów sieciowych. (Reis, Housley, 2023)

Działania hakerów, które mogą wyrządzić szkody, często są wykrywane dopiero po fakcie. W architekturach danych, aby zapewnić świadomość na temat potencjalnych ataków, stosowane są mechanizmy automatycznego monitorowania, logowania i ostrzegania o zdarzeniach. Najczęściej monitorowane obszary to: dostęp, który sprawdza uprawnienia użytkowników oraz uzasadnia, dlaczego je posiadają, oraz zasoby, w których monitorowane są dyski, pamięci i operacje wejścia-wyjścia w poszukiwaniu wzorców różniących się od tych powszechnych.

(Reis, Housley, 2023)

Bezpieczeństwo powinno stać się nawykiem. Dane należy traktować jako cenne zasoby, które można porównać do posiadania telefonu. Znajomość podstawowych zasad bezpieczeństwa i prywatności oraz dbałość o ochronę środowiska oznacza zgłaszanie potencjalnych wycieków danych, ataków hakerskich i problemów związanych z aktualizacjami oprogramowania. Takie podejście pozwala znacząco zredukować ryzyko oraz zwiększyć bezpieczeństwo i prywatność danych, które stają się coraz bardziej istotne w codziennym życiu.

Mimo tak ogromnego rozwoju danych, dalej pojawiają się wyzwania na przyszłość oraz przyszłość architektur danych, które są filarem zarządzania danymi. Architektura danych musi zapewniać jednocześnie przetwarzanie, przechowywanie, bezpieczeństwo oraz szeroką dostępność.

W związku z rosnącą ilością danych, architektura danych musi być stale aktualizowana, musi być w stanie aktualizować i obsługiwać zarówno dane strukturalne jak i niestukturalne. To wyzwania stawia przed architektami potrzebę tworzenia nowoczesnych modeli zarządzania danymi, takich jak Data Mesh oraz Data Fabric. (Reis, Housley, 2023)

Podsumowując, przyszłość architektury to okres dynamicznych zmian oraz rozwoju, gdzie innowacje technologiczne będą odgrywać kluczową rolę, ponieważ połączą bezpieczeństwo, elastyczność oraz zdolność do obsługi danych w czasie rzeczywistym. Organizacje będą

musiały inwestować w nowe modele do zarządzania danymi w celu sprostania nowym oczekiwaniom danych.

1.9 Nowoczesne podejście do architektury danych

Nowoczesna hurtownia danych staje się centrum życia danych. Hurtownie danych umożliwiają dostęp do danych w czasie rzeczywistym, zarządzania organizacją oraz analizą danych. Dane w hurtowniach są zazwyczaj zoptymalizowane pod kątem analityki, co oznacza, że są ustrukturyzowane i przetworzone. Nowoczesne architektury łączą ze sobą relacyjne hurtownie danych wraz z jeziorami danych, czego rezultatem są elastyczne modyfikacje na danych oraz wartościowe analizy.

Pierwszy model działa na zasadzie „odgórnej”, oznacza to, że przed rozpoczęciem procesu, musi zostać szczegółowo zaplanowany sposób działania. Podejście to w znaczący sposób ułatwia pracę z danymi historycznymi przy raportowaniu, gdyż pozwala analitykom na wykonanie statystyk opisowych oraz diagnostycznej. (Serra, 2024)

Jeziora danych są oparte na zasadzie „oddolnej”, cechuje się rozpoczęciem procesy przy niewielkich działaniach, co ułatwia szybkie wdrażanie modeli uczenia maszynowego, analityki predykcyjnej. Dzięki poznaniu nowoczesnej hurtowni danych, powstały architektury Data Fabric oraz Data Mesh, które stają się przyszłością pracy z danymi. Data Fabric jako kolejny filar przyszłości danych, cechuje się zwiększonym poziomem bezpieczeństwa, dostępu do baz danych oraz dostępnością. Jest to jedna z najpopularniejszych architektur stosowanych do ulepszania chmury oraz działania na dużych bazach danych powyżej 10TB. Zaletami tej architektury jest, integracja danych z wielu źródeł, dzięki czemu może obsługiwać dane różnych typów, dodatkowo umożliwiając adaptowaną wizualizację danych. (Serra, 2024)

Skalowalność projektu została tak zaprojektowana, żeby rozwijała się razem z organizacją. Umożliwia również pracę na danych w czasie rzeczywistym, czego rezultatem jest podejmowanie decyzji bez długiego oczekiwania na wyniki. Lepsza wydajność, elastyczność oraz zwiększone bezpieczeństwo stają się priorytetem tej architektury.

Zalety - Data Fabric
– Zwiększone bezpieczeństwo danych i dostępu do baz danych
– Integracja danych z różnych źródeł i obsługa wielu typów danych
– Możliwość pracy w czasie rzeczywistym, co skraca czas reakcji biznesowej
– Wysoka skalowalność, dostosowująca się do rozwoju organizacji
– Ułatwiona wizualizacja i analiza danych

Tabela 3. Zalety Data Fabric

źródło: opracowane na podstawie: Nowoczesne architektury danych: przewodnik po hurtowni danych, siatce danych oraz Data Fabric i Data Mesh.

Zalety - Data Mesh
– Zdecentralizowana struktura zarządzania danymi
– Dane traktowane jako produkt, co zwiększa ich jakość i użyteczność
– Automatyzacja dostarczania infrastruktury i zgodność z regulacjami
– Własność danych przypisana konkretnym zespołom, co zwiększa odpowiedzialność i specjalizację

Tabela 4. Zalety Data Fabric

źródło: opracowanie na podstawie: Nowoczesne architektury danych: przewodnik po hurtowni danych, siatce danych oraz Data Fabric i Data Mesh.

Niestety wraz z tak szeroką gamą zalet pojawiają się wyzwania, które stają się piętą achillesową przyszłości pracy z danymi. Wysokie koszty wprowadzenia oraz utrzymania, złożoność podczas wprowadzania oraz zależność od dostawcy usługi chmurowej może zastraszyć rozwój architektury. Data Mesh w odróżnieniu od Data Fabric, cechuje się zdecentralizowaną architekturą danych. Dane są traktowane jako produkt, który konsument ma wykorzystać do osiągnięcia swojego celu, dodatkowo zautomatyzowane dostarczanie

infrastruktury oraz nadzór, dzięki któremu dane są bezpieczne oraz zgodne z regulacjami globalnymi. (Serra, 2024)

Kolejną istotną różnicą jest sposób zarządzania w Data Fabric, wszystkie procesy są tworzone przez zespół informatyków będących ich właścicielami. W przypadku Data Mesh wszystkie dane stanowią własność poszczególnych zespołów wewnątrz organizacji, które są odpowiedzialne za gromadzenia, przetwarzania oraz zarządzanie nimi.

Wady - Data Fabric
– <i>Wysokie koszty wdrożenia i utrzymania</i>
– <i>Złożoność architektury, szczególnie na etapie implementacji</i>
– <i>Silna zależność od dostawcy chmurowego</i>

Tabela 5. Wady Data Fabric

źródło: opracowanie na podstawie: Nowoczesne architektury danych: przewodnik po hurtowni danych, siatce danych oraz Data Fabric i Data Mesh.

Wady – Data Mesh
– <i>Wysokie wymagania organizacyjne (kompetencje, odpowiedzialność zespołów)</i>
– <i>Potrzeba zmiany kultury organizacyjnej, trudność we wdrożeniu w dużych firmach</i>
– <i>Potencjalna niespójność danych między zespołami</i>

Tabela 6. Wady Data Mesh

źródło: opracowanie na podstawie: Nowoczesne architektury danych: przewodnik po hurtowni danych, siatce danych oraz Data Fabric i Data Mesh.

Przyszłością architektur danych jest sposób obsługi danych, wprowadzania, administracji oraz ciągłego skalowania w celu dogonienia wymagań klientów na świecie. Mimo zwiększającej się wiedzy, ludzie potrzebują wsparcia w pracy z danymi. Automatyzacja oraz sztuczna inteligencja są w stanie odciążyć w pracy związanej z zarządzaniem danymi.

1.9.1 Automatyzacja i sztuczna inteligencja w zarządzaniu danymi

DataOPS, czyli Data Operations oraz MLOps czyli Machine Learning Operations to nowoczesne podejścia do zarządzania danymi przy pomocy modeli uczenia maszynowego, którym celem jest zwiększenie efektywności oraz analiz danych, jednocześnie wdrażając modele w środowiska produkcyjne.

DataOps jest to sposób na automatyzację procesu pozyskiwania danych, co więcej może pomóc podczas transformacji oraz analizach danych. DataOps jest również używany w celu zarządzania danymi, czyszczeniem, utrzymaniem oraz kontrolowaniem jakości i integracji danych, mimo, że jest dopiero w stanie ciągłego rozwoju. (Databricks, 2024)

MLOPS to skrót od Machine Learning Operation, proces skupia się na automatyzacji, przy pomocy zapożyczania modeli uczenia maszynowego, wdrażania ich oraz utrzymywania w organizacji oraz łączenia z operacjami informatycznymi.

Według artykułu DataBricks, pożytkiem z Mlops jest skalowalność oraz redukcja ryzyka. MLOPS umożliwia zespołom zajmujących się danymi usprawnienie i przyspieszenie wdrażania uczenia maszynowego. Modele uczenia maszynowego wymagają aktywnej kontroli, zarządzania oraz optymalizacji. (Databricks, 2024)

MLOPS oraz DataOps powinny być wykorzystywane przy tworzeniu architektury danych jednocześnie, ponieważ pozytywnie wpływają na siebie. DataOps odpowiada za przetwarzanie, jakość oraz dostępność danych z kolei Mlops skupia się na wdrażaniu i utrzymaniu modeli uczenia maszynowego. W celu maksymalizacji potencjału modeli, kluczowym aspektem jest dostarczanie wysokiej jakości danych. Dodatkowo, integracja między dwoma procesami jest wpływowa, ponieważ stały dostęp do danych, które są automatycznie przetwarzane ze stanu surowego na strukturalne, zapewniają stabilność systemu. (Databricks, 2024)

DataOps działa również jako kontroler danych, czego rezultatem jest zapewnienie wysokiej jakości danych przez cały czas.

Organizacje mierzą się również z wyzwaniami regulacyjnymi, które stają się łatwiejsze do przestrzegania dzięki synergii tych dwóch procesów, zapewniają ład oraz porządek.

1.9.2 Wykorzystanie chmury obliczeniowej

Chmura obliczeniowa to serwis obliczeniowy oferowany przez firmy zewnętrzne oraz modyfikowalna dostępność, która zależy od zleceniodawcy, nie jak w przypadku serwerów fizycznych, gdzie firma generuje dodatkowe koszty za utrzymanie serwera, gdy jest nieużywany.

Pięcioma głównymi filarami w wykorzystaniu chmury obliczeniowej do przetwarzania danych są: pula zasób jako dostępność dla każdego użytkownika, który jest zarejestrowany. Wirtualizacja, czyli wykorzystywanie sprzętu w jak najbardziej efektywny sposób. Elastyczność, która polega na skalowaniu dynamicznym bez zwiększenia wydatków inwestycyjnych, naliczanie opłat, które są uzależnione od realnego wykorzystywania, co oznacza, że za niewykorzystane zasoby nie są pobierane dodatkowe koszty. Ostatnim filarem

jest automatyzacja, która w głównej mierze skupia się na budowaniu, wdrażaniu, konfiguracji, zabezpieczeniu oraz zarządzaniu bez ingerencji ludzkiej. (Mateos, Rosenberg, 2012) Różne typy chmur, mają zastosowanie w różnych sektorach gospodarczych. Istnieją chmury publiczne, prywatne oraz hybrydowe.

Chmura publiczna to dostarczanie usług przez zewnętrznych dostawców, której zasoby IT są udostępniane między wszystkimi użytkownikami. Zaletami jej są: skalowalność, ponieważ w odróżnieniu od serwera fizycznego, nie znajduje się ona w zasięgu odpowiedzialności organizacji korzystającej z usług, modyfikowanie jest bardzo łatwe i trwa bardzo krótki czas. Płaci się tylko za wykorzystane zasoby oraz nie ma konieczności zarządzania infrastrukturą. Wadami są mały wpływ nad kontrolą danych oraz bezpieczeństwa, dodatkowo w godzinach szczytu mogą występować opóźnienia. (Mateos, Rosenberg, 2012)

Chmura prywatna w odróżnieniu od publicznej jest wykorzystywana przez jedną organizację w jej centrum danych lub również przez zewnętrznego dostawcę jak w przypadku chmury publicznej. Zaletami wiodącymi tej chmury jest pełna kontrola nad danymi oraz bezpieczeństwa, dostosowywanie infrastruktury do specyfikacji modelu biznesowego klienta oraz wyższy poziom zgodności z regulacjami w przypadku polski np. RODO. Wadami są, większe koszty utrzymania oraz wdrożenia do organizacji oraz zarządzanie, które w odróżnieniu od chmury publicznej leży w rękach specjalistów z organizacji klienta.

Chmura hybrydowa, łączy chmurę publiczną i prywatną, czego rezultatem jest możliwość przechowywania danych wrażliwych w chmurze prywatnej a mniej istotnych procesów bądź zasobów w chmurze publicznej. Zaletami jest elastyczność, która wyróżnia się możliwością wyboru chmury, która jest bardziej opłacalna w danym procesie, bezpieczeństwo, które jest takie same jak w chmurze prywatnej jednak jest możliwość optymalizacji kosztów oraz wydajności, które były wadami nieoptymalizowanymi w przypadku chmury prywatnej. Wadami chmury hybrydowej jest złożony proces zarządzania chmurą oraz integracji obu chmur. Ostatnią wadą jest zapotrzebowanie na zaawansowane narzędzia do monitorowania oraz synchronizacji danych. (Mateos, Rosenberg, 2012)

Integracja chmury obliczeniowej z strategiami biznesowymi stała się przymusowym komponentem, który musi zostać wdrożony jednocześnie z powodu szybkich zmian w świecie cyfrowym. Organizacje, które wykorzystują jej potencjał, stają się liderami w branży oraz będą przygotowani na wyzwania mogące się pojawić w przyszłości z danymi.

