## PODSTAWY PROGRAMOWANIA

**-Podaj koncepcję stosu[41].**

Stos to abstrakcyjna struktura danych w której elementy ułożone są w kolejności LIFO (last in first out) ostatnie przyszło pierwsze wyjdzie. Operacje jakie możemy wykonać na stosie to push (odłóż na stos), pop (zdejmij ze stosu), peek (podgląd wierzchołka bez zdejmowania), isEmpty (spr czy stos jest pusty), size (rozmiar stosu).
Implementacja stosu może odbywać się za pomocą tablic lub listy jednokierunkowej.

**-Co to jest rekurencja?[42]**

Rekurencja to technika programowania polegająca na wywoływaniu funkcji samej siebie w celu rozwiązania problemu. Funkcja rekurencyjna składa się z dwóch części: warunku końcowego, który określa moment zakończenia rekurencji i zwrócenia wyniku, oraz części rekurencyjnej, która wywołuje funkcję samej siebie z nowymi argumentami i łączy wyniki. Jednym z przykładów użycia rekurencji jest obliczanie silni liczby. W matematyce, silnia liczby n to iloczyn wszystkich liczb naturalnych od 1 do n. Silnia liczby może być obliczona rekurencyjnie z wykorzystaniem następującego wzoru: n! = n * (n-1)!. W przypadku gdy n=1, silnia wynosi 1.

## GRAFIKA I KOMUNIKACJA CZŁOWIEK-KOMPUTER

**-Jaka jest różnica między grafiką wektorową a rastrową?[18]**

*Grafika wektorowa* jest tworzona przez matematyczne wzory i składa się z wektorów, które są zdefiniowane przez ich punkty końcowe. Dzięki temu grafika wektorowa jest skalowalna i nie traci na jakości nawet przy zwiększaniu lub zmniejszaniu rozmiaru. Grafika wektorowa jest idealna do tworzenia logo, ikon, elementów interfejsu użytkownika i innych elementów, które wymagają precyzyjnego kształtu i skalowalności.

*Grafika rastrowa* składa się z pikseli (małych kwadratów), które są ułożone w siatkę. Każdy piksel ma przypisaną wartość koloru, co tworzy obraz. Grafika rastrowa jest najlepsza do tworzenia fotografii, obrazów realistycznych, oraz innych elementów, w których ważne jest oddanie szczegółów i kolorów.

**-Podaj i omów znane ci modele barw związane ze sprzętem.[19]**

*RGB (Red-Green-Blue)* - jest to model kolorów używany w monitorach, telewizorach, aparatach fotograficznych i innych urządzeniach elektronicznych. W tym modelu każdy kolor jest reprezentowany przez kombinację trzech podstawowych barw: czerwonej, zielonej i niebieskiej, co pozwala na uzyskanie szerokiej gamy kolorów. Przykładowo, kolor żółty jest uzyskiwany poprzez połączenie składowych zielonej i czerwonej.
*CMY (Cyan-Magenta-Yellow)* - jest to model kolorów stosowany w druku, który opiera się na odejmowaniu światła białego. W tym modelu każdy kolor jest reprezentowany przez kombinację trzech podstawowych barw: cyjanowej, magentowej i żółtej. Po nałożeniu na siebie tych trzech barw, kolor czarny nie zawsze jest idealnie czarny, dlatego często stosuje się dodatkową czarną barwę.

*CMYK (Cyan-Magenta-Yellow-Key)* - jest to rozwinięcie modelu CMY, które dodaje czwartą barwę - kluczową (Key), czyli czarną. W tym modelu każdy kolor jest reprezentowany przez

kombinację czterech podstawowych barw: cyjanowej, magentowej, żółtej i czarnej. Czarna barwa jest stosowana w celu uzyskania bardziej intensywnych i wyrazistych kolorów, a także dla zmniejszenia kosztów druku, ponieważ stosowanie samego CMY może być droższe.

## METODOLOGIA PROGRAMOWANIA

**-Omów podstawowe założenia programowania strukturalnego.[43]**
Programowanie strukturalne to paradygmat programowania, który zakłada, że program powinien być zbudowany z prostych, logicznie zorganizowanych struktur, takich jak sekwencje, pętle i instrukcje warunkowe. Podstawowymi założeniami programowania strukturalnego są:

- Dekompozycja programu na mniejsze moduły – program powinien być podzielony na mniejsze fragmenty, które są bardziej zrozumiałe i łatwiejsze do utrzymania.

- Struktura sekwencji – program powinien być zbudowany z sekwencji instrukcji wykonywanych jeden po drugim, tak aby wynikające z nich obliczenia były przewidywalne i zawsze wykonywane w tej samej kolejności.

- Instrukcje warunkowe – program powinien umożliwiać warunkowe wykonanie określonych instrukcji, w zależności od spełnienia określonych warunków.

- Pętle – program powinien umożliwiać powtarzanie określonych instrukcji w celu wykonania ich wielokrotnie, tak długo jak spełnione są określone warunki.

- Brak skoków bezpośrednich – program powinien unikać skoków bezpośrednich, takich jak skoki bezpośrednie do innych fragmentów kodu lub wyjścia z pętli, ponieważ mogą one utrudniać zrozumienie i utrzymanie programu.

Celem programowania strukturalnego jest zwiększenie czytelności i zrozumiałości kodu oraz ułatwienie jego utrzymania. Aby osiągnąć ten cel, programiści powinni stosować określone konwencje nazewnictwa, dokumentować swoje rozwiązania oraz unikać złożonych konstrukcji i nadmiernego zagnieżdżania instrukcji warunkowych i pętli.

**-Omów strategię strukturalną tworzenia danych testowych.[44]**
Strategia strukturalna tworzenia danych testowych polega na wyborze takich danych, które reprezentują różne możliwe przypadki, warianty i sytuacje, które mogą wystąpić podczas działania programu. W tym celu analizuje się strukturę programu oraz jego wymagania funkcjonalne i niefunkcjonalne, aby wyznaczyć kluczowe cechy, które powinny być uwzględnione w testach.

W przypadku strategii strukturalnej tworzenia danych testowych wykorzystuje się informacje o strukturze programu, takie jak instrukcje warunkowe, pętle, gałęzie i metody, aby stworzyć testy, które wnikliwie badają każdą ścieżkę i wariant działania programu.

W celu stworzenia danych testowych zgodnie z tą strategią stosuje się różne techniki, takie jak analiza pokrycia kodu (code coverage), analiza wartości brzegowych (boundary value

analysis), analiza równoważności (equivalence partitioning) czy analiza przypadków granicznych (edge case analysis).

Analiza pokrycia kodu polega na ustaleniu, które linie kodu i gałęzie warunkowe zostały wykonane podczas testów i jakie wartości były przetwarzane. Dzięki temu można określić, które części programu wymagają dalszej uwagi i testowania.

Analiza wartości brzegowych polega na określeniu ekstremalnych wartości, które mogą wpłynąć na działanie programu, takie jak wartości minimalne i maksymalne dla określonych typów danych. Dzięki temu można zidentyfikować błędy i problemy, które mogą wystąpić, gdy program przetwarza skrajne wartości.

Analiza równoważności polega na podzieleniu danych wejściowych na grupy, które powinny generować podobne wyniki, co pozwala zredukować liczbę testów wymaganych do przetestowania całego programu.

Analiza przypadków granicznych polega na przetestowaniu skrajnych przypadków, które mogą prowadzić do awarii lub błędów programu.

Wszystkie te techniki pozwalają na stworzenie kompleksowych danych testowych, które uwzględniają różne przypadki, sytuacje i warianty działania programu. Dzięki temu można zwiększyć skuteczność testowania i zmniejszyć ryzyko wystąpienia błędów i problemów w programie.

## JĘZYKI PROGRAMOWANIA

**-Wyjaśnij pojęcia hermetyzacji, dziedziczenia i polimorfizmu.[39]**

*Hermetyzacja (ang. encapsulation)* - jest to mechanizm programowania obiektowego, który polega na ukryciu wewnętrznych danych obiektu przed światem zewnętrznym. Hermetyzacja pozwala na lepsze zarządzanie stanem obiektu oraz kontrolę nad dostępem do jego danych. W programowaniu obiektowym hermetyzacja osiągana jest przez stosowanie modyfikatorów dostępu, takich jak public, private i protected.

*Dziedziczenie (ang. inheritance)* - to mechanizm, który pozwala na tworzenie nowych klas na podstawie już istniejących. Klasa dziedzicząca (potomna) dziedziczy po klasie dziedziczonej (bazowej) jej pola, metody i zachowanie. Dziedziczenie pozwala na unikanie powielania kodu, co skraca czas programowania oraz upraszcza rozwijanie i modyfikowanie kodu.

*Polimorfizm (ang. polymorphism)* - to mechanizm, który umożliwia jednej nazwie metody lub funkcji posiadanie wielu różnych implementacji. Polimorfizm pozwala na tworzenie uniwersalnych metod, które są w stanie działać na różnych typach obiektów, co zwiększa elastyczność kodu i ułatwia jego rozwijanie. W programowaniu obiektowym polimorfizm osiągany jest dzięki stosowaniu dziedziczenia i interfejsów.

**-Omów i zobrazuj na przykładzie pojęcia klasy abstrakcyjnej i interfejsu.[40]**

Klasa abstrakcyjna i interfejs to dwa powiązane ze sobą pojęcia w programowaniu obiektowym, które pozwalają na wprowadzenie pewnego stopnia abstrakcji i jednocześnie definiowania pewnych wymagań dla klas dziedziczących.

Klasa abstrakcyjna to klasa, która nie może być bezpośrednio tworzona jako obiekt, ale służy jako wzorzec dla innych klas, które dziedziczą po niej. Klasa abstrakcyjna może zawierać metody abstrakcyjne, które nie posiadają implementacji, ale muszą być zaimplementowane przez każdą klasę dziedziczącą po klasie abstrakcyjnej. Metody abstrakcyjne definiują interfejs, który musi być zaimplementowany przez każdą klasę dziedziczącą po klasie abstrakcyjnej.

