Notatki z baz danych

Małgorzata Dymek

Spis treści

1 Podstawowe pojęcia

Baza danych - kolekcja powiązanych ze sobą danych.

- Reprezentuje pewne **aspekty świata rzeczywistego** (mini świat) lub obszar analizy, czasem dziedzinę problemu.
- Jest **logicznie spójnym zbiorem** danych o pewnym znaczeniu.
- Stworzona **w określonym celu**, dla pewnej grupy użytkowników.

Logika biznesowa - implementacja **procesów biznesowych**.

System Zarządzania Bazą Danych (SZBD) - system oprogramowania, który zapewnia:

- definiowanie, tworzenie, utrzymywanie bazy danych.
- operowanie na danych, kontrolę **spójności**, dbanie o **integralność**.
- odporność na błędy, **mechanizmy bezpieczeństwa**, kontrola dostępu.
- udostępnianie **metadanych**, dostępność przez różne interfejsy, komunikacja z innymi systemami.
- obsługa transakcji (ACID), sterowanie współbieżnością.

System bazy danych (SBD) = bazy + SZBD. Czasami zalicza się również aplikację użytkowników.

Relacyjne systemy baz danych

- **Niezależność** między danymi a programami.
- **Samoopisujący charakter SZBD** – zawiera **meta-dane**.
- Dostarczanie **wielu widoków** tych samych danych dla różnych użytkowników.
- Zapewnienie **integralności** danych.
- Współdzielenie danych, **przetwarzanie współbieżnych transakcji** wielu użytkowników.
- Wysoka **niezawodność i bezpieczeństwo**.
- Generalnie relacyjne systemy baz danych **źle się skalują poziomo**.

Bazy danych NoSQL

- Gromadzenie i **szybkie przetwarzanie** dużych ilości danych.
- Stosunkowo **łatwe skalowanie poziome**.
- Często: **struktura rozproszona**.
- Często: brak schematu określonego na poziomie samego systemu baz danych (ale schemat jest często narzucany przez aplikacje klienckie).
- Na ogół: **brak lub niepełna obsługa transakcji**.

1.1 Architektury systemów baz danych

1.1.1 Trójwarstwowa architektura ANSI-SPARC

- **Warstwa zewnętrzna** - dostarcza **niezależne, spersonalizowane widoki** użytkownikom. Użytkownicy nie muszą mieć dostępu do szczegółów.
- **Warstwa konceptualna** - dostarcza **jednolitego, globalnego widoku** danych i ich wzajemnych powiązań. Jest **niezależna od użytkowników**. Dostęp do niej mają zazwyczaj administratorzy.
- **Warstwa wewnętrzna** - opisuje jak dane są **fizycznie przechowywane**.

Zmiany wewnątrz warstwy niekoniecznie naruszają pozostałe.

1.1.2 Architektury komunikacyjne klient-serwer

- **Architektura dwuwarstwowa** - logika biznesowa w kliencie bądź serwerze bądź podzielona.
- **Architektura trójwarstwowa** - logika biznesowa w warstwie środkowej, na serwerze aplikacji.
- **Architektury wielowarstwowe** - wyspecjalizowane warstwy.

1.2 Modelowanie danych

- **Modele konceptualne** - umożliwiają opis konkretnego, **wybranego wycinka rzeczywistości** poprzez analizę obiektów i powiązań między tymi obiektami.
- **Modele implementacyjne** - organizują struktury danych (modele relacyjne, obiektowo-relacyjne, obiektowe, NoSQL).

1.2.1 Koncepcyjny model związków encji E/R

- **Encja - jednostka** odpowiadająca „obiektowi" (zbiorowi obiektów).
- **Atrybut** – **cecha** encji.
- **Związek - zależność** między encjami.

2 Relacyjne bazy danych

Systemy oparte o model relacyjny.

- Dane w **tabelach**. Tabela może przedstawiać **relację bazodanową**.
- **Wiersz (rekord)** - dane reprezentujące pewną **encję lub związek**. Powinny być **unikalne** wewnątrz tabeli.
- **Atrybuty - kolumny**. Są nazwane i mają określony logicznie niepodzielny typ danych związany z dziedziną atrybutu.
- **Skalar** - pojedyncza wartość, najmniejsza semantycznie jednostka danych. Są **atomowe w ramach modelu** relacyjnego. **Dziedzina** to **zbiór wartości skalarnych**.
- **Zmienna relacji - nazwany obiekt, którego wartość zmienia się w czasie**. **Wartość zmiennej** w danej chwili nazywa się **wartością relacji**.
- Relacyjna baza danych to **zestaw znormalizowanych relacji różnych stopni**.
- Relacyjny model ma **zastosowanie na zewnętrznym i pojęciowym poziomie**, nie na wewnętrznym.