1.10 Podsumowanie

W pierwszym rozdziale przedstawione zostały główne aspekty architektury danych, jej definicje, znaczenia oraz wyzwania z jakimi muszą się mierzyć przedsiębiorstwa. Architektura

danych jest ważnym elementem architektury korporacyjnej, w której dane są traktowane jako bogata infrastruktura, która wymaga odpowiedniego zarządzania oraz ochrony.

Definicja architektury danych obejmuje struktury, procesy i technologie, które po połączeniu umożliwiają efektywne przechowywanie, przetwarzanie oraz analizy danych. Integracja danych w organizacjach jest filarem do zapewnienia pełnego potencjału danych.

Problemy związane z architekturą danych, takie jak magazynowanie oraz pozyskiwanie danych w odpowiednim czasie oraz zwiększająca się ilość danych w zatrważająco szybkim tempie, które muszą być odpowiednio zarządzane. Zapewnianie wysokiej jakości i spójności danych oraz integracja ich w różnych systemach organizacyjnych to praktyki, które są identyfikowane jako fundamentalne wyzwania.

W kontekście rosnącego znaczenia ochrony danych osobowych (RODO), kwestie bezpieczeństwa i prywatności stają się priorytetowym elementem zarządzania danymi.

Nowoczesne podejście do architektury danych uwzględniając automatyzację i wykorzystanie sztucznej inteligencji, zwłaszcza w obszarze DataOps i MLOps, które wspierają zarządzaniem cykl życia danych oraz procesy analityczne. Dodatkowo, technologie chmurowe zyskują na znaczeniu w prowadzeniu biznesu, umożliwiając organizacjom elastyczność w zarządzaniu oraz skalowaniu danych w zależności od potrzeb.

Podsumowując, architektura danych stanowi fundament skutecznego zarządzania danymi, umożliwiając im dostosowanie się do zmieniających się technologii i wyzwań rynkowych.

Rozdział 2. Architektura Lambda i Kappa – struktura, zalety i porównanie

2.1 Charakterystyka architektur lambda i kappa

Współczesne systemy przetwarzania danych muszą sprostać rosnącym oczekiwaniom w analizie na dużych zbiorach danych w czasie rzeczywistym. Architektura Lambda i Kappa są uznawane za popularne w tej dziedzinie podejścia, głównie w podejściu systemów do przetwarzania danych strumieniowych. Mimo, że oba podejścia służą temu samemu rezultatowi, ich budowa oraz sposób działania różnią się od siebie w znaczący sposób. W niniejszym rozdziale zostaną opisane obie architektury wraz z ich charakterystykami oraz zaletami i wadami.

Celem rozdziału jest wprowadzenie do szczegółowych analiz mechanizmów funkcjonowania architektur Lambda i Kappa, zalety, wady oraz różnicę między nimi. Przedstawienie charakterystyk architektur umożliwi lepsze zrozumienie używania ich oraz wzrost popularności w środowisku zarządzania danymi.

Architektura Lambda, wprowadzona przez Nathana Marza, opiera się na kooperacji warstwy wsadowej z warstwą strumieniową, umożliwiając równocześnie wysoką dokładność danych historycznych jak i dostępność wyników w czasie rzeczywistym. Z kolei architektura Kappa, zaproponowana przez Jaya Krepsa, upraszcza ten model poprzez wykorzystywanie jedynie warstwy strumieniowej, redukując tym złożoność systemu.

W dalszych częściach rozdziału, szczegółowo przedstawione zostaną mechanizmy funkcjonowania, zastosowania oraz wskazanie mocnych i słabych stron architektur.

2.1.1 Podstawowe założenia architektury lambda

Architektura Lambda umożliwia przetwarzanie dużych zbiorów danych (Big Data) oraz jednoczesną analizę zarówno danych wsadowych, jak i strumieniowych w czasie rzeczywistym. Celem tej architektury jest zapewnienie zarówno precyzyjnych wyników na podstawie pełnych zbiorów danych, jak i natychmiastowe odpowiedzi na nowe informacje. Ten typ architektury, został zaprojektowany głównie dla systemów analitycznych działających na dużych

wolumenach danych. Wcześniej, gdy klient chciał wydobyć informacje z obszernego zbioru danych, musiał długo czekać na wynik, który po kilkugodzinnym generowaniu mógł być już nieaktualny. (Databrick)

Gdy pojawia się wiele źródeł danych – takich jak hurtownie danych, relacyjne bazy danych, pliki statystyczne czy dzienniki serwerów – dane są zbierane i przekazywane do systemu przetwarzania Lambda. Następnie dane trafiają do magazynu, gdzie są przechowywane w rozproszonych plikach o różnych formatach. Takie repozytoria, określane jako „data lake”, są dostępne w usługach chmurowych, takich jak AWS czy Microsoft Azure.

Jednocześnie może wystąpić potrzeba przetwarzania i analizy danych w czasie rzeczywistym, zwłaszcza gdy rozwiązanie obejmuje źródła danych strumieniowych, wymagające natychmiastowego magazynowania i analizy. Podstawą tej metody jest podział na trzy warstwy:

1. Warstwę wsadową (*batch layer*), która jest odpowiedzialna za przetwarzanie dużych zbiorów danych w trybie wsadowym.
2. Warstwę strumieniową (*speed layer*), obsługującą dane strumieniowe w czasie rzeczywistym
3. Warstwę serwującą (*serving layer*), która udostępnia wyniki użytkownikom oraz aplikacjom końcowym, w celu szybkiego dostępu do przetworzonych danych.

Dzięki tej strukturze architektura Lambda łączy dokładność przetwarzania wsadowego z szybkością analizy strumieniowej, umożliwiając efektywne podejmowanie decyzji na podstawie aktualnych danych. W celu lepszego zrozumienia działania poszczególnych warstw, warto przeanalizować poszczególne komponenty i ich współdziałanie.

2.1.2 Warstwa wsadowa

Warstwa wsadowa odpowiada za przetwarzanie danych – strumień informacji napływa w stałym tempie i wymaga sprawnej obróbki, aby uniknąć zatorów na etapie przetwarzania. W ramach batch processingu operacje są wykonywane na całym zbiorze danych, co wydłuża czas przetwarzania, ale pozwala uzyskać pełny obraz historyczny i wysoką wydajność. Dane w tej warstwie nie podlegają usunięciu – można jedynie aktualizować istniejące rekordy, co ułatwia analizę danych przetworzonych w architekturze Lambda.

Według Denny'ego Lee z DataBricks: „Ta warstwa jest budowana przy użyciu wstępnie zdefiniowanego harmonogramu, zwykle raz lub dwa razy dziennie. Warstwa wsadowa ma dwie bardzo ważne funkcje: zarządzanie zestawem danych głównych oraz wstępne obliczanie widoków wsadowych”. Oznacza to, że głównym zadaniem tej warstwy jest nie tylko

zarządzanie zestawem danych, ale również uporządkowanie, czego rezultatem jest usprawnienie dalszych analiz. (Databricks)

Przykładem praktycznego zastosowania warstwy wsadowej jest system reklamowy Amazon Ads, który raz w ciągu doby zbiera i aktualizuje ogromne ilości danych o użytkownikach. W skład tych danych wchodzi m.in. historia zakupów, wyszukiwania, aktywność na stronie oraz opinie. Następnie, przy wykorzystaniu algorytmu uczenia maszynowego (ML), system analizuje te informacje, aby prezentować bardziej dopasowane reklamy dla każdego klienta indywidualnie. (Amazon Ads)

Mimo, że warstwa wsadowa zapewnia wysoką dokładność oraz kompletność analiz, główną jej wadą są pojawiające się opóźnienia w dostępie do wyników. W momencie, w którym celem jest szybka reakcja na nowe dane, ważne staje się wykorzystanie podejścia opartego na przetwarzaniu strumieniowym. To właśnie zadanie warstwy szybkościowej, która umożliwia analizę danych w czasie rzeczywistym.

2.1.3 Warstwa strumieniowa

Warstwa strumieniowa zajmuje się wyłącznie danymi czasu rzeczywistego, nie jak w przypadku warstwy wsadowej danymi historycznymi w celu rekompensacji wysokiej latencji aktualizacji warstwy serwującej, oznacza to, że zajmują się redukcją czasu wykorzystywanego przy aktualizacji warstwy przedstawiającej. W przeciwieństwie do warstwy wsadowej, która przetwarza dane w dużych partiach w określonych odstępach czasu, warstwa strumieniowa działa na danych napływających w sposób ciągły. Korzysta ona z modelu inkrementalnego, w którym nowe dane są na bieżąco dodawane do istniejących wyników, a nie przetwarzanie od zera. Tego typu podejście pozwala na uzyskanie danych w czasie rzeczywistym, co jest kluczowe w aplikacjach wymagających szybkiej reakcji ze strony analityków. (Reis, Housley, 2023)

Według Lee (2015) warstwa strumieniowa pełni rolę uzupełniającą względem warstwy wsadowej, ponieważ „obsługuje dane, które nie zostały jeszcze przetworzone w widoku wsadowym ze względu na opóźnienia związane ze specyfiką działania tej warstwy” (Lee, 2015).

Innymi słowy, umożliwia ona na bieżące przetwarzanie jedynie najnowszych danych, co pozwala użytkownikom na natychmiastowy dostęp do aktualnych danych. System jest zdolny do generowania widoków danych w czasie rzeczywistym, stanowiących cenne uzupełnienie pełnych, lecz opóźnionych zestawów informacji generowanych w procesie wsadowym. Ten

rodzaj rozwiązań jest niezwykle ważny w sytuacjach, gdy dostęp do aktualnych danych odgrywa znaczącą rolę w podejmowaniu decyzji. (Twardowski i Ryzko, 2015)

Mimo to należy mieć na uwadze, że prędkość w jakiej warstwa strumieniowa pracuje może prowadzić do niedokładnych rezultatów.

Jednak warstwa ta nie zawiera danych historycznych, co może utrudniać głębszą analizę. Dodatkowo jakość przetwarzanych informacji bywa niższa w porównaniu do danych z warstwy wsadowej oraz dane, które zostały przetworzone przez warstwę wsadową i serwującą, są usuwane z tej warstwy. Z tego powodu warstwa nie jest odpowiednia do przechowywania pełnej historii danych. Mimo to, główną zaletą tej warstwy jest szybki dostęp do najnowszych danych, choć istotne jest, by kontrolować czas ich przechowywania, aby nie stracić ich dostępności. (Reis, Housley, 2023)

Przykładem zastosowania warstwy strumieniowej jest aplikacja Uber Eats, która wykorzystuje ją do lokalizacji najbliższego kierowcy oraz wyznaczenia najkrótszej trasy do celu. Zgodnie z informacjami podanymi przez Uber, firma stworzyła wewnętrzną platformę, AthenaX, do analizy strumieniowej, aby sprostać wymaganiom analitycznym. Platforma ta umożliwia zarówno technicznym, jak i nietechnicznym użytkownikom korzystanie z kompleksowych analiz za pomocą języka SQL. (Uber, 2017)

Warstwa strumieniowa, mimo że nie oferuje dostępu do danych historycznych, umożliwia szybki dostęp do najnowszych danych, co jest filarem poprawnego funkcjonowania aplikacji wymagających szybkich reakcji. Jest to technologia, która zyskała na znaczeniu w erze big data i analizy w czasie rzeczywistym.

2.1.4 Warstwa serwująca

Warstwa serwująca jest uznawana za końcowy etap przetwarzania danych w architekturze Lambda. Dane, które przeszły przez wcześniejsze warstwy przetwarzania, trafiają do niej, gdzie są odpowiednio indeksowane. Dzięki temu czas wyszukiwania tych danych jest minimalizowany, co umożliwia szybkie zapytania ad-hoc.

Jak wskazują Zirija Hasani, Margita Kon-Popovska i Goran Velinov w pracy „Lambda Architecture for Real Big Data Analytics”, „Warstwa serwująca indeksuje widoki wsadowe (zwykle pliki płaskie), co pozwala na szybkie zapytania z niskim opóźnieniem”. Indeksowanie jest więc ważnym procesem umożliwiającym szybkie i efektywne wyszukiwanie danych, które zostały wcześniej przetworzone. (Hasani, Kon-Popovska, Velinov, 2014)

Ponadto, warstwa strumieniowa również trafia do warstwy serwującej, ponieważ jej głównym zadaniem jest dostarczanie gotowych danych do analiz w jak najkrótszym czasie. Jest to jedna z najlepiej zoptymalizowanych warstw w architekturze Lambda. W kontekście technologii wykorzystywanych w warstwie serwującej, popularnymi rozwiązaniami są systemy bazodanowe i narzędzia do przetwarzania danych takie jak SploutSQL, Oracel, HBase, Cassandra czy Storm. Każde z tych narzędzi ma swój unikatowy sposób działania z danymi oraz na danych. (Hasani, Kon-Popovska, Velinov, 2014)

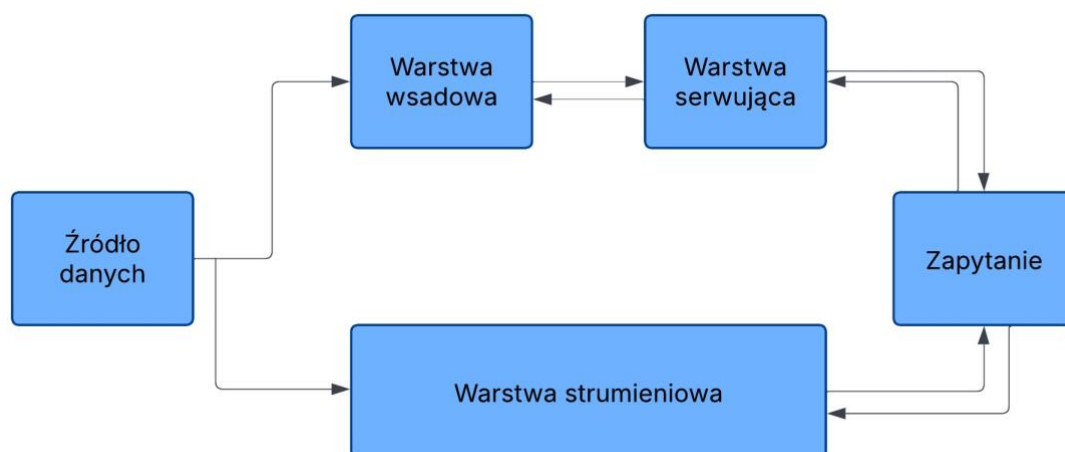
Zgodnie z opisem na stronie Snowflake w artykule „Lambda Architecture”, „Widoki wsadowe są indeksowane, aby mogły być udostępniane do zapytań. Gdy jedno zadanie indeksujące jest w trakcie realizacji, warstwa serwująca ustawia kolejne dane w kolejkach, aby włączyć je do kolejnego procesu indeksowania”. Indeksy są wykorzystywane w celu usprawnienia wyszukiwania pożądaných danych i zmniejszenia obciążeń serwerów z danymi, dodatkowo ustawia kolejne dane w kolejce w celu przygotowania do włączania ich przy kolejnych procesach. (Snowflake)

Podsumowując, warstwa serwująca pełni rolę „portu”, w którym dane przetworzone w wcześniejszych warstwach, jak i potwierdzenie ich przetworzenia, spotykają się i są gotowe do dalszego wykorzystania w analizach i aplikacjach.