Przykładem klasy abstrakcyjnej może być klasa "Zwierzę", która może zawierać metody abstrakcyjne, takie jak "poruszaj się", "jedz" i "oddychaj". Każda klasa dziedzicząca po klasie "Zwierzę" musi zaimplementować te metody w sposób odpowiedni dla konkretnego zwierzęcia.

Interfejs to natomiast abstrakcyjny typ danych, który definiuje zestaw metod, które muszą być zaimplementowane przez każdą klasę, która implementuje dany interfejs. Interfejs nie zawiera implementacji, ale tylko definicje metod, które muszą być zaimplementowane przez każdą klasę, która dziedziczy po danym interfejsie. Interfejs pozwala na tworzenie abstrakcyjnych typów danych, które mogą być implementowane przez różne klasy.

Przykładem interfejsu może być interfejs "Połączenie internetowe", który może zawierać metody "nawiąż połączenie", "prześlij dane" i "zamknij połączenie". Każda klasa, która implementuje ten interfejs, musi zaimplementować te metody w sposób odpowiedni dla swojego połączenia internetowego.

Podsumowując, klasy abstrakcyjne i interfejsy są narzędziami programowania obiektowego, które pozwalają na wprowadzenie abstrakcji i definiowania wymagań dla klas dziedziczących. Klasy abstrakcyjne służą jako wzorce dla innych klas, a interfejsy definiują zestaw metod, które muszą być zaimplementowane przez każdą klasę, która dziedziczy po danym interfejsie.

## BAZY DANYCH

**-Omów 6 podstawowych właściwości baz danych, które odróżniają je od aplikacji innych typów.[67]**

1)*Organizacja danych:* bazy danych umożliwiają organizację danych w sposób zgodny z ich semantyką i strukturą. Dzięki temu dane są łatwiejsze w utrzymaniu i wyszukiwaniu.

2)*Redundancja danych*: bazy danych minimalizują redundancję danych. Zamiast przechowywać tę samą informację w wielu miejscach, informacja ta jest przechowywana tylko raz. Dzięki temu zmiany w danych są łatwiejsze i mniej podatne na błędy.

3)*Kontrola spójności danych:* bazy danych umożliwiają kontrolowanie spójności danych, co oznacza, że dane w bazie są zawsze poprawne. Bazy danych zapewniają mechanizmy, które uniemożliwiają wprowadzenie do bazy danych niepoprawnych informacji.

4)*Współbieżny dostęp do danych:* bazy danych umożliwiają wielu użytkownikom jednoczesny dostęp do danych. Dzięki temu wiele osób może pracować nad tą samą bazą danych w tym samym czasie, co zwiększa wydajność i skuteczność pracy.

5)*Bezpieczeństwo danych:* bazy danych umożliwiają kontrolę dostępu do danych, co oznacza, że tylko uprawnione osoby mają dostęp do informacji w bazie danych. Bazy danych umożliwiają również tworzenie kopii zapasowych danych, co minimalizuje ryzyko utraty informacji.

6)*Skalowalność:* bazy danych są skalowalne, co oznacza, że mogą obsługiwać rosnącą ilość danych i użytkowników. Dzięki temu bazy danych są w stanie sprostać rosnącym wymaganiom biznesowym i technologicznym.

**-Podaj definicję i znaczenie kluczy w relacyjnych bazach danych (podstawowy, kandydujący, obcy, prosty, złożony). Na czym polega integralność referencyjna?[68]**
Relacyjna baza danych to zbiór połączonych tabel, które przechowują dane w postaci krotek (wierszy) i atrybutów (kolumn). Klucze są ważnymi pojęciami w relacyjnych bazach danych, które pomagają w identyfikacji rekordów i zapewnieniu spójności danych.

*Klucz podstawowy (Primary Key):* Jest to klucz, który jednoznacznie identyfikuje każdy wiersz w tabeli. Każda tabela powinna mieć klucz podstawowy. Klucz podstawowy musi być unikalny w tabeli i nie może zawierać wartości NULL.

*Klucz kandydujący (Candidate Key):* Jest to zbiór jednego lub więcej atrybutów, który również jednoznacznie identyfikuje każdy wiersz w tabeli. Różni się od klucza podstawowego tylko tym, że może istnieć więcej niż jeden klucz kandydujący w tabeli.

*Klucz obcy (Foreign Key):* Jest to atrybut lub zbiór atrybutów, które odwołują się do klucza podstawowego w innej tabeli. Klucz obcy umożliwia tworzenie powiązań między tabelami.

*Klucz prosty (Simple Key):* Jest to pojedynczy atrybut, który może służyć jako klucz, ale nie jest unikalny.

*Klucz złożony (Composite Key):* Jest to klucz składający się z dwóch lub więcej atrybutów, których kombinacja jest unikalna.

Integralność referencyjna to zasada zapewnienia spójności danych w relacyjnej bazie danych za pomocą kluczy obcych. Oznacza to, że każdy klucz obcy musi odwoływać się do istniejącego klucza podstawowego w innej tabeli, a każdy klucz podstawowy musi mieć odpowiadający mu klucz obcy. Integralność referencyjna uniemożliwia usunięcie rekordu, który ma powiązania z innymi rekordami przez klucz obcy, zanim usunięte zostaną powiązane rekordy z innymi tabelami, chroniąc w ten sposób spójność danych.

**-Wyjaśnij pojęcie Encji w modelu konceptualnym. Jakie cechy musi posiadać? Czym jest hierarchia encji?[69]**
W modelu konceptualnym bazy danych, encja jest podstawową jednostką reprezentującą obiekt lub zdarzenie o których system bazy danych musi przechowywać informacje np. Klient, Zamówienie, Produkt, Dostawca, Pracownik, Wizyta, Lot;
Encje wyszukuje się z opisu problemu (świata rzeczywistego): „Klient składa Zamówienie", „Pracownik sprzedaje
Produkt", „Dostawca dostarcza nam Produkty";

Cechy, jakie musi posiadać encja w modelu konceptualnym to między innymi:
Unikalność: każda encja musi mieć unikalną nazwę, która jednoznacznie identyfikuje tę encję w całym modelu.
Atrybuty: encja musi posiadać atrybuty, czyli cechy, które opisują tę encję. Atrybuty te muszą być odpowiednio nazwane i zdefiniowane.

Relacje: encja może mieć zdefiniowane relacje z innymi encjami w modelu, co oznacza, że jest w jakiś sposób powiązana z innymi obiektami lub zdarzeniami.

Nazwa encji powinna być rzeczownikiem w liczbie pojedynczej.

*Hierarchia encji* w modelu konceptualnym bazy danych oznacza zależności między encjami. Możemy wyróżnić encje nadrzędne i podrzędne. Encje nadrzędne to te, które są na wyższym poziomie hierarchii, czyli posiadają atrybuty wspólne dla innych encji. Encje podrzędne natomiast odwołują się do atrybutów encji nadrzędnej, ale posiadają również swoje własne atrybuty. Dzięki takiej hierarchii możemy tworzyć bardziej złożone modele konceptualne bazy danych, które dokładniej oddają strukturę i zależności między różnymi obiektami w systemie.

**-Na czym polega proces normalizacji relacyjnej bazy danych? Jakie znasz postacie normalne? Czym się różnią?[70]**
Proces normalizacji relacyjnej bazy danych polega na organizowaniu danych w taki sposób, aby były one bardziej spójne, trwałe i niezależne od innych danych w bazie. Celem normalizacji jest uniknięcie redundancji danych i minimalizacja szans na błędy związane z konsystencją danych.

Istnieją różne formy normalizacji, a każda kolejna forma normalna (1NF, 2NF, 3NF, BCNF, 4NF, 5NF, 6NF) odnosi się do bardziej szczegółowych wymagań dla danych w bazie.

Pierwsza postać normalna (1NF) - wymaga, aby każda kolumna tabeli posiadała tylko pojedynczą wartość i była atomowa. Wartości w każdej kolumnie muszą być unikalne dla każdego rekordu.

Druga postać normalna (2NF) - wymaga, aby każda kolumna w tabeli była w pełni zależna od klucza podstawowego i niezależna od innych kolumn.

Trzecia postać normalna (3NF) - wymaga, aby każda kolumna tabeli była zależna tylko od klucza podstawowego i niezależna od innych kolumn nie będących w kluczu podstawowym.

Postać normalna Boyce'a-Codda (BCNF) - to postać normalna, w której każda determinanta jest kluczem podstawowym.

Czwarta postać normalna (4NF) - wymaga, aby w tabeli nie było wielowartościowych zależności funkcyjnych i aby dane były przechowywane w osobnych tabelach.

Piąta postać normalna (5NF) - dotyczy sytuacji, w której w bazie danych istnieją dwie lub więcej niezależnych wielowartościowych zależności funkcyjnych.

Szósta postać normalna (6NF) - dotyczy sytuacji, w której baza danych nie zawiera żadnych niedeterministycznych danych.

Każda kolejna postać normalna jest bardziej restrykcyjna i wymagająca niż poprzednia, ale także bardziej odporna na błędy i redundancję danych.

**-Wyjaśnij główne różnice między przetwarzaniem danych typu OLTP i OLAP[71]**
OLTP (Online Transaction Processing) i OLTP (Online Analytical Processing) to dwa różne podejścia do przetwarzania danych w kontekście systemów bazodanowych.

OLTP jest typowo stosowany w systemach transakcyjnych, w których istotna jest szybkość i dokładność przetwarzania pojedynczych transakcji. OLTP umożliwia użytkownikom wprowadzanie, modyfikowanie i usuwanie danych w czasie rzeczywistym. Przykłady systemów opartych na OLTP to bankowość internetowa, sklepy internetowe, systemy rezerwacji biletów itp.

OLAP jest z kolei stosowany w systemach analitycznych, w których istotne są analizy i przetwarzanie dużych ilości danych. OLAP umożliwia użytkownikom przeprowadzanie złożonych analiz wielowymiarowych, takich jak raportowanie, wykresy, prognozowanie, wizualizacja danych itp. Przykłady systemów opartych na OLAP to hurtownie danych, systemy wspomagania decyzji itp.

