1. System informatyczny a system komputerowy.

System informatyczny i system komputerowy to dwa pojęcia, które często są mylone ze sobą. System komputerowy to fizyczne urządzenie, takie jak komputer lub serwer, które jest wyposażone w oprogramowanie i sprzęt do przetwarzania i przechowywania danych. Z kolei system informatyczny to złożona kombinacja oprogramowania, sprzętu i danych, które działają razem, aby umożliwić określone zadanie, np. zarządzanie finansami, zarządzanie produkcją itp.

W skrócie, system komputerowy to narzędzie, a system informatyczny to narzędzie w połączeniu z danymi i oprogramowaniem, które razem realizują określone zadanie.

1. Struktura i zadania systemu operacyjnego.

Każdy z systemów operacyjnych składa się z trzech, podstawowych elementów, w skład których wchodzą: jądro, system plików i powłoki. Każdy element, ma odmienną charakterystykę oraz pełni inną rolę. Dzięki czemu system operacyjny spełnia tak różnorodne funkcje. dające możliwość efektywnemu zarządzaniu systemem komputerowym.

Główne zadania Systemu operacyjnego:

* Zarządzanie efektywnością komputera
* Zarządzanie oraz zbieranie danych umieszczonych na dyskach
* Udostępnianie aplikacji (jako wirtualne maszyny)-Udostępnianie przez System operacyjny uproszczonego obrazu w komputerze gdzie pracuje aplikacja.
* Jednoczesne działanie wielu aplikacji (Wielozadaniowość)-Umożliwia uruchamianie w tym samym czasie wielu procesów
* Kontakt z użytkownikiem - Wyróżniamy:System tekstowy (np.Dos) -Komunikacja z użytkownikiem poprzez wydawane komendy System graficzny (np. Windows, Linux) - Komunikacja z użytkownikiem poprzez graficzne symbole oraz okna
* Łączność z innymi urządzeniami bądź komputerami

1. Struktura systemu komputerowego.

Struktura systemu komputerowego składa się z pięciu zasadniczych warstw tj. warstwa sprzętowa, system operacyjny, programy narzędziowe, programy użytkowe i użytkownicy:

* Sprzęt – zapewnia podstawowe możliwości obliczeniowe (procesor, pamięć, urządzenia wejścia/wyjścia) – podstawowe zasoby systemu komputerowego;
* Oprogramowanie systemowe – kontroluje i koordynuje działanie zasobów sprzętowych przez zastosowanie różnych programów użytkowych dla różnych użytkowników. Warstwa tworzona przez twórców systemu operacyjnego – są to zazwyczaj wysoko wyspecjalizowani eksperci;
* Oprogramowanie narzędziowe – dogodne interfejsy użytkowe wspomagające zarządzanie zasobami sprzętowymi oraz usprawniające, modyfikujące oprogramowanie systemowe, zazwyczaj pisane przez niezależnych programistów, którzy mają na celu usprawnienia wykonywania programów w bardziej wygodny i wydajny sposób, a przy tym często eliminują błędy czy też niedociągnięcia oprogramowania systemowego;
* Oprogramowanie użytkowe – określają sposoby użycia zasobów systemowych do rozwiązywania problemów obliczeniowych zadanych przez użytkownika (kompilatory, systemy baz danych, gry, oprogramowanie biurowe), tworzone przez programistów;
* Użytkownicy – ludzie, maszyny, inne komputery, mający bezpośredni kontakt z oprogramowaniem użytkowym. Realizują różne zadania za pomocą programów użytkowych na sprzęcie komputerowym pod nadzorem systemu operacyjnego.

1. Omów podstawowe typy danych i wskaż przykładowe zastosowanie

* Liczby całkowite (Integer)

Opis: Liczby całkowite to liczby bez części ułamkowych, mogą być dodatnie, ujemne lub zerowe.

Zastosowanie: Przechowywanie ilości, liczników, indeksów w pętlach.

* Liczby zmiennoprzecinkowe (Floating-point)

Opis: Liczby zmiennoprzecinkowe to liczby, które mają część ułamkową.

Zastosowanie: Obliczenia wymagające precyzyjnych wartości, np. w fizyce, finansach.

* Znaki (Character)

Opis: Typ danych reprezentujący pojedynczy znak.

Zastosowanie: Przechowywanie pojedynczych znaków, np. w przetwarzaniu tekstu.

* Łańcuchy znaków (String)

Opis: Sekwencja znaków używana do przechowywania tekstu.

Zastosowanie: Przechowywanie i manipulowanie tekstem, np. w aplikacjach webowych.

* Wartości logiczne (Boolean)

Opis: Typ danych, który może przyjmować tylko dwie wartości: prawda (true) lub fałsz (false).

Zastosowanie: Warunki logiczne, sterowanie przepływem programu (np. instrukcje if).

* Tablice (Array)

Opis: Kolekcja elementów tego samego typu, przechowywana pod wspólną nazwą.

Zastosowanie: Przechowywanie wielu wartości jednego typu, np. listy ocen.

* Listy (List)

Opis: Dynamiczna kolekcja elementów, podobna do tablicy, ale z możliwością zmiany rozmiaru.

Zastosowanie: Przechowywanie dynamicznych list danych, np. kolekcje obiektów.