2.2 Integracja warstw w architekturze Lambda

Integracja warstw w architekturze Lambda jest głównym filarem, który umożliwia skuteczne oraz spójne funkcjonowanie systemu. W architekturze tej współistnieją trzy warstwy przetwarzania danych: wsadowa (batch), strumieniowa (speed) oraz warstwa serwująca (serving). Zadaniem warstwy serwującej jest zapewnienie integracji wyników przetwarzania z warstw strumieniowej oraz wsadowej w sposób, który pozwoli końcowemu użytkownikowi na otrzymanie jednolitego i kompletnego zestawu danych, który jest dostępny praktycznie w czasie rzeczywistym. (Marz i Warren, 2015).

Integracja tych warstw odbywa się poprzez ciągłej synchronizacji danych pochodzących z dwóch niezależnych od siebie źródeł. Warstwa wsadowa, skupia się na generowaniu danych dokładnych, lecz z opóźnieniem ze względu na specyfikę wsadowego przetwarzania dużych zbiorów danych. Z drugiej strony warstwa strumieniowa dostarcza informacje natychmiast, jednak może generować dane z marginesem błędu bądź niedokładnością wynikającą z konieczności szybszego przetwarzania. (Kiran i in., 2015)



Rysunek 9. Proces działania architektura lambda

źródło: opracowanie na podstawie: Microsoft, *Big data architectures*

Proces funkcjonowania Architektury Lambda został przedstawiony na Rysunku 3. Pierwszą fazą poprawnego działania jest przetwarzanie danych. Dane ze źródeł są wysyłane jednocześnie do warstwy wsadowej, w której dane są przetwarzane wolniej i nie ma dostępu do nich w bardzo szybkim czasie, jednak ich jakość i rzetelność jest na bardzo wysokim poziomie, co więcej umożliwiają sprawdzanie danych historycznych. (Marz i Warren, 2015)

W warstwie strumieniowej (speed) w odróżnieniu od warstwy wsadowej, zapewnia przetwarzanie danych na bieżąco, umożliwia dostęp do rezultatów w czasie rzeczywistym. Mimo, że ten sposób jest wykorzystywany tymczasowo, ponieważ daje dostęp do aktualnych danych w ograniczonym czasie, wiąże ze sobą możliwe ryzyka niedokładności oraz błędy ze względu ograniczeń na pełną analizę. Wyniki przetwarzania strumieniowego mają charakter tymczasowy i zostają zastąpione pełniejszymi danymi po zakończeniu procesu wsadowego. (Twardowski i Ryzko, 2015)

Warstwa serwująca pełni rolę integrującą dane z warstwy strumieniowej oraz wsadowej. Jej głównym zadaniem jest dostarczenie danych z obu warstw, tworząc jednolity zestaw danych, który uwzględnia zarówno aktualne dane z warstwy strumieniowej, jak i dane historyczne z warstwy wsadowej. Warstwa ta rozpoczyna proces w momencie otrzymania konkretnego zapytania od użytkownika, tworząc analizy danych lub ich wizualizacje zgodnie z jego wymaganiami. (Twardowski i Ryzko, 2015)

Proces integracji danych w architekturze Lambda jest niezwykle istotny, ponieważ umożliwia połączenie optymalne szybkiego dostępu do informacji z dokładnością do analiz historycznych. Jednak implementacja tego procesu napotyka wyzwania, takie jak duplikujące dane

pochodzących z dwóch różnych źródeł lub rozwiązywanie konfliktów. W praktyce stosuje się mechanizmy synchronizacji, które pozwalają utrzymać spójność wyników końcowych. (Marz i Warren, 2015)

Przykład implementacji architektury Lambda w kontekście analizy danych sieciowych w Energy Science Network (ESnet). Sieć komputerowa Esnet, która cechuje się wysoką przepustowością, wykorzystywana jest na potrzeby naukowców z USA. Została stworzona w celu monitorowania i analizy ruchu sieciowego, zastosowana została architektura Lambda, aby zapewnić zarówno dokładność i dostęp do danych historycznych – warstwa wsadowa oraz zredukowanie opóźnień w analizie danych. (Kiran, Monga, 2015)

Ta implementacja architektury umożliwiła ESnet na efektywne zarządzanie dużymi wielkościami danych sieciowych. Podejście te pozwoliło na sprawne identyfikowanie oraz reagowanie na odchylenia w ruchu sieciowym, co jest filarem do utrzymywania odpowiedniego poziomu jakości usług sieciowych. (Kiran, Monga, 2015)

Podsumowując, integracja warstw w architekturze Lambda pozwala wykorzystywać efektywnie przetwarzanie wsadowe oraz strumieniowe. Mimo, że jest to proces złożony, który wymaga odpowiedniego zarządzania. Integracja ta, mimo swoich zalet, posiada również ograniczenia i potencjalne wady, które zostaną omówione w następnych w rozdziale.

2.3 Zalety i wady architektury Lambda

W ostatnim latach wzrosło zapotrzebowanie systemów przetwarzania dużych zbiorów danych, które umożliwiają równoczesną analizę oraz wizualizacje danych strumieniowych i wsadowych. W tym problemie, ważną rolę odgrywa Architektura Lambda, która ma na celu usprawnienie obsługi systemów danych, zredukowanie błędów ludzkich oraz dążenie do wyższej spójności i efektywności przetwarzania danych w czasie rzeczywistym.

Nathan Marz stworzył architekturę Lambda, w celu uproszczenia operacji związanych z zarządzaniem dużych zbiorów danych, których budowa często jest skomplikowana z powodu systemów wykorzystujących jedynie inkrementalnego, strumieniowego przetwarzania. Ograniczenie problemów, takich jak kompaktowanie danych w czasie rzeczywistym lub zarządzanie spójnością danych w systemach charakteryzujących się wysoką dostępnością.

Marz uwzględnił podczas projektowania architektury odporność systemu na błędy ludzkie, takie jak niezamierzone usunięcie zbioru danych lub inne operacje, które mogły naruszyć integracje danych. (Behera, 2022)

Zalety architektury Lambda
<i>Usprawnia obsługę dużych zbiorów danych</i>
<i>Redukuje ryzyko błędów ludzkich, np. przypadkowe usunięcia danych</i>
<i>Zapewnia wysoką spójność i efektywność przetwarzania danych w czasie rzeczywistym</i>
<i>Umożliwia analizę zarówno danych strumieniowych, jak i historycznych</i>
<i>Oferuje wysoką dokładność oraz większą przepustowość przy jednoczesnym zmniejszeniu opóźnień</i>
<i>Wysoka odporność i niezawodność dzięki wykorzystaniu rozproszonych systemów plików</i>
<i>Każda warstwa systemu może być skalowana niezależnie</i>
<i>Elastyczność i możliwość wdrożenia przy stosunkowo niskim koszcie utrzymania</i>

Tabela 7. Zalety architektury Lambda

źródło: opracowanie na podstawie: Rajat Kumar Behera, *Big Data Architectures: A detailed and application oriented analysis*

Wady architektury Lambda
<i>Wymaga ciągłej synchronizacji między warstwą wsadową a strumieniową</i>
<i>Potencjalne błędy operacyjne związane z integracją warstw</i>
<i>Brak obsługi operacji transakcyjnych w warstwie serwującej</i>
<i>Konieczność utrzymywania dwóch baz kodu (dla batch i stream)</i>
<i>Większe wymagania dotyczące kompetencji zespołu technicznego</i>
<i>Duplikacja logiki przetwarzania, co zwiększa złożoność i koszty utrzymania</i>

Tabela 8. Wady architektury Lambda

źródło: opracowanie na podstawie: Rajat Kumar Behera, *Big Data Architectures: A detailed and application oriented analysis*

Porównując typowe przyrostowe rozwiązania, architektura Lambda umożliwia uzyskiwanie wyższej dokładności, zwiększoną przepustowość oraz zmniejszone opóźnienia jednocześnie, zachowując przy tym ich spójność. Zaletą systemów opartych na LA jest odporność oraz niezawodność, które zapewniają przechowywanie danych w rozproszonym systemie plików. Operacja ta w znaczącym stopniu redukuje ryzyko pojawiania się błędów powstających z działań ludzkich. Dodatkowo, każda warstwa architektury jest w stanie niezależnie skalować się, czego rezultatem jest zwiększona elastyczność oraz wykorzystanie w problemach przy niewielkim nakładzie na konserwację. (Behera, 2022)

Architektura Lambda pozwala na przeprowadzanie analiz zarówno w czasie rzeczywistym, poprzez szybkie zapytania danych strumieniowych, jak i szczegółowych analiz historycznych danych przechowywanych w warstwie wsadowej. Wyzwaniem przy stosowaniu architektury Lambda staje się konieczność regularnej synchronizacji między warstwą strumieniową a warstwą wsadową, co może prowadzić do błędów operacyjnych. Drugim ograniczeniem jest wykorzystywanie jedynie zapytań analitycznych, wykluczających operacje transakcyjne w warstwie serwującej. Problemem staje się utrzymanie dwóch oddzielnych baz kodu dla warstwy

wsadowej oraz strumieniowej, które działają na zbliżonej zasadzie na różnych zestawach danych, co powoduje nadmierny poziom pracy specjalistów w branży programistycznej oraz wymaga wysokich kompetencji personelu w obu obszarach przetwarzania danych. (Behera, 2022).

2.4 Podstawowe założenia architektury Kappa

Architektura Kappa, która jest uznawana za uproszczoną wersję architektury Lambda, została zaproponowana przez pracownika LinkedIn Jaya Krepsa w 2014 roku. W odróżnieniu od architektury Lambda, w architekturze Kappa zrezygnowano z warstwy wsadowej, na rzecz całkowitego przesyłania danych za pomocą warstwy strumieniowej, co umożliwia szybszy i bardziej bezpośredni przepływ informacji. (Ounacer, 2017)

W Systemach opartych na architekturze Kappa, zrezygnowano z zapisywania danych w klasycznych relacyjnych bazach danych (SQL) oraz z magazynów klucz – wartość (Cassandra). W odróżnieniu od architektury Lambda, stosuje się niezmienny, dołączany na końcu dziennik (append-only log), który stanowi rolę głównego repozytorium danych. Z dziennika, dane są przesyłane do systemu obliczeniowego, które później przemieszczane są do pomocniczych magazynów danych używanych w warstwie serwującej. (Ounacer, 2017)

Architektura Kappa w odróżnieniu od architektury Lambda jest przeznaczona głównie do przetwarzania danych i nie umożliwia użytkownikowi przechowywania danych permanentnie. Dodatkowo redukuje złożoność systemu z powodu usunięcia warstwy wsadowej, dzięki czemu ograniczana się do jednej ścieżki kodu. Architektura Kappa przedstawiona na rysunku 11 jest wykonana z dwóch warstw, warstwy strumieniowej oraz serwującej, która jest wykorzystywana do wykonywania zapytań do wyników. (Behera, 2022)

W celu lepszego zrozumienia działania warstw oraz funkcjonowania architektury Kappa, szczegółowo zostaną opisane poszczególne warstwy, sposób działania, zalety, wady, przykłady oraz porównanie architektur Kappa oraz Lambda.