Główne różnice między OLTP i OLAP można podsumować w następujący sposób:
Cel: OLTP skupia się na przetwarzaniu pojedynczych transakcji w czasie rzeczywistym, natomiast OLAP na przetwarzaniu dużych ilości danych i ich analizie.
Struktura danych: OLTP zwykle korzysta z relacyjnych baz danych, w których dane są uporządkowane w tabelach, natomiast OLAP korzysta z hurtowni danych, w których dane są zorganizowane w taki sposób, aby ułatwić analizę wielowymiarową.
Rodzaj zapytań: OLTP obsługuje proste zapytania, takie jak wprowadzanie, modyfikowanie i usuwanie danych, natomiast OLAP obsługuje zapytania złożone, takie jak agregacja, grupowanie, filtrowanie, sortowanie itp.
Wydajność: OLTP musi zapewnić szybkość przetwarzania pojedynczych transakcji, natomiast OLAP musi zapewnić wydajność przy przetwarzaniu dużych ilości danych i złożonych zapytań.

Przykładem transformacji z OLTP do OLAP może być system bankowy, w którym OLTP odpowiada za przetwarzanie pojedynczych transakcji, takich jak wypłata lub wpłata pieniędzy, a OLAP odpowiada za generowanie raportów finansowych, takich jak bilans czy rachunek zysków i strat, na podstawie danych zgromadzonych w systemie OLTP.

**-Wyjaśnij na czym polega operacja ETL w hurtowni danych, podaj przykłady[72]**
Operacja ETL (ang. Extract, Transform, Load) to proces wyodrębniania, transformacji i ładowania danych z różnych źródeł do hurtowni danych.
ETL jest kluczowy dla funkcjonowania hurtowni danych, ponieważ pozwala na integrację danych pochodzących z wielu różnych źródeł i dostosowanie ich do wspólnego schematu, dzięki czemu dane te mogą być wykorzystane w analizie biznesowej. Proces ETL zazwyczaj składa się z trzech etapów:

*Ekstrakcja (Extract)* - polega na pobraniu danych z różnych źródeł, takich jak bazy danych, pliki CSV, systemy ERP, CRM, itp. W tym etapie dane są kopiowane z oryginalnego źródła do hurtowni danych.

*Transformacja (Transform)* - w tym etapie dane są poddawane różnym operacjom przetwarzania, które umożliwiają ich harmonizację i spójność. Na przykład, można usuwać duplikaty, wypełniać brakujące wartości, zmieniać formatowanie danych, łączyć różne źródła danych, itp.

*Ładowanie (Load)* - w końcowym etapie, przetworzone dane są zapisywane do hurtowni danych. Może to odbywać się na kilka sposobów, w zależności od potrzeb, na przykład: pełne ładowanie (ang. Full Load), przy którym cała tabela jest usuwana i ładowana od nowa lub inkrementalne ładowanie (ang. Incremental Load), które aktualizuje jedynie zmienione rekordy.

Przykłady ETL mogą obejmować przetwarzanie danych finansowych, takie jak transakcje kredytowe lub transakcje giełdowe. Dzięki ETL możliwe jest zintegrowanie tych danych z różnych źródeł, a następnie przeprowadzenie analizy, która pozwala np. na identyfikację trendów lub analizę ryzyka.

Innym przykładem ETL może być przetwarzanie danych sprzedażowych. Dane te mogą pochodzić z wielu źródeł, takich jak sklepy internetowe, punkty sprzedaży stacjonarnej, itp. ETL umożliwia integrację tych danych w jednym miejscu i zapewnia spójność danych, dzięki czemu można przeprowadzić analizę sprzedaży na poziomie całej organizacji.

**-Wyjaśnij transformację związku M:N z modelu ERD do RBD. Podaj przykład[73]**
Transformacja związku M:N (wielu do wielu) z modelu diagramu związków encji (ERD) do schematu relacyjnej bazy danych (RBD) wymaga utworzenia tabeli pośredniczącej (łącznikowej).
Tabela ta będzie zawierać klucze obce odnoszące się do kluczy głównych tabel, które były połączone relacją M:N w modelu ERD. W ten sposób umożliwia to utworzenie połączenia wielu do wielu między rekordami w tabelach.

Na przykład, załóżmy, że mamy dwie encje: "Studenci" i "Kursy", gdzie każdy student może być
zapisany na wiele kursów, a każdy kurs może mieć wielu studentów. W modelu ERD ta relacja
byłaby oznaczona jako związek M:N.
Aby zaimplementować tę relację w RBD, musimy utworzyć trzecią tabelę pośredniczącą, np. "Zapisy na kursy". Ta tabela będzie zawierać dwa klucze obce: "ID_studenta" odwołujący się do
tabeli "Studenci" i "ID_kursu" odwołujący się do tabeli "Kursy".
Tabela "Zapisy na kursy" będzie miała wiele rekordów, z każdym rekordem reprezentującym jedno
połączenie między studentem i kursami, w których jest zapisany. W ten sposób ta tabela umożliwi
nam reprezentowanie połączenia wielu do wielu między studentami a kursami w RBD.

**-Wyjaśnij transformację związku 1:N z modelu ERD do RBD. Podaj przykład[74]**
Transformacja związku 1:N z modelu ERD (Entity-Relationship Diagram) do RBD (Relacyjnego Modelu Danych) polega na utworzeniu tabeli dla każdej encji (1) i dodaniu klucza obcego do tabeli encji z drugiej strony związku (N).

Przykładem może być związek pomiędzy encjami "Klient" i "Zamówienie", gdzie jeden klient może mieć wiele zamówień, ale jedno zamówienie przypisane jest tylko do jednego klienta. W modelu ERD związek ten byłby oznaczony jako "1:N" i wyglądałby jak linia łącząca encje "Klient" i "Zamówienie", z jednym końcem linii oznaczonym "1" (dla encji "Klient") a drugim końcem "N" (dla encji "Zamówienie").

Aby przetransformować ten związek do RBD, należy utworzyć dwie tabele: "Klient" i "Zamówienie". W tabeli "Klient" tworzony jest klucz główny, np. ID_klienta, a w tabeli "Zamówienie" dodawany jest klucz obcy, np. ID_klienta, który odnosi się do klucza głównego w tabeli "Klient". Tabela "Zamówienie" może zawierać dodatkowe pola, takie jak np. ID_zamówienia, data zamówienia, itp.

## ZAAWANSOWANE BAZY DANYCH

**-Wyjaśnij pojęcie Master Data i zasady ich przechowywania[92]**
Master Data to kluczowe dane dotyczące podstawowych obiektów w organizacji, takich jak klienci, produkty, usługi, dostawcy, lokalizacje, a także inne dane charakterystyczne dla danej branży lub organizacji. Są to dane, które są wykorzystywane w różnych systemach w całej organizacji, a ich integralność i dokładność są kluczowe dla biznesu.

Przechowywanie Master Data powinno odbywać się zgodnie z kilkoma zasadami:

Jedno źródło prawdy (Single Source of Truth) - oznacza to, że każdy element Master Data powinien mieć tylko jedno źródło prawdy, które jest jednoznaczne, aktualne i dostępne dla wszystkich systemów.

Konsekwentność i jednolitość - oznacza to, że dane Master Data powinny być konsekwentne i jednolite we wszystkich systemach. Nazwy, kody, klasyfikacje i inne atrybuty powinny być zgodne z ustalonymi standardami.

Bezpieczeństwo i poufność - dane Master Data powinny być chronione przed nieuprawnionym dostępem i manipulacją.

Odporność na błędy - systemy przechowujące dane Master Data powinny być odpornie na błędy i awarie, aby uniknąć utraty danych.

Synchronizacja i integracja - dane Master Data powinny być synchronizowane i zintegrowane między różnymi systemami w organizacji, aby zapewnić spójność i integralność danych.

Przechowywanie Master Data jest kluczowe dla organizacji, ponieważ umożliwia integrację i zarządzanie danymi w organizacji oraz poprawia dokładność, spójność i jakość danych, co przekłada się na lepszą jakość procesów biznesowych i decyzji.

**-Wyjaśnij pojęcie instancji. Jakie mogą być powody, dla których należy instalować więcej instancji?[93]**
Instancją bazy danych nazywamy konkretne wcielenie bazy danych, które działa na określonym serwerze bazy danych i jest dostępne dla użytkowników. Innymi słowy, instancja to pojedynczy proces bazy danych działający w pamięci operacyjnej serwera bazy 0danych, wraz z jej plikami i strukturami danych.

Istnieją różne powody, dla których należy instalować więcej niż jedną instancję bazy danych:

Izolacja danych - Jeśli korzystamy z jednej bazy danych do różnych celów, możemy utworzyć osobną instancję dla każdego celu, aby izolować dane między nimi i uniknąć konfliktów. Na przykład możemy utworzyć oddzielną instancję bazy danych dla produkcji, testów i rozwoju.

Skalowalność - Jeśli nasza baza danych rośnie wraz z rozwojem naszej firmy, możemy utworzyć dodatkowe instancje, aby zwiększyć jej przepustowość i obsługiwać większą liczbę użytkowników.

Wsparcie dla różnych wersji oprogramowania - Jeśli korzystamy z różnych wersji oprogramowania bazy danych dla różnych aplikacji, możemy utworzyć osobne instancje dla każdej wersji, aby uniknąć problemów z zgodnością.

Ochrona przed awariami - Jeśli jedna z instancji bazy danych ulegnie awarii, to pozostałe instancje mogą nadal działać, co minimalizuje wpływ awarii na działanie naszej aplikacji.

Kontrola dostępu - Jeśli korzystamy z jednej bazy danych dla różnych grup użytkowników, możemy utworzyć osobne instancje, aby kontrolować dostęp do danych między grupami.

Instalowanie więcej niż jednej instancji bazy danych może zwiększyć złożoność zarządzania bazami danych, ale może również przynieść wiele korzyści w zakresie izolacji danych, skalowalności i bezpieczeństwa danych.

