**Przedmiot: Podstawy informatyki (dr inż. Witold Alda)**

1. Jednym z prostych sposobów zapisu algorytmu jest schemat blokowy. Proszę podać zasady tworzenia schematu blokowego.

***Jednym z prostych i intuicyjnych sposobów przedstawienia algorytmu jest schemat blokowy.*** *Schemat blokowy to graficzna forma zapisu algorytmu, która przedstawia kolejne kroki w postaci połączonych symboli. Aby poprawnie stworzyć schemat blokowy, należy przestrzegać kilku podstawowych zasad:*Elipsa Start / Stop Początek i koniec algorytmu  
Prostokąt Działanie (proces) Wykonanie operacji, np. obliczenia  
Równoległobok Wejście / Wyjście Odczyt danych, wypisanie wyniku  
Romb Decyzja (warunek) Sprawdzenie warunku logicznego  
Strzałka Przepływ Określa kierunek przepływu sterowania

2. Proszę omówić znaczenie pojęć: składnia i semantyka języka programowania.

**Składnia (ang. syntax)**  
Składnia określa **zasady poprawnego zapisu kodu źródłowego** w danym języku programowania. Obejmuje m.in.:  
kolejność słów kluczowych, sposób definiowania instrukcji, zmiennych, funkcji, użycie nawiasów, przecinków, średników itp.Można powiedzieć, że składnia to „gramatyka” języka programowania. Jeśli kod ma błędy składniowe, nie zostanie skompilowany lub uruchomiony.

**Semantyka (ang. semantics)**  
Semantyka dotyczy **znaczenia instrukcji** – czyli tego, **co dany fragment kodu robi**.  
Nawet jeśli składnia jest poprawna, kod może robić coś innego niż zamierzał programista – i wtedy mamy do czynienia z błędem semantycznym.

3. Jednym z podstawowych pojęć w informatyce jest pojęcie zmiennej. Proszę podać cechy charakterystyczne zmiennej oraz omówić przynajmniej dwie klasy zmiennych.

Zmienne służą do przechowywania danych w pamięci komputera, które mogą być wykorzystywane i modyfikowane podczas działania programu.

Cechy charakterystyczne zmiennej:  
Nazwa – unikalny identyfikator, za pomocą którego odwołujemy się do zmiennej.  
Typ danych – określa, jaki rodzaj danych może być przechowywany (np. liczba całkowita, tekst, liczba zmiennoprzecinkowa).  
Wartość – aktualna zawartość zmiennej (może się zmieniać podczas działania programu).  
Zakres (ang. scope) – określa, gdzie w programie zmienna jest dostępna.  
Czas życia (ang. lifetime) – określa, jak długo zmienna istnieje w pamięci.

Lokalna, Globalna

4. Stos i sterta. Proszę omówić te pojęcia i na przykładzie w języku ANSI C pokazać kiedy i w jaki sposób korzystamy ze stosu, a kiedy ze sterty.

Stos i sterta to dwa różne obszary pamięci operacyjnej wykorzystywane podczas działania programu, w szczególności w języku C. Służą one do przechowywania danych, ale działają na innych zasadach i mają różne zastosowania.

Stos (ang. stack)  
Pamięć przydzielana automatycznie przez kompilator.  
Używany do przechowywania zmiennych lokalnych i danych funkcji (np. parametrów).  
Działa na zasadzie LIFO – ostatni przydzielony element jest pierwszy zwalniany.  
Szybki i wydajny, ale ma ograniczony rozmiar.

Sterta (ang. heap)  
Pamięć przydzielana dynamicznie przez programistę za pomocą funkcji malloc, calloc, realloc.  
Wymaga ręcznego zwolnienia pamięci (free()).  
Używana do alokacji danych, które muszą istnieć dłużej niż czas działania jednej funkcji (np. duże tablice, struktury).  
Ma większą pojemność niż stos, ale dostęp jest wolniejszy.

5. Omówić metody przekazywania argumentów do funkcji i pokazać je na przykładach w języku ANSI C.

W języku ANSI C istnieją dwie główne metody przekazywania argumentów do funkcji:  
Przekazywanie przez wartość (pass by value):  
Domyślna metoda w C. Do funkcji przekazywana jest kopią wartości zmiennej – oryginał nie zostaje zmieniony.  
Przekazywanie przez wskaźnik (symulacja pass by reference):  
Przekazujemy adres zmiennej, a nie jej wartość. Pozwala funkcji zmienić oryginalną zmienną. zmien(int \*x); zmien(&a);

**6. Rekurencja. Proszę omówić na czym polega i pokazać prosty przykład – można posłużyć się metajęzykiem.**

Rekurencja to technika programistyczna, w której funkcja wywołuje sama siebie, aby rozwiązać mniejsze instancje danego problemu. Jest to metoda dzielenia problemu na podproblemy tego samego typu, aż do osiągnięcia tzw. warunku brzegowego (czyli takiego, który nie wymaga dalszego wywoływania funkcji).

Warunek brzegowy (bazowy) – moment zatrzymania wywołań rekurencyjnych.  
Wywołanie rekurencyjne – funkcja wywołuje samą siebie z mniejszym lub zmienionym argumentem.

**Przedmiot: Metody probabilistyczne i statystyka**

**1. Miary tendencji centralnej i miary rozrzutu.**  
Podstawowe pojęcia w statystyce, które służą do opisu i analizy danych liczbowych.  
**Miary tendencji centralnej**  
Miary tendencji centralnej pokazują, wokół jakiej wartości koncentrują się dane – czyli jaka jest ich „środkowa” lub „typowa” wartość.  
Najczęściej stosowane miary:  
***Średnia arytmetyczna*** – suma wszystkich wartości podzielona przez ich liczbę.  
**Mediana** – wartość środkowa po uporządkowaniu danych rosnąco. (Dla danych: 1, 3, 5 → mediana = **3**)  
***Moda*** (dominanta) – wartość występująca najczęściej. (Dla danych: 1, 2, 2, 3, 4 → moda = **2**)

**Miary rozrzutu (zmienności)**  
Miary rozrzutu pokazują, jak bardzo dane różnią się od siebie – czyli jak bardzo są „rozproszone” wokół wartości centralnej.  
Najczęściej stosowane miary:  
***Rozstęp*** – różnica między największą a najmniejszą wartością.  
***Wariancja***– średnia z kwadratów odchyleń od średniej. Jest podstawą do obliczenia odchylenia standardowego. ***Odchylenie standardowe*** – informuje, jak bardzo wartości oddalają się od średniej.

Obliczanie wariancji i odchylenia standardowego – krok po kroku  
*Dane: 2, 4, 6; n=3  
Średnia arytmetyczna (mean): (2 + 4 + 6) / 3 = 4  
Odchylenia od średniej: 2 − 4 = −2; 4 − 4 = 0; 6 − 4 = 2 -> Kwadraty odchyleń (squared deviations): (−2)² = 4; 0² = 0; 2² = 4  
Suma kwadratów odchyleń: 4 + 0 + 4 = 8  
Wariancja próbki: Oznaczenie: s² = 4  
8 / (3 − 1) = 8 / 2 = 4  
Odchylenie standardowe (standard deviation):  
√4 = 2; s = 2*  
  
  
**2. Omówić problem estymacji przedziałowej.**

Estymacja przedziałowa to metoda statystyczna służąca do oszacowania nieznanej wartości parametru populacji na podstawie próby w taki sposób, że zamiast jednej wartości (punktowej), podajemy przedział, w którym z określonym prawdopodobieństwem ten parametr się znajduje.

Elementy estymacji przedziałowej  
Punktowa estymata – wartość obliczona z próby (np. średnia próby).  
Przedział ufności – przedział liczbowy, w którym z określonym prawdopodobieństwem znajduje się szacowany parametr.  
Poziom ufności (np. 90%, 95%, 99%) – prawdopodobieństwo, że przedział zawiera prawdziwą wartość parametru.  
Estymacja przedziałowa nie mówi, że „z 95% prawdopodobieństwem średnia leży w tym konkretnym przedziale” – mówi raczej, że gdybyśmy wiele razy powtarzali doświadczenie, to w 95% przypadków wyznaczone przedziały zawierałyby prawdziwą wartość parametru.

3. Scharakteryzować podstawowe pojęcia i omówić krótko procedurę weryfikacji hipotezy statystycznej.

Weryfikacja hipotezy statystycznej to jedna z podstawowych metod analizy danych, która pozwala sprawdzić, czy na podstawie wyników próby można potwierdzić przypuszczenia (hipotezy) dotyczące całej populacji.

Podstawowe pojęcia:  
Hipoteza zerowa (H0)  
– To założenie „braku efektu” lub „braku różnicy”.  
– Jest to domyślna teza, którą testujemy i próbujemy obalić.

Hipoteza alternatywna (H1 lub Ha)  
– Twierdzenie przeciwne do hipotezy zerowej.

Poziom istotności (α)  
– Określa maksymalne dopuszczalne ryzyko popełnienia błędu I rodzaju.  
– Najczęściej: α = 0,05 (5%).

Wartość p (p-value)  
– Prawdopodobieństwo uzyskania takiego wyniku przy założeniu, że H0 jest prawdziwa.  
– Jeśli p < α, to odrzucamy H0.

Procedura  
H0: hipoteza zerowa  
H1: hipoteza alternatywna  
Wybór poziomu istotności α  
Dobór odpowiedniego testu statystycznego  
– np. test t-Studenta, test Z, test chi-kwadrat – zależnie od danych  
Obliczenie statystyki testowej i wartości p

Podjęcie decyzji:  
Jeśli p < α, odrzucamy Ha (hipoteza alternatywna jest prawdopodobna).  
Jeśli p ≥ α, brak podstaw do odrzucenia Ha.

**4. Statystyczne miary ilości informacji (entropia Shannona i miary pochodne).**W informatyce często chcemy wiedzieć, ile informacji zawiera jakiś komunikat. Do tego służą statystyczne miary informacji, a najważniejszą z nich jest entropia Shannona.  
Entropia Shannona  
Entropia to najważniejsza miara ilości informacji, wprowadzona przez Claude’a Shannona.  
Określa średnią ilość informacji, jaką niesie jedno zdarzenie lub symbol (np. znak w tekście).  
Co oznacza: Im bardziej losowe i nieprzewidywalne są dane, tym większa entropia.  
Im dane są powtarzalne i przewidywalne, tym entropia jest mniejsza.  
  
Miary pochodne od entropii  
Entropia warunkowa  
– Mierzy, ile niepewności zostaje, jeśli znamy inne dane.  
– Przykład: znajomość poprzedniego słowa może pomóc przewidzieć następne w zdaniu.

Wzajemna informacja  
– Mierzy, ile informacji jedno źródło przekazuje o drugim.  
– Przykład: ile wiedzy o X daje nam znajomość Y.

Dywergencja Kullbacka-Leiblera (KL)  
– Porównuje dwa rozkłady danych i mówi, jak bardzo się różnią.  
– Używana np. w uczeniu maszynowym.

Zastosowanie:  
Kompresja danych (np. ZIP, JPG), Szyfrowanie i bezpieczeństwo, Kodowanie sygnałów i transmisja danych

**5. Kompresja, kodowanie i szyfrowanie w informatyce.**

W informatyce pojęcia kompresji, kodowania i szyfrowania dotyczą przetwarzania i transmisji danych, ale każde z nich ma inny cel i zastosowanie.  
**Kompresja danych**  
Kompresja to proces zmniejszania rozmiaru danych w celu zaoszczędzenia miejsca na dysku lub przyspieszenia transmisji.  
Bezstratna – pozwala na dokładne odtworzenie oryginału. (ZIP, PNG, FLAC)  
Stratna – usuwa część danych, by zmniejszyć rozmiar kosztem jakości.(MP3, JPEG, MP4)  
zastosowanie: Przesyłanie plików przez Internet, Archiwizacja danych, Multimedia (dźwięk, obraz, wideo)  
**Kodowanie danych**  
Kodowanie polega na przekształcaniu danych do innej postaci – czytelnej dla maszyn lub ludzi, bez ukrywania treści.  
Przykłady: ASCII / Unicode – zamiana znaków na liczby binarne, Base64 – zamiana danych binarnych na ciąg znaków tekstowych (np. w e-mailach), Kodowanie Huffmana – używane w kompresji bezstratnej  
zastosowanie: Przechowywanie i transmisja danych, Formatowanie tekstów, plików, protokołów sieciowych  
**Szyfrowanie danych**  
Szyfrowanie to proces przekształcania danych w taki sposób, aby osoby nieuprawnione nie mogły ich odczytać. Stosuje się klucze (tajne lub publiczne) do zabezpieczania informacji.  
Symetryczne – ten sam klucz do szyfrowania i deszyfrowania (np. AES)  
Asymetryczne – różne klucze: publiczny i prywatny (np. RSA)  
zastosowanie: Bezpieczeństwo transmisji (np. HTTPS, VPN), Ochrona danych osobowych, Kryptografia

**Przedmiot: Języki formalne i kompilatory**

KOMPOLATOR

## ⚙️ 1. Wykonanie poleceń preprocesora  
- Obsługa dyrektyw preprocesora (np. `#include`, `#define`)  
- Usunięcie komentarzy  
- Wstawienie plików nagłówkowych  
## 🔎 2. Analiza leksykalna  
- Przekształca kod na tokeny (słowa kluczowe, identyfikatory, liczby)  
- Wykrywanie błędnych symboli  
## 🧩 3. Analizator syntaktyczny, Analiza składniowa (parsing)  
- Sprawdzenie zgodności z gramatyką języka  
- **Tworzenie drzewa składniowego** (AST – *abstract syntax tree*)  
## 🧠 4. Analiza semantyczna  
- Sprawdzenie typów danych  
- Sprawdzenie, czy zmienne są zadeklarowane  
- Weryfikacja poprawności wyrażeń  
## 🚀 5. Optymalizacja kodu wynikowego  
- Usprawnianie kodu (np. eliminacja martwego kodu)  
- Redukcja liczby instrukcji  
## 🛠️ 6. Generowanie kodu  
- Tworzenie kodu maszynowego lub pośredniego (np. bytecode)  
- Przygotowanie do linkowania i wykonania

leksykalna->syntaktyczny->semantyczna

**1. Kompilacja i interpretacja, języki skryptowe i kompilowane, język Java.**

Kompilacja to proces, w którym cały kod źródłowy programu jest przetwarzany na kod maszynowy przed jego uruchomieniem. Kompilowane języki, takie jak C czy C++, wymagają wcześniejszego przetworzenia kodu źródłowego na plik wykonywalny. Interpretacja natomiast oznacza, że program jest przetwarzany i uruchamiany w trakcie jego działania przez interpreter, jak ma to miejsce w językach skryptowych, takich jak Python czy JavaScript. Java łączy cechy obu podejść: kod źródłowy jest najpierw kompilowany do kodu pośredniego (bytecode), który następnie jest interpretowany przez maszynę wirtualną JVM (Java Virtual Machine).