2.4.1 Warstwy w architekturze Kappa

W odróżnieniu od wcześniej omawianej architektury Lambda, w której występowały trzy warstwy: wsadowa, strumieniowa oraz serwująca, w przypadku Kappy zrezygnowano z warstwy wsadowej. Architektura Kappa została stworzona w celu przetwarzania danych w czasie rzeczywistym, co oznacza, że zdarzenia są przetwarzane natychmiast po ich otrzymaniu, a nie w późniejszym terminie w trybie wsadowym. To działanie zostało wprowadzone na rzecz zmniejszenia opóźnień oraz umożliwienie systemowi na szybką reakcję w przypadku zmieniających się warunków. (Serra, 2024)

2.5 Integracja warstw w architekturze Kappa

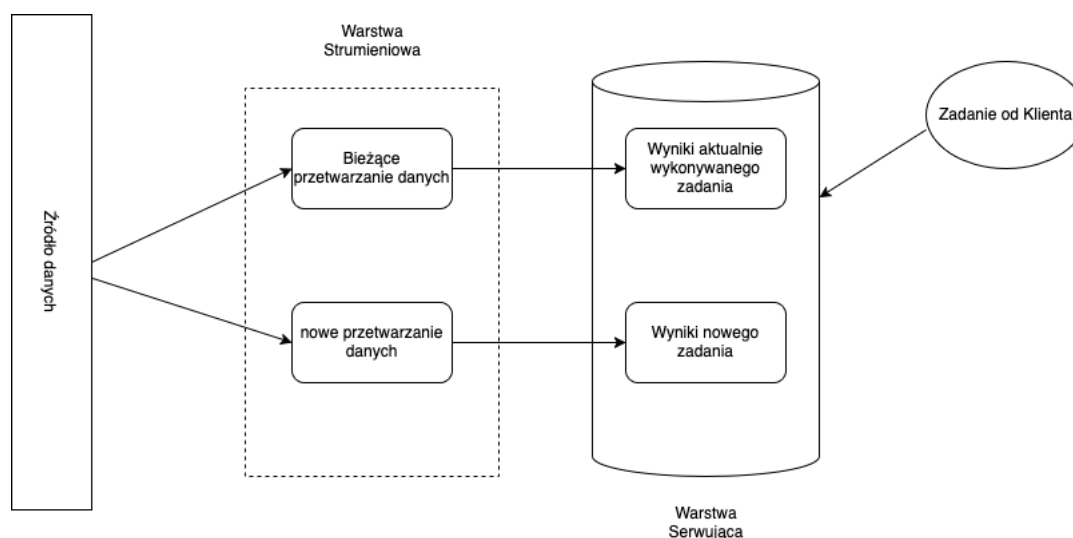
Architektura Kappa jest uznawana jako uproszczenie architektury Lambda. W odróżnieniu od architektury Lambda, gdzie występuje warstwa wsadowa, która jednocześnie z warstwą strumieniową przetwarzały dane w dwóch oddzielnych kodach, dane w Kappie są wysyłane do warstwy strumieniowej są przesyłane w szybkim tempie, umożliwiającym na odczytywanie danych w czasie rzeczywistym. (Hassan, Hassan, 2022)

Dane w momencie przyjęcia ze źródła danych do strumieniowej warstwy zostają wpisane do specjalnego dziennika (np. Apache Kafka log), który dopisuje dane, ale nie nadpisuje ich. Dzięki tej opcji w przypadku ponownego przetworzenia danych np. podczas zmiany algorytmu, wystarczy odczytać log z początku. (Hassan, Hassan, 2022)

Przykładem praktycznym append-only jest system do monitorowania zakupów w sklepie online, gdzie klient kupując produkt, zostaje dopisany do dziennika. W momencie dodania, zamiast usuwania wcześniejszych pozycji, nowe zdarzenie zostaje dopisane na koniec listy z id klienta, w przypadku zwrotu nie kasujemy pozycji kupna a dodajemy nową wiadomość na temat usunięcia zlecenia.

Trzema kluczowymi zasadami architektury Kappa to, przetwarzanie w czasie rzeczywistym – jest to możliwe dzięki usunięciu warstwy wsadowej, w celu zmniejszenia opóźnień i szybsze reagowanie na zmieniające się warunki. Pojedynczy strumień zdarzeń, który przechowuje wszystkie dane oraz umożliwia łatwe skalowanie, oraz przetwarzanie bezstanowe, który oznacza, że dane są przetwarzane niezależnie, bez uwzględniania wcześniejszych zdarzeń, ponieważ nie ma powodu do utrzymywania stanu między wieloma węzłami. (Serra, 2024)

Warstwy w architekturze Kappa są takie same jak w przypadku architektury Lambda z różnicą ilości warstw, ponieważ w architekturze Kappa występują tylko dwie bez warstwy wsadowej występującej w architekturze Lambda.



Rysunek 10. Działanie architektury Kappa

źródło: opracowanie na podstawie: Alaa Abdelraheem Hassan and Tarig Mohammed Hassan, 2022, *Real-Time Big Data Analytics for Data Stream Challenges: An Overview*

Przykładem implementacji architektury Kappa jest zastosowanie jej w rolnictwie precyzyjnym. W celu zwiększenia wydajności przetwarzania danych przez urządzenia IoT (np. maszyny rolnicze, roboty, sieci czujników) zwrócono szczególną uwagę na Apache Kafka, wykorzystywanego w procesie jako kolejka komunikatów. Podczas dostosowywania parametrów, takich jak podział pamięci RAM oraz okres zatwierdzania offsetu, które mają istotny wpływ na szybkość przetwarzania danych. Porównane zostały również kombinacje oprogramowań przetwarzających takich jak Apache Samza, Apache Storm oraz baz danych Apache Druid i Apache HBase w celu wybrania najefektywniejszego rozwiązania dla najlepszych rezultatów rolniczych. (Penka, 2021)

Rozumiejąc mechanizm działania architektury Kappa, ważnym zadaniem jest zauważanie zalet oraz wad architektury w celu poprawnego wyboru przy implementacji odpowiedniej architektury w organizacjach.

2.6 Zalety i wady architektury Kappa

Podczas wyboru najefektywniejszej architektury do przetwarzania danych, duży wpływ na funkcjonowanie systemu, jednocześnie pod względem wydajnościowym, jak i utrzymanie ma odpowiednia budowa. Architektura Kappa, jest uznawana jako jedna z nowszych rozwiązań, popularność uzyskała dzięki odejściu od złożoności systemu w porównaniu do architektury Lambda. Zrezygnowanie z warstwy wsadowej dało rezultat, który ogranicza złożoność, ułatwia implementacje oraz obniża koszty.

Niestety, mimo tak rozszerzonego planu, architektura Kappa również nie jest wolna od ograniczeń. Zależnie od miejsca oraz czasu implementacji, mogą pojawiać się konkretne korzyści, jak i również powstawać wady.

Potencjalnym wyzwaniem na drodze architektury Kappa to złożoność. Jest to spowodowane rezygnacją z warstwy wsadowej, co oznacza również rezygnację z dwóch oddzielnych ścieżek kodu. Wdrażanie przetwarzania bezstanowego jest trudnym przedsięwzięciem dla wyspecjalizowanej kadry, a w momencie dodania do tego procesu utrzymania, staje się bardzo zaawansowanym zadaniem w odróżnieniu od architektury Lambda. (Serra, 2024)

Ograniczenie architektury do warstwy strumieniowej tworzy problem przy obsługiwaniu danych historycznych, mimo posiadania logów dla każdej wartości danych, niektóre firmy z powodu złożoności odzwierciedlenia warstwy wsadowej, warstwą strumieniową, decydują się na przejście do rozwiązań architektury Lambda. (Serra, 2024)

Kolejnym wyzwaniem dla architektury Kappa jest możliwość wykonywania jedynie operacji analitycznych, a nie transakcyjnych. Przez wykorzystywanie jedynie warstwy strumieniowej, klasyfikuje się ten typ przetwarzania danych jako z długim czasem życia (TTL – Time to Live), czego skutkiem jest brak możliwości wykorzystywania natywnych usług chmurowych, oznacza to, że nie ma możliwości automatycznego skalowania, elastyczności oraz dostępności na żądanie w rezultacie nie może wykorzystywać gotowych usług chmurowych. (Behera, 2022)

Ostatnią wadą dla architektury Kappa są zapytania ad-hocowe. Z powodu zaprojektowania architektury z myślą o przetwarzaniu danych w czasie rzeczywistym niektóre ad-hocowe polecenia, które wymagają wykonania zapytania na dużym obszarze danych historycznych mogą być niemożliwe, ponieważ dane są dostępne przez ograniczony czas. Nawet jeżeli dane są dostępne, to Kappa nie jest przygotowana pod kątem ciężkich, złożonych zapytań analitycznych. (Behera, 2022)

Wady
<i>Wysoka złożoność wdrożenia i utrzymania bezstanowego przetwarzania danych</i>
<i>Brak warstwy wsadowej utrudnia pełne odtworzenie danych historycznych</i>
<i>Trudności w przetwarzaniu dużych zbiorów danych z przeszłości w warstwie strumieniowej</i>
<i>Obsługa wyłącznie operacji analitycznych, brak wsparcia dla operacji transakcyjnych</i>
<i>Brak zgodności z natywnymi usługami chmurowymi (ograniczenia TTL, brak elastyczności i skalowalności)</i>
<i>Ograniczona możliwość realizacji zapytań ad hoc, zwłaszcza dla dużych zbiorów danych historycznych</i>
<i>Niska wydajność w przypadku złożonych zapytań analitycznych</i>

Tabela 9. Wady architektury Kappa

źródło: opracowanie na podstawie: Rajat Kumar Behera, 2022, *Big Data Architectures: A detailed and application oriented analysis*

Z drugiej strony architektura Kappa umożliwia zredukowanie złożoności, dzięki rezygnacji z warstwy wsadowej, w której dane historyczne były przechowywane na rzecz warstwy strumieniowej. Jednocześnie ułatwia skalowanie i zwiększenie odporności na pojawiające się błędy, ponieważ dane mogą w łatwy sposób rozproszyć się między wieloma węzłami. (Serra, 2024)

W architekturze Kappa przetwarzanie jest niezależne, oznacza to, że nowe zdarzenia nie polegają na stanie wcześniejszych zapisanych zdarzeń. W rezultacie system cechuje się wysoką skalowalnością systemu z powodu braku obowiązku utrzymywania relacji między wieloma węzłami. (Serra, 2024)

Przetwarzanie w czasie rzeczywistym jest bardzo przydatne w organizacjach zajmujących się bankowością, e-commerce oraz inteligentnymi miastami. Architektura Kappa została ściśle zaprojektowana w celu przetwarzania danych w czasie rzeczywistym, oznacza to, że w momencie otrzymania danych przechodzi się do przetwarzania. Redukuje to pojawiające się opóźnienia oraz pozwala systemowi na szybkie reagowanie w środowisku pojawiających się nowych warunków. (Serra, 2024)

Architektura Kappa umożliwia redukcję obciążenia serwerów, ponieważ dane nie są permanentnie przechowywane w magazynie jak w przypadku warstwy wsadowej, a otrzymują logi, po których na określony czas jest możliwe wyszukiwanie ich. (Serra, 2024)

Zalety
<i>Zmniejszenie złożoności systemu poprzez eliminację warstwy wsadowej</i>
<i>Ułatwione skalowanie i większa odporność na błędy dzięki rozproszeniu danych między węzłami</i>
<i>Niezależne przetwarzanie zdarzeń, bez konieczności śledzenia stanu historycznego</i>
<i>Wysoka skalowalność dzięki brakowi zależności między węzłami</i>
<i>Przystosowanie do przetwarzania danych w czasie rzeczywistym (natychmiastowa reakcja na nowe zdarzenia)</i>
<i>Zmniejszone opóźnienia oraz szybsze reagowanie systemu w dynamicznych warunkach</i>
<i>Redukcja obciążenia serwerów dzięki rezygnacji z trwałego przechowywania danych</i>
<i>Wykorzystanie logów do tymczasowego przechowywania danych, co pozwala na ich wyszukiwanie przez określony czas</i>

Tabela 10. Zalety architektury Kappa

źródło: opracowanie na podstawie: Rajat Kumar Behera, 2022, *Big Data Architectures: A detailed and application oriented analysis*

Podsumowując architektura Kappa dobrze się sprawdza w organizacjach, gdzie szybkie przetwarzanie danych w czasie rzeczywistym jest znaczące. Dzięki rezygnacji z warstwy wsadowej uproszczono złożoność systemu, co skutkuje ułatwieniem w utrzymaniu oraz zredukowanie ryzyka błędu z powodu złożoności kodu. Dodatkowo brak zapotrzebowania na zarządzanie danymi historycznymi pozwala na wyższą skalowalność oraz zwiększoną elastyczność w środowiskach rozproszonych. Nie jest ona jednak wystarczająca w sytuacjach, gdy główną rolę gra analiza danych archiwalnych lub zapotrzebowanie na łączenie różnych trybów przetwarzania. W kolejnej części pracy, przedstawione zostaną główne różnice między architekturą Kappa i Lambda, a także przedstawione zostaną metadane, które ułatwią w określeniu odpowiedniej architektury do scenariuszy.

2.7 Porównanie Lambda vs Kappa

Ciągle zapotrzebowanie oraz rozwój systemów przetwarzania danych, szczególnie działających na dużych wolumenach danych oraz zbiorach informacji, które w szybkim tempie zmieniają

się, doprowadziły do stworzenia różnych architektur. Wśród nich najbardziej popularne to architektury Lambda i Kappa, których różnice są zauważalne na poziomie budowy wewnętrznej jak i sposobem obsługi danych.

Jako pierwsza pojawiła się architektura Lambda, która łączyła przetwarzanie strumieniowe z przetwarzaniem wsadowym. Pozwoli to na osiąganie wysokiej dokładności przy jednoczesnym analizowaniu danych w czasie rzeczywistym. Jednak wraz z rozwijającą się gospodarką, powstały firmy, które w swoich środowiskach stawiały na szybki czas reakcji oraz uproszczenia złożoności systemu. W taki sposób powstała architektura Kappa, która działa wyłącznie na przetwarzaniu strumieniowym, bez konieczności tworzenia oddzielnej ścieżki kodu dla warstwy wsadowej, co skutkowałoby utrudnieniem utrzymywania architektury.

Najważniejsze różnice pomiędzy obiema architekturami zostały przedstawione na tabeli. Analiza została wykonana w celu ułatwienia analiz, których rezultat wspomaga wybór odpowiedniej architektury do problemu oraz przygotuje zbiór metadanych, które wspomogą trenowanie modelu uczenia maszynowego, którego zadaniem będzie podjęcie decyzji wyboru odpowiedniej architektury.