* Słowniki (Dictionary/Map)

Opis: Kolekcja par klucz-wartość, gdzie każdemu kluczowi przypisana jest określona wartość.

Zastosowanie: Przechowywanie danych z unikalnymi kluczami, np. książka adresowa.

* Krotki (Tuple)

Opis: Niezmienialna (immutable) kolekcja elementów różnych typów.

Zastosowanie: Grupowanie kilku różnych typów danych, np. wynik funkcji zwracający wiele wartości.

* Wyliczenia (Enum)

Opis: Specjalny typ danych reprezentujący zbiór stałych wartości.

Zastosowanie: Przechowywanie stałych, które mogą być grupowane, np. dni tygodnia, miesiące.

5. Charakterystyka architektury komunikacyjnej systemu bazy danych(1-,2-,3- warstwowa).

Poziom wewnętrzny – inaczej jest określany fizycznym, ponieważ jest związany z „fizycznym” sposobem przechowywania danych oraz tym, jak wygląda dostęp do nich. Przykładowo, w lokalnych bazach danych cały zbiór informacji jest przechowywany na jednym komputerze. Istnieją jednak również np. chmurowe bazy danych, w których informacje są przechowywane w modelu rozproszonym, a więc na co najmniej dwóch (a zazwyczaj więcej) urządzeniach fizycznych.

Poziom zewnętrzny – jest określany także poziomem użytkownika. W tym kontekście istotne jest to, w jaki sposób dane są prezentowane konkretnym odbiorcom. Na tym poziomie nie są prezentowane zbędne dane oraz te, do których użytkownik końcowy nie ma prawa dostępu.

Poziom pojęciowy, czyli koncepcyjny – to aspekt związany z opisem zawartości bazy danych oraz powiązań i relacji pomiędzy poszczególnymi jej elementami.

1. Zastosowanie i charakterystyka modelu ER w procesie projektowania pojęciowego modelu danych.

Model ER (Entity-Relationship) jest jednym z najważniejszych narzędzi w procesie projektowania koncepcyjnego modelu danych. Służy do przedstawiania danych w sposób abstrakcyjny, ale zrozumiały zarówno dla specjalistów technicznych, jak i biznesowych.

Zastosowanie Modelu ER

Koncepcyjne Modelowanie Danych: Model ER jest wykorzystywany na wczesnym etapie projektowania systemów informacyjnych. Pomaga w zrozumieniu i zdefiniowaniu wymagań dotyczących danych poprzez tworzenie diagramów, które przedstawiają obiekty rzeczywistego świata oraz ich wzajemne relacje.

Komunikacja z Interesariuszami: Diagramy ER są narzędziem komunikacyjnym, które ułatwia dialog między analitykami, projektantami systemów, programistami oraz użytkownikami końcowymi. Dzięki swojej graficznej formie są łatwiejsze do zrozumienia niż opis tekstowy.

Podstawa dla Fizycznego Modelowania Danych: Model ER służy jako fundament dla tworzenia fizycznych modeli baz danych. Na jego podstawie można generować schematy relacyjne dla baz danych, co przyspiesza proces implementacji.

Dokumentacja Systemu: Diagramy ER pełnią rolę dokumentacji systemu, która może być wykorzystywana na różnych etapach cyklu życia systemu, od analizy wymagań, przez implementację, aż po utrzymanie i rozwój systemu.

Charakterystyka Modelu ER

Encje (Entities): Encje to podstawowe elementy modelu ER. Reprezentują obiekty rzeczywistego świata, które mają znaczenie dla systemu informacyjnego, takie jak Klient, Produkt, Zamówienie. Każda encja posiada atrybuty, które opisują jej właściwości (np. encja Klient może mieć atrybuty takie jak imię, nazwisko, adres).

Atrybuty (Attributes): Atrybuty to właściwości opisujące encje. Mogą to być atrybuty proste (np. imię, nazwisko) lub złożone (np. adres, który może składać się z ulicy, miasta, kodu pocztowego).

Klucze (Keys): Klucze to specjalne atrybuty lub zestawy atrybutów, które jednoznacznie identyfikują encję. Najczęściej stosowany jest klucz główny (Primary Key), który musi być unikalny dla każdej instancji encji.

Relacje (Relationships): Relacje opisują powiązania między encjami. Mogą być różnego typu: jeden-do-jednego (1:1), jeden-do-wielu (1), wiele-do-wielu (M). Relacje mogą mieć swoje własne atrybuty (np. data zawarcia umowy w relacji między Klientem a Produktem).

Kardynalność (Cardinality): Kardynalność określa liczbę instancji jednej encji, które mogą być związane z instancją innej encji w relacji. Przykładowo, relacja jeden-do-wielu oznacza, że jedna instancja encji A może być związana z wieloma instancjami encji B, ale każda instancja encji B może być związana tylko z jedną instancją encji A.

Stopień (Degree): Stopień relacji określa, ile encji uczestniczy w danej relacji. Najczęściej spotykane są relacje binarne (między dwiema encjami), ale mogą istnieć również relacje ternarne (między trzema encjami) i bardziej złożone.