**2. Podstawowe moduły funkcjonalne kompilatora, rola kodu pośredniego przy kompilacji.**

Kompilator składa się z kilku podstawowych modułów funkcjonalnych: analizatora leksykalnego, analizatora syntaktycznego, analizatora semantycznego, generatora kodu pośredniego oraz optymalizatora i generatora kodu maszynowego. Kod pośredni, generowany po analizie syntaktycznej i semantycznej, stanowi abstrakcyjną formę programu, która może być optymalizowana i przekształcana na kod maszynowy odpowiedni dla różnych architektur sprzętowych. Użycie kodu pośredniego pozwala na większą przenośność i efektywność kompilacji.

**3. Wyjaśnić pojęcia: analiza leksykalna, wyrażenie regularne, automat skończony.**

Analiza leksykalna to proces, w którym kod źródłowy jest przetwarzany na mniejsze elementy zwane leksemami (tokeny). Analizator leksykalny grupuje ciągi znaków w logiczne jednostki, takie jak słowa kluczowe, identyfikatory, operatory czy literały. Wyrażenie regularne to formalny sposób opisywania wzorców w ciągach znaków i są one często wykorzystywane w analizie leksykalnej do identyfikowania leksemów. Automat skończony to model matematyczny służący do reprezentacji procesów o skończonej liczbie stanów, wykorzystywany w analizatorach leksykalnych do przetwarzania wejściowego ciągu znaków.

**4. Wyjaśnić pojęcia: analiza syntaktyczna, gramatyka, automat ze stosem.**Analiza syntaktyczna (parsing) to etap przetwarzania kodu źródłowego, w którym analizator syntaktyczny sprawdza, czy ciąg tokenów generowany przez analizator leksykalny spełnia zasady gramatyki języka programowania. Gramatyka to zbiór reguł definiujących poprawną strukturę wyrażeń w języku formalnym. Automat ze stosem to zaawansowany model automatu, który korzysta ze stosu do przechowywania dodatkowych informacji o stanie analizy, co pozwala mu na skuteczne przetwarzanie bardziej złożonych struktur językowych, takich jak zagnieżdżone nawiasy.

**5. Przykładowe techniki optymalizacji kodu pośredniego.**

Optymalizacja kodu pośredniego ma na celu poprawę efektywności kodu wynikowego poprzez redukcję nadmiarowych operacji lub wykorzystanie bardziej efektywnych instrukcji. Przykłady technik optymalizacji obejmują eliminację martwego kodu (usuwanie instrukcji, które nie wpływają na wynik), minimalizację liczby przejść do innych bloków kodu (redukowanie skoków) oraz zastępowanie złożonych wyrażeń prostszymi, które mają taką samą funkcjonalność (optymalizacja siłowa). Inne techniki to propagacja stałych, gdzie wartości stałe są wstawiane bezpośrednio do kodu zamiast zmiennych, oraz unikanie powtarzających się obliczeń.

**1. Relacyjny Model Danych – teoria, podstawowe elementy i związki logiczne.**

**Relacyjny Model Danych** (RMD) to jeden z najpowszechniej stosowanych modeli organizacji danych w bazach danych. Został zaproponowany przez Edgara F. Codda w 1970 roku. Jego fundamentalna idea polega na reprezentowaniu danych w postaci dwuwymiarowych tabel, zwanych **relacjami**.

**Teoria Relacyjnego Modelu Danych:**  
- Opiera się na matematycznej teorii zbiorów i logice pierwszego rzędu.  
- Dane są postrzegane jako zbiory krotek (wierszy), gdzie każda krotka reprezentuje fakt o świecie rzeczywistym.  
- Zapewnia wysoką niezależność danych od aplikacji, co oznacza, że zmiany w strukturze bazy danych (np. dodanie nowej kolumny) nie wymagają zazwyczaj modyfikacji kodu aplikacji.  
- Umożliwia operowanie na danych za pomocą języka wysokiego poziomu (np. SQL)

**Podstawowe elementy Relacyjnego Modelu Danych:**

**Relacja (Tabela):** Podstawowy element strukturalny, przedstawiający dane w postaci dwuwymiarowej siatki. Składa się z nagłówka (schematu) i ciała.

**Atrybut (Kolumna):** Nazwana właściwość, która opisuje pewien aspekt encji. W każdej kolumnie dane są jednorodne pod względem typu (np. tekst, liczba całkowita, data).

**Krotka (Wiersz):** Pojedynczy rekord w tabeli, składający się z wartości dla wszystkich atrybutów. Reprezentuje pojedynczy obiekt lub zdarzenie.  
  
**Klucz główny (Primary Key - PK):** Jeden lub więcej atrybutów, których wartości jednoznacznie identyfikują każdą krotkę w relacji. Nie może zawierać wartości NULL.  
**Klucz obcy (Foreign Key - FK):** Jeden lub więcej atrybutów w jednej relacji, których wartości odwołują się do klucza głównego innej relacji (lub tej samej). Służy do ustanawiania związków logicznych między tabelami i zapewniania integralności referencyjnej.

**Związki logiczne (relacje) w Relacyjnym Modelu Danych:**

Związki między tabelami są realizowane poprzez dopasowywanie wartości kluczy obcych do wartości kluczy głównych. Wyróżniamy trzy podstawowe typy związków:

**Jeden do jednego (1:1):** Każdy wiersz w tabeli A jest powiązany z co najwyżej jednym wierszem w tabeli B, i na odwrót. Jest to rzadki typ związku, często oznacza, że dane mogłyby być w jednej tabeli.  
**Jeden do wielu (1:N):** Każdy wiersz w tabeli A może być powiązany z wieloma wierszami w tabeli B, ale każdy wiersz w tabeli B jest powiązany z co najwyżej jednym wierszem w tabeli A.  
**Wiele do wielu (N:M):** Każdy wiersz w tabeli A może być powiązany z wieloma wierszami w tabeli B, i na odwrót. Ten typ związku jest realizowany za pomocą dodatkowej tabeli pośredniczącej (zwanej również tabelą połączeniową lub asocjacyjną), która zawiera klucze obce z obu powiązanych tabel, tworząc złożony klucz główny.

**2. Systemy Zarządzania Bazą Danych – rola kluczowych modułów funkcjonalnych**

System Zarządzania Bazą Danych (DBMS, ang. *Database Management System*) to oprogramowanie pośredniczące pomiędzy użytkownikiem (lub aplikacją) a bazą danych. Jego głównym celem jest efektywne i bezpieczne zarządzanie danymi, zapewniając ich spójność, dostępność oraz integralność.

DBMS składa się z wielu modułów funkcjonalnych, z których najważniejsze to:

1. Moduł Języka Definicji Danych (DDL Processor)
   * Rola: Odpowiada za przetwarzanie poleceń DDL (ang. *Data Definition Language*), takich jak CREATE, ALTER, DROP. Umożliwia definiowanie schematu bazy danych – struktury tabel, widoków, indeksów oraz ograniczeń integralności.
   * Działanie: Tłumaczy te polecenia na wewnętrzne struktury danych, które są przechowywane w katalogu systemowym (słowniku danych) zawierającym metadane o bazie danych.
2. Moduł Języka Manipulacji Danymi (DML Processor)
   * Rola: Przetwarza polecenia DML (ang. *Data Manipulation Language*), takie jak SELECT, INSERT, UPDATE, DELETE. Umożliwia użytkownikom i aplikacjom odczytywanie, dodawanie, modyfikowanie oraz usuwanie danych.
   * Działanie: Interpretuje zapytania DML, współpracując z optymalizatorem zapytań i menedżerem transakcji w celu wykonania operacji na danych.
3. Optymalizator Zapytań (Query Optimizer)
   * Rola: Kluczowy moduł odpowiedzialny za analizę zapytań DML (szczególnie SELECT) i wybór najbardziej efektywnego planu ich wykonania.
   * Działanie: Analizuje dostępne indeksy, statystyki dotyczące danych, struktury tabel i ograniczenia, aby znaleźć optymalną ścieżkę dostępu do danych. Generuje plan wykonania zapytania, który jest następnie przekazywany do wykonawcy zapytań.
4. Menedżer Danych (Data Manager / Storage Manager)
   * Rola: Zarządza fizycznym przechowywaniem danych na dysku oraz ich buforowaniem w pamięci operacyjnej. Odpowiada za efektywne odczytywanie i zapisywanie bloków danych.
5. Menedżer Transakcji (Transaction Manager)
   * Rola: Zapewnia właściwości ACID (atomowość, spójność, izolacja, trwałość) dla transakcji. Koordynuje ich wykonanie, gwarantując spójność danych nawet w przypadku awarii systemu lub współbieżnych operacji.
6. Menedżer Współbieżności (Concurrency Control Manager)
   * Rola: Często zintegrowany z menedżerem transakcji. Odpowiada za zarządzanie jednoczesnym dostępem wielu użytkowników lub procesów do tych samych danych. Zapobiega niespójnościom wynikającym z równoczesnych modyfikacji danych.
7. Menedżer Odzyskiwania (Recovery Manager)
   * Rola: Zapewnia trwałość danych i ich integralność po awariach systemu (np. zaniku zasilania, błędach oprogramowania). Odtwarza stan bazy danych sprzed awarii na podstawie logów transakcji i kopii zapasowych.
8. Katalog Systemowy (System Catalog / Data Dictionary)
   * Rola: Choć nie jest aktywnym modułem funkcjonalnym, stanowi centralne repozytorium metadanych o bazie danych. Zawiera informacje o schemacie bazy, użytkownikach, uprawnieniach, indeksach, widokach, ograniczeniach, statystykach itp. Jest wykorzystywany przez inne moduły DBMS w trakcie przetwarzania zapytań i operacji.

**3. Zasada ACID – rola jaką spełnia i podstawowe sposoby jej realizacji.**

Zasada **ACID** to akronim opisujący cztery podstawowe właściwości, które muszą być spełnione przez transakcje w systemach zarządzania bazami danych (SZBD). Jej celem jest zapewnienie, że dane pozostają spójne i poprawne, nawet w sytuacjach, gdy wielu użytkowników jednocześnie modyfikuje dane lub gdy dochodzi do awarii systemu. Bez spełnienia tych warunków nie byłoby możliwe utrzymanie integralności danych w złożonych środowiskach informatycznych.

**A – Atomowość (Atomicity)**  
Gwarantuje, że transakcja jest niepodzielna – albo wszystkie operacje w jej ramach zostaną wykonane, albo żadna. W przypadku błędu cała transakcja zostaje wycofana (rollback) do stanu początkowego. Realizuje się to poprzez dziennik transakcji (Transaction Log), w którym DBMS zapisuje zmiany przed ich wprowadzeniem do bazy, oraz przez mechanizmy wycofywania, które cofają zmiany przy błędach lub awariach.

**C – Spójność (Consistency)**  
Oznacza, że każda transakcja musi przekształcać bazę danych z jednego spójnego stanu w inny, również spójny. System dba o to poprzez sprawdzanie ograniczeń integralności (np. kluczy głównych i obcych, NOT NULL, CHECK) przed i po wykonaniu transakcji. Dodatkowo, aplikacje mogą implementować reguły biznesowe, których naruszenie powinno skutkować wycofaniem transakcji.

**I – Izolacja (Isolation)**  
Zapewnia, że równolegle wykonywane transakcje nie wpływają negatywnie na siebie nawzajem. Transakcje powinny być wykonywane tak, jakby były realizowane jedna po drugiej. W praktyce osiąga się to poprzez stosowanie mechanizmów takich jak blokady (locki), wielowersyjność (MVCC – Multi-Version Concurrency Control) oraz odpowiednie poziomy izolacji zdefiniowane w standardzie SQL (np. Read Committed, Repeatable Read, Serializable).

**D – Trwałość (Durability)**  
Gwarantuje, że po zatwierdzeniu transakcji (COMMIT), wszystkie jej zmiany są trwale zapisane i nie zostaną utracone nawet w przypadku awarii systemu. Realizacja tej właściwości opiera się m.in. na technice zapisu do dziennika (Write-Ahead Logging – WAL), stosowaniu punktów kontrolnych (checkpointing), a także tworzeniu regularnych kopii zapasowych bazy danych.

**4. Definiowanie warunków integralności bazy danych w relacyjnych systemach baz danych.**

**Warunki integralności w relacyjnych bazach danych**

Warunki integralności bazy danych to zbiór reguł i ograniczeń, które zapewniają poprawność, spójność i niezawodność danych przechowywanych w relacyjnej bazie danych. Definiuje się je na poziomie schematu bazy, a ich egzekwowanie realizowane jest automatycznie przez system zarządzania bazą danych (DBMS). Ich głównym celem jest zapobieganie wprowadzaniu niepoprawnych danych oraz utrzymanie wysokiej jakości i spójności informacji.