Kryterium	Architektura Lambda	Architektura Kappa
Model przetwarzania	Wsadowy i strumieniowy	Strumieniowy
Złożoność implementacji	Konieczność utrzymania dwóch baz kodu	Utrzymanie jednej ścieżki kodu
Obsługa danych historycznych	Tak – warstwa batch	Ograniczona
Reakcja na dane w czasie rzeczywistym	Wspierana, ale opóźnienie z warstwy batch	Dane przetwarzane od razu
Zapytania ad hoc	Brak ograniczeń	Ograniczona
Skalowalność	Wysoka – zależna od synchronizacji warstw	Wysoka – dzięki bezstanowości i niezależności węzłów
Zastosowania	Analiza historyczna, raportowanie, łączenie wielu źródeł	Monitoring, analiza strumieniowa, systemy wymagające natychmiastowej reakcji
Zgodność z chmurą	Zgodna	Ograniczona
Koszty utrzymania	Wyższe	Niższe

Tabela 11. Porównanie architektur

źródło: opracowanie własne

Tabela 11 prezentuje zestawienie najważniejszych cech, które umożliwiają odróżnienie architektury Lambda i Kappa, uwzględniając takie kryteria jak sposób przetwarzania danych,

złożoność systemu, możliwości analityczne, skalowalność czy zgodność z rozwiązaniami chmurowymi.

Główną różnicą, która wyróżnia architektury od siebie to sposób przetwarzania danych – Lambda opiera się na dwóch niezależnych warstwach: wsadowej oraz strumieniowej, gdy Kappa zredukował to do jedynie warstwy strumieniowej. Ta decyzja przekłada się na złożoność systemów, implementacje oraz utrzymanie ich. W przypadku Lambdy istotne jest utrzymanie dwóch oddzielnych ścieżek kodu, które zwiększają koszt i ryzyko niespójności, gdzie kappa ogranicza się do jednej ścieżki, czego rezultatem jest zmniejszenie czasu potrzebnego na rozwój oraz redukcja kosztów w porównaniu do Lambdy. (Talhaoui, 2017)

Wyróżniającą cechą jest możliwość pracy na danych historyczny w przypadku architektury Lambda. Przy pomocy warstwy wsadowej możliwe jest tworzenie analiz opartych na dużych wolumenach danych archiwalnych. Z drugiej strony, architektura Kappa pozwala na nieutrzymywaniu relacji między przetwarzanymi zdarzeniami, co umożliwia łatwiejsze rozdzielanie obciążeń między węzły systemu. Ta właściwość jest przydatna w środowisku, gdzie ważnym aspektem jest szybka reakcja systemu.

Środowisko chmurowe, które stało się nieodłącznym środowiskiem do pracy z danymi jest jednocześnie kompatybilne z architekturami, jednak Lambda współpracuje z natywnymi usługami wielu dostawców, czyli gotowymi usługami, natomiast Kappa spotyka ograniczenia głównie ze względu na sposób przechowywania danych w logach oraz brak wsparcia dla długiego czasu życia strumienia (TTL).

Podsumowując, powyższa tabela, która stanowi jednocześnie narzędzie umożliwiające analizę porównawczą cech architektur. W kolejnym etapie zostanie przybliżona definicja metadanych, przygotowanie kryteriów, które zostaną wykorzystane jako pomoc przy tworzeniu systemu rekomendacji wyboru architektur.

2.8 Identyfikacja metadanych

W dotychczasowych częściach pracy przedstawione zostały główne charakterystyki rozróżniające architekturę Lambda od Kappy wraz z ich zaletami oraz ograniczeniami. Rozbudowanie analizy przez próbę automatyzacji procesu wyboru odpowiedniej architektury w zależności od cech systemu. Podejście to wykorzystywane jest głównie w złożonych oraz dynamicznych środowiskach, w których manualne podejmowanie decyzji staje się czasochłonne, kosztowne oraz narażone na błędy. Czynnikiem, który odgrywa znaczącą rolę w budowie tej automatyzacji jest rozpoznawanie użytecznych metadanych, które będą wykorzystane jako dane wejściowe w modelu uczenia maszynowego.

2.8.1 Czym są metadane?

Metadane są identyfikowane jako warstwa opisowa, która odnosi się do danych, przy pomocy których umożliwia się rozpoznanie, klasyfikację, wyszukiwanie oraz przetwarzanie. Metadane są definiowane jako ustrukturyzowane informacje opisujące inne zasoby informacyjne, ich kontekst, strukturę, źródło oraz sposób wykorzystania. Określany są również jako „dane o danych” lub „informacje o informacjach”. (European Data Portal, 2021)

Zakres zastosowania metadanych jest obszerny, od plików tekstowych, przez bazy danych, aż po schematy klasyfikacyjne czy jednostki organizacyjne. W środowisku informatycznym, metadane stanowią funkcję porządkującą, która pozwala na automatyzację procesów, kontrole dostępu, przetwarzanie informacji oraz poprawianie jakości danych.

Podczas wyboru odpowiedniej architektury danych, metadane mogą wspomóc podczas wyboru odpowiedniego systemu oraz pełnią rolę cech decyzyjnych, na których podstawie kwalifikuje się dany przypadek do odpowiedniej kategorii: Lambda lub Kappa. W trakcie tego procesu wymagane jest wcześniejsze zidentyfikowanie zmiennych, których wpływ na działanie systemu jest zwiększenie efektywności w zależności od zastosowanego modelu przetwarzania.

2.8.2 Zastosowanie metadanych w klasyfikacji architektur

W niniejszej pracy metadane traktowane są jako opisowe właściwości API lub systemu, które wykorzystuje się jako dane wejściowe do modelu uczenia maszynowego. Celem metadanych jest wsparcie procesu decyzyjnego, którego rezultatem jest dobranie odpowiedniej architektury danych do przetwarzanych informacji. W kolejnym podrozdziale przedstawione zostaną proponowane metadane, które zostaną uwzględnione przy budowie modelu uczenia maszynowego.

2.8.3 Propozycje metadanych

Poniższa tabela przedstawia przykładowe metadane, które mogą zostać wykorzystane przy klasyfikacji odpowiedniej architektury do przetworzonych danych. Zostały one stworzone w oparciu o analizę porównawczą architektury Lambda i Kappa oraz praktycznymi aspektami związanymi z przetwarzaniem danych.

Metadana	Opis	Wpływ na wybór architektury
Liczba zapytań na sekundę	Miara obciążenia systemu – ile żądań przetwarzanych jest w jednostce czasu	Wysokie wartości sprzyjają Kappa
Czas życia danych (TTL)	Okres, przez który dane muszą być dostępne	Długi czas przechowywania – Lambda
Dostęp do danych historycznych	Czy system wymaga analiz danych z przeszłości	Tak – Lambda
Typ przetwarzania	Czy dane są analizowane w trybie strumieniowym, wsadowym czy mieszanym	Mieszane – Lambda; Strumieniowe – Kappa
Częstotliwość aktualizacji danych	Jak często dane są aktualizowane	Częste aktualizacje – Kappa
Typ zapytań	Proste zapytania, złożone analizy ad hoc	Złożone analizy – Lambda
Skalowalność systemu	Wymaga elastyczności rozproszenia danych i mocny obliczeniowej	Wysoka – Kappa
Integracja z usługami chmurowymi	Stopień zależności od platform cloud-native	Duża integracja – korzystniejsza lambda

Tabela 12. Metadane do odróżniania architektur

źródło: opracowanie własne

2.9 Podsumowanie

W rozdziale drugim przeprowadzona została szczegółowa analiza dwóch popularnych podejść do przetwarzania danych – architektury Lambda oraz architektury Kappa. Omówiona została ich struktura, sposób działania, zalety oraz wyzwania stojące przed ich implementacją. Analizy zostały dopełnione praktycznymi zastosowaniami architektur w różnych środowiskach, czego rezultatem jest zrozumienie wyzwań technologicznych, takich jak analiza strumieni danych, przetwarzanie informacji w czasie rzeczywistym czy potrzeba zachowania danych historycznych.

W dalszej części została wykonana analiza porównawcza obu podejść, pokazująca w jakich sytuacjach lepszym wyborem jest architektura Lambda, a kiedy korzystniejszym jest wykorzystanie architektury Kappa.

Porównanie obu architektur zostało przedstawione w tabeli i szczegółowo omówione, które ułatwia zrozumienie omawianego problemu.

Końcowo wprowadzono pojęcie metadanych, przedstawianych jako dane opisujące inne dane, oraz wytłumaczono ich rolę w nowoczesnych systemach informacyjnych. Dodatkowo opisana została wartość metadanych przy budowie modelu uczenia maszynowego. Zaproponowane zostały przykładowe cechy do identyfikacji odpowiedniej architektury do przetwarzanych danych. W kolejnym rozdziale wybrane zostaną najistotniejsze cechy, które umożliwią stworzenie modelu uczenia maszynowego bez przetrenowania go oraz niedotrenowania.

Rozdział 3. Budowa modelu uczenia maszynowego

3.1 Cel i założenia modelu

Budowa efektywnego modelu uczenia maszynowego do rozpoznawania odpowiedniej architektury danych: Lambda lub Kappa przy klasyfikacji przypadków przetwarzania danych. Celem tego narzędzia jest wspomaganie projektantów systemów informacyjnych w podejmowaniu decyzji architektonicznych w kontekście rozwijającego się oraz zmiennego środowiska technologicznego.

Ważnym aspektem podczas budowy modelu było założenie, że zestaw metadanych jest w stanie opisać charakterystykę architektur. W skład metadanych wchodzi cechy takie jak: liczba zapytań na sekundę, przechowywanie danych historycznych, czas życia danych (TTL), sposób przetwarzania, częstotliwość aktualizacji danych, typ zadań, skalowalność systemu oraz integracje z usługami chmurowymi.

W powyższym algorytmie uczenia maszynowego, z uwagi na cel klasyfikacji odpowiedniej architektury z dwóch powyższych, problem przedstawiony został jako problem o charakterze binarnym, dodatkowo będzie to oparte o nadzorowanym uczeniu maszynowym. Model był uczony na syntetycznym zbiorze danych, składającym się z wyżej wymienionych metadanych, które pomogą w rozpoznaniu odpowiedniej architektury dla danego problemu, który każdy unikatowo będzie prezentowany w wierszu. Dane wejściowe przyjmowały wartości cech numerycznych (int) oraz kategoriowych, które następnie zostały poddane procesowi kodowania.

Do klasyfikacji algorytmicznej wybrane zostało drzewo decyzyjne. Model ten umożliwia wysoką skuteczność klasyfikacji oraz łatwe zobrazowanie kroków, które stoją za podjęciem decyzji, takich jak wielkości progów oraz wybranych cech w podejmowaniu decyzji. Zrozumienie drzewa stanowi dodatkową zaletę algorytmu klasyfikacyjnego, szczególnie, gdzie przejrzystość projektu jest bodźcem przewodnim do poprawnej realizacji projektów w momencie, gdy jest on przetwarzany przez inne zespoły i czas gra istotną rolę.

3.1.1 Drzewo decyzyjne – definicja i uzasadnienie wyboru

Drzewo decyzyjne to wszechstronne algorytmy uczenia maszynowego, które są wykorzystywane zarówno do klasyfikacji, regresji jak i operacji wielowyjściowych. Modele uzyskiwane dzięki tym algorytmom wyróżniają się zdolnościami do uczenia się wobec złożonych zbiorów danych. Drzewa decyzyjne są czynnikami wchodzącymi w skład budowy losowych lasów, które są obecnie identyfikowane jako jedno z najlepszych algorytmów uczenia maszynowego. (Geron, 2023)

W odniesieniu do niniejszej pracy, wybranie drzewa decyzyjnego jako algorytm klasyfikujący został uznany za najlepiej dopasowany z powodu kilku czynników. Pierwszym i zarazem najważniejszym powodem jest transparentność algorytmu oraz intuicyjne tworzenie drzewa, co jest ważne w środowiskach technologicznych, gdzie zrozumienie końcowego projektu przez użytkownika gra kluczową rolę. Co więcej są efektywne obliczeniowo i wykorzystywane są przy danych numerycznych jak i kategoriowych, co odpowiada zbiorem danych wykorzystywanych w klasyfikacji architektury.

Dodatkowo, drzewa decyzyjne pozwalają na rozróżnienie najważniejszych metadanych, które mają znaczący wpływ na podejmowanie decyzji, co pomaga w analizie oraz optymalizacji procesów. Struktura drzew pozwala na przekształcenie w zestaw reguł decyzyjnych, które wykorzystywane są przy wprowadzaniu systemów wspomagania decyzji. (Dudek, 2014)

3.2 Dane wejściowe

W trakcie budowy modelu uczenia maszynowego do klasyfikacji, którego założeniem było pogrupowanie analizowanych przypadków do jednej z dwóch architektur: Lambda lub Kappa. Dane zostały wygenerowane syntetycznie przy pomocy środowiska Python z uwzględnieniem opisanych metadanych obu architektur. Struktura modelu oraz sposób przygotowania został zaprojektowany w sposób najlepiej przedstawiający rzeczywiste uwarunkowania, w których wykorzystuje się systemy przetwarzania danych w środowiskach biznesowych oraz technologicznych.

3.2.1 Charakterystyka zbioru danych

Zbiór danych, na którym model był trenowany oraz testowany zawiera 4000 rekordów, gdzie każdy przedstawia hipotetyczny przypadek wdrażania systemu przetwarzania danych. Dane zostały przygotowane w formacie tabelarycznym (CSV – comma-separated values) i zawierają zarówno dane numeryczne jak i kategoriowe. Każdy wiersz, przedstawia zestaw metadanych opisujących dane API, gdzie kolumna docelowa (architektura) przyjmuje wartość 0 dla architektury Lambda i 1 dla architektury Kappa.