**-Przedstaw zasady przechowywania danych w plikach, rodzaje plików, Filegroups, cechy związane z ich konfiguracją i lokalizacją fizyczną.[94]**
Przechowywanie danych w plikach to popularny sposób przechowywania danych w systemach zarządzania bazą danych. Oto kilka zasad dotyczących przechowywania danych w plikach:

- Każdy plik powinien przechowywać tylko jeden rodzaj danych.

- Pliki powinny być umieszczane na różnych dyskach fizycznych w celu zwiększenia wydajności systemu.

- Wszystkie pliki muszą być dostępne dla systemu w momencie uruchamiania bazy danych.

- Wszystkie pliki muszą być zgodne z zasadami systemu plików.

- Pliki muszą być odpowiednio skalowane w celu zapewnienia optymalnej wydajności systemu.

Rodzaje plików w bazach danych:

- Plik danych (data file) - przechowuje dane tabel i indeksów.
- Plik dziennika transakcji (transaction log file) - przechowuje informacje o zmianach dokonywanych w bazie danych.
- Filegroups to grupy plików, które służą do przechowywania określonych danych. Pozwalają one na lepszą organizację przechowywania danych i zarządzanie nimi, szczególnie w dużych bazach danych. Dzięki filegroups możemy określić, gdzie należy przechowywać określone dane, co pozwala na optymalizację wydajności systemu.

Cechy związane z konfiguracją i lokalizacją fizyczną plików w bazie danych:

- Rozmiar plików powinien być odpowiednio skalowany w celu zapewnienia optymalnej wydajności.
- Lokalizacja plików powinna być starannie wybrana, aby zapewnić optymalną wydajność i bezpieczeństwo danych.
- Jeśli chcemy osiągnąć wysoką wydajność systemu, pliki powinny być umieszczone na różnych dyskach fizycznych.
- W przypadku serwerów z wieloma dyskami fizycznymi można skonfigurować grupy plików, aby każda grupa była umieszczona na innym dysku fizycznym.
- Konfiguracja plików może wpłynąć na wydajność systemu, a także na szybkość wykonywania operacji I/O, takich jak odczyt i zapis danych.

**-Przedstaw fukcje i działanie pliku *.ldf oraz zasady jego przechowywania.[95]**
Plik z rozszerzeniem .ldf to plik dziennika transakcji (transaction log) w bazie danych SQL Server. Jego główną funkcją jest utrzymywanie informacji o zmianach wprowadzanych do bazy danych w formie sekwencji transakcji.

Gdy dane są wprowadzane, zmieniane lub usuwane w bazie danych, informacje o tych operacjach są zapisywane w pliku dziennika transakcji. Plik ten zawiera informacje o każdej operacji, włączając w to czas i datę, nazwę użytkownika, który wykonał operację, nazwę tabeli, na której operacja została wykonana, a także wartości wprowadzone do tabeli.

Plik .ldf jest jednym z dwóch plików używanych przez bazę danych SQL Server, drugim plikiem jest plik .mdf (master data file). Plik .ldf jest automatycznie generowany przez bazę danych i jest przechowywany w określonym katalogu na serwerze baz danych. Zasady przechowywania pliku .ldf są takie same jak dla pliku .mdf, tzn. plik ten powinien być przechowywany na dysku, który jest wystarczająco wydajny i bezpieczny.

Główne funkcje pliku .ldf to:

Zapewnienie integralności danych poprzez przechowywanie informacji o zmianach wprowadzanych do bazy danych w formie sekwencji transakcji.

Umożliwienie odzyskiwania danych po awarii lub błędzie systemu poprzez umożliwienie przywrócenia bazy danych do stanu sprzed awarii.

Umożliwienie odzyskiwania danych w przypadku wykonywania operacji nieodwracalnych, np. usuwania danych.

Umożliwienie replikacji bazy danych, czyli utworzenia jej kopii na innym serwerze.

W przypadku błędów w pliku .ldf mogą wystąpić problemy z integralnością danych i utratą informacji o transakcjach. W takiej sytuacji konieczne może być przywrócenie bazy danych z kopii zapasowej lub przeprowadzenie procedury naprawy pliku dziennika transakcji.

**-Przedstaw 4 typowe błędy związane z realizacją współbieżności w bazie danych.[96]**
Konflikty zapisu (write conflicts) - występują, gdy dwa lub więcej wątków próbuje jednocześnie zmodyfikować ten sam rekord w bazie danych. W efekcie jedna z transakcji zostanie odrzucona, co może prowadzić do nieprzewidywalnego zachowania aplikacji.

Zatory (deadlocks) - to sytuacja, w której dwa lub więcej wątków czekają na siebie nawzajem, blokując sobie dostęp do zasobów bazy danych. W efekcie żadna z transakcji nie może się zakończyć, co prowadzi do blokady systemu.

Brak izolacji transakcji (transaction isolation) - gdy jedna transakcja zmienia dane, które są widoczne dla innej transakcji, która jeszcze ich nie zakończyła. To może prowadzić do błędów i niestabilnego zachowania systemu.

Zjawisko brudnego odczytu (dirty reads): Współbieżne operacje odczytu i zapisu mogą prowadzić do zjawiska brudnego odczytu, w którym jedna transakcja odczytuje dane, które zostały zapisane przez inną transakcję, ale jeszcze nie zostały zatwierdzone. W przypadku, gdy druga transakcja zostanie wycofana, to odczytane przez pierwszą transakcję dane mogą być nieprawidłowe.

**-Co to są transakcje w bazach danych? Omów na wybranym przykładzie podstawowe właściwości transakcji (ACID).[97]**
Transakcje w bazach danych to operacje, które są wykonywane na danych w systemie zarządzania bazą danych (DBMS). Transakcje służą do utrzymania spójności i integralności danych w bazie danych, co oznacza, że dane są zawsze poprawne i spójne, nawet w przypadku awarii systemu lub innych błędów.

Podstawowe właściwości transakcji są ściśle powiązane z akronimem ACID:

Atomicity (Atomowość) - Transakcja jest atomowa, co oznacza, że jest traktowana jako pojedyncza i niepodzielna operacja. To oznacza, że wszystkie operacje w transakcji muszą zostać zakończone lub żadna z nich nie zostanie wykonana.

Consistency (Spójność) - Transakcja musi utrzymać spójność danych w bazie danych. Oznacza to, że po zakończeniu transakcji dane muszą być zawsze poprawne i spójne.

Isolation (Izolacja) - Transakcje powinny być izolowane od siebie, co oznacza, że jedna transakcja nie powinna wpływać na wynik innej transakcji. To zapewnia, że wyniki transakcji są niezależne od innych transakcji, które mogą być wykonywane równolegle.

Durability (Trwałość) - Transakcje muszą być trwałe, co oznacza, że po zakończeniu transakcji wynik musi zostać zapisany na stałe i pozostać w bazie danych, nawet w przypadku awarii systemu.

Przykład: Zakładamy, że chcemy zmienić stan konta w bazie danych. Operacja ta będzie wykonywana w transakcji. W trakcie transakcji następują następujące operacje:
- pobranie aktualnego stanu konta
- zmiana stanu konta o określoną wartość
- zapisanie zmienionego stanu konta do bazy danych

Jeśli którakolwiek z operacji zakończy się niepowodzeniem, cała transakcja zostanie wycofana (operacje wycofania zostaną wykonane) i dane w bazie danych pozostaną niezmienione. Jeśli natomiast wszystkie operacje zostaną wykonane pomyślnie, dane zostaną zapisane w bazie danych i zostaną uznane za trwałe. W ten sposób transakcja utrzymuje spójność i integralność danych w bazie danych.

**-Omów główne zasady działania protokołu blokowania zasobów Strict 2PL[98]**

Protokół blokowania zasobów Strict 2PL (Strict Two-Phase Locking) to protokół transakcyjny stosowany w zaawansowanych bazach danych do zarządzania współbieżnym dostępem do danych. Główną ideą protokołu jest użycie blokad, aby zapewnić spójność danych oraz uniknąć niepożądanych zjawisk, takich jak zagubienie modyfikacji lub czytanie brudnych danych.

Zasady działania protokołu blokowania zasobów Strict 2PL są następujące:

1)Faza blokowania (Locking Phase): w tej fazie transakcja blokuje (przydziela) wszystkie potrzebne do wykonania operacje zasoby, takie jak wiersze w tabelach, bloki dyskowe lub pliki. Wszystkie te zasoby są zablokowane na czas trwania transakcji.

2)Faza wykonywania (Execution Phase): po zablokowaniu zasobów transakcja rozpoczyna wykonywanie operacji na danych. W tej fazie transakcja może odczytywać i modyfikować dane tylko wtedy, gdy ma do nich zablokowany dostęp.

3)Faza zwalniania (Unlocking Phase): po zakończeniu wykonywania operacji transakcja zwalnia wszystkie zablokowane zasoby. W tym momencie inne transakcje mogą uzyskać dostęp do tych zasobów.

W protokole Strict 2PL wszystkie blokady zasobów są trzymane do końca transakcji. Oznacza to, że żadna inna transakcja nie może uzyskać dostępu do zasobu aż do czasu, gdy transakcja zakończy swoje działanie i zwolni blokadę.
Protokół Strict 2PL gwarantuje, że transakcje wykonywane w sposób współbieżny są zgodne z izolacją na poziomie szeregowym (ang. serializability), co oznacza, że wynik działania wielu transakcji współbieżnych jest taki sam, jakby były one wykonywane sekwencyjnie w jakiejś kolejności. Jednakże, protokół ten ma tendencję do blokowania zasobów przez długi czas, co może prowadzić do spadku wydajności systemu przy dużych obciążeniach.

**-Omów różnicę między logami UNDO i REDO. W jakich sytuacjach znajdują zastosowanie?[99]**
Logi UNDO i REDO to dwa różne typy logów wykorzystywanych w zaawansowanych bazach danych.

Logi UNDO służą do cofania zmian wykonanych w bazie danych. W przypadku, gdy transakcja zostanie anulowana, log UNDO pozwala na przywrócenie stanu bazy danych sprzed rozpoczęcia transakcji. Zawiera on informacje o wszystkich zmianach dokonanych w bazie danych przez transakcję i pozwala na cofnięcie ich wszystkich w sposób odwrotny do wykonanego przez nią działania.