W relacyjnym modelu danych wyróżnia się trzy główne kategorie integralności:

**1. Integralność encji (Entity Integrity)**   
Integralność encji zapewnia, że każda krotka (wiersz) w tabeli może być jednoznacznie zidentyfikowana. Podstawą tej zasady jest klucz główny, czyli atrybut (lub zestaw atrybutów), który musi być unikalny i nie może przyjmować wartości NULL. Klucz główny jest definiowany za pomocą deklaracji PRIMARY KEY.

**2. Integralność referencyjna (Referential Integrity)**  
Dotyczy spójności relacji pomiędzy tabelami. Gwarantuje, że wartości w kolumnach klucza obcego (foreign key) w jednej tabeli muszą odpowiadać wartościom klucza głównego w innej tabeli lub być puste (NULL). Klucz obcy jest definiowany za pomocą ograniczenia FOREIGN KEY, a jego zachowanie można dostosować poprzez akcje:

**3. Integralność domeny (Domain Integrity)**  
Gwarantuje, że dane w kolumnie są zgodne z jej zdefiniowaną domeną, czyli typem i zakresem dopuszczalnych wartości. Realizowana jest poprzez:

- określenie **typu danych** kolumny (np. INT, VARCHAR, DATE),  
- użycie ograniczenia **NOT NULL**, które wymusza podanie wartości,  
- zastosowanie **UNIQUE**, zapewniającego unikalność danych (dopuszcza jednak NULL),  
- użycie ograniczenia **CHECK**, które pozwala zdefiniować własne warunki logiczne (np. CHECK (wiek >= 0)).

**Inne mechanizmy wspierające integralność danych**

Oprócz wbudowanych ograniczeń, relacyjne systemy bazodanowe umożliwiają stosowanie bardziej zaawansowanych mechanizmów:

**Indeksy** – choć ich główną funkcją jest przyspieszanie zapytań, unikalne indeksy (tworzone np. z kluczem głównym lub ograniczeniem UNIQUE) bezpośrednio egzekwują unikalność danych.

**Wyzwalacze (triggery)** – umożliwiają automatyczne wykonanie określonego kodu SQL w odpowiedzi na zdarzenia (takie jak INSERT, UPDATE czy DELETE). Pozwalają na wdrożenie niestandardowych zasad integralności, trudnych do wyrażenia przy użyciu standardowych ograniczeń.

**Procedury składowane (stored procedures)** – zawierają logikę biznesową związaną z walidacją i manipulacją danymi. Ich zastosowanie umożliwia centralizację reguł walidacyjnych i większą kontrolę nad spójnością danych w aplikacjach korzystających z bazy.

**Podsumowując**, warunki integralności są fundamentem każdej solidnej bazy danych. Ich poprawne zdefiniowanie i konsekwentne egzekwowanie zapewniają nie tylko bezpieczeństwo i spójność danych, ale także zwiększają niezawodność całego systemu informatycznego.

**5. Proces normalizacji w relacyjnych bazach danych – cel i charakterystyka etapów procesu.**

Normalizacja to metodyczna technika projektowania relacyjnych baz danych, której celem jest eliminacja redundancji danych oraz poprawa ich spójności i integralności. Polega na dekompozycji dużych, często nadmiarowych tabel na mniejsze, logicznie powiązane struktury. Proces ten został zaproponowany przez Edgara F. Codda, twórcę relacyjnego modelu danych.

Cele normalizacji

**usunąć redundancję** (czyli powielanie danych),  
**zapobiegać anomaliom** (błędom podczas wstawiania, aktualizacji lub usuwania danych),  
**zwiększyć spójność** i czytelność struktury danych.

Formy normalne

Normalizacja dzieli się na kolejne etapy nazywane formami normalnymi (*Normal Forms* – NF). Każda wyższa forma obejmuje reguły poprzednich i wprowadza dodatkowe ograniczenia. W praktyce najczęściej stosuje się pierwsze trzy formy normalne.

**Pierwsza Forma Normalna (1NF)**

Tabela spełnia 1NF, jeśli każda kolumna zawiera wyłącznie wartości atomowe, czyli niepodzielne. Niedozwolone są listy wartości, zagnieżdżone struktury ani grupy powtarzające się. Każdy wiersz musi być unikalny.

| **StudentID** | **Imię** | **Kursy** |
| --- | --- | --- |
| 1 | Anna | Matematyka, Fizyka |
| 2 | Jan | Informatyka |

**Druga Forma Normalna (2NF)**

Warunkiem 2NF jest spełnienie 1NF   
oraz pełna zależność wszystkich kolumn niekluczowych od całego klucza głównego. W przypadku klucza złożonego (składającego się z kilku kolumn), żadna kolumna niekluczowa nie może zależeć tylko od jego części.

| **ZamówienieID (klucz1)** | **ProduktID (klucz2)** | **NazwaProduktu** | **Cena** | **Ilość** |
| --- | --- | --- | --- | --- |
| 1 | 101 | Myszka | 50 | 2 |
| 1 | 102 | Klawiatura | 100 | 1 |

**Trzecia Forma Normalna (3NF)**

Tabela jest w 3NF, jeśli spełnia 2NF i nie zawiera zależności przechodnich, czyli takich, w których jedna kolumna niekluczowa zależy od innej kolumny niekluczowej.

| **PracownikID** | **Imię** | **DziałID** | **NazwaDziału** | **Lokalizacja** |
| --- | --- | --- | --- | --- |
| 1 | Anna | 10 | IT | Warszawa |
| 2 | Jan | 20 | HR | Kraków |

Praktyczna równowaga: normalizacja vs. denormalizacja

W większości systemów biznesowych osiągnięcie 3NF lub BCNF jest wystarczające. Nadmierna normalizacja (np. do 4NF lub 5NF) może prowadzić do konieczności częstego łączenia tabel, co wpływa na złożoność zapytań i wydajność. W takich przypadkach stosuje się czasem denormalizację, czyli celowe przywrócenie pewnych nadmiarowości dla poprawy wydajności odczytu.

Przedmiot: Systemy Baz Danych 1. Dwu i trzy warstwowa architektura systemów baz danych - porównanie typowych cech. 2. Realizacja transakcji rozproszonej w rozproszonych systemach baz danych (nie w NoSQL). 3. Cel tworzenia i zasady realizacji (rola ETL) systemów typu Hurtownie Danych. 4. Model wielowymiarowej kostki danych w systemach typu OLAP. 5. Obiektowe bazy danych – cechy charakterystyczne i rola procesu Mapowania Obiektowo-Relacyjnego (ORM). 6. Rola i zasadnicze cechy realizacji systemów NoSQL.

++++++++++++++++++++++++++++++++++++++

Oto odpowiedzi na pytania egzaminacyjne dotyczące Podstaw Technologii Baz Danych:

**Przedmiot: Systemy Baz Danych //** mgr Grzegorz Stolecki

**1. Dwu- i trzy-warstwowa architektura systemów baz danych - porównanie typowych cech.**

Architektura systemu baz danych odnosi się do sposobu organizacji komponentów systemu, które współpracują w celu zarządzania danymi. Główne typy to architektura dwu- i trzy-warstwowa, każda z nich ma swoje typowe cechy, zalety i wady.

**Architektura Dwu-Warstwowa (Klient-Serwer)**

W architekturze dwu-warstwowej system jest podzielony na dwie główne warstwy:

1. **Warstwa Klienta (Prezentacji i Logiki Biznesowej):** Odpowiada za interfejs użytkownika oraz zawiera część logiki biznesowej i logiki dostępu do danych. Jest to aplikacja, która działa bezpośrednio na komputerze użytkownika.
2. **Warstwa Serwera (Bazy Danych):** Odpowiada za przechowywanie i zarządzanie danymi. Jest to system zarządzania bazą danych (SZBD).

**Typowe Cechy Architektury Dwu-Warstwowej:**

* **Prostota:** Łatwiejsza w implementacji i zarządzaniu dla mniejszych aplikacji.
* **Bezpośrednie połączenie:** Klient łączy się bezpośrednio z serwerem bazy danych.
* **Rozproszona logika biznesowa:** Logika biznesowa jest często rozproszona między klientem a serwerem bazy danych (np. za pomocą procedur składowanych na serwerze i logiki w aplikacji klienckiej).
* **Skalowalność:** Ograniczona skalowalność. Wzrost liczby klientów może szybko przeciążyć serwer bazy danych, ponieważ każdy klient utrzymuje bezpośrednie połączenie i wymaga zasobów serwera.
* **Zależność od klienta:** Zmiany w logice biznesowej często wymagają aktualizacji aplikacji klienckich.
* **Bezpieczeństwo:** Potencjalne luki bezpieczeństwa, ponieważ klient ma często bezpośredni dostęp do bazy danych. Loginy i hasła do bazy danych mogą być przechowywane na kliencie.
* **Złożoność zarządzania zmianą:** Wdrażanie nowych wersji aplikacji klienckich na wielu stacjach roboczych może być kosztowne i czasochłonne.
* **Przeznaczenie:** Często stosowana w mniejszych systemach, aplikacjach desktopowych, intranetowych o ograniczonej liczbie użytkowników.

**Architektura Trzy-Warstwowa (lub N-Warstwowa)**

W architekturze trzy-warstwowej (lub N-warstwowej, gdzie warstw może być więcej niż trzy) system jest podzielony na co najmniej trzy odrębne warstwy:

1. **Warstwa Prezentacji (Klient):** Odpowiada wyłącznie za interfejs użytkownika. Nie zawiera logiki biznesowej ani logiki dostępu do danych. Może to być przeglądarka internetowa, aplikacja mobilna, cienki klient desktopowy.
2. **Warstwa Logiki Biznesowej (Serwer Aplikacji):** Centralna warstwa zawierająca całą logikę biznesową aplikacji. Jest to serwer, który pośredniczy między warstwą prezentacji a warstwą danych. Obsługuje żądania od klientów, przetwarza je zgodnie z regułami biznesowymi i przekazuje do warstwy danych.
3. **Warstwa Danych (Serwer Bazy Danych):** Odpowiada za przechowywanie i zarządzanie danymi, tak jak w architekturze dwu-warstwowej.

**Typowe Cechy Architektury Trzy-Warstwowej:**

* **Modułowość i elastyczność:** Wyraźne rozdzielenie ról i odpowiedzialności poszczególnych warstw.
* **Skalowalność:** Znacznie lepsza skalowalność. Warstwa logiki biznesowej może być skalowana niezależnie od warstwy prezentacji czy danych (np. poprzez dodawanie kolejnych serwerów aplikacji).
* **Centralizacja logiki biznesowej:** Cała logika biznesowa znajduje się na serwerze aplikacji, co ułatwia zarządzanie, modyfikację i wdrażanie zmian.
* **Bezpieczeństwo:** Zwiększone bezpieczeństwo, ponieważ klient nie łączy się bezpośrednio z bazą danych. Serwer aplikacji może pełnić rolę bramy bezpieczeństwa i maskować dane uwierzytelniające do bazy.
* **Łatwiejsze zarządzanie zmianą:** Zmiany w logice biznesowej wymagają aktualizacji tylko serwera aplikacji, bez konieczności wdrażania na stacjach klienckich (szczególnie w przypadku klientów webowych).
* **Złożoność:** Większa złożoność implementacji i konfiguracji początkowej.
* **Redundancja:** Może wymagać większej infrastruktury (dodatkowe serwery).
* **Przeznaczenie:** Idealna dla dużych, rozproszonych systemów o wielu użytkownikach, aplikacji internetowych, mobilnych, korporacyjnych.

Podsumowując, wybór architektury zależy od wymagań projektu, jego skali, budżetu i złożoności. Architektura trzy-warstwowa stała się standardem dla większości współczesnych, dużych aplikacji ze względu na jej skalowalność, elastyczność i bezpieczeństwo.

**2. Realizacja transakcji rozproszonej w rozproszonych systemach baz danych (nie w NoSQL).**

W klasycznych rozproszonych systemach baz danych (RDBMS) transakcje rozproszone polegają na tym, że pojedyncza transakcja obejmuje wiele węzłów (baz danych) znajdujących się na różnych serwerach, często w różnych lokalizacjach. Celem jest zagwarantowanie spójności i integralności danych, mimo że operacje są wykonywane na niezależnych fragmentach danych.

🔐 Wymagania dla transakcji rozproszonych

Transakcja rozproszona musi spełniać zasady ACID:

🧠 Kluczowy mechanizm: protokół dwufazowego zatwierdzania (2PC – Two-Phase Commit)

➤ Faza 1: Przygotowanie (prepare)

1. Koordynator transakcji (Transaction Coordinator) wysyła do wszystkich zaangażowanych węzłów żądanie prepare to commit.
2. Każdy węzeł (uczestnik) sprawdza, czy może zatwierdzić operacje (np. czy nie ma błędów, konfliktów).
3. Każdy węzeł odpowiada:
   * TAK (OK) – gotowy do zatwierdzenia,
   * NIE (FAIL) – nie może zatwierdzić.

➤ Faza 2: Zatwierdzenie (commit) lub wycofanie (rollback)

1. Jeśli wszyscy uczestnicy odpowiedzieli „TAK”, koordynator wysyła COMMIT do wszystkich węzłów.
2. Jeśli choć jeden uczestnik odpowiedział „NIE” – wysyłane jest ROLLBACK, aby wszyscy wycofali operację.
3. Każdy węzeł wykonuje polecenie (commit lub rollback) i informuje koordynatora.