Dane kategoriowe zostały przekształcone za pomocą funkcji OneHotEncoder z biblioteki sklearn.preprocessing. Rezultatem tego procesu było przedstawienie zmiennych nominalnych w postaci binarnej, co pozwoliło na efektywniejsze trenowanie modelu klasyfikacyjnego.

3.2.2 Opis zmiennych wejściowych

Podczas budowania modelu, wykorzystane zostały zmienne oryginalne jak i cechy utworzone sztucznie w procesie inżynierii cech. Poniżej przedstawiono zestawienie wszystkich zmiennych:

- typ_przetwarzania – zmienna kategoryczna, opisująca typ przetwarzania danych (0 – strumieniowe, 1 – wsadowe, 2 – hybrydowe)
- typ_zapytan – rodzaj zapytań występujących w systemie (0 – proste, 1 – ad-hoc, 2 – złożone)
- skalowalnosc – skala rozproszenia systemu (0 – niska, 1 – średnia, 2 – wysoka)
- historia_danych – zmienna binarna przedstawiająca, czy system działa na danych historycznych (0 – nie, 1 – tak)
- chmura – informacja o wykorzystywaniu usług chmurowych (0 – brak, 1 – obecność)
- zapytania_na_sekunde – liczba zapytań obsługiwanych na sekundę przez system (zmienna numeryczna)
- ttl_dni – liczba dni, gdy dane są dostępne w systemie (ang. *Time to Live*)
- czy_wysoki_ttl – zmienna binarna utworzona na podstawie ttl_dni, (1 – gdy TTL >60, 0 gdy ≤ 60)
- czy_duzo_zapytan – cecha wskazująca, czy system przetwarza więcej niż 1000 zapytań na sekundę. (0 – nie, 1 – tak)
- czy_chmura_i_historia – zmienna złożona, przyjmuje wartość 1, gdy system korzysta jednocześnie z chmury i danych historycznych

Przy pomocy dodatkowych cech binarnych model zyskał możliwość efektywniejszego rozróżniania sytuacji granicznych, w których klasyfikacja była utrudniona na podstawie surowych danych przed konwertowaniem.

```
print(df.head())
```

	zapytania_na_sekunde	historia_danych	ttl_dni	typ_przetwarzania	\
0	1499	0	25	2	
1	667	0	49	2	
2	1393	0	44	1	
3	363	0	92	0	
4	1185	0	44	2	

	typ_zapytan	skalowalnosc	chmura	czy_wysoki_ttl	czy_duzo_zapytan	\
0	1	2	0	0	1	
1	0	1	1	0	0	
2	1	1	0	0	1	
3	0	1	0	1	0	
4	1	0	0	0	1	

	czy_chmura_i_historia	architektura
0	0	0
1	0	1
2	0	0
3	0	0
4	0	0

Rysunek 11. Prezentacja zbioru danych

```
print(df.dtypes)
```

zapytania_na_sekunde	int64
historia_danych	int64
tłł_dni	int64
typ_przetwarzania	int64
typ_zapytan	int64
skalowalnosc	int64
chmura	int64
czy_wysoki_tłł	int64
czy_duzo_zapytan	int64
czy_chmura_i_historia	int64
architektura	int64
dtype: object	

Rysunek 12. Typ danych w poszczególnych kolumnach

3.2.3 Zmienna docelowa

Zmienną klasyfikacyjną stanowi kolumna *architektura*, która przyjmuje wartości:

- 0 – w przypadku architektury Lambda,
- 1 – w przypadku architektury Kappa,

Wartość ta została przydzielona każdemu opisanemu przypadkowi. Oznacza to, że każdy wiersz reprezentuje osobny przypadek do klasyfikacji. Umożliwiło to stworzenie etykiet w zbiorze danych, które odpowiadają przewidywalnym scenariuszom użycia odpowiedniej architektury.

3.2.4 Przygotowanie danych i jakość

Na zbiorze danych wykonane zostały operacje przekształcające:

- Konwersję zmiennych kategoriycznych za pomocą OneHotEncoder,
- Dodanie sztucznych cech binarnych dla lepszego ujęcia progowych właściwości systemu,
- Usunięcie potencjalnych duplikatów i sprawdzenie poprawności typów danych
- Podział na zbiór treningowy (80%) i testowy (20%)

Biorąc pod uwagę syntetyczny rodzaj danych, zbiór nie posiada brakujących wartości (NULL), a rozkład klas został w znaczącym stopniu wyrównany za pomocą parametru `class_weight = balanced` w modelu.

3.2.5 Podsumowanie

W celu utworzenia transparentnego oraz dokładnego modelu klasyfikacyjnego, dane wejściowe zostały zaprojektowane syntetycznie. Metadane dobrane zostały w sposób odzwierciedlający realistyczne problemu przy wyborze architektur przetwarzania danych, a ich ustrukturyzowana postać oraz proces czyszczenia umożliwiają wysoką jakość predykcji. Zastosowanie cech

binarnych oraz poprawne zaprogramowanie zmiennych kategorycznych pozwoliły na poprawę efektywności uczenia modelu oraz szybkość jego uczenia.

3.3 Budowa modelu klasyfikacyjnego

W kolejnym etapie projektu przygotowany został model klasyfikacyjny, którego celem było przydzielenie jednej z dwóch architektur: Lambda lub Kappa dla konkretnego przypadku (np. API lub systemu danych). W skład procesu budowy modelu wchodziło dobór odpowiedniego algorytmu klasyfikacyjnego, przygotowanie danych wejściowych oraz skonfigurowanie środowiska uczenia maszynowego.

3.3.1 Wybór algorytmu

Do realizacji celu projektu, w którym stara się ustalić przy pomocy metadanych pasującą architekturę danych wybrano algorytm drzewa decyzyjnego (DecisionTreeClassifier), który jest dostępny w bibliotece scikit-learn. Wybór ten został wybrany przede wszystkim z powodu wysokiej transparentności wyników oraz stosunkowo niewielką złożonością obliczeniową w porównaniu do np. lasów losowych. Drzewa decyzyjne pozwalają użytkownikowi na zrozumienie funkcjonowania modelu, który podejmuje decyzję na podstawie poszczególnych cech, co jest często spotykane oraz cenne w systemach rekomendacyjnych.

3.3.2 Struktura pipeline'u

Mechanizm pipeline został wykorzystany w projekcie w celu zapewnienia przejrzystości oraz powtarzalności przetwarzania danych wraz z trenowaniem modelu, który łączy przyszłe etapy w jedną spójną strukturę. Pierwszym krokiem pipeline'u było zakodowanie zmiennych kategorycznych wykorzystując narzędzie OneHotEncoder, co przekonwertowało dane na liczbowe w celu ułatwienia uczenia. Pozostałe cechy zostały ominięte, dzięki argumentowi `remainder = passthrough`.

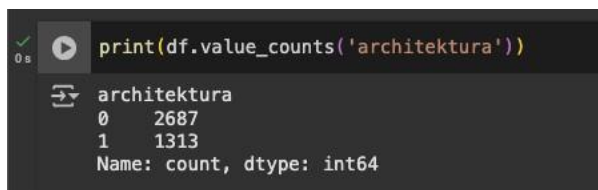
W rezultacie, uniknięto wielokrotnego aktualizowania danych wejściowych przy każdorazowym treningu modelu.

```
encoder = ColumnTransformer(  
    transformers=[('kat', OneHotEncoder(drop='first'), kolumny_kat)],  
    remainder='passthrough'  
)  
  
model = Pipeline([  
    ('preprocessing', encoder),  
    ('clf', DecisionTreeClassifier(max_depth=7, class_weight='balanced', random_state=42))  
])
```

Rysunek 13. przygotowanie oraz implementacja pipeline'u

3.3.3 Podział pracy

Przed rozpoczęciem trenowania modelu dane zostały podzielone na zbiór treningowy oraz testowy w proporcji 80:20. Do losowego podziału wykorzystano funkcję `train_test_split` z ustalonym parametrem `random_state`, który zapewnia powtarzalność wyników w przypadku testowania kodu na innym komputerze bądź w nowej sesji. Dodatkowo, z uwagi na brak równowagi wśród klas, w modelu uwzględniono parametr `class_weight = 'balanced'`, który automatycznie dostosowuje wagę próbek w zależności od ich częstotliwości.



```
print(df.value_counts('architektura'))
```

architektura	count
0	2687
1	1313

Name: count, dtype: int64

Rysunek 14. Przedstawienie braku równowagi architektur na zbiorze danych

3.3.4 Parametry modelu

Dla zwiększonej przejrzystości oraz dokładności klasyfikacji modelu, maksymalną głębokość drzewa (`max_depth`) ograniczono do wartości 6. Ten proces umożliwił nie tylko dokładne przewidywanie klasy, ale również transparentną interpretację funkcjonowania modelu. Dodatkowe parametry, takie jak `random_state` oraz `class_weight`, zostały dobrane w sposób zapewniający stabilność oraz odporność modelu na potencjalne szumy w danych.

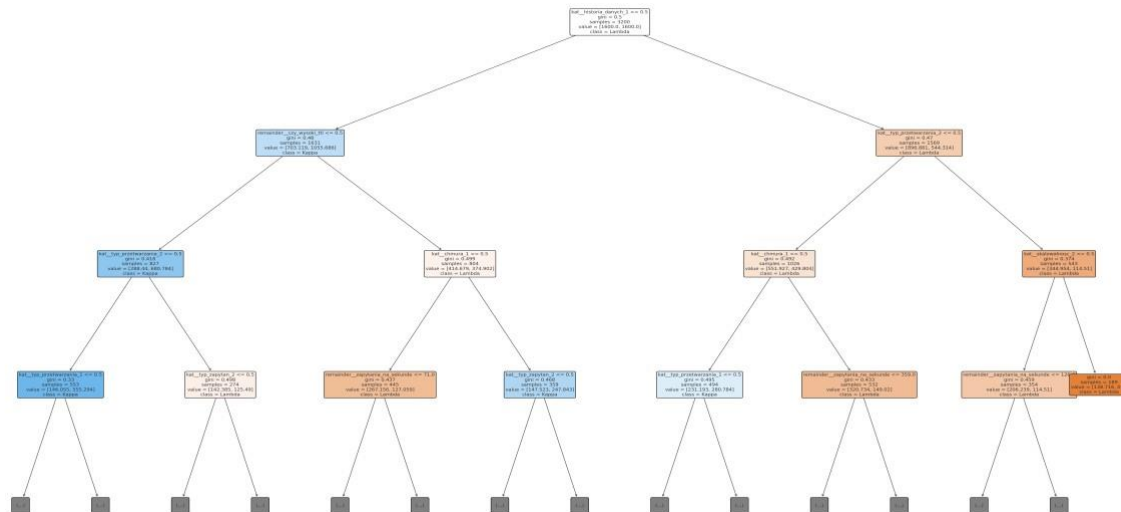
3.3.5 Podsumowanie

Wynikiem przeprowadzonych działań był model klasyfikacyjny oparty na algorytmie drzewa decyzyjnego, który umożliwił przeprowadzenie ewaluacji. W kolejnym podrozdziale przedstawiona zostanie wizualizacja struktury drzewa decyzyjnego oraz opisane czynniki ją tworzące.

3.4 Wizualizacja struktury drzewa decyzyjnego

W celu lepszego zrozumienia procesu podejmowania decyzji przez model klasyfikacyjny, wykonana została wizualizacja wytrenowanego drzewa decyzyjnego. Narzędzie `plot_tree` z biblioteki `scikit-learn` pozwolił na stworzenie czytelnego diagramu, który przedstawia strukturę podziału danych na odpowiednie architektury dzięki zdefiniowanym metadanych.

Na diagramie każda wewnętrzna gałąź reprezentuje warunek logiczny, który jest podstawą do dalszego rozgałęziania, natomiast każdy liść przedstawia końcową decyzję modelu, co oznacza przypisanie wartości 0 dla Lambda lub 1 dla Kappa.



Rysunek 15. Drzewo decyzyjne z (depth = 3)

3.4.1 Opis elementów drzewa decyzyjnego

Na potrzeby prezentacji drzewa, głębokość drzewa na diagramie została zredukowana do 3 poziomów, co umożliwia lepszą wizualizację oraz zrozumienie algorytmu.

Każdy węzeł drzewa reprezentuje zestaw informacji:

- Nazwa kolumny \leq wartość – warunek logiczny, którego zadaniem jest pogrupowanie danych (np. `czy_duzo_zapytan <= 0.5`),
- Gini – współczynnik czystości węzła. Gini index przyjmuje wartości z przedziału $<0;0.5>$ - im mniejsza jest wartość, tym jednoznaczna decyzja w danym węźle (np. `gini = 0.0` oznacza, że wszystkie próbki w tym węźle należą do jednej klasy),
- Samples – liczba próbek, które docierają do danego węzła
- Value = [a,b] – liczba obserwacji z danej grupy przypisanych do klasy 0 (Lambda) i klasy 1 (Kappa), wartości w drzewie są rodzaju zmiennoprzecinkowego (float) a nie całkowite (int) z powodu wykorzystania `class_weight = balanced`, oznacza to, że drzewo nie pokazuje dokładnej ilości a sumę wag próbek,
- Class = architektura – etykieta klasy, która zostaje przypisana w tym węźle na podstawie przewagi liczby próbek (np. `class = Lambda`)

W rezultacie, wizualizacja pozwala na prześledzenie funkcjonowania algorytmu klasyfikacyjnego, który wybiera dane cechy wejściowe z największym wpływem na podejmowane decyzje. Przykładowo, jeżeli warunki związane ze zmiennymi `czy_duzo_zapytan` i `typ_przetwarzania_hybrydowe` pojawiają się wysoko w drzewie,

sugeruje to, że były one uznane przez model jako najważniejsze z punktu widzenia klasyfikacji.