2.1 Definicja Relacji wg Date'a

Relacja R (w znaczeniu **wartość relacji**) na zbiorze niekoniecznie różnych dziedzin $D_1, D_2, \ldots, D_n$ składa się z dwóch części: **nagłówka** (heading) i **treści** (body).

Nagłówek jest to ustalony **zbiór atrybutów**, a ściśle rzecz biorąc zbiór par $< nazwa\_atrybutu : nazwa\_dziedziny >$, takich, że każdy unikalny atrybut odpowiada dokładnie jednej wyjściowej dziedzinie. Atrybuty **nie są uporządkowane**.

Treść jest to **zbiór krotek**. Każda krotka jest zbiorem par $< nazwa\_atrybutu : wartosc\_atrybutu >$. W każdej takiej krotce jest jedna para dla każdego atrybutu z nagłówka. Krotki są **unikalne, nieuporządkowane** oraz **istnieje przynajmniej jeden klucz kandydujący**.

Tak zdefiniowana relacja jest też określana terminem **relacja bazodanowa**.

Liczebność relacji - liczba krotek w zbiorze.
Stopień relacji - ilość dziedzin.

2.2 Rodzaje relacji

- **Relacja nazwana** jest **nazwaną zmienną relacji** zdefiniowana w SZBD (CREATE TABLE/VIEW).
- **Relacja podstawowa** (autonomiczna) - wystarczająco ważna, by stanowić **nazwany niezależny byt** w bazie danych. "Przechowuje dane".

- **Relacja wyprowadzana** (pochodna) - relacja, którą można otrzymać ze zbioru nazwanych relacji za pomocą jakiegoś **wyrażenia relacyjnego**.
- **Perspektywa** (widok) - **nazwana relacja pochodna**.Są **wirtualnymi zmiennymi** relacji.
- **Migawka** (perspektywa znormalizowana, snapshot) - **rzeczywista nazwana relacja pochodna**.
- **Wynik zapytania** - **nietrwała, nienazwana relacja pochodna** powstająca w wyniku realizacji **zapytania**.
- **Wynik pośredni** - **nietrwała, nienazwana relacja pochodna**, która powstaje w wyniku realizacji **podzapytania**.

Predykat - rodzaj założenia pilnujący, żeby baza zawierała **dane zgodne z rzeczywistością**.

2.3 Klucze

Klucz kandydujący (potencjalny) w relacji R jest podzbiorem K (zbioru atrybutów relacji R), mającym:

- **Własność jednoznaczności** - żadne dwie różne krotki z R nie mają tej samej wartości dla atrybutów z K.
- **Własność nieredukowalności** - żaden właściwy podzbiór K nie ma własności jednoznaczności.

Nadklucz/zbiór identyfikujący - zbiór atrybutów posiadający **własność jednoznaczności**.

Klucz złożony - więcej niż jeden atrybut, **prosty** - jeden atrybut.

Relacja może mieć więcej niż jeden klucz kandydujący. Jeden z nich powinien być wybrany jako **klucz główny**. Reszta to **klucze alternatywne**.

2.4 Problem integralności referencyjnej

Reguła integralności referencyjnej - w bazie danych nie mogą występować żadne niedopasowane wartości kluczy obcych. Wchodzi ona w skład modelu relacyjnego danych. **Każdy stan** (wartości przechowywanych danych), **który nie spełnia tej reguły jest z definicji niepoprawny**.
Więzy referencyjne - warunki utrzymania integralności referencyjnej.

Sposoby **rozwiązania** problemu **stanów niepoprawnych**:

- **ON DELETE/UPDATE RESTRICT** - niedopuszczenie do utworzenia niepoprawnych wartości.
- **ON DELETE/UPDATE CASCADE** - kaskadowe kasowanie lub aktualizowanie wartości tak, by zapewnić integralność referencyjną.

2.5 Schematy relacji

Schemat relacji - zbiór atrybutów, ozn. $R(A_1, A_2, \ldots, A_n)$. Ewentualnie zbiór atrybutów i zbiór wszystkich reguł integralności danych.
Schemat relacyjnej bazy danych S jest **zbiorem schematów relacji**, $S = \{R_1, R_2, \ldots, R_m\}$ wraz ze zbiorem wszystkich **więzów integralności** danych.