**3. Cel tworzenia i zasady realizacji (rola ETL** Extract, Transform, Load**) systemów typu Hurtownie Danych.**

Hurtownia danych (Data Warehouse, DW) to scentralizowane repozytorium informacji pochodzących z różnych, często rozproszonych i heterogenicznych źródeł danych. Została zaprojektowana z myślą o wspieraniu analiz biznesowych (Business Intelligence, BI), raportowania i podejmowania decyzji. W przeciwieństwie do systemów transakcyjnych (OLTP), hurtownie danych koncentrują się na analizie danych historycznych i zagregowanych.

Główne cele tworzenia hurtowni danych:

1. Wsparcie procesów decyzyjnych – dostarczanie spójnego, zintegrowanego i wiarygodnego obrazu danych, który ułatwia menedżerom i analitykom podejmowanie trafnych decyzji.
2. Integracja danych z różnych źródeł – łączenie danych z systemów takich jak ERP, CRM, HR czy systemy sprzedażowe, które często różnią się formatem, strukturą i technologią.
3. Standaryzacja i oczyszczanie danych – usuwanie duplikatów, błędów oraz niespójności, co przekłada się na wyższą jakość danych dostępnych do analizy.
4. Gromadzenie danych historycznych – możliwość przechowywania danych przez wiele lat w celu analizy trendów, porównań międzyokresowych i prognozowania. Typowe systemy OLTP zwykle przechowują tylko dane bieżące.
5. Poprawa wydajności zapytań analitycznych – hurtownie danych są zoptymalizowane pod kątem analiz (OLAP), dzięki czemu zapytania są szybsze i bardziej efektywne w porównaniu z systemami transakcyjnymi.
6. Ujednolicony widok danych (single source of truth) – zapewnienie jednego wspólnego źródła danych dla całej organizacji, co eliminuje rozbieżności między raportami pochodzącymi z różnych systemów.

⚙️ Zasady realizacji hurtowni danych – rola procesu ETL

Centralnym elementem budowy i działania hurtowni danych jest proces ETL (Extract, Transform, Load), czyli Ekstrakcja, Transformacja i Załadunek danych. To cykliczny i automatyzowany proces, który przenosi dane ze źródeł operacyjnych do hurtowni danych.

Rola procesu ETL:

* Extract (Ekstrakcja) – polega na pobieraniu danych z różnych źródeł, takich jak relacyjne bazy danych, pliki płaskie (CSV), systemy ERP, CRM czy arkusze kalkulacyjne.
* Transform (Transformacja) – przekształcanie danych: oczyszczanie, formatowanie, łączenie, standaryzacja i dostosowanie ich do struktury hurtowni danych. Jest to najważniejszy i najbardziej złożony etap.
* Load (Załadunek) – wczytanie przetworzonych danych do hurtowni danych, gdzie będą one dostępne do analiz i raportów.

🧱 Inne kluczowe zasady realizacji hurtowni danych

* Orientacja tematyczna – dane w hurtowni są pogrupowane według obszarów biznesowych (np. sprzedaż, klienci, produkty), a nie według systemów źródłowych.
* Integracja – dane pochodzące z różnych źródeł są ujednolicone i spójne.
* Zmienność w czasie (time-variant) – dane w hurtowni posiadają znacznik czasu (np. data wprowadzenia), co umożliwia analizę zmian i trendów historycznych.
* Nielotność (non-volatile) – po załadowaniu dane nie są modyfikowane ani usuwane. Każde nowe dane są zapisywane jako nowe rekordy, co pozwala zachować historię zmian.

**4. Model wielowymiarowej kostki danych w systemach typu OLAP.**

**Cel i kontekst: Systemy OLAP**

**Systemy OLAP (Online Analytical Processing)** są zaprojektowane do szybkiej analizy dużych wolumenów danych, wspierając złożone zapytania analityczne i raportowanie, a także eksplorację danych. Są one kluczowym komponentem systemów Business Intelligence i zazwyczaj opierają się na **hurtowniach danych** jako swoim źródle danych.

W przeciwieństwie do systemów OLTP (Online Transaction Processing), które są zoptymalizowane pod kątem szybkich operacji transakcyjnych (INSERT, UPDATE, DELETE) na danych bieżących, systemy OLAP są zoptymalizowane pod kątem szybkich, złożonych zapytań typu SELECT na danych historycznych i zagregowanych.

**Model Wielowymiarowej Kostki Danych (Multidimensional Data Cube)**

**Model wielowymiarowej kostki danych** jest podstawową koncepcją organizacji danych w systemach OLAP. Dane są postrzegane nie jako tabele (jak w modelu relacyjnym), lecz jako kostka (lub hiperkostka) złożona z **faktów** i **wymiarów**.

* **Fakty (Measures):** Reprezentują ilościowe, numeryczne dane, które są przedmiotem analizy. Są to wartości, które możemy agregować, takie jak:
  + Sprzedaż (ilość sprzedanych produktów, wartość sprzedaży), Zysk, Liczba transakcji, Koszty
  + Czas spędzony na stronie Fakty są przechowywane w tabeli faktów, która jest centralnym punktem kostki.
* **Wymiary (Dimensions):** Reprezentują perspektywy, z których fakty są analizowane. Są to cechy opisujące fakty. Każdy wymiar ma swoją hierarchię atrybutów. Przykładowe wymiary:
  + **Czas:** Rok, Kwartał, Miesiąc, Dzień
  + **Produkt:** Kategoria, Podkategoria, Nazwa produktu
  + **Lokalizacja:** Kraj, Region, Miasto, Sklep
  + **Klient:** Grupa klientów, Płeć, Wiek
  + **Sprzedawca:** Nazwisko, Dział

**Struktura Kostki Danych:**

Wyobraźmy sobie kostkę danych jako sześcian. Każda oś kostki reprezentuje jeden wymiar, a komórki w kostce zawierają wartości faktów dla danej kombinacji wymiarów. W rzeczywistości, dla więcej niż trzech wymiarów, mówimy o "hiperkostce", która jest abstrakcyjną reprezentacją.

* **Tabela Faktów:** Centralna tabela w hurtowni danych, zawierająca klucze obce do wszystkich tabel wymiarów oraz kolumny z faktami. Jest to często bardzo duża tabela.
* **Tabele Wymiarów:** Mniejsze tabele, które zawierają atrybuty opisujące poszczególne wymiary. Każda tabela wymiaru ma klucz główny, który jest kluczem obcym w tabeli faktów.

**Przykład: Kostka Sprzedaży**

* **Fakt:** Wartość Sprzedaży
* **Wymiary:**
  + Czas (Rok, Miesiąc, Dzień)
  + Produkt (Kategoria, Nazwa produktu)
  + Lokalizacja (Kraj, Miasto)

Możemy analizować "Wartość Sprzedaży" dla "Konkretnego Produktu" w "Danym Miesiącu" w "Określonym Mieście".

**Operacje OLAP na Kostce Danych:**

Systemy OLAP umożliwiają szybkie wykonywanie różnorodnych operacji analitycznych na kostce danych, m.in.:

1. **Slice (Krojenie):** Wybieranie podzbioru danych poprzez ustalenie wartości dla jednego lub więcej wymiarów.
   * *Przykład:* Wyświetl sprzedaż dla "produktów elektronicznych".
2. **Dice (Kostkowanie):** Wybieranie podzbioru danych poprzez określenie zakresów wartości dla co najmniej dwóch wymiarów.
   * *Przykład:* Wyświetl sprzedaż dla "produktów elektronicznych" w "Europejskich miastach" w "Q1 2024".
3. **Drill Down (Zagłębianie):** Przechodzenie od bardziej ogólnych danych do bardziej szczegółowych, schodząc w dół hierarchii wymiaru.
   * *Przykład:* Od sprzedaży rocznej do sprzedaży kwartalnej, następnie miesięcznej, a na końcu dziennej.
4. **Roll Up (Zwijanie):** Przechodzenie od bardziej szczegółowych danych do bardziej ogólnych, wznosząc się w górę hierarchii wymiaru. Jest to odwrotność Drill Down.
   * *Przykład:* Od sprzedaży dziennej do miesięcznej, a następnie kwartalnej i rocznej.
5. **Pivot (Obracanie/Przestawianie):** Zmiana perspektywy widoku danych poprzez obracanie wymiarów wzdłuż różnych osi.
   * *Przykład:* Zmiana widoku z produktów w wierszach i czasu w kolumnach na czas w wierszach i produkty w kolumnach.

**Rodzaje implementacji OLAP:**

* **MOLAP (Multidimensional OLAP):** Dane są przechowywane bezpośrednio w wielowymiarowej bazie danych, zoptymalizowanej pod kątem zapytań OLAP. Wartości faktów są często wstępnie agregowane. (Microsoft Analysis Services (SSAS), Oracle OLAP, IBM Cognos TM1)
* **ROLAP (Relational OLAP):** Dane są przechowywane w relacyjnej bazie danych (model gwiazdy lub płatka śniegu), a operacje OLAP są wykonywane poprzez złożone zapytania SQL.

Model wielowymiarowej kostki danych jest niezwykle efektywny w analizie danych biznesowych, umożliwiając szybkie i intuicyjne przeglądanie danych z wielu perspektyw, co jest kluczowe dla podejmowania strategicznych decyzji.

**5. Obiektowe bazy danych – cechy charakterystyczne i rola procesu Mapowania Obiektowo-Relacyjnego (ORM).**

**Obiektowe Bazy Danych (OODBMS - Object-Oriented Database Management Systems)**

**Obiektowe bazy danych** to rodzaj systemów baz danych, które przechowują dane jako **obiekty**, w sposób bezpośrednio zgodny z paradygmatem programowania obiektowego. Zostały zaprojektowane jako alternatywa dla relacyjnych baz danych, aby lepiej radzić sobie ze złożonymi strukturami danych i relacjami, które występują w programowaniu obiektowym.

**Cechy Charakterystyczne Obiektowych Baz Danych:**

1. **Obsługa obiektów:**
   * **Tożsamość Obiektu (Object Identity):** Każdy obiekt ma unikalny, niezmienny identyfikator (OID), niezależny od jego wartości. Tożsamość obiektu jest zachowywana w całej bazie danych.
   * **Hermetyzacja:** Obiekty zawierają zarówno dane (atrybuty), jak i metody (operacje) działające na tych danych.
   * **Złożone Obiekty:** OODBMS bezpośrednio obsługują złożone obiekty, które mogą zawierać inne obiekty (wartościowe lub poprzez referencje). Nie ma potrzeby dekompozycji złożonych obiektów na wiele tabel, jak w bazach relacyjnych.
   * **Dziedziczenie (Inheritance):** Obsługa mechanizmu dziedziczenia klas, gdzie klasy pochodne dziedziczą atrybuty i metody po klasach bazowych. Baza danych może przechowywać hierarchie klas.
   * **Polimorfizm:** Możliwość definiowania metod o tej samej nazwie, ale z różnym zachowaniem w zależności od typu obiektu.
2. **Bezpośrednie odwzorowanie:** Brak "rozbieżności impedancji" (impedance mismatch) między modelem obiektowym języka programowania a modelem danych bazy danych. Obiekty z aplikacji mogą być bezpośrednio przechowywane w bazie danych bez skomplikowanego mapowania.
3. **Nawigacyjny dostęp do danych:** Dane są często dostępne poprzez nawigację po związkach między obiektami (poprzez referencje), co może być bardziej naturalne i wydajne dla niektórych typów zapytań niż operacje łączenia tabel (JOIN) w bazach relacyjnych.
4. **Wersjonowanie obiektów:** Często oferują wbudowaną obsługę wersjonowania obiektów, co jest przydatne w systemach CAD, CAM, zarządzania dokumentami.
5. **Dedykowane języki zapytań:** Niektóre OODBMS mają własne języki zapytań, które są obiektowo zorientowane (np. OQL - Object Query Language).

**Zalety OODBMS:**

* Lepsza obsługa złożonych, powiązanych danych (np. grafy, struktury drzewiaste).
* Eliminacja rozbieżności impedancji, co upraszcza rozwój aplikacji.
* Potencjalnie lepsza wydajność dla nawigacyjnych zapytań.

**Wady OODBMS:**

* Brak standaryzacji (w przeciwieństwie do SQL).
* Mniejsza dojrzałość rynkowa i ekosystem niż RDBMS.
* Trudności w obsłudze ad-hoc zapytań i raportowania.
* Trudniejsza integracja z istniejącymi narzędziami BI.

Obiektowe bazy danych nie zdobyły tak szerokiej popularności jak relacyjne, ale znalazły niszę w specyficznych zastosowaniach, takich jak projektowanie wspomagane komputerowo (CAD), systemy informacji geograficznej (GIS), telekomunikacja czy biologia molekularna.

**Rola procesu Mapowania Obiektowo-Relacyjnego (ORM - Object-Relational Mapping)**

Ponieważ relacyjne bazy danych (RDBMS) stały się dominującym modelem, a programowanie obiektowe jest standardem w tworzeniu aplikacji, pojawił się problem **rozbieżności impedancji** (impedance mismatch) między dwoma paradygmatami:

* **Relacyjne bazy danych:** Dane przechowywane w płaskich tabelach, operacje na zbiorach (JOIN, SELECT), brak dziedziczenia, brak bezpośrednich referencji między wierszami.
* **Programowanie obiektowe:** Dane jako obiekty ze stanem i zachowaniem, złożone obiekty, hierarchie dziedziczenia, referencje między obiektami, grafy obiektów.