Logi REDO, z kolei, służą do odtwarzania zmian wykonanych w bazie danych. W przypadku awarii systemu, log REDO pozwala na przywrócenie stanu bazy danych do momentu poprzedzającego awarię. Zawiera on informacje o wszystkich zmianach dokonanych w bazie danych i pozwala na ich powtórzenie, w celu przywrócenia bazy danych do stanu sprzed awarii.

Obydwa typy logów są bardzo ważne dla zapewnienia bezpieczeństwa i integralności danych w bazach danych. Logi UNDO są szczególnie przydatne w przypadku, gdy transakcje nie zostały ukończone, a logi REDO są przydatne w przypadku awarii systemu lub innych sytuacji, w których należy przywrócić bazę danych do stanu sprzed określonego momentu.

**-Omów różnicę między typami Recovery Model: Full, Bulk Logged i Simply[100]**
Recovery Model to jedna z cech bazy danych w systemie zarządzania relacyjnymi bazami danych (RDBMS), która określa sposób w jaki baza danych będzie wykonywać backup i recovery.

W systemie Microsoft SQL Server istnieją trzy różne typy Recovery Model: Full, Bulk-Logged i Simple, które różnią się między sobą sposobem, w jaki zarządzane są transakcje i jak rejestrowane są zmiany w bazie danych.

*Full Recovery Model:* w tym modelu, baza danych rejestruje każdą transakcję w dzienniku transakcji (transaction log), co umożliwia przywrócenie bazy danych do dowolnego punktu w czasie. Dziennik transakcji zawiera informacje o każdej zmianie w bazie danych, która została wykonana od czasu ostatniego backupu. Backup bazy danych jest wykonywany wraz z kopią dziennika transakcji. Model ten zapewnia najwyższy poziom ochrony danych, ale wymaga również najwięcej miejsca na dysku dla dziennika transakcji oraz wymaga stałych backupów.
*Bulk-Logged Recovery Model:* w tym modelu, baza danych rejestruje tylko niektóre transakcje w dzienniku transakcji, co skutkuje mniejszą ilością informacji, która musi być backupowana. Model ten ma większe ryzyko utraty danych niż Full Recovery Model, ale wymaga mniejszej ilości miejsca na dysku dla dziennika transakcji. Ten model jest zwykle używany w sytuacjach, gdy wykonuje się operacje ładowania danych (bulk loads).

*Simple Recovery Model:* w tym modelu, baza danych nie rejestruje każdej transakcji w dzienniku transakcji, co oznacza, że nie ma możliwości przywrócenia bazy danych do dowolnego punktu w czasie. Zamiast tego, backup bazy danych jest wykonywany bezpośrednio, a dziennik transakcji jest automatycznie obcinany. Model ten zapewnia najmniejszą ilość miejsca na dysku dla dziennika transakcji, ale oferuje najmniejszą ochronę danych.

W zależności od potrzeb biznesowych oraz polityki backupu i recovery, wybór jednego z powyższych Recovery Model może mieć duże znaczenie dla utrzymania integralności i dostępności bazy danych.

## ALGORYTMY I ZŁOŻONOŚCI

**-Omów sposoby definiowania złożoności obliczeniowej algorytmów.[45]**
Złożoność obliczeniowa algorytmu odnosi się do ilości zasobów (takich jak czas i pamięć) potrzebnych do wykonania algorytmu w zależności od rozmiaru danych wejściowych. Istnieją różne sposoby definiowania złożoności obliczeniowej algorytmów, w tym:

*Złożoność czasowa* - odnosi się do liczby operacji wykonywanych przez algorytm w zależności od rozmiaru danych wejściowych. Najczęściej wyrażana jest w notacji asymptotycznej, takiej jak O(n), gdzie n oznacza rozmiar danych wejściowych.

*Złożoność pamięciowa* - odnosi się do ilości pamięci potrzebnej do przechowywania danych i wyników pośrednich algorytmu w zależności od rozmiaru danych wejściowych. Najczęściej wyrażana jest również w notacji asymptotycznej, takiej jak O(n).

*Złożoność algorytmu w najgorszym przypadku* - odnosi się do maksymalnej złożoności czasowej lub pamięciowej algorytmu dla każdego możliwego wejścia.

*Złożoność algorytmu w przypadku oczekiwanym* - odnosi się do średniej złożoności czasowej lub pamięciowej algorytmu dla losowych danych wejściowych.

*Stała złożoność* - odnosi się do stałej liczby operacji lub pamięci potrzebnej przez algorytm, niezależnie od rozmiaru danych wejściowych.

*Złożoność obliczeniowa w sensie redukcji* - odnosi się do ilości zasobów potrzebnych do rozwiązania problemu, który jest redukowany do problemu rozwiązywanego przez algorytm.

Wszystkie te sposoby definiowania złożoności obliczeniowej algorytmów są ważne i pozwalają na określenie, jak dobrze algorytm działa dla różnych rozmiarów danych wejściowych i w różnych przypadkach.

**-Podaj i przedyskutuj złożoność obliczeniowa algorytmów sortowania: przez wstawianie i Quick-sort[46]**
Sortowanie przez wstawianie i Quick-sort to dwa popularne algorytmy sortowania, które różnią się pod względem złożoności obliczeniowej.

*Sortowanie przez wstawianie:*
Algorytm sortowania przez wstawianie polega na wstawianiu elementów z nieposortowanej części tablicy na swoje miejsce w posortowanej części. Dla każdego elementu z nieposortowanej części, algorytm przegląda posortowaną część i wstawia element w odpowiednie miejsce.
Złożoność czasowa:
Najlepsza złożoność czasowa wynosi O(n), gdy tablica jest już posortowana. W najgorszym przypadku, gdy tablica jest posortowana w odwrotnej kolejności, złożoność czasowa wynosi O(n^2). W przypadku średnim, złożoność wynosi również O(n^2), ale dla małych rozmiarów tablicy jest to skuteczny algorytm.

Złożoność pamięciowa:
Złożoność pamięciowa wynosi O(1), ponieważ algorytm sortowania przez wstawianie wykonuje tylko jedną zamianę elementów w pamięci.

*Quick-sort:*

Algorytm Quick-sort polega na rekurencyjnym dzieleniu tablicy na mniejsze podzbiory i sortowaniu każdego z nich. Algorytm wykorzystuje strategię "dziel i zwyciężaj", która pozwala na osiągnięcie złożoności czasowej O(n*log(n)).
Złożoność czasowa:
W najlepszym przypadku, gdy dzielone podzbiory są równe, złożoność czasowa wynosi O(n*log(n)). W najgorszym przypadku, gdy podział jest bardzo nierówny, złożoność czasowa wynosi O(n^2). Jednak szansa na to, że taki podział nastąpi, jest bardzo mała, szczególnie jeśli losowo wybieramy punkt podziału. Dlatego algorytm Quick-sort jest uważany za jedno z najbardziej efektywnych rozwiązań dla problemów sortowania.

Złożoność pamięciowa:
Złożoność pamięciowa wynosi O(log(n)), ponieważ algorytm Quick-sort wykonuje rekurencję, dzieląc tablicę na mniejsze podzbiory, ale zawsze działa na oryginalnej tablicy.

Podsumowując, sortowanie przez wstawianie jest bardziej skuteczne dla małych rozmiarów tablic, ale dla większych rozmiarów tablic, algorytm Quick-sort jest bardziej wydajny ze względu na złożoność czasową

**-Omów korzyści z wykorzystanie kopca do realizacji kolejek priorytetowych.[47]**
Kopiec jest strukturą danych, która może być wykorzystana do realizacji kolejek priorytetowych. Kolejki priorytetowe są strukturami danych, w których elementy są przechowywane w pewnym porządku, a element o najwyższym priorytecie jest zawsze na szczycie kolejki.

Korzyści z wykorzystania kopca do realizacji kolejek priorytetowych to:

Efektywność: Kopiec pozwala na szybkie wstawianie i usuwanie elementów z kolejki, co sprawia, że jest bardzo efektywny w porównaniu z innymi strukturami danych, takimi jak listy czy tablice.

Stały czas dostępu: Czas dostępu do najwyższego priorytetu elementu w kolejce jest stały i wynosi O(1), co oznacza, że nie zależy on od liczby elementów w kolejce.

Łatwa implementacja: Implementacja kolejki priorytetowej przy użyciu kopca jest stosunkowo prosta i nie wymaga złożonych operacji na strukturze.

Możliwość dynamicznego dostosowania kolejki: Kopiec pozwala na dynamiczne dostosowanie rozmiaru kolejki, co oznacza, że można łatwo dodawać lub usuwać elementy w trakcie działania programu.

Wiele zastosowań: Kolejki priorytetowe znajdują zastosowanie w wielu dziedzinach, takich jak algorytmy sortowania, planowanie zadań, przetwarzanie danych, grafy, systemy operacyjne itp.

Podsumowując, wykorzystanie kopca do realizacji kolejek priorytetowych ma wiele korzyści, takich jak efektywność, stały czas dostępu, łatwa implementacja, możliwość dynamicznego dostosowania kolejki oraz wiele zastosowań w różnych dziedzinach.

**-Omów sposób wyszukiwania kluczy w słownikach przy użyciu haszowania i przedyskutuj korzyści ze stosowania tego sposobu.[48]**
Wyszukiwanie kluczy w słownikach przy użyciu haszowania polega na przypisaniu unikalnego kodu haszującego do każdego klucza w słowniku. Ten kod haszujący może być użyty do szybkiego wyszukiwania i odwoływania się do elementów słownika.

Proces haszowania polega na przetwarzaniu klucza za pomocą funkcji haszującej, która przekształca klucz w unikalny kod haszujący. Następnie ten kod haszujący jest używany do indeksowania elementu w słowniku. Gdy chcemy odszukać klucz, ponownie przetwarzamy go funkcją haszującą i porównujemy wynik z kodami haszującymi zapisanymi w słowniku. Jeśli kod haszujący pasuje do klucza, możemy uzyskać dostęp do odpowiadającej mu wartości.