2.6 Zależności funkcyjne

Wartość danej relacji - definicja:

Niech R wartość relacji, X i Y dowolne podzbiory atrybutów R. Mówimy, że **Y jest funkcyjnie zależny od X** (ozn. $X \rightarrow Y$) wtw gdy:
jeśli dwie krotki z R mają **takie same wartości wszystkich atrybutów z X**, to mają również **takie same wartości wszystkich atrybutów ze zbioru Y**.

Przykład: $\{PESEL\} \longrightarrow \{Nazwisko, Imie\}$

Rodzaje zależności funkcyjnych
Niech $A = \{A_1, A_2, \ldots, A_n\}$, $B = \{B_1, B_2, \ldots, B_m\}$ będą zbiorami atrybutów pewnej relacji.
Zależność funkcyjna jest:

- **trywialna**, jeśli $B \subset A$.
- **nietrywialna**, jeśli $\exists b \in B : b \notin A$.

- **całkowicie nietrywialna**, jeśli $A \cap B = \emptyset$.

Zależność funkcyjna pojedynczego atrybutu od zbioru atrybutów - **nietrywialna**, jeśli atrybut z prawej strony nie należy do zbioru atrybutów z lewej strony. W zależności funkcyjnej zbiór wartości atrybutów z prawej strony jest zawsze zbiorem jednoelementowym.

2.7 Zależności wielowartościowe

Mówimy, że zbiór atrybutów $Y = \{Y_1, \ldots, Y_n\}$ schematu relacji R jest **wielowartościowo zależny** ($X \twoheadrightarrow Y$) od zbioru atrybutów $X = \{X_1, \ldots, X_m\}$, jeśli dla każdej pary krotek t i u, które mają takie same wartości atrybutów ze zbioru X, można znaleźć w tej instancji relacji krotkę w, której składowe mają wartości równe:

- wartościom atrybutów ze zbioru X w krotkach t oraz u,
- wartościom atrybutów ze zbioru Y krotki t,
- wartościom tych składowych krotki u, które nie należą ani do X, ani do Y.

Niech X, Y, Z będą niepustymi zbiorami atrybutów, niech schemat relacji R będzie sumą tych zbiorów. Zależność $X \twoheadrightarrow Y$ zachodzi wtw, gdy zachodzi również $X \twoheadrightarrow Z$. ($X \twoheadrightarrow Y|Z$)

Zależność wielowartościową X -» Y określa się jako **trywialną**, jeśli:

- $Y \subset X$,
- $X \cup Y$ zawiera wszystkie atrybuty relacji.

Każda zależność funkcyjna jest zależnością wielowartościową.

Twierdzenie Fagina
Niech X, Y, Z będą niepustymi zbiorami atrybutów, niech schemat relacji R będzie sumą tych zbiorów. Wówczas relacja jest **równa złączeniu swoich rzutów** na {X, Y} oraz {X, Z} wtw, gdy w relacji **zachodzi zależność wielowartościowa** $X \twoheadrightarrow Y|Z$.

3 Projektowanie tabel

Analiza rzeczywistości → Model związków encji ER → Diagram związków encji ERD

Zasady projektowania tabel

- Każda **tabela** (relacja) ma **jednoznaczną nazwę**. Porządek w tabeli nie jest istotny. Powinna przechowywać informacje o **obiektach jednego rodzaju**.
- Każda **kolumna** (atrybut) ma **jednoznaczną nazwę, unikalną** w obszarze tabeli. Zawiera wartości **skalarne** jednego określonego **typu**.
- **Wiersze** są **unikalne**.
- W tabeli powinien istnieć (**klucz główny**).
- Unikanie redundancji.

Anomalia wstawiania - nie można wprowadzić nowej wartości, bo nie istnieje inna powiązana z nią (np. nowego działu bez pracownika, jeśli informacja o działach jest w tabeli pracownicy).
Anomalia usuwania - usunięcie jakiejś danej powoduje usunięcie innej (usunięcie wszystkich pracowników działu usuwa dział).

3.1 Dekompozycja relacji

Rodzielenie tabeli opisującej obiekty wielu rodzajów na pojedyncze tabele obiektów jednego rodzaju.

Niech relacja R ma zbiór atrybutów $\{A_1, \ldots, A_n\}$. Relację R dekomponujemy na relację S o schemacie $\{B_1, \ldots, B_m\}$ oraz relację T o schemacie $\{C_1, \ldots, C_k\}$ tak, by spełnione były następujące zasady:

- $\{A_1, \ldots, A_n\} = \{B_1, \ldots, B_m\} \cup \{C_1, \ldots, C_k\}$.
- Krotki S powstają przez rzutowanie wszystkich krotek relacji R na zbiór atrybutów $\{B_1, \ldots, B_m\}$.