**Mapowanie Obiektowo-Relacyjne (ORM)** to technika i narzędzia programistyczne (tzw. frameworiki ORM, np. Hibernate dla Javy, Entity Framework dla .NET, SQLAlchemy dla Pythona), które umożliwiają \*\*odwzorowanie obiektów z języka programowania na struktury relacyjne b

**6. Rola i zasadnicze cechy realizacji systemów NoSQL**

Systemy NoSQL powstały, aby radzić sobie z wyzwaniami, których tradycyjne relacyjne bazy danych (SQL) nie obsługują dobrze, szczególnie w kontekście:

- wielkich ilości danych (Big Data),  
- danych nieustrukturyzowanych lub półstrukturyzowanych (np. JSON, dokumenty, grafy),  
- wysokiej skalowalności poziomej (rozproszenie danych na wiele serwerów),  
- szybkiego dostępu w czasie rzeczywistym.

🌟 Zasadnicze cechy systemów NoSQL

1. Brak sztywnego schematu

Nie musisz z góry definiować kolumn jak w SQL.  
Można przechowywać różne struktury danych w tej samej kolekcji (np. MongoDB).

2. Wysoka skalowalność pozioma

Łatwo dodajesz nowe serwery (shardy), by obsługiwać więcej danych.

3. Elastyczność danych

Obsługa różnych formatów: dokumenty (JSON), klucz-wartość, kolumny, grafy.

4. Wydajność i szybkość

Dostosowane do zapytań o konkretny element, a nie do złożonych relacji.  
Idealne do zapytań typu „daj mi wszystkie zamówienia klienta X”.

5. Model CAP (Consistency, Availability, Partition tolerance)

NoSQL często wybiera dostępność i odporność na rozdzielenie kosztem spójności (np. w MongoDB).

6. Brak standardowego języka zapytań (jak SQL)

* Każdy system ma własny interfejs (np. zapytania MongoDB to operacje JSON-owe).

📊 Typy baz NoSQL:

| Typ | Opis | Przykłady |
| --- | --- | --- |
| Dokumentowe | Dane jako dokumenty JSON/BSON | MongoDB, CouchDB |
| Klucz-wartość | Proste pary klucz → wartość | Redis, DynamoDB |
| Kolumnowe | Dane przechowywane w kolumnach | Cassandra, HBase |
| Grafowe | Dane i relacje jako grafy | Neo4j, ArangoDB |

🧠 Podsumowanie:

Systemy NoSQL:

są nowoczesną alternatywą dla SQL,  
wspierają Big Data, IoT, analitykę, web i mobilne aplikacje,  
oferują elastyczność i wysoką wydajność, kosztem czasem spójności lub złożoności relacji.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Oto odpowiedzi na pytania egzaminacyjne dotyczące Sztucznej Inteligencji i Systemów Ekspertowych:

**1. Probabilistyczne bazy wiedzy – metoda i podstawowe cechy tej reprezentacji**

**Probabilistyczne bazy wiedzy** to metoda reprezentacji wiedzy w systemach sztucznej inteligencji, która radzi sobie z niepewnością i niekompletnością danych, wykorzystując narzędzia teorii prawdopodobieństwa. W przeciwieństwie do logiki klasycznej, która operuje na wartościach prawda/fałsz, bazy probabilistyczne przypisują stopnie pewności (prawdopodobieństwa) zdarzeniom lub twierdzeniom.

To rodzaj systemu, który łączy fakty i reguły logiczne z prawdopodobieństwem. Zamiast mówić „to jest prawda”, mówi:  
👉 „to jest prawdopodobne z szansą 80%”.

🧠 Metoda działania

Zamiast klasycznego „prawda/fałsz”, baza używa rachunku prawdopodobieństwa, by:  
modelować niepewność,  
uczyć się z danych (np. poprzez statystyki, AI),  
wnioskować, gdy dane są niepełne, sprzeczne lub niepewne.

🌟 Podstawowe cechy probabilistycznej reprezentacji wiedzy

1. Modelowanie niepewności  
Każdy fakt, reguła lub zależność ma przypisaną wartość prawdopodobieństwa (np. „jeśli kaszel, to grypa z prawdopodobieństwem 70%”).

2. Wnioskowanie probabilistyczne  
System nie tylko odpowiada „tak” lub „nie”, ale też: „na 85% to prawda”.

3. Uczenie się z danych  
Systemy te mogą aktualizować prawdopodobieństwa na podstawie obserwacji (np. Bayes, modele Markowa, sieci Bayesowskie).

4. Elastyczność w analizie niepełnych danych  
Potrafią odpowiadać, nawet jeśli nie znają wszystkich faktów (np. przewidywanie choroby mimo braku wyników wszystkich badań).

5. Wysoka złożoność obliczeniowa  
Wnioskowanie może być kosztowne obliczeniowo, szczególnie w dużych bazach.

🔍 Przykłady zastosowań:

Diagnostyka medyczna, Systemy rekomendacyjne (np. filmy, zakupy), Rozpoznawanie mowy i języka naturalnego, Bezpieczeństwo IT i analiza ryzyka.

✅ Podsumowanie:

Probabilistyczne bazy wiedzy to narzędzia do inteligentnego wnioskowania przy niepewności. Łączą logikę z statystyką, co pozwala lepiej odwzorować rzeczywisty świat, gdzie nie wszystko jest czarno-białe.

**2. Regułowe bazy wiedzy – budowa reguł i zasady wnioskowania regułowego (CLIPS, Prolog)**

**Regułowe bazy wiedzy** to jedna z najpopularniejszych i najbardziej intuicyjnych metod reprezentacji wiedzy w systemach ekspertowych i systemach opartych na wiedzy. Wiedza jest w nich kodowana w postaci zbioru reguł, które odzwierciedlają heurystyki i zależności znane ekspertom dziedzinowym.

**Budowa Reguł:**

Podstawową jednostką wiedzy w systemach regułowych jest **reguła produkcyjna**, która ma postać: **JEŻELI (warunek) WÓWCZAS (wniosek)** lub bardziej ogólnie: **JEŻELI (przesłanka P1 I przesłanka P2 I ... I przesłanka Pn) WÓWCZAS (konkluzja K1 I konkluzja K2 I ... I konkluzja Km)**

Regułowe bazy wiedzy – budowa reguł i zasady wnioskowania  
Regułowe bazy wiedzy to systemy, które przechowują wiedzę w postaci reguł typu "jeśli – to", czyli IF–THEN.

Każda reguła składa się z:

przesłanki – czyli warunku, który musi być spełniony,  
konkluzji – czyli wniosku, który jest wyciągany, gdy warunek jest prawdziwy.

Na przykład: „Jeśli pacjent ma gorączkę i kaszel, to może mieć grypę.”

System analizuje fakty i stosuje reguły, aby wyciągać nowe wnioski.

Wyróżniamy dwa główne sposoby wnioskowania:

Wnioskowanie w przód – zaczynamy od znanych faktów i sprawdzamy, jakie wnioski można wyciągnąć.  
Wnioskowanie wstecz – zaczynamy od celu (np. diagnozy) i sprawdzamy, czy znane fakty ją potwierdzają.

Takie systemy są stosowane np. w medycynie, diagnostyce technicznej czy systemach doradczych.

* + *Przykład:* Cel: Pacjent\_ma\_grypę? Szukam reguły: JEŻELI temperatura\_jest\_wysoka AND pacjent\_ma\_kaszel WÓWCZAS pacjent\_ma\_grype. Teraz pytam o podcele: temperatura\_jest\_wysoka?, pacjent\_ma\_kaszel?.

W wielu zaawansowanych systemach ekspertowych stosuje się kombinację obu strategii wnioskowania.

**3. Zasadnicze różnice pomiędzy dziedzinowym systemem ekspertowym a Shell-em systemem ekspertowego**

Aby zrozumieć różnice, najpierw zdefiniujmy oba pojęcia w kontekście systemów ekspertowych.

**System Ekspertowy (Dziedzinowy System Ekspertowy / Pełny System Ekspertowy)**

**System ekspertowy** to program komputerowy, który emuluje proces podejmowania decyzji i rozumowania człowieka-eksperta w pewnej, specyficznej dziedzinie wiedzy (np. diagnoza medyczna, konfiguracja komputerów, prognozowanie pogody). Składa się z wielu komponentów, w tym silnika wnioskującego, bazy wiedzy, modułu pozyskiwania wiedzy, modułu wyjaśnień i interfejsu użytkownika.

**Shell Systemu Ekspertowego (Expert System Shell)**

**Shell systemu ekspertowego** to środowisko programistyczne lub "szkielet" systemu eksperckiego, który zawiera gotowy **silnik wnioskujący** i inne mechanizmy (np. do zarządzania bazą wiedzy, moduł wyjaśnień, podstawowy interfejs), ale **nie zawiera żadnej wiedzy dziedzinowej**. Jest to pusta struktura, którą należy "zapełnić" wiedzą, aby stworzyć konkretny system ekspertowy.

**Zasadnicze Różnice:**

|  |  |  |
| --- | --- | --- |
| **Cecha** | **Dziedzinowy System Ekspertowy (Pełny SE)** | **Shell Systemu Ekspertowego** |
| **Zawartość Wiedzy** | **Zawiera wiedzę dziedzinową** (fakty i reguły) dla konkretnego zastosowania. | **Nie zawiera żadnej wiedzy dziedzinowej.** Jest "pusty". |
| **Cel** | Rozwiązywanie konkretnego problemu w określonej dziedzinie (np. diagnozowanie chorób, konfigurowanie sieci). | Stanowi **platformę/narzędzie** do budowy różnych systemów ekspertowych. |
| **Użyteczność** | Gotowy do użycia przez użytkownika końcowego do rozwiązywania problemów. | Używany przez inżynierów wiedzy do tworzenia systemów ekspertowych. |
| **Kompletność** | Kompletny system, zawierający wszystkie niezbędne komponenty, w tym bazę wiedzy. | Niekompletny; brakuje mu najważniejszego elementu: wiedzy dziedzinowej. |
| **Faza Rozwoju** | Wynik końcowej fazy rozwoju (po zapełnieniu Shell-a wiedzą). | Punkt wyjścia do tworzenia systemów ekspertowych. |
| **Złożoność Koncepcji** | Jest to aplikacja rozwiązywania problemów. | Jest to narzędzie do tworzenia aplikacji. |
| **Przykłady** | MYCIN (diagnoza chorób krwi), XCON (konfiguracja komputerów VAX), PROSPECTOR (geologia). | CLIPS, OPS5, Exsys Corvid (przykłady komercyjnych/akademickich shelli). |

**4. Zasada działania i przykładowe zastosowania algorytmów ewolucyjnych**

**Algorytmy ewolucyjne (EA - Evolutionary Algorithms)** to klasa algorytmów optymalizacyjnych i heurystycznych inspirowanych mechanizmami ewolucji biologicznej (dobór naturalny, dziedziczenie, mutacja, rekombinacja). Są one szczególnie przydatne do rozwiązywania złożonych problemów, dla których brakuje efektywnych algorytmów deterministycznych, lub gdy przestrzeń rozwiązań jest zbyt duża do przeszukania w sposób wyczerpujący.

**Główne etapy działania algorytmu ewolucyjnego (iteracyjnego):**

1. **Inicjalizacja Populacji:**
   * Algorytm rozpoczyna się od stworzenia początkowej **populacji** potencjalnych rozwiązań problemu. Każde rozwiązanie nazywane jest **osobnikiem** (lub chromosomem) i jest reprezentowane jako struktura danych (np. ciąg bitów, wektor liczb rzeczywistych, drzewo).
   * Populacja jest zazwyczaj generowana losowo.
2. **Ocena Funkcji Przystosowania (Fitness Function Evaluation):**
   * Dla każdego osobnika w populacji obliczana jest jego **wartość przystosowania (fitness)**. Funkcja przystosowania mierzy, jak "dobrze" dany osobnik (rozwiązanie) radzi sobie z problemem. Im wyższa wartość przystosowania, tym lepsze rozwiązanie.
   * To jest kluczowy element, ponieważ to ona kieruje ewolucją.
3. **Selekcja (Selection):**
   * Na podstawie wartości przystosowania, wybierana jest grupa osobników, które mają zostać **rodzicami** dla następnego pokolenia. Osobnik z wyższym przystosowaniem ma większą szansę na bycie wybranym.
   * Popularne metody selekcji to: ruletka, turniejowa, rankingowa.
4. **Operatory Genetyczne (Genetic Operators) – Tworzenie Nowej Populacji:**
   * Wybrani rodzice tworzą nowe **potomstwo** (nowe rozwiązania) za pomocą operatorów genetycznych:
     + **Rekombinacja / Krzyżowanie (Crossover):** Dwa osobniki-rodzice wymieniają się częściami swoich "genotypów", tworząc nowe osobniki-potomstwo, które dziedziczą cechy od obu rodziców.
     + **Mutacja (Mutation):** Losowe, niewielkie zmiany są wprowadzane w genotypie jednego lub więcej osobników. Mutacja wprowadza różnorodność do populacji i pomaga algorytmowi uniknąć utknięcia w lokalnych ekstremach.
   * Często stosuje się również operator **reprodukcji** (bez zmian), który polega na bezpośrednim kopiowaniu najlepiej przystosowanych osobników do następnego pokolenia (tzw. elitaryzm).
5. **Zastępowanie / Zakończenie:**
   * Nowo utworzona populacja zastępuje starą populację.
   * Proces powtarza się od kroku 2 przez określoną liczbę **generacji** lub do momentu spełnienia kryterium zatrzymania (np. osiągnięcie zadowalającej wartości przystosowania, brak poprawy przez wiele generacji, upływ czasu).

**Przykładowe Zastosowania Algorytmów Ewolucyjnych:**

Algorytmy ewolucyjne są uniwersalne i znajdują zastosowanie w wielu dziedzinach, szczególnie tam, gdzie problemy są złożone, a przestrzeń rozwiązań ogromna:

* **Optymalizacja:**

**Problemy optymalizacji kombinatorycznej:** Problem komiwojażera (TSP), problem plecaka, **Projektowanie inżynierskie:** Optymalizacja kształtów samolotów, samochodów, anten, obwodów elektronicznych.  
**Finanse:** Optymalizacja portfela inwestycyjnego, strategie handlowe.