Korzyści ze stosowania haszowania do wyszukiwania kluczy w słownikach to przede wszystkim szybkość i wydajność. Ponieważ kod haszujący jest unikalny dla każdego klucza, proces wyszukiwania jest bardzo szybki i nie wymaga przeszukiwania całego słownika. W przypadku dużych słowników haszowanie może znacznie skrócić czas wyszukiwania.

Ponadto, stosowanie haszowania może zwiększyć wydajność pamięciową, ponieważ słownik nie musi przechowywać całego klucza wraz z jego wartością. Zamiast tego, tylko unikalny kod haszujący jest przechowywany w słowniku.

Jednakże, w przypadku słowników o dużym rozmiarze i wysokiej liczbie kolizji haszowania, czyli sytuacji, gdy dwa różne klucze mają ten sam kod haszujący, wydajność haszowania może spadać, co może prowadzić do konieczności ponownego haszowania lub szukania kluczy w inny sposób. W takich przypadkach konieczne jest zastosowanie specjalnych algorytmów rozwiązywania kolizji, które pomogą utrzymać wydajność haszowania.

**-Omów wybrany przez siebie algorytm grafowy (wstawiania elementu do drzewa poszukiwań binarnych, znajdowania minimalnego drzewa rozpinającego, Dikstry, A∗).[49]**
Wybrałem algorytm Dijkstry, który jest algorytmem służącym do wyszukiwania najkrótszej ścieżki w grafie ważonym. Algorytm ten działa w czasie O(|E| + |V|log|V|), gdzie |E| to liczba krawędzi a |V| to liczba wierzchołków w grafie.

Algorytm Dijkstry działa w sposób iteracyjny, przetwarzając wierzchołki w kolejności od najmniejszej do największej wartości dotychczas znanej odległości od źródła. Dla każdego wierzchołka z listy wierzchołków do przetworzenia algorytm wykonuje następujące kroki:

Sprawdza, czy odległość od źródła do tego wierzchołka jest mniejsza niż dotychczas znana odległość.
Jeśli tak, aktualizuje odległość i przypisuje nowego poprzednika.
Dodaje wierzchołek do listy przetworzonych.
Powtarza cały proces dla każdego sąsiada danego wierzchołka, którego nie ma na liście przetworzonych.
Algorytm kończy pracę, gdy wszystkie wierzchołki zostaną przetworzone lub gdy zostanie znaleziona ścieżka do celu.

Algorytm Dijkstry jest użyteczny w przypadku, gdy chcemy znaleźć najkrótszą ścieżkę w grafie ważonym, ale nie mamy informacji o tym, jakie krawędzie są najważniejsze. W przeciwieństwie do algorytmu Bellmana-Forda, który może działać na grafach z ujemnymi wagami, algorytm Dijkstry działa tylko dla grafów z nieujemnymi wagami. Algorytm ten jest również użyteczny, gdy chcemy znaleźć najkrótszą ścieżkę z jednego wierzchołka do wielu innych, a nie tylko do jednego celu, co czyni go jednym z najważniejszych algorytmów w dziedzinie sieci komputerowych i telekomunikacji.

## SIECI BEZPRZEWODOWE

**-Scharakteryzuj zasadę działania roamingu w sieciach ESS, podaj warunki konfiguracji, które zapewnią użytkownikowi swobodne poruszanie się między AP[29]**
Roaming w sieciach ESS (Extended Service Set) polega na automatycznym przełączaniu się urządzenia końcowego między punktami dostępu (AP), tak aby użytkownik mógł swobodnie poruszać się w obszarze sieci bez utraty połączenia z siecią.

Zasada działania roamingu polega na tym, że urządzenie końcowe, takie jak laptop czy telefon komórkowy, porównuje sygnały otrzymywane od różnych punktów dostępu i przełącza się na AP z silniejszym sygnałem. Oznacza to, że urządzenie końcowe musi być w stanie wykryć sygnały od różnych punktów dostępu, a także być w stanie ocenić ich jakość, aby dokonać wyboru.

Aby zapewnić swobodne poruszanie się między AP, konieczne jest skonfigurowanie sieci w odpowiedni sposób. Oto niektóre z warunków konfiguracji, które zapewnią prawidłowe działanie roamingu:

Punkty dostępu muszą być skonfigurowane w jednej sieci ESS z tym samym SSID i kanałem radiowym.
AP muszą mieć tę samą nazwę i hasło sieci, aby użytkownik mógł bez przeszkód przechodzić między nimi.
Wszystkie punkty dostępu muszą być skonfigurowane w trybie infrastruktury, a nie Ad-hoc.
Każdy punkt dostępu powinien być skonfigurowany z takim samym typem zabezpieczeń.
Powinno być dostępnych wystarczająco dużo punktów dostępu, aby zapewnić pokrycie sieci w całym obszarze i uniknąć przeciążenia.
Dzięki powyższym warunkom roaming będzie działał sprawnie, a użytkownik będzie mógł swobodnie przechodzić między punktami dostępu, bez utraty połączenia z siecią.

**-Na czym polega problem „ukrytego węzła” (hidden node) w sieciach bezprzewodowych[30]**
Problem "ukrytego węzła" (hidden node) występuje w sieciach bezprzewodowych, kiedy dwa węzły znajdujące się w tej samej sieci nie są w stanie bezpośrednio komunikować się ze sobą,

ponieważ są zbyt daleko od siebie lub są zablokowane przez przeszkody, takie jak ściany lub budynki. W tym przypadku, dwa węzły mogą jednocześnie komunikować się z trzecim węzłem, ale nie wiedzą o sobie nawzajem.

Przykładem może być sytuacja, w której dwa urządzenia mobilne próbują się komunikować w sieci bezprzewodowej, ale są zbyt daleko od siebie, aby odbierać sygnały bezpośrednio. W tym przypadku, oba urządzenia mogą wysłać sygnały do punktu dostępu (AP), ale nie są w stanie wykryć sygnałów, które są wysyłane przez drugie urządzenie.

To powoduje problemy związane z transmisją danych w sieci bezprzewodowej, ponieważ węzeł, który znajduje się między ukrytymi węzłami, może nie wiedzieć, że inny węzeł już korzysta z kanału transmisji. W takim przypadku, może wystąpić kolizja, a dane mogą nie zostać prawidłowo przesłane.

Rozwiązaniem problemu "ukrytego węzła" może być zastosowanie odpowiednich mechanizmów kontroli dostępu do medium (MAC), takich jak mechanizm RTS/CTS (Request to Send/Clear to Send) lub mechanizm CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance), które pozwalają na lepszą koordynację i kontrolę transmisji danych w sieci bezprzewodowej.

**-Scharakteryzuj proces przyłączania stacji do sieci bezprzewodowej[31]**
Proces przyłączania stacji do sieci bezprzewodowej (WiFi) składa się z kilku kroków i wymaga pewnej interakcji między stacją a punktem dostępu (AP). Poniżej przedstawiam ogólny opis procesu:

- Skanowanie sieci: Stacja przeszukuje okolicę w poszukiwaniu dostępnych sieci bezprzewodowych. Wszystkie znalezione sieci są wyświetlane w menu wyboru sieci.

- Wybór sieci: Użytkownik wybiera sieć, do której chce się podłączyć.

- Autoryzacja: Stacja wysyła żądanie autoryzacji do punktu dostępu. W przypadku zabezpieczenia sieci hasłem, stacja musi podać poprawne hasło.

- Asocjacja: Po poprawnej autoryzacji, stacja wysyła żądanie asocjacji (przyłączenia) do punktu dostępu. W odpowiedzi, punkt dostępu wysyła stacji informacje o sieci, w tym adres IP, adres MAC i inne parametry.

- Uzyskanie adresu IP: Stacja uzyskuje adres IP, który jest niezbędny do komunikacji w sieci.

- Ustanowienie sesji: Stacja ustanawia sesję z punktem dostępu, co oznacza, że jest gotowa do przesyłania i odbierania danych.

Warto zauważyć, że proces ten może się różnić w zależności od rodzaju zabezpieczenia sieci i konfiguracji punktu dostępu. Na przykład, jeśli sieć jest zabezpieczona hasłem WPA2, to kroki autoryzacji i asocjacji będą bardziej złożone i wymagają podania hasła przez użytkownika.

**-Omów różnice między standardami n (WiFi 4), ac (WiFi 5) i ax (WiFi 6)** [32]
Standardy n (WiFi 4), ac (WiFi 5) i ax (WiFi 6) są kolejnymi generacjami standardów sieci bezprzewodowych, a każdy z nich wprowadza pewne ulepszenia i zmiany w stosunku do poprzedniego. Poniżej przedstawiam najważniejsze różnice między tymi standardami:

- Prędkość: Standard n umożliwia przesyłanie danych z prędkością do 600 Mbps, podczas gdy standard ac zwiększa tę prędkość do 1,3 Gbps, a standard ax do 9,6 Gbps.

- Pasmo: Standard n działa na pasmie 2,4 GHz i/lub 5 GHz, podczas gdy standard ac obsługuje jednocześnie pasma 2,4 GHz i 5 GHz. Standard ax natomiast działa na pasmach 2,4 GHz, 5 GHz i 6 GHz.

- Liczba kanałów: Standard n obsługuje do 40 kanałów, standard ac do 80 kanałów, a standard ax do 160 kanałów.

- Technologia MIMO: Standard n wykorzystuje technologię MIMO (Multiple Input Multiple Output), która umożliwia jednoczesną transmisję danych na kilku strumieniach. Standard ac wprowadza technologię MU-MIMO (Multi-User Multiple Input Multiple Output), która umożliwia jednoczesną transmisję danych do wielu urządzeń. Standard ax natomiast wprowadza technologię MU-MIMO z większą liczbą strumieni.

- Technologia beamforming: Standard ac i ax wprowadzają technologię beamforming, która umożliwia kierowanie sygnału w kierunku konkretnych urządzeń, co poprawia jakość połączenia i zasięg sieci.