- Analogicznie krotki T na zbiór atrybutów $\{C_1, \ldots, C_k\}$

Odzyskiwanie danych po dekompozycji
Chcemy, by w wyniku dekompozycji była możliwość takiego połączenia krotek powstałych w wyniku rzutowania, aby uzyskany zbiór krotek zawierał wszystkie i tylko te krotki, które należały do relacji przed dekompozycją.
Jeśli mamy $\{A, B, C\}$ i $\{A \to B\}$, $\{B \to C\}$ zależności funkcyjne to odtworzymy, jeżeli tylko $A \to B$ to niekoniecznie.

4 Normalizacja, postaci normalne

Poziomy normalizacji relacji:

- 1NF
- 2NF
- 3NF
- BCNF
- 4NF
- 5NF
- DKNF

W praktyce w większości przypadków doprowadzenie do 3NF usuwa redundancję i relacja już jest w wszystkich 'wyższych' postaciach. Absolutnym minimum jest 2PN lub 3PN.
Są systemy z nieznormalizowanymi tabelami, np. systemu analityczne, gdzie redundancja nie jest tak niekorzystna, ważniejsze są jakieś zestawienia danych, o spójność dbają systemy relacyjne (z normalizacją) dostarczające te dane.

4.1 Pierwsza postać normalna

- Krotki są **unikalne, nieuporządkowane** oraz **istnieje przynajmniej jeden klucz kandydujący**.
- Atrybuty **nie są uporządkowane**, są **atomowe**.
- Wartości atrybutów są skalarne.
- Brak atrybutów cyklicznych, ew. gdy jesteśmy pewni dokładnej ilości danego atrybutu.

Relację spełniającą taki warunek nazywa się relacją **znormalizowaną**.

4.2 Druga postać normalna

Każdy atrybut wtórny jest w pełni zależny funkcyjnie od klucza kandydującego. Żaden niekluczowy (wtórny) atrybut nie jest zależny funkcyjnie od żadnego podzbioru właściwego klucza kandydującego.

Jeśli **wszystkie klucze** kandydujące są **proste** to relacja jest w **2PN**.

4.3 Trzecia postać normalna

- Dla każdej nietrywialnej zależności funkcyjnej $\{A_1, \ldots, A_n\} \to \{B\}$ zbiór atrybutów $\{A_1, \ldots, A_n\}$ jest nadkluczem lub atrybut B jest elementem pewnego klucza kandydującego.
- **Żaden atrybut niekluczowy nie jest zależny funkcyjnie od innego atrybutu niekluczowego**.

Dekompozycja do 3PN

- Szukamy wszystkich nietrywialnych, całkowitych zależności funkcyjnych $\{A_1, \ldots, A_n\} \to \{B_i\}$, które naruszają warunek trzeciej postaci normalnej.
- Załóżmy, że otrzymujemy zależność (*)$\{A_1, \ldots, A_n\} \to \{B_1, \ldots, B_m\}$. Dzielimy schemat relacji na dwa nierozłączne podzbiory:
  - Pierwszy, zawierający wszystkie atrybuty występujące w zależności (*).
  - Drugi, zawierający atrybuty z lewej strony rozważanej zależności (*) oraz atrybuty nie występujące ani z lewej ani z prawej strony tej zależności.

4.4 Postać normalna Boyce'a-Codda PNBC, BCNF

- Dla każdej całkowicie nietrywialnej zależności funkcyjnej $A \rightarrow B$ zbiór **A jest nadkluczem**.
- Każda relacja binarna jest w BCNF.

Dekompozycja do postaci BCNF - jak dekompozycja do 3PN. **Nie zawsze zachowuje zależności funkcyjne**.

4.5 Czwarta postać normalna

Relacja jest w 4PN, jeśli dla każdej nietrywialnej zależności wielowartościowej X -» Y **zbiór X zawiera klucz relacji**. **Dekompozycja do 4PN analogicznie**.

4.6 Piąta postać normalna

4.6.1 Warunek 3D (3-rozkładalności)

Rozważmy relację R o schemacie $\{P, C, W\}$. Jeśli wartość p_1 wiąże się z c_1, a c_1 wiąże się z w_1 oraz w_1 wiąże się z p_1, to p_1, c_1 i w_1 muszą występować w tym samym wierszu.
Uogólnienie do n-rozkładalności – relacja jest n-rozkładalna, gdy spełnia kilka takich cyklicznych warunków.