1. Operacje algebry relacyjnej w systemie bazy danych.

Operacje algebry relacyjnej są podstawą teoretyczną dla operacji na bazach danych. Algebra relacyjna to zestaw operacji służących do manipulacji relacjami (tabelami) w relacyjnych bazach danych. Oto główne operacje algebry relacyjnej:

1. Selekcja (Selection)

Symbol: σ

Opis: Wybiera wiersze z relacji, które spełniają określony warunek.

Przykład: σ\_{wiek > 30}(Pracownicy) wybiera wszystkich pracowników, którzy mają więcej niż 30 lat.

2. Projekcja (Projection)

Symbol: π

Opis: Wybiera określone kolumny z relacji, usuwając duplikaty.

Przykład: π\_{imie, nazwisko}(Pracownicy) wybiera tylko kolumny 'imie' i 'nazwisko' z tabeli Pracownicy.

3. Złączenie (Join)

Symbol: ⋈

Opis: Łączy dwie relacje na podstawie określonego warunku.

Przykład: Pracownicy ⋈\_{Pracownicy.id = Działy.pracownik\_id} Działy łączy tabelę Pracownicy z tabelą Działy na podstawie zgodności kolumn 'id' z 'pracownik\_id'.

4. Iloczyn kartezjański (Cartesian Product)

Symbol: ×

Opis: Łączy każdą krotkę z jednej relacji z każdą krotką z drugiej relacji, tworząc wszystkie możliwe kombinacje.

Przykład: Pracownicy × Działy tworzy wszystkie możliwe kombinacje wierszy z tabel Pracownicy i Działy.

5. Różnica (Difference)

Symbol: −

Opis: Zwraca wiersze, które są w jednej relacji, ale nie ma ich w drugiej relacji.

Przykład: Pracownicy − Menedżerowie zwraca wszystkich pracowników, którzy nie są menedżerami.

6. Sum (Union)

Symbol: ∪

Opis: Zwraca wiersze, które są w jednej z relacji lub w obu, eliminując duplikaty.

Przykład: Pracownicy ∪ Nowi\_Pracownicy zwraca wszystkich pracowników, włączając nowych pracowników, bez duplikatów.

7. Przecięcie (Intersection)

Symbol: ∩

Opis: Zwraca wiersze, które są wspólne dla obu relacji. Przykład: Pracownicy ∩ Menedżerowie zwraca wszystkich pracowników, którzy są również menedżerami.

8. Renaming (Zmiana nazwy)

Symbol: ρ

Opis: Zmienia nazwę relacji lub jej atrybutów.

Przykład: ρ\_{Nowi\_Pracownicy(imie, wiek)}(Pracownicy) zmienia nazwę relacji Pracownicy na Nowi\_Pracownicy oraz kolumny na 'imie' i 'wiek'.

1. Normalizacja danych oraz ich charakterystyka: 0, 1, 2, 3 NF oraz BCNF.

Normalizacja danych to proces organizowania danych w bazie danych w taki sposób, aby minimalizować redundancję i zapewnić integralność danych. Proces ten polega na podziale tabel na mniejsze, bardziej złożone struktury bez utraty danych oraz na ustanowieniu relacji między nimi. Normalizacja zwykle odbywa się w kilku etapach, z których każdy jest nazywany postacią normalną (ang. normal form). Poniżej omówię poszczególne postacie normalne oraz ich cechy charakterystyczne.

0 Normalna Forma (0 NF)

Charakterystyka: Dane w bazie danych są nieuporządkowane. Mogą zawierać powtarzające się grupy i redundancje.

Przykład: W jednej tabeli przechowywane są informacje o klientach i ich zamówieniach, gdzie każde zamówienie klienta jest przechowywane jako oddzielny rekord w tej samej tabeli.

1 Normalna Forma (1 NF)

Warunki:

Każda kolumna zawiera tylko jedną wartość (atomowość).

Wszystkie wartości w kolumnie są tego samego typu.

Każdy rekord jest unikalny.

Charakterystyka: Eliminacja powtarzających się grup, brak powtarzających się kolumn.

Przykład: Tabela, w której dane o klientach są przechowywane w osobnych kolumnach, a każde zamówienie klienta jest oddzielnym rekordem.

2 Normalna Forma (2 NF)

Warunki:

Spełnia wszystkie wymagania 1NF.

Każdy atrybut niekluczowy jest w pełni zależny od całego klucza głównego, a nie tylko jego części (pełna zależność funkcjonalna).

Charakterystyka: Usunięcie częściowych zależności.

Przykład: Tabela zamówień, w której wszystkie kolumny zależą od klucza głównego (np. numer zamówienia i ID klienta), bez zależności tylko od jednej części klucza złożonego.

3 Normalna Forma (3 NF)

Warunki:

Spełnia wszystkie wymagania 2NF.

Każdy atrybut niekluczowy jest bezpośrednio zależny od klucza głównego (bezpośrednia zależność funkcjonalna).

Żadne atrybuty niekluczowe nie są wzajemnie zależne (brak przechodnich zależności funkcjonalnych).

Charakterystyka: Usunięcie przechodnich zależności.

Przykład: Tabela klientów zawiera tylko kolumny bezpośrednio zależne od klucza głównego (np. ID klienta, imię, nazwisko, adres), bez dodatkowych informacji zależnych od innych atrybutów.