3.4.2 Przykład interpretacji fragmentu drzewa

W jednym z pierwszych rozgałęzień model sprawdza warunek:

$$Kat_historia_Danych_1 \leq 0.5$$

Jeżeli warunek jest spełniony, czyli system nie cechuje się magazynowaniem oraz tworzeniem analiz na danych historycznych, dane przekierowywane są w stronę architektury Kappa. Jeżeli nie, następuje dalsze sprawdzanie warunków.

Podejście to pozwala na klasyfikację oraz zrozumienie logiki działania algorytmu w modelu, co wyróżnia drzewa decyzyjne wyjątkową użytecznością w zastosowaniu, gdzie wymagana jest transparentność decyzji.

3.4.3 Podsumowanie

Wizualizacja drzewa decyzyjnego umożliwiła dostarczenie praktycznego wglądu w sposób klasyfikacji przypadków na podstawie ich metadanych. Umożliwiła rozpoznanie najważniejszych atrybutów oraz ocenę czy model podejmuje decyzje zgodne z oczekiwaniami wynikającymi z wiedzy dziedzinowej. Podejście to wzmacnia wartość modelu na poziomie narzędzia predykcyjnego, ale również w analizie biznesowej oraz technologicznej.

3.5 Ewaluacja modelu klasyfikacyjnego

Kolejnym etapem po procesie budowy oraz wizualizacji drzewa decyzyjnego była ewaluacja jakości działania modelu. Założeniem tego etapu była weryfikacja skuteczności algorytmu klasyfikacyjnego, którego zadaniem było przydzielenie prób do odpowiednich klas: architektura Lambda oraz architektura Kappa. Przeprowadzenie szczegółowej analizy jakości klasyfikacji umożliwia ocenę skuteczności modelu oraz rozpoznanie potencjalnych ograniczeń i obszarów wymagających dodatkowej optymalizacji.

3.5.1 Metody oceny modelu

Przy ocenie modelu wykorzystane zostały podstawowe metryki klasyfikacyjne:

- Accuracy (dokładność) – określa, jaki procent wszystkich przykładów ze zbioru danych został prawidłowo sklasyfikowany. Jest to wskaźnik uznawany za podstawowy wyznacznik skuteczności modelu. Wynik jest wyrażony jako iloraz liczby poprawnych predykcji do całości przypadków testowych.
- Macierz pomyłek (confusion matrix) – Wyznacznik przedstawia dokładne zrozumienie, które klasy najczęściej mylono. Zawiera informacje o liczbie przypadków prawidłowych i błędnie przypisanych do każdej z klas.
- Precision, Recall i F1-score – metryki te pozwalają na głębszą analizę skuteczności klasyfikacji w przypadku nierównomiernego rozkładu klas.

3.5.2 Wyniki modelu

Skuteczność modelu została przetestowana na zbiorze testowym stanowiącym 20% całego zbioru, który nie był wykorzystany przy trenowaniu modelu. Umożliwiło to na obiektywną ocenę zdolności generalizacji klasyfikatora.

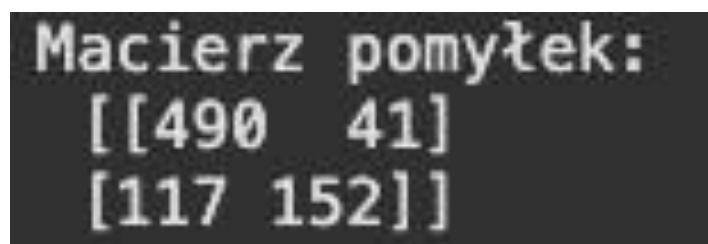
Na podstawie przeprowadzonych obliczeń uzyskano wynik dokładności (accuracy) na poziomie:

Accuracy: 0.803%

Wynik ten przedstawia, że model prawidłowo sklasyfikował 80% przypadków testowych. Jest to wartość uznawana za bardzo dobrą w przypadku problemów binarnej klasyfikacji, biorąc pod uwagę, że model skupia się jedynie na logicznych cechach decyzyjnych opisujących strukturę danych wejściowych.

3.5.3 Analiza macierzy pomyłek

Macierz pomyłek umożliwia rozpoznanie, w jakim stopniu klasy Lambda i Kappa były wyraźnie różne. Przypadki należące do klasy Kappa były częściej błędnie klasyfikowane jako Lambda. Skutkiem błędnej klasyfikacji mógł być fakt, że przypadki architektury Lambda w zbiorze uczącym (X_{train}) dominowały ilościowo lub miały bardziej zrozumiałe reguły decyzyjne dla modelu.



```
Macierz pomyłek:  
[[490 41]  
[117 152]]
```

Rysunek 16. Macierz pomyłek

Dodatkowo obserwacje wynikające z analizy struktury drzewa decyzyjnego wskazują, że model wykazywał większą pewność w klasyfikacji przypadków reprezentujących architekturę Lambda. Metadane takie jak wysoka skalowalność, obecność danych historycznych oraz typ przetwarzania „hybrydowe” były często wykorzystywane w wyższych węzłach drzewa, co oznacza, że miały ważny wpływ na podejmowane decyzje klasyfikujące. Występowanie ich w pierwszych podziałach drzewa decyzyjnego wskazują na wysoką wartość informacji oraz sugerują, że dla przypadków spełniających te kryteria model był w stanie bez problemowo przypisać poprawną klasę.

3.5.4 Wnioski z ewaluacji

Uzyskany wynik dokładności klasyfikacji powyżej 0.80 świadczy o wysokiej skuteczności modelu oraz potwierdza tezę, że zaprojektowany zbiór cech przedstawia w istotny sposób różnice między architekturami Lambda i Kappa. Jednocześnie model zachowuje wysoką interpretowalność, czego skutkiem może być wykorzystywanie go w analizach wspierających do podejmowania decyzji dotyczących wyboru odpowiedniej architektury przetwarzania danych.

Dalsze ulepszenia modelu mogłyby obejmować zwiększenie liczby przypadków klasy Kappa w zbiorze treningowym, rozszerzenie zbioru metadanych, czego skutkiem byłoby powiększenie głębokości drzewa decyzyjnego oraz testowanie bardziej zaawansowanych algorytmów klasyfikujących np. las losowy, który byłby w stanie lepiej uchwycić nieliniowe zależności między zmiennymi.

3.6 Wnioski końcowe z części praktycznej

Założeniem praktycznej części pracy było zaprojektowanie oraz przetestowanie klasyfikatora, którego zadaniem było pogrupowanie przypadków do jednej z dwóch architektur przetwarzania danych: Lambda lub Kappa. W skład procesu wchodziło zarówno przygotowanie syntetycznych danych oraz wykorzystanie najlepiej pasującego algorytmu klasyfikacyjnego do projektu.

Przy pomocy dobranych metadanych, takich jak typ przetwarzania, rodzaj zapytań, historia danych, TTL, liczba zapytań na sekundę, skalowalność i wykorzystanie chmury umożliwiło na stworzenie zbioru danych, który pozwolił na utworzenie zdarzeń z wyróżniającymi się różnicami architektonicznymi. Wprowadzenie inżynierii cech (np. `czy_duzo_zapytan`) pozwoliła na ustrukturyzowanie danych w sposób czytelny dla modelu oraz zwiększenie jego skuteczności.

Zastosowany algorytm drzewa decyzyjnego okazał się najlepszym wyborem. Umożliwił on na osiągnięcie satysfakcjonującego poziomu skuteczności klasyfikacji ($\text{accuracy} = 0.8025$) i zapewnił ułatwienie zrozumienia procesu decyzyjnego. Przy pomocy wizualizacji struktury drzewa z głębokością na poziomie trzy, możliwe było rozpoznanie kluczowych reguł klasyfikacyjnych i prześledzenie, jakie warunki najskuteczniej prowadzą do przypisywania odpowiednich architektur.

Ewaluacja modelu przedstawiła, że przypadki typowe dla architektury Lambda były częściej klasyfikowane poprawnie, czego przyczyną mogły być jednoznaczne cechy decyzyjne, takie jak wysoka skalowalność, obecność danych historycznych oraz typ przetwarzania hybrydowy. Przypadki przypisane do architektury Kappa wyróżniały się większą różnorodnością, czego skutkiem był niższy poziom dokładności w ich rozpoznaniu.

Podsumowując, wnioski wynikające z przeprowadzonego eksperymentu potwierdzają, że na podstawie odpowiednio przetworzonym zbiorze danych oraz przygotowanych metadanych możliwe jest zbudowanie prostego oraz skutecznego modelu klasyfikacyjnego, którego zadaniem jest wsparcie inżynierów danych w doborze odpowiedniej architektury na dużych wolumenach danych.

Rozdział 4. Podsumowanie

W pracy wykonana została szczegółowa analiza dwóch najpopularniejszych architektur przetwarzania danych: Lambda i Kappa. W okresie rosnącego popytu na przetwarzanie danych w czasie rzeczywistym, obie architektury uznawane są za podstawę w dużej ilości nowoczesnych systemów informacyjnych. Celem pracy było zaprezentowanie ich struktur, zalet i wad oraz stworzenie modelu klasyfikacyjnego, który umożliwi wsparcie dla specjalistów podczas wyboru odpowiedniej architektury dla klientów w zależności od charakterystyki danych.

W części teoretycznej zaprezentowane zostały charakterystyki obu architektur z uwzględnieniem ich założeń, struktury oraz praktycznego zastosowania. Stworzona została również szczegółowa analiza porównawcza architektur, która umożliwiła rozpoznanie charakterystycznych metadanych. Opracowanie to pozwoliło na stworzenie solidnej podstawy teoretycznej do budowy modelu klasyfikacyjnego.

W części praktycznej przedstawiony został klasyfikujący model uczenia maszynowego z wykorzystaniem drzewa losowego do rozpoznania oraz przydzielenia odpowiedniej architektury dla badanych obserwacji na podstawie wcześniej przygotowanego zbioru metadanych z części teoretycznej. Zastosowanie tego algorytmu pozwoliło na uzyskanie wysokiego wyniku istotności wyników na poziomie accuracy 80%, co potwierdziło skuteczność jego działania. Otrzymane wyniki pozwoliły na wyróżnienie takich cech jak skalowalność danych, obecność danych historycznych oraz typ przetwarzania jako najbardziej kluczowe.

Mimo otrzymanych wyników, trzeba pamiętać o nałożonych ograniczeniach modelu. Dane użyte w klasyfikacji były symulowane, co może wpływać na poprawność jego działania na danych rzeczywistych w warunkach produkcyjnych. Dodatkowo, zastosowanie jedynie algorytmu drzewa decyzyjnego w celu klasyfikacji nie musi być uznawane jako najlepsze rozwiązanie. Rozsądne będzie sprawdzenie poprawności jego działania na pozostałych algorytmach, takich jak Random Forest lub Gradient Boosting, które mogą pozwolić uzyskać większą dokładność oraz stabilność predykcji.

Model ten może stanowić krok w stronę automatyzacji procesu doboru architektury danych w środowisku analitycznym, jednakże trzeba zauważyć, że kolejnymi procesami powinna być optymalizacja algorytmu oraz adaptacja do pracy na danych produkcyjnych. W przyszłości warto zwrócić uwagę na rozbudowanie zbioru metadanych oraz integrację z systemami uczenia maszynowego w czasie rzeczywistym, co pozwoli na regularną aktualizację klasyfikacji.

Podsumowując, niniejsza praca w sposób kompleksowy zaprezentowała problem wyboru architektury danych w środowiskach przetwarzania danych w czasie rzeczywistym, dostarczając ważnych wskazówek funkcjonowania oraz narzędzi, które pozwolą na wsparcie procesu decyzyjnego.

Bibliografia

1. The Open Group, *TOGAF 9.1 Documentation*
<https://pubs.opengroup.org/architecture/togaf91-doc/arch/index.html> (data dostępu: 02.03.2025)
2. Gartner, *Enterprise Architecture and technology Leaders*
<https://www.gartner.com/en/information-technology/role/enterprise-architecture/technology-leaders> (data dostępu: 02.03.2025)
3. EABOK, <https://eabok.org/> (data dostępu: 02.03.2025)
4. Helion, Joe Reis, Matt Housley, 2023, *Inżyniera danych w praktyce*
5. IBM, Chuck Ballard, Dirk Herreman, Don Schau, Rhonda Bell, Eunsang Kim, Ann Valencic, 1998, *Data Modeling Techniques for Data Warehousing* http://e-ammr.net/download/ibm_warehouse%20model.pdf (data dostępu 03.03.2025)
6. IEE, Amr A. Munshi, Yasser Abdel -Rady I. Mohammed , 2018, *Data Lake Lambda Architecture for Smart Grids Big Data Analytics*
<https://ieeexplore.ieee.org/abstract/document/8417407> (data dostępu: 03.03.2025)
7. ERP-view, *Architektura Big Data – Lambda i Kappa*.<https://www.erpview.pl/artykuly-business-intelligence/26031-architektura-big-data-lambda-ikappa.html> (data dostępu: 03.03.2025)
8. Researchgate, 2010, G. Satyanarayana Reddy, Rallabandandi Srinivasu, M. Poorna chander RAO oraz Srikanth Reddy Rikkula, *Data warehousing, data mining, olap and oltp technologies are essential elements to suport decision-making proces in industries*, https://www.researchgate.net/publication/50235148_DATA_WAREHOUSING_DATA_MINING_OLAP_AND_OLTP_TECHNOLOGIES_ARE_ESSENTIAL_ELEMENTS_TO_SUPPORT_DECISION-MAKING_PROCESS_IN_INDUSTRIES (data dostępu: 04.03.2025)
9. Databricks, *Lambda Architecture*, https://www-databrickscom.translate.google/glossary/lambdarchitecture?_x_tr_sl=en&_x_tr_tl=pl&_x_tr_hl=pl&_x_tr_pto=rq (data dostępu: 04.03.2025)
10. Microsoft, *Big data architectures*
<https://learn.microsoft.com/plpl/azure/architecture/databases/guide/big-data-architectures> (data dostępu: 04.03.2025)
11. Amazon Advertising, *Amazon Attribution*,
<https://advertising.amazon.com/plpl/solutions/products/amazon-attribution> (data dostępu: 04.03.2025)
12. Uber, *AthenaX*, <https://www.uber.com/en-PL/blog/athenax/> (data dostępu: 05.03.2025)
13. ICT, *Lambda Architecture for Real Time Big Data Analytic* 2014,
https://d1wqtxts1xzle7.cloudfront.net/93244575/lambda-architecture-for-realtime-big-data-analytic-libre.pdf?1667025155=&response-contentdisposition=inline%3B+filename%3DLambda_Architecture_for_Real_Time_Big_