Warunek 3D jest spełniony wtw, gdy **relacja jest równa złączeniu trzech pewnych swoich rzutów**.

4.6.2 Zależność złączenia

Niech R będzie relacją, {A, B, . . . , Z} będą dowolnymi podzbiorami właściwymi zbioru atrybutów R. Mówimy, że relacja **R spełnia zależność złączenia** ozn. *(A, B, ..., Z) wtw, gdy **R jest równa złączeniu naturalnemu swoich rzutów na {A, B, ..., Z}**.

Zależność złączenia *(A, B, ..., Z) w relacji **wynika z zależności atrybutów schematu tej relacji od klucza** wtw, gdy **złączenia** naturalne rzutów są **realizowane względem** (pewnych) **kluczy** kandydujących relacji (**jest implikowana kluczami**).

4.6.3 Piąta postać normalna

Mówimy, że relacja jest w 5PN wtw, gdy **każda zależność złączenia w relacji jest implikowana kluczami kandydującymi** tej relacji.
Fagin udowodnił, że dowolna relacja może być bezstratnie zdekomponowana na równoważny zestaw relacji w piątej postaci normalnej (**postać 5NF jest zawsze osiągalna**).

4.7 Postać normalna klucza dziedziny DKNF

Wszystkie więzy i zależności, które powinny być zachowane w poprawnych stanach relacji, mogą być wymuszane przez wymuszanie więzów domenowych oraz więzów kluczy na relacji. Tylko związki wyrażane za pomocą kluczy obcych.
W praktyce użyteczność DKNF jest problematyczna, gdyż określenie w ten sposób różnych więzów może się okazać bardzo trudne.

4.8 Denormalizacja

W pewnych przypadkach projektant godzi się na doprowadzenie np. tylko do 2PN, mimo że występuje nadal redundancja. Nazywa się to **denormalizacją**. Może być korzystna ona z uwagi na **większą wydajność wyszukiwania**, bo złączenie tabel może być kosztowne. Jednak i tak nie powinno się tego robić.

5 Transakcje

5.1 Wymagania (ACID)

- **Atomicity** (niepodzielność) transakcja jest wykonywana **w całości albo wcale**.
- **Consistency** (spójność) po zakończeniu transakcji baza musi być w stanie spójnym, tzn muszą być **zachowane wszystkie więzy** narzucone na dane.
- **Isolation** (izolacja) wykonywana w **izolacji od innych** transakcji.
- **Durability** (trwałość) po zakończeniu transacji jej **efekty** muszą być **trwałe** w systemie nawet w przypadku awarii.

5.2 Stany transakcji

- **Aktywna** – podczas wykonywania operacji.
- **Częściowo zatwierdzona** – ostatnia operacja transakcji została wykonana. Teraz protokół zatwierdzania musi zapewnić **trwałość zmian**. Ew. sprawdzenie czy może zostać zatwierdzona. Jeśli może, to osiągnęła swój **punkt zatwierdzenia**.
- **Nieudana** – wykonywanie transakcji **nie może być kontynuowane**.
- **Przerwana, wycofana** – baza jest **odtworzona** do stanu sprzed rozpoczęcia transakcji.
- **Zatwierdzona** – wszelkie **zmiany** wykonane przez transakcje muszą być **trwałe**. Zatwierdzonej transakcji **nie można już wycofać** (ew. **transakcja kompensująca**).

5.3 Implementacja atomowości i trwałości

Strategie zarządców buforów

- **Fix** - nie może być zapisów przed końcem transakcji.
- **No-Fix** - mogą być zapisy przed końcem transakcji. Synchronizacja może być realizowana **cyklicznie** w ramach procesu zwanego **punktem kontrolnym**.

- **Flush** - synchronizacja zmienionych bloków po zakończeniu transakcji.
- **No-Flush** - brak obowiązku synchronizowania zmienionych bloków na koniec transakcji. Synchronizacja może być wykonana później.

W większości relacyjnych systemów baz danych, wykorzystujących tzw. **dzienniki transakcji** stosowana jest **strategia No-Fix/No-Flush**.

5.4 Dziennik transakcji

- Zawiera **informacje o wszystkich wprowadzonych przez transakcje zmianach**.
- Mogą pojawić się informacje o operacji kompensującej, dane dotyczące operacji odczytu.
- Transakcja nie zostaje uznana za zakończoną, dopóki fizycznie na dysku w pliku dziennika nie znajdą się wpisy opisujące wszystkie przeprowadzone przez transakcję zmiany oraz informacja o zatwierdzeniu transakcji.
- W przypadku przerwania transakcji, dzięki informacjom z dziennika można **wycofać zmiany wykonane przez transakcję**.