Boyce-Codd Normal Form (BCNF)

Warunki:

Spełnia wszystkie wymagania 3NF.

Dla każdej zależności funkcjonalnej X -> Y, X jest nadkluczem tabeli (czyli zbiór atrybutów X jest kluczem kandydującym).

Charakterystyka: Bardziej rygorystyczna wersja 3NF, eliminuje pewne anomalie, które mogą występować w 3NF.

Przykład: W tabeli, w której istnieją zależności wielokrotne, BCNF wymaga, aby każda taka zależność była związana z nadkluczem.

Normalizacja danych ma na celu eliminację redundancji i zapewnienie spójności danych, co ułatwia zarządzanie bazą danych i poprawia jej wydajność. Jednak nadmierna normalizacja może prowadzić do skomplikowanych zapytań i pogorszenia wydajności odczytu, dlatego w praktyce często stosuje się kompromisy w zależności od konkretnych wymagań aplikacji i bazy danych.

1. Organizacja plików w systemie bazy danych (plik uporządkowany, nieuporządkowany oraz haszowy).

Organizacja plików w systemie bazy danych odnosi się do sposobu, w jaki dane są fizycznie przechowywane i zarządzane na nośnikach pamięci, takich jak dyski twarde czy SSD. Efektywna organizacja plików jest kluczowa dla wydajności operacji bazy danych, takich jak wyszukiwanie, wstawianie, usuwanie i aktualizowanie danych. Istnieją trzy główne typy organizacji plików: pliki uporządkowane, nieuporządkowane oraz haszowe.

1. Plik uporządkowany (ang. Ordered File)

W pliku uporządkowanym, zwanym również plikiem sekwencyjnym, rekordy są przechowywane w określonym porządku, zazwyczaj według wartości kluczowej.

Zalety:

Efektywne wyszukiwanie binarne: Możliwe jest zastosowanie wyszukiwania binarnego, co znacząco przyspiesza operacje odczytu dla dużych zbiorów danych.

Łatwe sortowanie: Rekordy są już posortowane, co upraszcza niektóre operacje, takie jak generowanie raportów czy sortowanie wyników zapytań.

Wady:

Kosztowne wstawianie: Dodanie nowego rekordu wymaga często przesunięcia wielu innych rekordów, aby zachować porządek.

Kosztowne usuwanie: Usuwanie rekordu również może wymagać przesunięcia wielu innych rekordów, aby wypełnić lukę.

Zastosowanie:

Pliki uporządkowane są używane w sytuacjach, gdzie przeważają operacje odczytu nad operacjami zapisu i usuwania.

2. Plik nieuporządkowany (ang. Unordered File)

Pliki nieuporządkowane, znane również jako pliki heapowe, przechowują rekordy w kolejności, w jakiej są one dodawane. Nie ma tutaj żadnego określonego porządku.

Zalety:

Szybkie wstawianie: Nowe rekordy mogą być dodawane szybko, zazwyczaj na końcu pliku.

Proste zarządzanie: Struktura jest prosta, co ułatwia implementację i zarządzanie.

Wady:

Niska efektywność wyszukiwania: Wyszukiwanie rekordu wymaga przeszukiwania całego pliku (przeszukiwanie sekwencyjne).

Fragmentacja: Może wystąpić fragmentacja pliku, zwłaszcza po wielu operacjach usuwania.

Zastosowanie:

Pliki nieuporządkowane są używane tam, gdzie częste są operacje wstawiania i usuwania, a mniej ważna jest szybkość wyszukiwania.

3. Plik haszowy (ang. Hashed File)

W plikach haszowych rekordy są przechowywane według wartości wyznaczonej przez funkcję haszującą. Ta funkcja przekształca wartość klucza w lokalizację w pliku.

Zalety:

Bardzo szybkie wyszukiwanie: Rekordy mogą być odczytywane niemal natychmiast, ponieważ funkcja haszująca bezpośrednio wskazuje lokalizację rekordu.

Szybkie wstawianie i usuwanie: Dzięki funkcji haszującej, operacje te są zazwyczaj bardzo szybkie.

Wady:

Problemy z kolizjami: Funkcja haszująca może czasem przypisać dwóm różnym rekordom tę samą lokalizację, co wymaga dodatkowych mechanizmów do obsługi kolizji.

Brak porządku: Rekordy nie są przechowywane w żadnym określonym porządku, co utrudnia wykonywanie operacji wymagających sortowania.

Zastosowanie:

Pliki haszowe są idealne dla aplikacji, gdzie kluczowe jest szybkie wyszukiwanie i gdzie operacje sortowania są rzadkie lub nieistotne.

1. Charakterystyka „modelu kosztów” systemu bazy danych.

„Model kosztów” w systemie bazy danych jest kluczowym elementem optymalizacji zapytań. Jest to mechanizm używany przez optymalizator zapytań (query optimizer) do oceny różnych planów wykonania zapytań i wyboru najlepszego z nich na podstawie oszacowanych kosztów wykonania. Charakterystyka modelu kosztów obejmuje kilka istotnych aspektów:

1. Definicja Modelu Kosztów

Model kosztów to zestaw reguł i algorytmów używanych przez optymalizator zapytań do oszacowania zasobów wymaganych do wykonania danego planu zapytania. Zasoby te mogą obejmować czas procesora, operacje dyskowe, pamięć oraz przepustowość sieci.