* **Sztuczna Inteligencja:**

**Uczenie maszynowe:** Optymalizacja wag sieci neuronowych, dobór cech, ewolucja struktur sieci.  
**Gry:** Rozwój strategii gry, np. w szachy, gry planszowe.  
**Programowanie ewolucyjne (Genetic Programming):** Automatyczne generowanie kodu programów w celu rozwiązania problemów.

* **Projektowanie artystyczne i muzyczne:** Generowanie innowacyjnych wzorów, tekstur, melodii.

Dzięki swojej naturze przeszukiwania globalnego i zdolności do unikania lokalnych ekstremów, algorytmy ewolucyjne są potężnym narzędziem w optymalizacji i uczeniu maszynowym.

**5. Zasada działania podstawowych operatorów algorytmu ewolucyjnego (reprodukcja, rekombinacja, mutacja)**

Podstawowe operatory algorytmu ewolucyjnego są inspirowane mechanizmami dziedziczenia i zmienności genetycznej w biologii. Są one kluczowe dla tworzenia nowych, potencjalnie lepszych rozwiązań w kolejnych pokoleniach.

**1. Reprodukcja (Selection) - w kontekście algorytmów ewolucyjnych**

**Reprodukcja** (lub selekcja reprodukcyjna) odnosi się do etapu **wyboru osobników z bieżącej populacji, które będą miały szansę na przekazanie swoich genów do następnego pokolenia.** Osobnik, który zostanie wybrany, staje się "rodzicem".

* **Zasada działania:**
  + Osobniki z wyższą wartością **funkcji przystosowania (fitness)** mają większe prawdopodobieństwo bycia wybranymi. To mechanizm odpowiadający "doborowi naturalnemu" – najlepiej przystosowane jednostki mają większe szanse na przeżycie i rozmnażanie.
  + Istnieje wiele metod selekcji, np.:

**Selekcja ruletkowa:** Szanse na wybór są proporcjonalne do przystosowania osobnika. Wyobraź sobie ruletkę, gdzie rozmiar pola odpowiada przystosowaniu.  
**Selekcja turniejowa:** Losowo wybiera się małą grupę osobników (turniej), a następnie wybiera się najlepszego z tej grupy do roli rodzica.  
**Elitaryzm:** Bezpośrednie kopiowanie kilku najlepiej przystosowanych osobników z obecnej populacji do następnego pokolenia, bez modyfikacji. Zapewnia to, że najlepsze rozwiązania nie zostaną utracone przez mutację czy rekombinację.

* **Rola:** Zapewnia, że dobre rozwiązania mają większy wpływ na przyszłe pokolenia, kierując algorytm w stronę optymalnego rozwiązania.

**2. Rekombinacja / Krzyżowanie (Crossover)**

* Operator ten polega na **wymianie fragmentów informacji genetycznej (cech)** między dwoma (lub więcej) wybranymi osobnikami-rodzicami, w celu stworzenia jednego lub dwóch nowych osobników-potomstwa. Jest to odpowiednik rozmnażania płciowego w biologii.
* **Mechanizm:**

Dla każdej pary rodziców losowo wybierane są **punkty krzyżowania** (lub punkty podziału) w ich reprezentacji genetycznej.

Następnie "genotypy" rodziców są dzielone w tych punktach, a powstałe fragmenty są zamieniane, tworząc nowe kombinacje cech.

* **Przykłady:**
  + **Krzyżowanie jednopunktowe:** Losowo wybierany jest jeden punkt. Część genotypu przed punktem pochodzi od jednego rodzica, a część po punkcie od drugiego.
    - Rodzic 1: [1 0 | 1 1 0], Rodzic 2: [0 1 | 0 1 1], Potomek 1: [1 0 | 0 1 1], Potomek 2: [0 1 | 1 1 0]
  + **Krzyżowanie dwupunktowe, wielopunktowe:** Wybiera się odpowiednio dwa lub więcej punktów krzyżowania.
* **Rola:** Pozwala na **tworzenie nowych kombinacji cech** z istniejących dobrych rozwiązań. Sprzyja eksploracji przestrzeni rozwiązań poprzez łączenie korzystnych podrozwiązań.

**3. Mutacja (Mutation)**

* **Zasada działania:** Operator mutacji wprowadza **losowe, niewielkie zmiany** w genotypie jednego osobnika (potomka lub, rzadziej, rodzica). Jest to odpowiednik mutacji genetycznych w biologii. Mutacja jest zazwyczaj stosowana z niskim prawdopodobieństwem.
* **Mechanizm:**
  + Dla każdej pozycji w genotypie lub dla całego genotypu, z pewnym małym prawdopodobieństwem, losowo zmieniana jest wartość (np. bit 0 na 1, liczba rzeczywista w zakresie, węzeł w drzewie).
* **Przykłady:**

**Mutacja bitowa:** Jeśli genotyp jest ciągiem bitów, losowo wybrany bit jest odwracany (0 na 1 lub 1 na 0).

* + - Osobnik: [1 0 1 1 0], Po mutacji (np. trzeci bit): [1 0 0 1 0]

**Mutacja wymiany:** Wymiana pozycji dwóch wartości w genotypie.  
**Mutacja zmiennych rzeczywistych:** Dodanie małej losowej wartości do atrybutu liczbowego.

* **Rola:**
  + **Wprowadza różnorodność** do populacji, zapobiegając jej przedwczesnemu zbiegnięciu się do lokalnego ekstremum (utknięciu w suboptymalnym rozwiązaniu).
  + Pozwala na **odkrywanie nowych obszarów** w przestrzeni rozwiązań, które nie były dostępne poprzez samą rekombinację. Jest to mechanizm eksploracji.
  + Bez mutacji populacja mogłaby szybko stać się jednorodna, a dalsza ewolucja by zanikła.

Współdziałanie tych trzech operatorów – selekcji (kierującej algorytm w stronę lepszych rozwiązań), rekombinacji (łączącej dobre rozwiązania) i mutacji (wprowadzającej różnorodność i eksplorację) – pozwala algorytmom ewolucyjnym efektywnie przeszukiwać złożone przestrzenie rozwiązań i znajdować dobre, często optymalne lub bliskie optymalnym, rozwiązania.

**6. Społeczne aspekty wdrażania rozwiązań sztucznej inteligencji w praktyce**

Wdrażanie rozwiązań sztucznej inteligencji (AI) w praktyce to nie tylko kwestia techniczna, ale także proces głęboko wpływający na społeczeństwo, gospodarkę i etykę. Wiąże się z nim szereg kluczowych aspektów społecznych, które muszą być brane pod uwagę.

**1. Rynek pracy i zatrudnienie:**

**Automatyzacja i utrata miejsc pracy:** AI może zautomatyzować powtarzalne i rutynowe zadania, co może prowadzić do utraty miejsc pracy w niektórych sektorach (np. produkcja, usługi).  
**Tworzenie nowych miejsc pracy:** Jednocześnie AI tworzy nowe zawody i specjalizacje (np. inżynierowie AI, analitycy danych, specjaliści od etyki AI).  
**Zmiana charakteru pracy:** Praca stanie się bardziej zogniskowana na kreatywności, rozwiązywaniu problemów, myśleniu krytycznym i współpracy z systemami AI.  
**Potrzeba przekwalifikowania:** Konieczność inwestowania w edukację i szkolenia, aby siła robocza mogła dostosować się do zmieniających się wymagań rynku pracy.

**2. Etyka i odpowiedzialność:**

**Uprzedzenia (Bias):** Systemy AI są trenowane na danych, które mogą zawierać uprzedzenia historyczne lub społeczne. Jeśli AI uczy się z takich danych, może powielać i wzmacniać dyskryminację (np. w rekrutacji, udzielaniu kredytów, systemach sprawiedliwości).  
**Transparentność i wyjaśnialność (Explainability / Interpretability):** Wiele zaawansowanych modeli AI (tzw. "czarne skrzynki") jest trudnych do zrozumienia. Brak możliwości wyjaśnienia, dlaczego system podjął daną decyzję, rodzi problemy z zaufaniem, odpowiedzialnością i zgodnością z prawem (np. w medycynie, finansach).  
**Odpowiedzialność za błędy:** Kto ponosi odpowiedzialność, gdy system AI popełni błąd, który wyrządzi szkodę (np. autonomiczny samochód)? Producent, deweloper, operator, użytkownik?  
**Prywatność danych:** Rozwiązania AI często wymagają dostępu do ogromnych ilości danych osobowych, co rodzi obawy o prywatność, bezpieczeństwo i nadużycia.

**3. Bezpieczeństwo i zaufanie:**

**Bezpieczeństwo cybernetyczne:** Systemy AI mogą być celem ataków (np. zatruwanie danych treningowych, ataki adversarialne) lub same mogą być wykorzystywane do ataków.  
**Zaufanie publiczne:** Brak zaufania do AI może wynikać z obaw o błędy, uprzedzenia, utratę kontroli czy niezrozumienie działania algorytmów. Budowanie zaufania wymaga transparentności i odpowiedzialności.  
**Autonomia systemów:** Wzrost autonomii systemów AI (np. w broni autonomicznej) rodzi poważne dylematy etyczne i kwestie kontroli.

**4. Nierówności społeczne:**

**Cyfrowy podział:** Korzyści z AI mogą być nierównomiernie rozłożone, pogłębiając podział między tymi, którzy mają dostęp do technologii i umiejętności, a tymi, którzy go nie mają.  
**Koncentracja władzy:** Rozwój AI może prowadzić do koncentracji władzy ekonomicznej i politycznej w rękach nielicznych firm lub państw dominujących w tej technologii.

**5. Regulacje i polityka:**

**Potrzeba regulacji:** Konieczność tworzenia adekwatnych ram prawnych i etycznych, które będą nadążać za tempem rozwoju AI (np. regulacje dotyczące prywatności danych, odpowiedzialności, stosowania AI w krytycznych obszarach).  
**Standardy bezpieczeństwa i testowania:** Opracowanie standardów dla bezpiecznego i wiarygodnego wdrażania systemów AI.

**6. Wpływ na interakcje międzyludzkie i psychikę:**

**Dehumanizacja:** Nadmierne poleganie na AI w obszarach wrażliwych (np. obsługa klienta, opieka zdrowotna) może prowadzić do zmniejszenia empatii i osobistego kontaktu.  
**Uzależnienie i manipulacja:** Algorytmy AI (np. w mediach społecznościowych) mogą być optymalizowane pod kątem utrzymywania uwagi użytkownika, co rodzi pytania o uzależnienie i możliwość manipulacji.

Wdrożenie AI wymaga **interdyscyplinarnego podejścia**, łączącego inżynierię, etykę, prawo, socjologię i politykę, aby maksymalizować korzyści i minimalizować ryzyka społeczne. Kluczowe jest prowadzenie otwartej debaty publicznej i edukacji na temat AI.

**Przedmioty: Administracja i utrzymanie systemów baz danych / Architektury baz danych / Modelowanie danych wielowymiarowych**

**1. Wymień podstawowe zadania administratora bazy danych.**

Administrator bazy danych (DBA - Database Administrator) to kluczowa rola w każdej organizacji, która wykorzystuje systemy baz danych. Jego zadania są różnorodne i obejmują zarządzanie całym cyklem życia bazy danych. Do podstawowych zadań DBA należą:

**Instalacja i Konfiguracja:** Instalowanie oprogramowania systemu zarządzania bazą danych (DBMS) oraz konfigurowanie go pod kątem optymalnej wydajności i bezpieczeństwa.  
**Projektowanie i Tworzenie Baz Danych:** Współpraca z deweloperami przy projektowaniu schematów baz danych, tworzenie tabel, indeksów, widoków, procedur składowanych i funkcji.  
**Zarządzanie Użytkownikami i Uprawnieniami:** Tworzenie kont użytkowników, przypisywanie ról i zarządzanie uprawnieniami dostępu do danych, zapewniając zgodność z polityką bezpieczeństwa.  
**Monitorowanie Wydajności (Performance Monitoring):** Ciągłe monitorowanie działania bazy danych, identyfikowanie i rozwiązywanie problemów z wydajnością, optymalizacja zapytań i indeksów.  
**Tworzenie Kopii Zapasowych i Odzyskiwanie Danych (Backup and Recovery):** Regularne tworzenie kopii zapasowych bazy danych oraz planowanie i testowanie procedur odzyskiwania danych po awarii. To kluczowe dla ciągłości działania biznesu.  
**Utrzymanie i Optymalizacja:** Przeprowadzanie rutynowych zadań konserwacyjnych, takich jak reorganizacja indeksów, optymalizacja przestrzeni dyskowej, usuwanie zbędnych danych.  
**Audyt i Bezpieczeństwo:** Monitorowanie dostępu do bazy danych, identyfikowanie potencjalnych zagrożeń bezpieczeństwa i wdrażanie mechanizmów ochrony danych.  
**Planowanie Pojemności (Capacity Planning):** Analiza trendów wzrostu danych i przewidywanie przyszłych potrzeb w zakresie pamięci masowej i zasobów sprzętowych.  
**Rozwiązywanie Problemów (Troubleshooting):** Diagnozowanie i naprawianie błędów i problemów związanych z bazą danych, od prostych zapytań po złożone awarie systemu.  
**Dokumentacja:** Prowadzenie szczegółowej dokumentacji dotyczącej struktury bazy danych, konfiguracji, procedur operacyjnych i polityk bezpieczeństwa.