5.5 Odtwarzanie systemu

Proces **odtwarzania bazy** (recovery) po awarii przy strategii **No-fix/No-Flush**

- **redo**
  Należy **wycofać wszystkie zmiany wprowadzone przez transakcje**, które w momencie awarii **jeszcze się nie zakończyły** (No-Fix).
- **undo** Należy **powtórzyć te operacje, których efekty nie zostały jeszcze trwale zapisane na dysku**, mimo, że **transakcja została zatwierdzona** (No-Flush).

W przypadku awarii dysków z danymi (bez utraty dysku z dziennikiem transakcji) należy najpierw **przywrócić pliki z kopii zapasowych** (restore). Potem należy wykonać recovery.

6 Współbieżność

6.1 Harmonogramy

Harmonogram - inaczej historia S zbioru n transakcji $T_1, \ldots, T_n$ jest takim ciągiem wszystkich operacji transakcji, że dla każdej transakcji $T_i \in S$ operacje tej transakcji w S muszą występować w takiej samej kolejności, w jakiej występują w T_i. Operacje z różnych transakcji mogą się przeplatać.

- **Harmonogram szeregowy** - operacje każdej transakcji są wykonywane kolejno, **bez przeplatania** operacji z różnych transakcji. Są **nieefektywne**.
- **Harmonogram szeregowalny** - jego **wpływ** na stan bazy danych jest taki sam jak pewnego harmonogramu szeregowego.

- **Harmonogram nieszeregowy** - harmonogram, który nie jest sekwencyjny.

6.2 Poziomy izolacji transakcji

- Read Uncommitted
- Read Commited (domyślny)
- Repeatable Read
- Serializable
- Read Committed Snapshot
- Snapshot

Poziom izolacji L_1 jest **słabszy** niż poziom izolacji L_2 (analogicznie mocniejszy), co oznaczamy $L_1 \ll L_2$ jeśli wszystkie nieszeregowalne harmonogramy, które są zgodne z L_2 są również zgodne z L_1 a istnieje przynajmniej jeden zgodny z L_1, ale nie zgodny z L_2.
Dwa poziomy izolacji L_1 i L_2 są **równoważne**, co oznaczamy $L_1 == L_2$ jeśli zbiory nieszeregowanych harmonogramów zgodnych odpowiednio z L_1 i L_2 są identyczne.
Dwa poziomy izolacji transakcji są **nieporównywalne**, co oznaczamy $L_1 \gg\ll L_2$, jeśli każdy z tych poziomów dopuszcza pewien harmonogram, którego nie dopuszcza drugi poziom.

Dwie operacje są w **stanie konfliktu**, jeśli spełnione są warunki:

- Należą do **różnych transakcji**,
- Uzyskują dostęp do **tego samego elementu**,
- **Przynajmniej jedna** z operacji jest **operacją zapisu** elementu.

Harmonogram pełny

- harmonogram, zawierający **wszystkie operacje z transakcji składowych**,
- dla dowolnych dwóch **operacji**, które są w **konflikcie**, jedna z nich musi **poprzedzać** drugą w harmonogramie

Mówimy, że dwa harmonogramy są **równoważne konfliktowo**, jeżeli **kolejność** wszystkich **operacji konfliktowych** jest w nich **taka sama**. **Wyniki** harmonogramów równoważnych konfliktowo są takie same.

6.3 Szeregowalność konfliktowa

Harmonogram S jest **szeregowalny konfliktowo**, jeżeli jest on **konfliktowo równoważny** pewnemu harmonogramowi **szeregowemu** S'. Możemy zmieniać kolejność niekonfliktowych operacji w S.
Szeregowalność konfliktowa stanowi **warunek wystarczający** zachowania spójności danych.

Graf poprzedzania (precedence graph): graf skierowany G (V, E), V - zbiór wierzchołków, każdy wierzchołek reprezentuje jedną transakcję w harmonogramie; E - zbiór krawędzi.
Krawędź $T_i \rightarrow T_j$ jest tworzona, jeśli jedna z operacji z T_i występuje w rozważanym harmonogramie przed pewną operacją konfliktową z transakcji T_j.
Graf poprzedzania **nie zawiera cykli** wtw, gdy harmonogram jest **szeregowalny konfliktowo**.