2. Czynniki Wpływające na Koszty

Kilka kluczowych czynników wpływa na oszacowanie kosztów w modelu:

Koszty I/O: Czas potrzebny na odczyt i zapis danych na dysku. Operacje I/O są często najdroższym komponentem.

Koszty CPU: Czas procesora potrzebny na przetworzenie danych, w tym operacje sortowania, łączenia i filtrowania.

Koszty pamięci: Ilość pamięci potrzebnej do przetworzenia zapytania, co może wpływać na decyzje dotyczące np. używania pamięci podręcznej.

Koszty sieci: Jeśli zapytanie dotyczy rozproszonych baz danych, koszty związane z transferem danych między różnymi węzłami sieci.

3. Elementy Modelu Kosztów

Statystyki danych: Informacje o rozkładzie danych, liczbie wierszy w tabelach, liczbie unikalnych wartości w kolumnach itp.

Plany wykonania: Różne możliwe sposoby realizacji zapytania, np. różne sposoby łączenia tabel (nested loop join, hash join, merge join).

Operatory: Poszczególne operacje w planie zapytania, takie jak skanowanie tabeli, indeksowanie, sortowanie, łączenie itp.

4. Szacowanie Kosztów

Optymalizator zapytań szacuje koszty różnych planów wykonania na podstawie:

Liczby operacji I/O: Szacowane na podstawie liczby bloków danych, które muszą zostać odczytane/zapisane.

Liczby operacji CPU: Szacowane na podstawie liczby operacji potrzebnych do przetworzenia danych.

Dostępności indeksów: Wykorzystanie istniejących indeksów może znacząco zmniejszyć koszty.

6. Rola w Optymalizacji Zapytania

Optymalizator używa modelu kosztów do:

Porównania różnych strategii wykonania zapytania.

Wybierania najbardziej efektywnego planu wykonania.

Minimalizowania zasobów zużywanych podczas wykonania zapytania.

7. Ewolucja Modeli Kosztów

Z biegiem lat modele kosztów ewoluowały, stając się bardziej zaawansowane i precyzyjne, integrując bardziej szczegółowe statystyki, lepsze modele predykcyjne i zaawansowane techniki optymalizacyjne.

1. Proces transakcyjny w systemie bazy danych (realizacja sekwencyjna i współbieżna).

Proces transakcyjny w systemie bazy danych odnosi się do realizacji operacji na danych w sposób, który zapewnia ich integralność, spójność oraz bezpieczeństwo. Transakcje to logiczne jednostki pracy, które składają się z jednego lub kilku zapytań lub operacji. Poniżej przedstawiam omówienie realizacji sekwencyjnej i współbieżnej transakcji w systemach baz danych.

Realizacja sekwencyjna

Realizacja sekwencyjna transakcji oznacza, że transakcje są wykonywane jedna po drugiej, w ścisłej kolejności. Każda transakcja musi się zakończyć przed rozpoczęciem kolejnej. Oto główne cechy i korzyści realizacji sekwencyjnej:

Prostota: Prostsze zarządzanie, brak konieczności synchronizacji między transakcjami.

Brak problemów współbieżności: Unika problemów takich jak zakleszczenia, anomalie współbieżności (np. read-write, write-read).

Łatwość debugowania: Łatwiejsze diagnozowanie błędów, ponieważ sekwencyjny charakter eliminuje problemy wynikające z jednoczesnych operacji.

Jednakże, realizacja sekwencyjna ma również swoje wady:

Niska wydajność: Ogranicza liczbę transakcji, które mogą być przetwarzane w jednostce czasu, co może prowadzić do spadku wydajności w systemach o dużym obciążeniu.

Mniejsze wykorzystanie zasobów: Zasoby systemowe (procesor, pamięć, dyski) mogą być niewykorzystywane w pełni, gdyż transakcje nie są przetwarzane równolegle.

Realizacja współbieżna

Realizacja współbieżna transakcji oznacza, że wiele transakcji może być wykonywanych jednocześnie, co zwiększa wydajność systemu. Kluczowe aspekty realizacji współbieżnej obejmują:

Zwiększona wydajność: Równoległe przetwarzanie transakcji pozwala na lepsze wykorzystanie zasobów systemowych i skrócenie czasu przetwarzania.

Większa skalowalność: Systemy mogą obsługiwać większą liczbę jednoczesnych użytkowników i operacji.

Jednak współbieżność wprowadza także pewne wyzwania:

Problemy współbieżności: Może dochodzić do konfliktów, takich jak zakleszczenia (deadlocki) i różne anomalie współbieżności, np.:

Brudny odczyt (dirty read): Jedna transakcja odczytuje dane, które zostały zmodyfikowane przez inną transakcję, ale nie zostały jeszcze zatwierdzone.

Niepowtórny odczyt (non-repeatable read): Dane odczytane przez jedną transakcję są zmieniane przez inną transakcję między dwoma odczytami w pierwszej transakcji.

Odczyt fantomowy (phantom read): W wyniku operacji wstawiania lub usuwania danych przez inne transakcje, zakres danych odczytywanych przez jedną transakcję może się zmieniać.