- Współpraca z IoT: Standard ax wprowadza poprawki w zakresie działania sieci bezprzewodowych w otoczeniu urządzeń IoT (Internet of Things), co poprawia stabilność sieci i umożliwia lepszą obsługę urządzeń takich jak inteligentne domy, inteligentne miasta itp.

Podsumowując, każdy z kolejnych standardów WiFi wprowadza poprawki i ulepszenia w zakresie prędkości, pasma, liczby kanałów, technologii MIMO, beamformingu oraz współpracy z urządzeniami IoT. Standardy ac i ax wprowadzają również ważne zmiany w zakresie technologii transmisji danych, co poprawia jakość połączenia i umożliwia jednoczesne przesyłanie danych do wielu urządzeń.

**-Omów znaczenie funkcji NAV (Network Allocation Vetor) w protokole komunikacyjnym CSMA/CA**[33]
Funkcja NAV (Network Allocation Vector) w protokole komunikacyjnym CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) jest bardzo ważnym elementem zapewniającym bezpieczne i wydajne działanie sieci bezprzewodowych.

W skrócie, funkcja NAV działa jako wirtualny timer, który informuje inne urządzenia w sieci o czasie trwania transmisji danych na kanale. Kiedy jedno urządzenie rozpoczyna transmisję, wysyła ono sygnał z informacją o długości czasu transmisji (NAV) do innych urządzeń w sieci. Wtedy te urządzenia wyłączają nadajniki i przestają nadawać przez czas trwania transmisji, aby uniknąć kolizji z sygnałem przesyłanym przez urządzenie rozpoczynające transmisję.

Dzięki funkcji NAV, urządzenia w sieci bezprzewodowej mogą bezpiecznie korzystać z jednego kanału radiowego, ponieważ informują się nawzajem o czasie trwania transmisji. Bez NAV, wiele urządzeń mogłoby jednocześnie próbować przesłać dane na kanale, co skutkowałoby kolizjami i spadkiem wydajności sieci.

Warto zauważyć, że funkcja NAV jest szczególnie ważna w przypadku sieci bezprzewodowych pracujących w trybie infrastruktury, gdzie wiele urządzeń korzysta z jednego punktu dostępowego. W przypadku sieci ad-hoc (bez punktu dostępowego) wykorzystanie NAV nie jest tak istotne, ponieważ urządzenia w takiej sieci komunikują się bezpośrednio między sobą.

**-Omów działanie funkcji WPS w urządzeniu dostępowym WLAN (dwie metody uwierzytelniania)[34]**
Funkcja WPS (Wi-Fi Protected Setup) jest protokołem służącym do łatwego i szybkiego konfigurowania urządzeń sieci bezprzewodowych. WPS pozwala użytkownikom na szybkie dodanie nowego urządzenia do sieci WLAN (Wireless Local Area Network), poprzez zminimalizowanie procesu ręcznego wprowadzania kluczy zabezpieczeń.

WPS oferuje dwie metody uwierzytelniania:

- Metoda PBC (Push Button Configuration) - ta metoda polega na tym, że użytkownik naciska przycisk WPS na urządzeniu dostępowym (np. punkt dostępowy), a następnie na urządzeniu klienckim (np. laptopie) w ciągu dwóch minut naciska przycisk WPS. Po naciśnięciu przycisku WPS na urządzeniach, punkt dostępowy i urządzenie klienckie automatycznie nawiązują połączenie i ustawiają zabezpieczenia bezprzewodowe.

- Metoda PIN (Personal Identification Number) - ta metoda polega na wprowadzeniu numeru PIN (8-cyfrowego) z urządzenia klienckiego na urządzenie dostępowe. Użytkownik wprowadza numer PIN w aplikacji konfiguracyjnej urządzenia klienckiego lub w przeglądarce internetowej, a następnie klikając na przycisk "Connect", urządzenie klienckie nawiązuje połączenie z punktem dostępowym.

Funkcja WPS jest łatwa w użyciu i pozwala na szybkie dodanie nowych urządzeń do sieci bezprzewodowej. Niemniej jednak, WPS jest znany ze swoich słabości związanych z bezpieczeństwem, takich jak ataki Brute Force, którym podatny jest numer PIN. Dlatego, wiele firm zajmujących się sieciami bezprzewodowymi zaleca wyłączenie funkcji WPS w urządzeniach sieciowych, aby zwiększyć poziom bezpieczeństwa.

## SZTUCZNA INTELIGENCJA

**-Omów język logiki pierwszego rzędu i wnioskowanie wykorzystujące rezolucję.[75]**
Język logiki pierwszego rzędu (FOL - First-Order Logic) to formalny język używany do reprezentowania informacji w sposób logiczny. Język ten składa się z elementów składniowych, takich jak stałe, zmienne, funkcje i predykaty, oraz reguł semantycznych, które opisują, jak wyrażenia w tym języku mają być interpretowane.

Predykaty są podstawowymi elementami języka FOL, które służą do opisywania własności i relacji między obiektami. Predykaty są zazwyczaj oznaczane literami lub symbolami, a ich argumentami mogą być stałe lub zmienne. Przykładem predykatu może być "jest mniejsze od", który opisuje relację między dwoma liczbami.

Wnioskowanie wykorzystujące rezolucję jest techniką służącą do dowodzenia twierdzeń w języku FOL. Rezolucja polega na tworzeniu klauzul, które są logicznymi formułami, zawierającymi predykaty i zmienne. Każda klauzula może być pozytywna lub negatywna. W procesie wnioskowania wykorzystuje się rezolucję, która polega na wykorzystaniu reguł logicznych, aby zredukować klauzule do prostszych form.

Proces rezolucji polega na łączeniu dwóch klauzul w celu utworzenia nowej klauzuli. Klauzule są łączone za pomocą reguły rezolucji, która mówi, że jeśli w dwóch klauzulach występuje taki sam literał, ale jeden jest pozytywny, a drugi negatywny, to można je zredukować do jednej klauzuli.

Wnioskowanie wykorzystujące rezolucję polega na próbie znalezienia sprzeczności między zestawem klauzul, które reprezentują jakieś twierdzenie lub zestaw faktów. Jeśli uda się znaleźć sprzeczność, to oznacza, że początkowy zestaw klauzul jest fałszywy. W ten sposób można wykorzystać wnioskowanie wykorzystujące rezolucję do dowodzenia twierdzeń i wnioskowania na temat danych w języku FOL.

Rezolucja jest jednym z podstawowych algorytmów stosowanych w dziedzinie sztucznej inteligencji i jest używana do rozwiązywania różnych problemów, takich jak problemy planowania, analizy obrazów, a także do projektowania systemów ekspertowych.

**-Scharakteryzuj metody reprezentacji wiedzy niepewnej (systemy rozmyte i probabilistyczne).[76]**
Wiedza niepewna to taka, której nie można jednoznacznie określić jako prawdziwej lub fałszywej. W takich przypadkach wykorzystuje się metody reprezentacji wiedzy niepewnej, które pozwalają na bardziej elastyczne i realistyczne podejście do modelowania i rozwiązywania problemów.

Jedną z metod reprezentacji wiedzy niepewnej są systemy rozmyte (fuzzy logic). Systemy rozmyte pozwalają na modelowanie wartości, które nie mają dokładnej wartości binarnej (prawda/fałsz), ale istnieją w zakresie wartości od 0 do 1. Na przykład, w przypadku opisu temperatury można powiedzieć, że jest "gorąco", "umiarkowanie" lub "zimno". Takie wartości można przedstawić za pomocą funkcji przynależności, które opisują stopień przynależności elementu do danej kategorii. Systemy rozmyte pozwalają na bardziej elastyczne podejście do modelowania problemów i pozwalają na bardziej precyzyjne wyniki.

Inną metodą reprezentacji wiedzy niepewnej są systemy probabilistyczne (probability theory). Systemy probabilistyczne pozwalają na modelowanie niepewności za pomocą prawdopodobieństwa. Na przykład, jeśli chodzi o diagnostykę medyczną, nie zawsze można jednoznacznie stwierdzić, czy dana osoba choruje na daną chorobę, ale można określić prawdopodobieństwo, z jakim dana osoba jest chora. Systemy probabilistyczne pozwalają na wykorzystanie różnych metod statystycznych i probabilistycznych do modelowania problemów, co pozwala na bardziej realistyczne i precyzyjne wyniki.

**-Omów rodzaje uczenia maszyn.[77]**
Uczenie maszynowe to dziedzina sztucznej inteligencji, która polega na tworzeniu algorytmów i modeli, które uczą się na podstawie danych i doświadczenia, aby potrafiły rozwiązywać problemy i podejmować decyzje na podstawie nowych sytuacji. Istnieją trzy podstawowe rodzaje uczenia maszynowego:

Uczenie nadzorowane (supervised learning) - polega na wykorzystaniu zestawu danych treningowych, które zawierają odpowiedzi (etykiety) dla każdego elementu. Algorytm uczy się na podstawie tych danych, a następnie używa tych informacji, aby przewidywać odpowiedzi dla nowych danych. Przykłady zastosowań uczenia nadzorowanego to: rozpoznawanie obrazów, klasyfikacja tekstu, przewidywanie cen akcji, itp.

Uczenie nienadzorowane (unsupervised learning) - polega na wykorzystaniu zestawu danych, które nie mają przypisanych etykiet. Algorytm uczy się samodzielnie na podstawie struktury danych i szuka ukrytych wzorców lub grup danych. Przykłady zastosowań uczenia nienadzorowanego to: klastrowanie danych, redukcja wymiarowości, wykrywanie anomalii, itp.

Uczenie ze wzmocnieniem (reinforcement learning) - polega na nauki poprzez interakcję z otoczeniem, w którym agent próbuje wykonać określone zadania. Za każde poprawne działanie agent otrzymuje nagrodę, a za błędne działanie otrzymuje karę. Algorytm uczy się, aby maksymalizować otrzymywane nagrody. Przykłady zastosowań uczenia ze wzmocnieniem to: sterowanie robotami, nauka gry w gry planszowe, itp.