6.4 Szeregowalność perpektywiczna

Równoważność perspektywiczna harmonogramów
Harmonogram S i S' zawierają te same instrukcje i dla każdego elementu danych Q:

- Jeśli w S T_k jest transakcją, która w harmonogramie czyta Q jako pierwsza, to w S' T_k musi być transakcją, która czyta Q jako pierwsza.
- Jeśli w S T_i czyta Q zapisany przez T_j, to w S' T_i czyta Q zapisany przez T_j.
- Jeśli w S T_m jest ostatnią transakcją, która zapisuje Q, to w S' T_m jest ostatnią transakcją, która zapisuje Q.

Harmonogram szeregowalny perspektywicznie jest harmonogramem **równoważnym perspektywicznie** jakiemuś harmonogramowi szeregowemu.

Szeregowalność perspektywiczna to **szeregowalność konfliktowa z dodatkowymi ograniczeniami** operacji zapisów:

- każda operacja w(x) jest poprzedzona r(x),
- wartość zapisana przez operację w(x) **jest nie stałą funkcją** tylko wartości r(x).

6.5 Inne rodzaje harmonogramów

- **Harmonogramy odtwarzalne**
  Dla każdej pary transakcji: T_i , T_k jeśli T_k czyta dane zapisane przez T_i , to T_i **musi zostać zatwierdzona zanim zostanie zatwierdzona** T_k. Jeśli T_i zostanie przerwana, T_k można też przerwać i wycofać.

- **Harmonogramy bezkaskadowe**
  Dla każdej pary transakcji T_i , T_k jeśli T_k czyta dane zapisane przez T_i , wówczas T_i musi zostać **zatwierdzona przed tą operacją odczytu** z transakcji T_k. Wycofanie jednej transakcji powoduje konieczność wycofania innych (**kaskadowe wycofanie**) - duży koszt.

- **Harmonogram ścisły**
  Transakcje **nie mogą odczytywać ani zapisywać elementu**, aż zostanie **zakończona** ostatnia **transakcja**, która go **zapisała**. Anulowanie transakcji - odtworzenie obrazu pierwotnego.

6.6 Transakcje debit-credit

Aktualizowanie danych przez dodawanie lub usuwanie wartości może mieć **poprawne harmonogramy szeregowalne, nie szeregowalne konfliktowo**, bo dodawanie i odejmowanie jest przemienne.

7 Poziomy izolacji transakcji

Problemy:

- **P0 (Dirty Write)**: T_1 modyfikuje daną. T_2 modyfikuje tą samą daną zanim T_1 zostanie zaakceptowana (lub anulowana).
- **A1 (Dirty Read)**: T_1 modyfikuje daną. T_2 czyta daną zanim T_1 zostaje zaakceptowana. Jeżeli T_1 zostanie wycofana, T_2 ma odczyt danej która "nigdy nie istniała".
- **A2 (Non-repeatable or Fuzzy Read)**: T_1 czyta daną. Następnie T_2 modyfikuje albo usuwa tą daną i zostaje zatwierdzona. Gdy T_1 próbuje powtórzyć odczyt, dostaje inną wartość albo okazuje się, że dana została usunięta.
- **A3 (Phantom)**: T_1 odczytuje zestaw danych zaspokajających klauzulę WHERE. Następnie T_2 dodaje rekordy które spełniają tą klauzulę i zostaje zaakceptowana. GDY T_1 próbuje powtórzyć odczyt dostaje inny zestaw danych.
- **P4 Lost update**: T_1 odczytuje daną i wylicza nową wartość. T_2 odczytuje daną i wylicza nową wartość. T_1 zapisuje wartość i zostaje zaakceptowana, T_2 nadpisuje tą wartość i zostaje zaakceptowana.

A - oryginalna definicja, P - rozszerzona.

P0: w1[x]...w2[x]...((c1 lub a1) i (c2 lub a2) w dowolnej kolejności)

A1: w1[x]...r2[x]...(a1 i c2 w dowolnej kolejności)
P1: w1[x]...r2[x]...((c1 lub a1) i (c2 lub a2) w dowolnej kolejności)

A2: r1[x]...w2[x]...c2...r1[x]...c1
P2: r1[x]...w2[x]...((c1 lub a1) i (c2 lub a2) w dowolnej kolejności)

A3: r1[P]...w2[y in P]...c2...r1[P]...c1
P3: r1[P]...w2[y in P]...((c1 lub a1) i (c2 lub a2) w dowolnej kolejności)

P4: r1[x]...w2[x]... c2 ...w1[x]...c1

Poziom izolacji	P0 Dirty Write	P1 Dirty Read	P2 Non-repeatable/Fuzzy Read	P3 Phantoms
READ UNCOMMITED	NIE	TAK	TAK	TAK
READ COMMITED	NIE	NIE	TAK	TAK
REPEATABLE READ	NIE	NIE	NIE	TAK
SERIAZABLE	NIE	NIE	NIE	NIE

8 Izolacja dla systemów z blokowaniem

Blokada (lock) jest **zmienną związaną z elementem danych**, która opisuje **stan** tego elementu pod względem **możliwości działań**, jakie mogą być na nim w danej chwili wykonywane.