Da.pdf&Expires=1745408627&Signature=WvyrosEspCOba7qH0y6uGDqzBVVVg9M1luvjF4saplpI3p3qAmPlvecaCO~pUcMOj5R-JJOeC7qMa9UI1PBqdGGPBNQRfX3gfUPwfEvea8upu6b4yRjOQjkVH2vROjQZv4qabtXbmtyuwg7c0vYlFzK73D6~tSqeCxcehljvenMj54Yl32mZHeblkQmVT5we59TIFDt0jJnXlFM3jYE~gMRi3gHksXwa~3Ss2v2eIQGZc6~3M5t9vLx9ldAxOl3p7nyMZIObF0SlK3yAl~YojRIlf7cBgR2T9i8KjDBXjnfFT3iSpVwqStfD9JbpkPWz7Ha7JhBDJNVuZlnaq4vA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA (data dostępu: 07.03.2025)

14. Snowflake, *Lambda*

Architecture <https://www.snowflake.com/guides/lambdaarchitecture/> (data dostępu: 08.03.2025)

15. Helion, Serra, J., 2023. *Nowoczesne architektury danych: przewodnik po hurtowni danych, siatce danych oraz Data Fabric i Data Mesh.* (data dostępu: 09.03.2025)

16. *Data Quality: The Accuracy Dimension,*

https://omnisgh.primo.exlibrisgroup.com/discovery/fulldisplay?docid=alma993572123307664&context=L&vid=48OMNIS_WSE:WSE&lang=pl&search_scope=MyInst_and_CI&adaptor=Local%20Search%20Engine&tab=Everything&query=any,contains,Data%20Quality:%20The%20Accuracy%20Dimension&offset=0 (data dostępu: 09.03.2025)

17. Synthesite, *Data Modeling Techniques for Data Warehousing*

, <https://eddyswork.synthesite.com/resources/Data%20Modeling%20Tech%20For%20Data%20Warehouseing.pdf> (data dostępu: 10.03.2025)

18. Iebsco, *RESEARCH ON THE DESIGN OF WEB DATA WAREHOUSE BASED ON ETL*

META DATA MODEL AND PARTICLE SWARM OPTIMISATION, <https://research-iebsco1com-1jphm95vp1f7a.han.sgh.waw.pl/c/4cswel/viewer/pdf/da4erbu7zz> (data dostępu: 10.03.2025)

19. ResearchGate, Reddy, G. Satyanarayana Srinivasu, Rallabandi, 2010,

https://www.researchgate.net/publication/50235148_DATA_WAREHOUSING_DATA_MINING_OLAP_AND_OLTP_TECHNOLOGIES_ARE_ESSENTIAL_ELEMENTS_TO_SUPPORT_DECISION-MAKING_PROCESS_IN_INDUSTRIES (data dostępu: 10.03.2025)

20. Apache beam, *Apache Beam Documentation,* <https://beam.apache.org/> (data dostępu: 12.03.2025)

21. Helion, Campbell, L. and Majors, C, 2018, *Inżynieria niezawodnych baz danych. Projektowanie systemów odpornych na błędy*

22. ResearchGate, Mainali, K., 2021. *Discovering DataOps: A Comprehensive Review of Definitions, Use Cases and Tools* str 61.

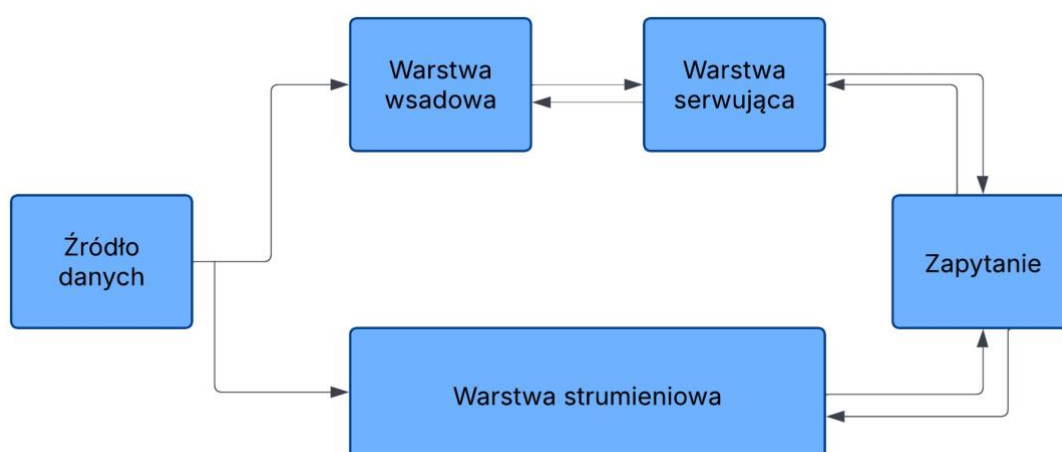
https://www.researchgate.net/profile/Kiran_Mainali2/publication/355107036_Discovering_DataOps_A_Comprehensive_Review_of_Definitions_Use_Cases_and_Tools/links/615dd703fbd5153f47e938a1/Discovering-DataOps-A-Comprehensive-Review-of-Definitions-Use-Cases-and-Tools.pdf (data dostępu: 16.03.2025)

23. Databricks, *MLOps.*

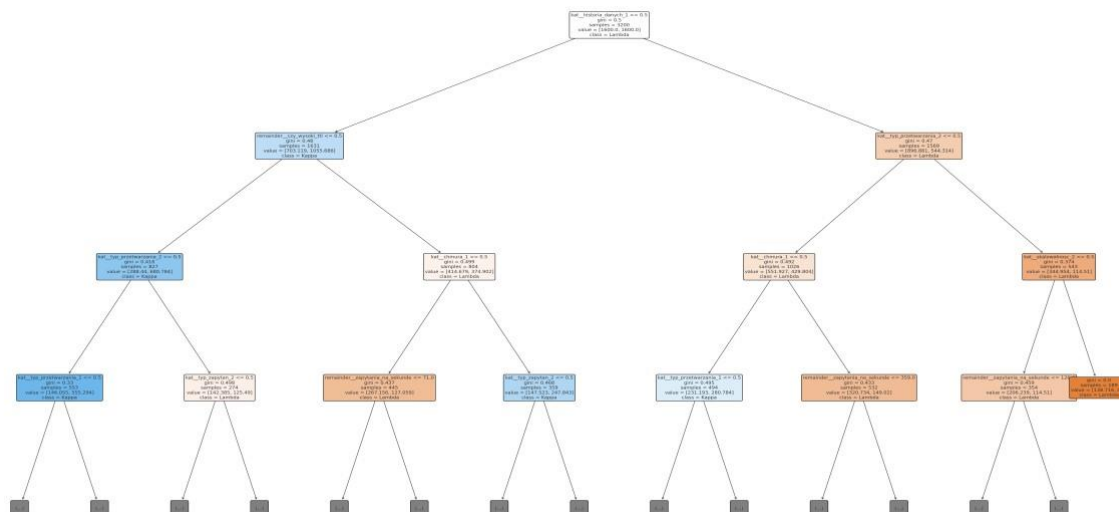
- https://www.databricks.com/glossary/mlops?utm_source=chatgpt.com (data dostępu: 18.03.2025)
24. ResearchGate, Kiran, Monga, 2015. *Lambda Architecture for cost-effective batch and speed big data processing*
https://www.researchgate.net/publication/308871780_Lambda_architecture_for_costeffective_batch_and_speed_big_data_processing (data dostępu: 20.03.2025)
25. Twardowski Bartłomiej, Ryżko Dominik Paweł:
real-time Big Data processing, In: The 2014 IEEE/WIC/ACM International Joint Conferences on Intelligent Agent Technologies / Ślęzak Dominik [i in.] (eds.), vol. 3, 2014, CPS, ISBN 978-1-4799-4143-8, pp. 333-337, DOI: 10.1109/WI-IAT.2014.185 (data dostępu: 21.03.2025)
26. ResearchGate, Behera, R., 2022. *Big Data Architectures: A Detailed and Application Oriented Analysis*.
https://www.researchgate.net/publication/364660407_Big_Data_Architectures_A_Detailed_and_Application_Oriented_Analysis (data dostępu: 24.03.2025)
27. ScienceDirect, Jean Bertin Nkamla Penka, Said Mahmoudi, Olivier Debauche, 2021, A new Kappa Architecture for IoT Data Management in Smart Farming,
https://www.sciencedirect.com/science/article/pii/S1877050921013983?ref=pdf_download&fr=RR-2&rr=92912c503d74ee3a (data dostępu: 27.03.2025)
28. EJ Compute Journal, Alaa Abdelraheem Hassan, Tarig Mohammed Hassan,
RealTime Big Data Analytics for Data Stream Challenges: An Overview.
<https://www.ejcompute.org/index.php/compute/article/view/62/29> (data dostępu: 04.04.2025)
29. European Data Portal, 2021. *Wprowadzenie do zarządzania metadanymi. Moduł szkoleniowy*
1.4. https://data.europa.eu/sites/default/files/d2.1.2_training_module_1.4_introduction_to_metadata_management_pl_edp.pdf (data dostępu: 10.04.2025)
30. Helion, Geron Aurelien, 2023, *Uczenie maszynowe z użyciem Scikit-Learn, Keras i TensorFlow*
31. Grzegorz Dudek, 2014, *Prognozowanie krótkoterminowe obciążeń systemów elektroenergetycznych z wykorzystaniem rozmytych drzew regresyjnych*
<https://pe.org.pl/articles/2014/4/24.pdf> (data dostępu: 12.04.2025)
32. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia, 2010, A view of cloud computing. *Communications of the ACM*, 53(4), 50–58str.
33. Carlo Batini, Monica Scannapieco, 2016, *Data and Information Quality*, 21-23str
34. Roberto Minerva, Abyi Biru, Domenico Rotondi, 2015, Towards a definition of the Intrnt of Things (IoT),
https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf (data dostępu: 15.04.2025)

35. Helion, Arthus Matoes, Jothy Rosenberg, 2012, *Chmura obliczeniowa – Rozwiązania dla biznesu*

Spis treści	2
Rysunek 1. Podział architektury korporacyjnej	3
Rysunek 2. Reprezentacja relacji w języku SQL	6
Rysunek 3. <i>Przykład struktury hierarchicznej w kontekście modelowania danych</i>	6
Rysunek 4. Proces pozyskiwania danych z systemu źródłowego do magazynu danych	11
Rysunek 5. Przykład danych niezwiązanych w relacyjnym modelu danych	12
Rysunek 6. Przykład danych związanych w relacyjnym modelu danych	12
Rysunek 7. Częstotliwość pozyskiwania danych ze źródeł: przetwarzanie wsadowe, mikropartyjne i w czasie rzeczywistym	13
Rysunek 8. Proces pozyskiwania synchronicznego	13



<i>Rysunek 9. Proces działania architektura lambda</i>	30
Rysunek 10. Działanie architektury Kappa	35
<i>Rysunek 11. Prezentacja zbioru danych</i>	47
<i>Rysunek 12. Typ danych w poszczególnych kolumnach</i>	47
<i>Rysunek 13. przygotowanie oraz implementacja pipeline'u</i>	48
<i>Rysunek 14. Przedstawienie braku równowagi architektur na zbiorze danych</i>	49



Rysunek 15. Drzewo decyzyjne z (depth = 3)	50
Rysunek 16. Macierz pomyłek	53

SPIS TABEL

Tabela 1. Charakterystyka systemu OLTP według wybranych kategorii	7
Tabela 2. Charakterystyka systemu OLAP według wybranych kategorii.....	7
Tabela 3. Zalety Data Fabric.	20
Tabela 4. Zalety Data Mesh.	20
Tabela 5. Wady Data Fabric.	21
Tabela 6. Wady Data Mesh.	21
Tabela 7. Zalety architektury Lambda	32
Tabela 8. Wady architektury Lambda	32
Tabela 9. Wady architektury Kappa	37
Tabela 10. Zalety architektury Kappa	38
Tabela 11. Porównanie architektur	39
Tabela 12. Metadane do odróżniania architektur	42

Streszczenie

Założeniem pracy jest wykonanie analizy porównawczej dwóch popularnych architektur danych pod względem zastosowania ich w środowiskach informatycznych, które wymagają przetwarzania danych w czasie rzeczywistym. W niniejszej pracy szczegółowo zostały przedstawione teoretyczne aspekty obu architektur, struktury, zalety oraz wady. Analiza porównawcza umożliwiła wskazanie optymalnych scenariuszy dla architektur Lambda oraz Kappa.

Na potrzeby części praktycznej został stworzony model klasyfikacyjny oparty na algorytmie drzewa decyzyjnego, który umożliwia automatyczny wybór odpowiedniej architektury danych w zależności od podanych metadanych, takich jak typ przetwarzania, obecność danych historycznych oraz skalowalność systemu. Skuteczność modelu klasyfikacyjnego wyniosła ponad 80%, co jest informacją o istotnym wpływie jego funkcjonowania w środowisku Big Data.

Praca stanowi wsparcie dla inżynierów danych oraz analityków w procesie automatyzacji wyboru architektury danych, oferując praktyczne wskazówki na temat wdrożenia i działania modelu w dynamicznych środowiskach analitycznych.