Istnieją również inne rodzaje uczenia maszynowego, takie jak uczenie pół-nadzorowane, uczenie transferowe, uczenie multi-tasking, uczenie aktywne, itp. Każdy rodzaj uczenia maszynowego ma swoje wady i zalety i jest wykorzystywany w zależności od problemu, który chcemy rozwiązać.

**-Omów wybrany przez siebie algorytm uczenia (gradientowy, symulowanego wyżarzania, ewolucyjny, roju).[78]**
Algorytmy ewolucyjne (EA - Evolutionary Algorithms) są to metaheurystyki inspirowane procesami biologicznymi, takimi jak ewolucja i selekcja naturalna. Algorytm ewolucyjny generuje populację losowych rozwiązań, a następnie w iteracyjny sposób wykonuje operacje takie jak krzyżowanie, mutacja i selekcja, aby tworzyć kolejne pokolenia populacji.

Poniżej przedstawione są kroki algorytmu ewolucyjnego:

1) Inicjalizacja populacji początkowej - algorytm rozpoczyna od wygenerowania losowej populacji początkowej, której rozmiar i wartości są określone przez użytkownika.

2) Ocena funkcji celu - każde rozwiązanie w populacji jest oceniane pod kątem jakości przez funkcję celu, która określa, jak dobrze dany osobnik rozwiązuje problem.

3) Selekcja - wybierane są najlepsze osobniki z populacji do kolejnego pokolenia. Najczęściej stosowaną metodą selekcji jest metoda ruletki, w której szansa na wybór danego osobnika jest proporcjonalna do jego jakości.

4) Krzyżowanie - wybrane osobniki są krzyżowane, aby stworzyć nowe rozwiązania. Krzyżowanie polega na wymianie części informacji genetycznej między rodzicami, aby stworzyć potomstwo z nowymi cechami.

5) Mutacja - nowo powstałe osobniki są mutowane, aby wprowadzić losowe zmiany w ich cechach. Mutacja zapewnia różnorodność w populacji i pomaga uniknąć zatrzymania w lokalnym minimum.

6) Ograniczenie rozmiaru populacji - jeśli populacja osiągnie maksymalny rozmiar, niektóre osobniki muszą zostać usunięte, aby zrobić miejsce dla nowych potomków.

7) Powtarzanie kroków 2-6 - proces tworzenia kolejnych pokoleń populacji jest powtarzany, aż osiągnięty zostanie warunek zakończenia (np. maksymalna liczba iteracji lub znalezienie wystarczająco dobrego rozwiązania).

Algorytm ewolucyjny jest stosowany w wielu dziedzinach, takich jak optymalizacja, projektowanie systemów, sztuczna inteligencja i robotyka. Jego zaletami są łatwa implementacja, zdolność do radzenia sobie z wieloma typami problemów optymalizacyjnych i zdolność do znalezienia globalnego minimum. Jednak czasami algorytm może wymagać dużych zasobów obliczeniowych, zwłaszcza przy dużych populacjach i złożonych problemach.

**-Co to jest sieć neuronowa i jak można ją uczyć?[79]**
Sieć neuronowa to model matematyczny, który jest inspirowany biologicznymi sieciami neuronów w mózgu. Składa się on z połączonych ze sobą sztucznych neuronów, które przetwarzają informacje wejściowe i generują wyjście na podstawie swojego wewnętrznego stanu i połączeń z innymi neuronami. Sieci neuronowe są używane w różnych dziedzinach, takich jak rozpoznawanie obrazów, przetwarzanie języka naturalnego, predykcja i analiza danych finansowych.

Aby nauczyć sieć neuronową, należy dostarczyć jej zestawu danych uczących, które składają się z wejść i odpowiadających im wyjść. Następnie sieć jest trenowana na tych danych, a wagi połączeń między neuronami są dostosowywane tak, aby minimalizować błąd między wyjściem sieci a oczekiwanym wyjściem dla danego wejścia.

Istnieją różne metody uczenia sieci neuronowej, takie jak:

Wsteczna propagacja błędów (backpropagation): najpopularniejsza metoda uczenia sieci neuronowej, polega na obliczeniu błędu sieci dla każdej próbki uczącej i propagacji tego błędu wstecz przez sieć, aby dostosować wagi połączeń.
Uczenie ze wzmocnieniem (reinforcement learning): polega na uczeniu sieci neuronowej poprzez nagradzanie lub karanie jej na podstawie jej zachowania w środowisku.
Uczenie nienadzorowane (unsupervised learning): polega na uczeniu sieci neuronowej bez używania par wejście-wyjście, ale zamiast tego poprzez analizę właściwości danych wejściowych i odkrycie struktury danych.
Wybór odpowiedniej metody uczenia zależy od problemu, który chcemy rozwiązać oraz od charakterystyki danych uczących.

**-Co to jest generalizacja, od czego zależy i jak ją można poprawić?[80]**
Generalizacja to zdolność modelu do dobrze wykonania na danych, których nie widział podczas treningu. Chodzi o to, aby model nie zapamiętywał danych treningowych, ale aby wyciągnął ogólne reguły, które można zastosować do nowych danych. Model, który nie jest w stanie generalizować, może wykazywać zjawiska takie jak overfitting, w którym model dostosowuje się zbyt dokładnie do danych treningowych i nie potrafi poprawnie przewidywać nowych danych.

Generalizacja zależy od wielu czynników, w tym od ilości i jakości danych treningowych, jakości modelu, jego architektury i hiperparametrów oraz od sposobu uczenia modelu. Istnieje kilka sposobów poprawienia generalizacji, takich jak:

- Zwiększenie ilości danych treningowych - większe zbiory danych mogą pomóc w zwiększeniu generalizacji poprzez pokrycie większej liczby przypadków.

- Zmniejszenie złożoności modelu - model z mniejszą ilością parametrów lub mniejszą liczbą warstw może być mniej skłonny do overfittingu.
- Używanie regularyzacji - regularyzacja może pomóc w kontrolowaniu wielkości wag modelu i zapobiec overfittingowi.

- Używanie walidacji krzyżowej - walidacja krzyżowa pozwala na ocenę ogólnej wydajności modelu na wielu zbiorach danych treningowych i testowych, co może pomóc w zwiększeniu generalizacji.

- Testowanie modelu na różnych danych testowych - testowanie modelu na różnych danych testowych może pomóc w ocenie jego zdolności do generalizacji.

**-Omów problem poszukiwania centrów klastrów.[81]**
Problem poszukiwania centrów klastrów to jeden z problemów analizy skupień (ang. clustering analysis) w dziedzinie uczenia maszynowego. Polega na znalezieniu optymalnej liczby i położenia centrów klastrów, tak aby minimalizować sumę kwadratów odległości pomiędzy punktami danych a ich przypisanymi centrami.

W zależności od algorytmu wykorzystywanego do rozwiązania problemu, podejście do znajdowania centrów klastrów może się różnić. Jednym z popularnych algorytmów jest algorytm k-średnich (k-means), który działa w następujący sposób:

1) Losowo wybiera k punktów danych jako początkowe centra klastrów.
2) Przypisuje każdy punkt danych do najbliższego centrum klastra.
3) Oblicza nowe położenia centrów klastrów na podstawie średniej arytmetycznej przypisanych do nich punktów danych.
4) Powtarza kroki 2 i 3, aż nie nastąpi zbieżność.

Innym popularnym algorytmem jest algorytm hierarchiczny, który tworzy drzewo hierarchii klastrów. Algorytm zaczyna od pojedynczych punktów danych i łączy je w klastry w oparciu o podobieństwo między nimi. Klastery te mogą następnie zostać połączone w większe klastry,

tworząc drzewo hierarchii. Ostateczne położenia centrów klastrów można określić na podstawie tego drzewa.

Problem poszukiwania centrów klastrów jest ważnym narzędziem w analizie danych, a jego rozwiązanie może mieć wiele zastosowań, takich jak grupowanie klientów w celach marketingowych, analiza zachowań użytkowników lub identyfikacja wzorców w danych.

**-Budowa drzew decyzyjnych metodą ID3.[82]**
Metoda ID3 (Iterative Dichotomiser 3) to jedna z najstarszych i najprostszych metod budowy drzew decyzyjnych, stosowana głównie w celu rozwiązania problemów klasyfikacyjnych. Algorytm ten polega na iteracyjnym budowaniu drzewa, gdzie w każdym kroku wybierany jest atrybut, który najlepiej podzieli zbiór danych na grupy o jak największej czystości (tj. jednorodności klasowej).

Poniżej przedstawione są kroki algorytmu ID3:

1) Określenie celu klasyfikacji oraz zbioru danych treningowych.
2) Obliczenie entropii dla całego zbioru danych. Entropia mierzy nieokreśloność klasyfikacji i określa stopień nieuporządkowania danych. Wartość entropii wynosi 0, gdy wszystkie próbki należą do tej samej klasy, a wartość ta rośnie wraz z liczbą klas i ich rozproszeniem w zbiorze danych.

3) Wybór atrybutu, dla którego zysk informacyjny (information gain) będzie największy. Zysk informacyjny obliczany jest jako różnica entropii zbioru przed i po podziale ze względu na dany atrybut.
4) Podział zbioru danych na podzbiory ze względu na wartości wybranego atrybutu.

5) Utworzenie nowych węzłów w drzewie decyzyjnym, które odpowiadają wartościom atrybutu, a następnie rekurencyjne wywołanie algorytmu dla każdego z podzbiorów danych, aż do osiągnięcia warunku stopu.
6) Przypisanie klas dla liści drzewa, np. poprzez wybranie najczęściej występującej klasy.

Ogólnie rzecz biorąc, celem budowy drzewa decyzyjnego jest stworzenie modelu predykcyjnego, który pozwoli na klasyfikację nowych obiektów na podstawie wartości atrybutów. Warto jednak pamiętać, że drzewa decyzyjne mają tendencję do przeuczenia (overfitting), czyli zbytniego dopasowania do danych treningowych. Dlatego ważne jest stosowanie odpowiednich technik regularyzacji oraz walidacji modelu.