Złożoność zarządzania współbieżnością: Konieczność stosowania mechanizmów takich jak blokady (locki), zarządzanie czasem transakcji, wielowersyjność (MVCC).

1. Algorytm zarządzania współbieżnym wykonywaniem transakcji (2PL).

Dwufazowy protokół blokowania (2PL, ang. Two-Phase Locking) jest algorytmem zarządzania współbieżnym wykonywaniem transakcji w systemach bazodanowych, który zapewnia właściwość izolacji transakcji, co jest jednym z kluczowych aspektów ACID (Atomicity, Consistency, Isolation, Durability). Algorytm ten składa się z dwóch faz: fazy rozprzestrzeniania blokad oraz fazy zwalniania blokad.

Fazy 2PL

Faza rozprzestrzeniania blokad (ang. Growing Phase):

W tej fazie transakcja może tylko zakładać blokady (read locks lub write locks) na zasobach, które chce zmodyfikować lub odczytać.

Transakcja nie może zwalniać żadnych blokad podczas tej fazy.

Faza ta trwa aż transakcja zarezerwuje wszystkie potrzebne blokady.

Faza zwalniania blokad (ang. Shrinking Phase):

Po zakończeniu fazy rozprzestrzeniania, transakcja przechodzi do fazy zwalniania blokad, gdzie może jedynie zwalniać blokady.

W tej fazie transakcja nie może zakładać żadnych nowych blokad.

Faza ta trwa do końca transakcji, aż wszystkie blokady zostaną zwolnione.

Właściwości 2PL

Zapewnienie izolacji:

Dzięki dwóm fazom, transakcje są chronione przed niepożądanym współbieżnym dostępem do tych samych zasobów, co minimalizuje ryzyko wystąpienia konfliktów oraz niepoprawnych stanów bazy danych.

Serializowalność:

Protokół 2PL gwarantuje, że wykonywanie transakcji jest serializowalne, czyli wyniki równoległych transakcji są takie same, jakby były wykonywane w pewnej kolejności sekwencyjnej.

Rodzaje 2PL

Ścisły 2PL (Strict 2PL):

Wszystkie blokady pisania (write locks) są przetrzymywane aż do zakończenia transakcji (commit lub abort).

Zapewnia, że żadna inna transakcja nie zobaczy niezakończonych zapisów innej transakcji, co pomaga w zapewnieniu właściwości trwałości (Durability) i izolacji (Isolation).

Silnie ścisły 2PL (Rigorous 2PL):

Wszystkie blokady (zarówno odczytu jak i zapisu) są przetrzymywane aż do zakończenia transakcji.

To najbezpieczniejsza wersja 2PL, zapewniająca maksymalną izolację między transakcjami.

1. Teoria informacji wg. C.E. Schannona.

Teoria informacji, stworzona przez Claude'a Elwooda Shannona w latach 40. XX wieku, to fundament współczesnej komunikacji i informatyki. Shannon, pracując dla Bell Labs, opublikował swój przełomowy artykuł "A Mathematical Theory of Communication" w 1948 roku. Oto kluczowe elementy jego teorii:

1. Model komunikacyjny

Shannon zaproponował abstrakcyjny model komunikacyjny, który składa się z następujących elementów:

Źródło informacji: generuje wiadomość, która ma być przekazana.

Nadawca (Transmitter): przekształca wiadomość w sygnał odpowiedni do transmisji.

Kanał: medium, przez które sygnał jest przesyłany. Może być podatne na zakłócenia.

Odbiorca (Receiver): przekształca odebrany sygnał z powrotem w wiadomość.

Odbiorca końcowy (Destination): końcowy adresat wiadomości.

2. Entropia

Entropia to miara niepewności związana z przypadkowym procesem źródła informacji. W kontekście teorii informacji . Entropia mierzy średnią ilość informacji (w bitach) produkowaną przez źródło informacji.

3. Szyfrowanie i dekodowanie

Shannon wprowadził pojęcie kodowania źródła i kodowania kanału:

Kodowanie źródła: proces przekształcania wiadomości do bardziej efektywnego formatu, redukując redundancję.

Kodowanie kanału: dodawanie nadmiarowości do sygnału, aby umożliwić detekcję i korekcję błędów powstałych podczas transmisji przez zakłócony kanał.

4. Pojemność kanału

Pojemność kanału (C) to maksymalna ilość informacji, która może być przesłana przez kanał bez błędów.

gdzie B to szerokość pasma kanału, a SNR (Signal-to-Noise Ratio) to stosunek mocy sygnału do mocy szumu.

5. Redundancja i wydajność

Teoria Shannona wprowadza również koncepcję redundancji w wiadomości oraz wydajności kodowania. W praktyce, systemy komunikacyjne starają się maksymalizować wydajność, minimalizując redundancję przy jednoczesnym zapewnieniu niezawodnej transmisji.

Zastosowania

Teoria informacji ma szerokie zastosowanie w wielu dziedzinach, takich jak:

Telekomunikacja: projektowanie efektywnych systemów przesyłu danych.

Kompresja danych: algorytmy kompresji, takie jak ZIP czy MP3, bazują na zasadach teorii informacji.

Kryptografia: bezpieczeństwo komunikacji poprzez szyfrowanie.