**2. Na czym polega semiaddytywność miary w modelu wielowymiarowym?**

W kontekście **modelu wielowymiarowego** (często używanego w hurtowniach danych i systemach OLAP), **miara** to numeryczny atrybut, który jest przedmiotem analizy (np. sprzedaż, zysk, liczba sztuk). **Semiaddytywność** to właściwość miary, która określa, w jaki sposób może być ona agregowana (sumowana) wzdłuż różnych wymiarów.

Miara jest **semiaddytywna**, jeśli może być sumowana wzdłuż **niektórych wymiarów**, ale **nie wszystkich**. Oznacza to, że jej wartość ma sens sumowana tylko dla określonych kombinacji wymiarów.

**Przykłady semiaddytywnych miar:**

* **Stan magazynu (Inventory Balance):** Możesz sumować stan magazynu dla różnych produktów w danym punkcie czasu lub dla różnych lokalizacji w danym punkcie czasu. Jednak **nie możesz sumować stanu magazynu wzdłuż wymiaru czasu**, ponieważ sumowanie stanu magazynu z początku miesiąca i z końca miesiąca dałoby bezsensowny wynik. Stan magazynu jest miarą "migawkową" dla konkretnego punktu w czasie.
* **Saldo konta bankowego:** Podobnie jak stan magazynu, możesz sumować saldo kont dla różnych klientów, ale nie możesz sumować salda konta jednej osoby z różnych punktów w czasie.
* **Liczba pracowników:** Możesz sumować liczbę pracowników w różnych działach w danym momencie, ale nie sumuje się liczby pracowników z różnych miesięcy, aby uzyskać całkowitą liczbę zatrudnionych w roku (chyba że szukasz sumy punktowej, a nie całościowej).

**Dla porównania:**

* **Miara addytywna:** Może być sumowana wzdłuż **wszystkich wymiarów**. Przykładem jest Wartość Sprzedaży. Suma sprzedaży produktów, w różnych lokalizacjach i w różnych okresach ma zawsze sens.
* **Miara nieaddytywna:** Nie może być sumowana wzdłuż **żadnego wymiaru**. Przykładem jest Średnia Cena. Sumowanie średnich cen nie ma sensu, a żeby uzyskać średnią dla większego zagregowania, trzeba by ponownie obliczyć ją z sumarycznych wartości.

Zrozumienie semiaddytywności jest kluczowe przy projektowaniu hurtowni danych i raportów, ponieważ pozwala na prawidłowe agregowanie danych i unikanie błędnych wniosków analitycznych. Systemy OLAP muszą być świadome tej właściwości, aby prawidłowo obsługiwać zapytania analityczne.

**3. Omów różnice pomiędzy wierszowym a kolumnowym składowaniem danych w bazach relacyjnych.**

W bazach relacyjnych dane mogą być fizycznie składowane na dwa podstawowe sposoby: wierszowo lub kolumnowo. Wybór metody składowania ma znaczący wpływ na wydajność zapytań, zwłaszcza w przypadku dużych zbiorów danych i specyficznych wzorców dostępu.

**Wierszowe Składowanie Danych (Row-Oriented Storage)**

W tradycyjnych bazach danych (np. większość systemów OLTP – Online Transaction Processing) dane są przechowywane **wiersz po wierszu**. Oznacza to, że wszystkie wartości należące do jednego rekordu (wiersza) są przechowywane obok siebie na dysku.

**Typowe cechy i zastosowania:**

**Optymalne dla transakcji (OLTP):** Szybkie wstawianie nowych rekordów i modyfikowanie istniejących, ponieważ cały wiersz jest dostępny w jednym bloku dyskowym.  
**Dobre dla zapytań, które pobierają wszystkie (lub większość) kolumn wiersza:** Kiedy potrzebujesz danych o całym obiekcie (np. wszystkich danych o konkretnym kliencie), odczytujesz tylko jeden blok.  
**Mniej efektywne dla zapytań analitycznych:** Kiedy zapytanie potrzebuje tylko kilku kolumn z wielu wierszy (np. sumy sprzedaży dla danej kolumny), system musi odczytać całe wiersze, w tym dane z kolumn, które nie są potrzebne, co prowadzi do większego obciążenia I/O.  
**Wysoka redundancja danych:** Jeśli wstawiasz wiele wierszy, powtarzające się wartości w kolumnach są przechowywane wielokrotnie.

**Kolumnowe Składowanie Danych (Column-Oriented Storage)**

**Zasada działania:** W bazach danych z kolumnowym składowaniem danych (często używanych w systemach OLAP – Online Analytical Processing i Big Data) dane są przechowywane **kolumna po kolumnie**. Oznacza to, że wszystkie wartości należące do jednej kolumny są przechowywane obok siebie na dysku.

**Typowe cechy i zastosowania:**

**Optymalne dla zapytań analitycznych (OLAP):** Kiedy zapytanie potrzebuje tylko kilku kolumn z wielu wierszy (np. SELECT SUM(WartośćSprzedaży) FROM Sprzedaz), system musi odczytać tylko bloki danych dla tych konkretnych kolumn, co drastycznie zmniejsza obciążenie I/O.  
**Dobre dla agregacji:** Szybkie obliczanie agregacji (sumy, średnie, liczniki) dla dużej liczby wierszy, ponieważ wszystkie wartości potrzebne do agregacji są przechowywane blisko siebie.  
**Wysoka kompresja danych:** Wartości w pojedynczej kolumnie są zazwyczaj tego samego typu danych i często mają powtarzające się wartości (np. kolumna Kraj). Pozwala to na zastosowanie bardzo efektywnych algorytmów kompresji, co zmniejsza zużycie miejsca na dysku i przyspiesza odczyt.  
**Mniej efektywne dla transakcji (OLTP):** Wstawianie nowego wiersza wymaga modyfikacji wielu bloków dyskowych (po jednym dla każdej kolumny), co jest mniej wydajne niż w systemach wierszowych. Podobnie, aktualizacja pojedynczego wiersza może wymagać odczytu i zapisu wielu bloków.  
**Przykłady baz danych:** Vertica, Amazon Redshift, Google BigQuery, ClickHouse.

**Podsumowanie różnic:**

|  |  |  |
| --- | --- | --- |
| **Cecha** | **Wierszowe Składowanie (Row-Oriented)** | **Kolumnowe Składowanie (Column-Oriented)** |
| **Główna jednostka przechowywania** | Wiersz (rekord) | Kolumna |
| **Optymalizacja dla** | OLTP (transakcje, szybkie INSERT/UPDATE/DELETE) | OLAP (analityka, szybkie SELECT na części kolumn) |
| **Typowe zapytania** | SELECT \* FROM Tabela WHERE ID = ... | SELECT SUM(KolumnaX) FROM Tabela WHERE KolumnaY = ... |
| **I/O dla analityki** | Wysokie (odczyt całych wierszy) | Niskie (odczyt tylko potrzebnych kolumn) |
| **Kompresja danych** | Mniejsza efektywność | Wysoka efektywność (powtarzające się wartości w kolumnach) |
| **Wstawianie/Aktualizacja** | Szybkie (zapis/odczyt jednego bloku) | Wolniejsze (zapis/odczyt wielu bloków) |

Eksportuj do Arkuszy

Wybór między wierszowym a kolumnowym składowaniem danych zależy od dominującego wzorca użycia bazy danych. Wiele nowoczesnych baz danych oferuje hybrydowe podejścia lub pozwala na wybór modelu składowania dla poszczególnych tabel.

**4. Czym różni się pełna kopia bazy danych (ang. full backup) od kopii przyrostowej (ang. incremental backup)?**

Tworzenie kopii zapasowych jest kluczowym elementem zarządzania bazami danych, zapewniającym bezpieczeństwo danych i możliwość ich odzyskania po awarii. Istnieją różne strategie tworzenia kopii, z których najczęściej wyróżnia się pełne i przyrostowe kopie zapasowe.

**Pełna kopia zapasowa** to kompletna kopia **wszystkich danych** zawartych w bazie danych w momencie jej wykonania. Zawiera wszystkie pliki danych, dzienniki transakcji (lub ich stan w momencie kopii) oraz metadane niezbędne do całkowitego odtworzenia bazy danych do stanu z chwili wykonania kopii.

**Typowe cechy:**

* **Zawartość:** Obejmuje *wszystkie* dane, niezależnie od tego, czy zmieniły się od ostatniej kopii.  
  **Rozmiar:** Zazwyczaj jest największa ze wszystkich typów kopii.  
  **Czas wykonania:** Trwa najdłużej, ponieważ kopiuje całą bazę danych.  
  **Odtwarzanie (Recovery):** Najprostszy i najszybszy sposób odzyskania danych. Aby przywrócić bazę danych, wystarczy użyć tylko jednej, ostatniej pełnej kopii zapasowej.  
  **Wymagane miejsce:** Wymaga największej przestrzeni dyskowej do przechowywania.

**Kopia przyrostowa (Incremental Backup)**

**Czym jest:** **Kopia przyrostowa** to kopia zapasowa, która zawiera **tylko te dane, które zmieniły się** od momentu wykonania **ostatniej dowolnej kopii zapasowej** (czyli od ostatniej pełnej lub ostatniej przyrostowej kopii).

**Typowe cechy:**

* **Zawartość:** Kopiuje *tylko zmiany*. System śledzi, które bloki danych zostały zmodyfikowane od poprzedniej kopii zapasowej i kopiuje tylko te zmienione bloki.  
  **Rozmiar:** Zazwyczaj jest najmniejsza z kopii zapasowych.  
  **Czas wykonania:** Trwa najkrócej, ponieważ kopiuje tylko niewielką część danych.  
  **Odtwarzanie (Recovery):** Bardziej złożony i czasochłonny proces. Aby przywrócić bazę danych, potrzebujesz **ostatniej pełnej kopii zapasowej** oraz **wszystkich kolejnych kopii przyrostowych** wykonanych od tej pełnej kopii, w odpowiedniej kolejności.  
  **Wymagane miejsce:** Wymaga najmniejszej przestrzeni dyskowej na przechowywanie pojedynczych kopii.

**Kluczowe różnice w porównaniu:**

|  |  |  |
| --- | --- | --- |
| **Cecha** | **Pełna Kopia (Full Backup)** | **Kopia Przyrostowa (Incremental Backup)** |
| **Kopiowana zawartość** | Wszystkie dane bazy danych | Tylko zmiany od ostatniej **dowolnej** kopii zapasowej |
| **Rozmiar** | Duży | Mały |
| **Czas wykonania** | Długi | Krótki |
| **Złożoność odtwarzania** | Niska (potrzebna tylko jedna kopia) | Wysoka (potrzebna pełna kopia + wszystkie kolejne przyrostowe) |
| **Czas odtwarzania** | Krótki | Długi |
| **Wymagane miejsce** | Dużo (na każdą kopię) | Mało (na pojedynczą kopię) |

W praktyce często stosuje się strategię mieszaną, np. pełne kopie raz w tygodniu, a kopie przyrostowe codziennie, co pozwala zbalansować czas tworzenia kopii, wymagane miejsce i czas odzyskiwania danych.

**5. Do czego służy w relacyjnej bazie danych ograniczenie =**(zasada, która pilnuje poprawności danych) **klucza obcego (ang. foreign key constraint)?**

**Ograniczenie klucza obcego (Foreign Key Constraint)** jest fundamentalnym elementem **relacyjnego modelu danych** i służy do utrzymania **integralności referencyjnej** między tabelami. Jest to jeden z najważniejszych mechanizmów zapewniających spójność danych w relacyjnej bazie danych.

**Definicja i Rola:**

* **Co to jest:** Klucz obcy to kolumna (lub zestaw kolumn) w jednej tabeli, która **odwołuje się do klucza głównego (Primary Key)** lub klucza kandydującego (Candidate Key) w innej tabeli (zwanej tabelą nadrzędną lub tabelą referencyjną). Kolumna klucza obcego w tabeli podrzędnej (tabeli odwołującej się) musi zawierać wartości, które istnieją w kolumnie klucza głównego tabeli nadrzędnej, lub wartości NULL.
* **Główna rola - Integralność Referencyjna:** Ograniczenie klucza obcego zapewnia, że **żadna wartość w kolumnie klucza obcego nie będzie odwoływać się do nieistniejącego wiersza** w tabeli nadrzędnej. Zapobiega to powstawaniu "wiszących referencji" (dangling references), czyli sytuacji, w której rekord w jednej tabeli odwołuje się do rekordu, który już nie istnieje w innej tabeli.
* **Ustanawianie związków:** Klucze obce są mechanizmem, za pomocą którego tworzymy i egzekwujemy **związki logiczne (relacje)** między tabelami (np. jeden do wielu, jeden do jednego).

**Przykładowe zastosowanie:**

Rozważmy dwie tabele:

1. Działy (tabela nadrzędna):

ID\_Dzialu (Klucz Główny)  
Nazwa\_Dzialu

1. Pracownicy (tabela podrzędna):
   * ID\_Pracownika (Klucz Główny)  
     Imie
   * Nazwisko  
     ID\_Dzialu (Klucz Obcy, odwołujący się do ID\_Dzialu w tabeli Działy)

Ograniczenie klucza obcego na kolumnie ID\_Dzialu w tabeli Pracownicy zapewni, że:

* Nie będzie można dodać nowego pracownika do tabeli Pracownicy z ID\_Dzialu, które **nie istnieje** w tabeli Działy. (Np. nie możesz przypisać pracownika do działu o ID 999, jeśli dział o ID 999 nie istnieje w tabeli Działy).
* Nie będzie można **usunąć** działu z tabeli Działy, jeśli istnieją pracownicy, którzy są do niego przypisani (chyba że zdefiniowano odpowiednią akcję kaskadową, o czym poniżej).
* Nie będzie można **zmienić** ID\_Dzialu w tabeli Działy, jeśli istnieją pracownicy, którzy są do niego przypisani.