Informatyka: algorytmy i struktury danych.

Shannon położył podwaliny pod rozwój nowoczesnych technologii komunikacyjnych i informatycznych, które są fundamentem współczesnego świata cyfrowego. Jego teoria pozostaje kluczowym narzędziem w analizie i optymalizacji procesów przesyłu informacji.

1. Różnice między programowaniem strukturalnym i obiektowym.

Programowanie strukturalne i obiektowe to dwa różne paradygmaty programowania, które oferują odmienne podejścia do organizowania i zarządzania kodem. Oto główne różnice między nimi:

Programowanie Strukturalne

Podejście do struktury programu:

Skupia się na podziale programu na funkcje lub procedury.

Programy są zorganizowane w sposób hierarchiczny, gdzie dane są przetwarzane przez funkcje.

Modularność:

Modularność jest osiągana poprzez funkcje i procedury.

Każda funkcja wykonuje określone zadanie i może być wielokrotnie używana.

Przechowywanie danych:

Dane są zazwyczaj przechowywane w zmiennych globalnych lub przekazywane między funkcjami jako argumenty.

Brak jest koncepcji łączenia danych z funkcjami w jeden logiczny blok.

Przykłady języków:

C, Pascal, Fortran.

Zalety:

Prostota i przejrzystość dla mniejszych programów.

Łatwość analizy i debugowania dla niewielkich projektów.

Wady:

Trudności w zarządzaniu dużymi projektami.

Problemy z utrzymaniem kodu, szczególnie gdy projekt staje się bardziej złożony.

Słaba modularność, utrudniająca ponowne użycie kodu.

Programowanie Obiektowe

Podejście do struktury programu:

Skupia się na obiektach, które łączą dane (atrybuty) i zachowania (metody) w jedną całość.

Programy są zorganizowane wokół obiektów i ich interakcji.

Modularność:

Modularność jest osiągana poprzez klasy i obiekty.

Klasy definiują strukturę danych i funkcje, które na tych danych operują.

Przechowywanie danych:

Dane są przechowywane jako atrybuty obiektów.

Dane i funkcje, które na nich operują, są zgrupowane razem, co ułatwia zarządzanie nimi.

Przykłady języków:

C++, Java, Python, C#.

Zalety:

Lepsza modularność i możliwość ponownego użycia kodu.

Łatwiejsze zarządzanie dużymi projektami.

Możliwość modelowania rzeczywistych problemów w sposób bardziej intuicyjny.

Silna enkapsulacja, zwiększająca bezpieczeństwo danych.

Wady:

Większa złożoność początkowa i krzywa uczenia się.

Czasami mniej wydajne niż programowanie strukturalne dla prostych problemów.

Możliwość nadmiernego skomplikowania projektu przy niewłaściwym zastosowaniu.

Porównanie

Styl programowania:

Strukturalne: Funkcje/procedury.

Obiektowe: Klasy/obiekty.

Zarządzanie danymi:

Strukturalne: Zmienne globalne, argumenty funkcji.

Obiektowe: Atrybuty obiektów, metody.

Modularność:

Strukturalne: Funkcje/procedury.

Obiektowe: Klasy/obiekty.

Skalowalność:

Strukturalne: Trudniej zarządzać większymi projektami.

Obiektowe: Łatwiejsze zarządzanie większymi projektami.

Podsumowując, programowanie strukturalne jest bardziej intuicyjne dla mniejszych, prostych projektów, natomiast programowanie obiektowe oferuje lepszą modularność i skalowalność, co czyni je bardziej odpowiednim dla dużych, złożonych systemów.

1. Dziedzina i istota nauki jaką jest Sztuczna Inteligencja.

Dziedzina i Istota Sztucznej Inteligencji

Definicja i Podstawowe Pojęcia

Sztuczna Inteligencja (SI, ang. Artificial Intelligence, AI) jest dziedziną nauki i technologii, która koncentruje się na tworzeniu systemów zdolnych do wykonywania zadań, które normalnie wymagałyby ludzkiej inteligencji. W skrócie, AI to zdolność maszyn do naśladowania, naśladowania i naśladowania ludzkich zdolności poznawczych.

Główne Obszary Sztucznej Inteligencji

Uczenie Maszynowe (Machine Learning, ML): Jest to poddziedzina SI, która polega na tworzeniu algorytmów i modeli, które uczą się na podstawie danych. Modele te mogą być następnie używane do przewidywania lub podejmowania decyzji bez potrzeby programowania w sposób bezpośredni.

Uczenie Głębokie (Deep Learning, DL): Jest to zaawansowana forma uczenia maszynowego, która wykorzystuje sieci neuronowe o wielu warstwach. Dzięki tym sieciom możliwe jest analizowanie skomplikowanych danych, takich jak obrazy czy dźwięk, z wysoką precyzją.

Przetwarzanie Języka Naturalnego (Natural Language Processing, NLP): Ta dziedzina zajmuje się interakcją między komputerami a ludzkim językiem, umożliwiając komputerom rozumienie, interpretowanie i generowanie ludzkiego języka w sposób wartościowy.

Sztuczne Sieci Neuronowe (Artificial Neural Networks, ANN): Struktury inspirowane biologicznymi mózgami, które są podstawą uczenia głębokiego. Składają się z jednostek (neuronów) połączonych wagami, które są dostosowywane w trakcie procesu uczenia.

Systemy Ekspertowe: Programy komputerowe, które naśladują decyzje ekspertów ludzkich w określonych dziedzinach. Wykorzystują one bazy wiedzy i zasady wnioskowania do podejmowania decyzji.

Robotyka: Zastosowanie SI w tworzeniu autonomicznych robotów zdolnych do wykonywania zadań w świecie fizycznym. Robotyka często integruje techniki z zakresu widzenia komputerowego, przetwarzania sygnałów i automatyzacji.

Istota Sztucznej Inteligencji

Sztuczna Inteligencja zmienia sposób, w jaki ludzie żyją i pracują, przynosząc korzyści w wielu dziedzinach:

Automatyzacja: SI pozwala na automatyzację rutynowych i powtarzalnych zadań, co zwiększa efektywność i redukuje koszty operacyjne.

Analiza Danych: Umożliwia analizowanie ogromnych zbiorów danych w sposób szybki i dokładny, co jest kluczowe w dziedzinach takich jak finanse, medycyna, marketing, i wielu innych.

Rozwój Technologiczny: SI jest siłą napędową innowacji technologicznych, prowadząc do powstania nowych produktów i usług.

Personalizacja: Wykorzystanie SI pozwala na dostosowywanie produktów i usług do indywidualnych potrzeb użytkowników, co poprawia doświadczenia klientów.

Badania Naukowe: SI wspiera badania naukowe, pomagając w odkrywaniu wzorców i zależności w danych, które mogą prowadzić do nowych odkryć.

Wyzwania i Etyka

Mimo licznych korzyści, SI niesie ze sobą także wyzwania i pytania etyczne:

Prywatność: Wykorzystanie danych osobowych do trenowania modeli SI budzi obawy dotyczące prywatności i ochrony danych.

Zastępowanie Miejsc Pracy: Automatyzacja i robotyka mogą prowadzić do utraty miejsc pracy, co wymaga przemyślenia polityk zatrudnienia i przekwalifikowania pracowników.

Przedsądy i Dyskryminacja: Modele SI mogą nieświadomie wzmocnić istniejące przedsądy, jeśli są trenowane na stronniczych danych.

Bezpieczeństwo: Zwiększenie autonomii systemów SI rodzi pytania o ich bezpieczeństwo i niezawodność, zwłaszcza w kontekście krytycznych infrastruktur.

Podsumowując, sztuczna inteligencja jest dynamicznie rozwijającą się dziedziną, która ma ogromny potencjał do transformacji wielu aspektów życia codziennego. Jednakże, jej rozwój musi być zrównoważony odpowiedzialnym podejściem do etyki i wpływu społecznego.

16. Test Turinga: przykład/analiza testu/wniosek.

17. Definicja systemu eksperckiego oraz jego budowa.

18. Biometria jako dziedzina nauki i techniki.

19. Budowa „sztucznego neuronu” (model SN), funkcja aktywacji oraz proces uczenia.

20. Przykładowych struktur SSN.

21. Metodyka UML i podstawowe diagramy.

22. Modele cyklu życia oprogramowania.

23. Podstawowe aspekty bezpieczeństwa systemu informatycznego.

24. Metody obliczania kosztów budowy i wdrożenia systemu komputerowego.

25. Pojęcie języka maszynowego.

26. Scharakteryzuj wybrany współczesny język programowania uwzględniając jego zalety i ograniczenia.

27. Omów obszary zastosowania kryptologii w informatyce.

28. Scharakteryzuj znaczenie MFA w bezpieczeństwie systemów informatycznych.

29. Kanały i metody transmisji danych.

30. Metody zarządzania projektem.

31. Scharakteryzuj różnicę między AR i VR

32. Potencjalne obszary zastosowania technologii VR

33. Scharakteryzuj sposoby zabezpieczania infrastruktury komputerowejprzed czynnikami środowiskowymi

34. Na czym polega atak typu DDoS

35. Jaki jest scenariusz przebiegu ataku Phishingowego

36. Czym jest Spoofing i jak przebiega

37. Omów metody kontroli dostępu do danych w systemach komputerowych

38. Różnica między identyfikacją, autoryzacją i uwierzytelnianiem

39. Pojęcie i znaczenie luki dnia zerowego

40. Wyjaśnij pojęcie „Big Data”

41. Wyjaśnij pojęcie „internet rzeczy”

42. Czym jest „GIT” i do czego służy github?

43. Omów pojęcie „wieloplatformowości” oprogramowania

44. Wskaż i omów potencjalne wyzwania podczas tworzenia Polityki Bezpieczeństwa Informacyjnego

45. Kto jest odpowiedzialny za bezpieczeństwo danych i informacji w systemach informatycznych podmiotu?

46. Na czym polega różnica między „robakami”i„wirusami” komputerowymi

47. Scharakteryzuj architekturę systemu komputerowego typu „punkt-punkt”

48. Czym była centralna szyna danych?

49. Na czym polega różnica między komputerami binarnymi a komputerami kwantowymi?

50. Wyjaśnij znaczenie zegara procesora dla wydajności systemu komputerowego