**Akcje Kaskadowe (Referential Actions):**

Klucze obce pozwalają również na zdefiniowanie, co ma się stać, gdy próbujemy usunąć lub zaktualizować wiersz w tabeli nadrzędnej, do którego istnieją odwołania w tabeli podrzędnej. Najczęściej używane akcje to:

Podsumowując, ograniczenie klucza obcego jest niezbędne do utrzymania **spójności, poprawności i integralności danych** w relacyjnych bazach danych, zapewniając, że relacje między tabelami są zawsze poprawne i logiczne.

Przedmioty: Analiza i wizualizacja danych / Zastosowanie metod uczenia maszynowego

**1. Omów różnice pomiędzy uczeniem nadzorowanym (supervised learning) i uczeniem nienadzorowanym (unsupervised learning).**

Uczenie maszynowe dzieli się na kilka głównych paradygmatów, z których uczenie nadzorowane i nienadzorowane są najbardziej podstawowe i powszechne. Różnią się one znacząco pod względem rodzaju danych wejściowych, celów i metod.

Uczenie nadzorowane to taki typ uczenia maszynowego, gdzie model uczy się na podstawie danych, które mają już przypisane prawidłowe odpowiedzi – tzw. etykiety. Celem jest nauczenie modelu, jak przewidywać te odpowiedzi dla nowych danych. Przykład: rozpoznawanie, czy zdjęcie przedstawia kota czy psa – model widzi obraz i zna prawidłową odpowiedź.

Z kolei uczenie nienadzorowane działa bez etykiet – model otrzymuje dane, ale nie wie, co one oznaczają. Jego zadaniem jest samodzielne znalezienie ukrytych wzorców, podobieństw czy grup. Przykład: pogrupowanie klientów sklepu według zachowań zakupowych, bez wcześniejszej wiedzy o ich typach.

Główna różnica polega więc na tym, że uczenie nadzorowane potrzebuje danych z odpowiedziami, a nienadzorowane – nie. Pierwsze służy do przewidywania, drugie do odkrywania struktury danych.

**2. Jakie są najważniejsze metody zapobiegania przeuczeniu (overfitting) modelu?**

**Przeuczenie (overfitting)** to sytuacja, gdy model uczy się danych treningowych **zbyt dokładnie**, łącznie z szumem i przypadkowymi zależnościami, przez co **słabo działa na nowych danych**.

Aby temu zapobiec, stosuje się kilka najważniejszych metod:

**1. Walidacja krzyżowa (cross-validation)**Dzielimy dane na części – uczymy model na jednych, testujemy na innych. Dzięki temu sprawdzamy, czy model generalizuje, a nie tylko zapamiętuje.

**2. Regularizacja (L1, L2)**Dodaje karę do funkcji kosztu za zbyt duże wagi modelu. Wymusza prostszy model, który mniej się dopasowuje do szumu.

**3. Zmniejszenie złożoności modelu**Użycie prostszej architektury – np. mniej neuronów, warstw lub gładszej funkcji – ogranicza ryzyko przeuczenia.

**4. Zwiększenie ilości danych treningowych**Więcej danych = trudniej zapamiętać wszystko = lepsze generalizowanie.

**5. Augmentacja danych**Sztuczne zwiększanie danych przez modyfikacje (np. obracanie zdjęć) – popularne w analizie obrazów.

**6. Wczesne zatrzymanie (early stopping)**Monitorujemy błąd walidacyjny i **przerywamy uczenie**, gdy zaczyna rosnąć, zanim model się przeuczy.

**7. Dropout (w sieciach neuronowych)**Losowo „wyłączamy” część neuronów w czasie uczenia, by zapobiec zależnościom między nimi.

**Podsumowanie:**

Aby zapobiec przeuczeniu, należy **kontrolować złożoność modelu, stosować walidację, regularizację i odpowiednio przygotowywać dane**. To pozwala uzyskać model, który lepiej działa na nowych, nieznanych danych.

**3. Co to jest funkcja aktywacji w komórce sieci neuronowej?**

W kontekście sztucznych sieci neuronowych, **funkcja aktywacji** (ang. activation function) to kluczowy element każdej **komórki (neuronu)**. Jej rola polega na wprowadzeniu **nieliniowości** do modelu, co umożliwia sieci neuronowej uczenie się i reprezentowanie złożonych, nieliniowych relacji w danych.

**Funkcja aktywacji w komórce sieci neuronowej** to mechanizm, który odpowiada za przekształcenie sumy ważonej sygnałów wejściowych neuronu w jego sygnał wyjściowy.

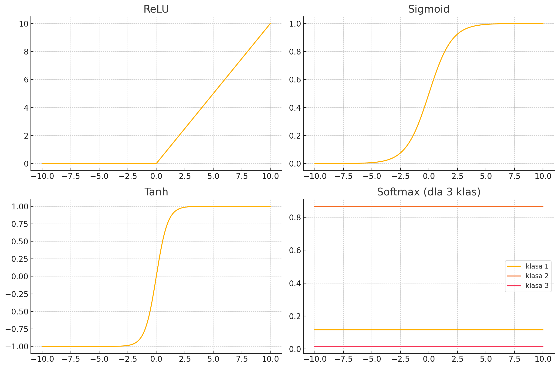
Najpierw neuron sumuje swoje wejścia pomnożone przez wagi i dodaje wartość biasu. Otrzymany wynik jest następnie przetwarzany przez funkcję aktywacji, która wprowadza **nieliniowość**. Dzięki temu sieć neuronowa może uczyć się złożonych, nieliniowych zależności – co nie byłoby możliwe, gdybyśmy używali tylko operacji liniowych.

Funkcja aktywacji wpływa również na zakres wyjściowy neuronu, a tym samym na działanie całej sieci i jej proces uczenia.

Najczęściej stosowane funkcje aktywacji to:

**ReLU** – szybka i prosta, zwraca 0 lub dodatnią wartość,  
**Sigmoid** – daje wynik w zakresie od 0 do 1, ale może powodować zanikający gradient,  
**Tanh** – zwraca wartości od -1 do 1,  
**Softmax** – używana w warstwie wyjściowej przy klasyfikacji wieloklasowej.

Dobór funkcji zależy od problemu i miejsca jej użycia w architekturze sieci.



**4. Do czego służy wykres skrzynkowy (box-plot)?**

**Wykres skrzynkowy (box-plot)**, znany również jako wykres pudełkowy z wąsami (box-and-whisker plot), to graficzna metoda prezentacji rozkładu danych numerycznych poprzez ich kwartyle. Jest niezwykle przydatny do wizualnego podsumowania rozkładu danych i porównywania rozkładów między różnymi grupami.

**Do czego służy:**

Główne cele i zastosowania wykresu skrzynkowego to:

1. **Wizualizacja Rozkładu Danych:** Szybkie przedstawienie kluczowych statystyk pięciu liczb (five-number summary):
   * **Minimum:** Najmniejsza wartość w zbiorze danych (z wyłączeniem wartości odstających).
   * **Pierwszy kwartyl (Q1 / 25. percentyl):** 25% danych znajduje się poniżej tej wartości. Dolna krawędź "skrzynki".
   * **Mediana (Q2 / 50. percentyl):** Środkowa wartość zbioru danych, która dzieli go na dwie równe części. Linia wewnątrz "skrzynki".
   * **Trzeci kwartyl (Q3 / 75. percentyl):** 75% danych znajduje się poniżej tej wartości. Górna krawędź "skrzynki".
   * **Maksimum:** Największa wartość w zbiorze danych (z wyłączeniem wartości odstających).
2. **Identyfikacja Centralnej Tendencji i Rozproszenia:**
   * **Mediana** wskazuje na centralną wartość rozkładu.
   * **Długość skrzynki (IQR - Interquartile Range = Q3 - Q1)** pokazuje rozpiętość środkowych 50% danych, co jest miarą rozproszenia i zmienności.
3. **Wykrywanie Skoszenia (Skewness):**
   * Położenie mediany w skrzynce oraz długość "wąsów" mogą wskazywać na asymetrię (skoszenie) rozkładu danych.
     + Jeśli mediana jest bliżej Q1, a górny wąs dłuższy, rozkład jest skoszony w prawo (dodatnie skoszenie).
     + Jeśli mediana jest bliżej Q3, a dolny wąs dłuższy, rozkład jest skoszony w lewo (ujemne skoszenie).
4. **Wykrywanie Punktów Odstających (Outliers):**
   * Wartości odstające (często reprezentowane jako pojedyncze punkty poza wąsami) są wyraźnie widoczne. Punktem odstającym jest zazwyczaj wartość leżąca poza zakresem Q1−1.5⋅IQR lub Q3+1.5⋅IQR.
5. **Porównywanie Rozkładów Między Grupami:**
   * Jest to jedna z największych zalet box-plotów. Można łatwo umieścić wiele wykresów skrzynkowych obok siebie (np. sprzedaż w różnych regionach, wyniki egzaminów w różnych klasach), aby wizualnie porównać mediany, zakresy, rozproszenia i obecność wartości odstających w różnych grupach.

**Elementy Wykresu Skrzynkowego:**

* **Skrzynka (Box):** Rozciąga się od pierwszego kwartyla (Q1) do trzeciego kwartyla (Q3). Wewnątrz skrzynki znajduje się linia reprezentująca medianę (Q2).
* **Wąsy (Whiskers):** Rozciągają się od skrzynki do najdalej położonych punktów danych, które nie są wartościami odstającymi. Zazwyczaj ich długość jest ograniczona do 1.5⋅IQR od Q1 i Q3.
* **Punkty Odstające (Outliers):** Indywidualne punkty danych, które leżą poza końcami wąsów, sugerując, że są one nietypowe dla reszty rozkładu.

Wykresy skrzynkowe są szczególnie użyteczne w eksploracyjnej analizie danych (EDA) do szybkiego zrozumienia podstawowych charakterystyk rozkładów numerycznych.

**5. W jaki sposób można określić czy dana wartość szeregu jest punktem odstającym (outlier)?**

Punkt odstający to taka wartość w szeregu danych, która **znacznie różni się od pozostałych** – może to oznaczać błąd lub ważne zjawisko.

Aby określić, czy dana wartość jest outlierem, można zastosować kilka metod:

**1. Reguła 1.5 IQR (rozstęp międzykwartylowy)**

* Obliczamy kwartyle: Q1 (25%) i Q3 (75%),  
  Liczymy IQR = Q3 – Q1,  
  Punkt jest odstający, jeśli:  
  jest mniejszy niż Q1 – 1.5×IQR,  
  lub większy niż Q3 + 1.5×IQR.

**2. Odchylenie standardowe (dla rozkładu normalnego)**

* Obliczamy średnią i odchylenie standardowe,  
  Punkt jest odstający, jeśli:

znajduje się dalej niż 2 lub 3 odchylenia od średniej.

**3. Wizualizacja (np. wykres pudełkowy / boxplot)**

Wartości odstające są widoczne jako punkty poza „wąsami” wykresu.

**4. Metody statystyczne i uczenia maszynowego**

np. DBSCAN, Isolation Forest – automatycznie wykrywają odchylenia w danych.

Punkt odstający to wartość wyraźnie różniąca się od innych. Można go wykryć za pomocą **statystyki opisowej** (IQR, odchylenie), **wizualizacji**, lub **algorytmów**.

- Metody probabilistyczne i statystyka (dr inż. Janusz Majewski; dr Beata Basiura)  
- Języki formalne i kompilatory (dr inż. Janusz Majewski; dr Edward Szczypka)  
- Bazy danych (dr inż. Robert Marcjan; dr inż. Marek Valenta)  
- Sztuczna inteligencja i systemy ekspertowe (dr inż. Marek Valenta; dr hab. inż. Rafał Dreżewski) **Framestick**  
- Zaawansowane systemy baz danych (dr inż. Marek Valenta ; dr inż. Robert Marcjan)  
- Architektury baz danych (mgr Grzegorz Stolecki)  
- Administracja i utrzymanie systemów baz danych (mgr Grzegorz Stolecki)  
- Grupowa sztuczna inteligencja (prof. dr hab. inż. Grzegorz Dobrowolski, dr inż. Adam Łuszpaj)  
JAVA Agent DEvelopment Framework (**JADE**)  
- Analiza i wizualizacja danych, Zastosowanie metod uczenia maszynowego Modelowanie danych wielowymiarow  
 (mgr Grzegorz Stolecki)

🅰️ A – Atomicity (Atomowość)ędzy z A, ale przed dodaniem do B → całość zostaje anulowana.  
🅲 C – Consistency (Spójność)  
🅸 I – Isolation (Izolacja)  
🅳 D – Durability (Trwałość)

klinet <->serwer aplikacji <-> baza danych

Średnia arytmetyczna <> Mediana <> Moda  
Rozstęp <> Wariancja <> Odchylenie standardowe

Średnia arytmetyczna  
Odchylenia od średniej  
Kwadraty odchyleń  
Suma kwadratów odchyleń: 4 + 0 + 4 = 8  
Wariancja próbki: Oznaczenie: s² = 4  
8 / (3 − 1) = 8 / 2 = 4  
Odchylenie standardowe (standard deviation):

Leksykalna <> syntaktyczna <> semantyczna

Moduł Języka Definicji Danych (DDL Processor) – CREATE, ALTER, DROP   
Moduł Języka Manipulacji Danymi (DML Processor) – INSERT, UPDATE   
Optymalizator Zapytań (Query Optimizer) – SELECT

Defmanzpytań  
DanTranWspółbiezności  
RecowerySystem

Menedżer Danych (Data Manager / Storage Manager)   
Menedżer Transakcji (Transaction Manager)   
Menedżer Współbieżności (Concurrency Control Manager)   
Menedżer Odzyskiwania (Recovery Manager)   
Katalog Systemowy (System Catalog / Data Dictionary)