Dobrze sformowane zapisy - przed zapisem **wymagane jest założenie blokady X** (ew. predykatowej).
Dobrze sformowane odczyty - do operacji odczytu **wymagane jest założenie blokady S** (ew. predykatowej).

Poziom izolacji	P0	P1	P2	P3	Blokady X	Blokady S
Locking READ UNCOMMITED	NIE	TAK	TAK	TAK	długie	nie
Locking READ COMMITED	NIE	NIE	TAK	TAK	długie	krótkie
Locking REPEATABLE READ	NIE	NIE	NIE	TAK	długie	długie
Locking SERIAZABLE	NIE	NIE	NIE	NIE	długie	długie predykatowe

9 Problemy współbieżności

9.1 Protokół dwufazowego blokowanie 2PL

Rygorystyczny protokół 2PL

- **Dostęp** do wiersza wymaga **blokady S**.
- **Modyfikacja** wiersza wymaga **blokady X**.
- Jeśli żądanie blokady zostanie odrzucone ze względu na inną blokadę transakcja **przechodzi w stan oczekiwania** do momentu, **aż blokada** konfliktowa **zdjęta**. System powinien nie dopuścić do zagłodzenia.
- Blokady S i X **długie**.

Wersja podstawowa 2PL

- Przed rozpoczęciem działania na obiekcie wymagane założenie **odpowiedniej blokady**.
- Po zwolnieniu blokady transakcja już **nie może zakładać żadnej nowej blokady** na **jakikolwiek obiekt**.

Wersja ścisła 2PL

- jak rygorystyczna, ale **blokady S krótkie**.

Konserwatywny protokoł 2PL - gwarancja braku zakleszczeń.

- określenie **zbioru elementów**, które transakcja **chce blokować**.
- jeśli można **zablokować wszystkie**, to elementy są blokowane,
- jeśli nie, to po pewnym czasie **próba jest ponawiana**.

Jeśli wszystkie transakcje spełniają którąś wersję protokołu dwufazowego blokowania, to wszystkie (przeplatane) harmonogramy są szeregowalne konfliktowo.

9.2 Zakleszczenia

Zakleszczenie (deadlock) występuje wówczas, gdy **każda transakcja oczekuje** na pewien element zablokowany przez inną transakcję.

9.2.1 Protokoły zapobiegania zakleszczeniom

Protokoły tego typu są rzadko wykorzystywane w praktyce.

- Zarządca transakcji sprawdza, czy transakcja może spowodować zakleszczenie.
- Jeśli **tak**, to **transakcja nie jest wykonywana**. Po pewnym czasie następuje **ponowna próba** wykonania transakcji.
- Jeśli **nie**, to jest **wykonywana**.

Np. **konserwatywny protokół 2PL**.

9.2.2 Protokoły unikania zakleszczeń

- **Uporządkowanie wszystkich elementów**. Każda transakcja wymagająca dostępu do kilku elementów realizowała dostępy i **blokowała elementy w tej samej kolejności**.
- Wykorzystanie **znaczników czasu**, wersja **Czekaj-kończ (Faworyzowanie młodszej)**
  Załóżmy, że transakcja T_i próbuje zablokować element danych, który jest już blokowany przez inną transakcję T_k.
  - Jeżeli $TS(T_i) < TS(T_k)$, to T_i czeka.
  - wpp T_i jest anulowana i ponawiana później z tym samym znacznikiem czasu.
- Wykorzystanie **znaczników czasu**, wersja **Zakończ-czekaj (Faworyzowanie starszej)**
  Załóżmy, że transakcja T_i próbuje zablokować element danych, który jest już blokowany przez inną transakcję T_k.
  - Jeżeli $TS(T_i) < TS(T_k)$ to T_k zostaje anulowana i ponawiana później z tym samym znacznikiem czasu.
  - wpp T_i czeka
- Strategia **bez oczekiwania**
  Jeśli transakcja nie może założyć blokady, to jest **wycofywana** (i potem wznawiana) **bez sprawdzania**, czy **zakleszczenie** rzeczywiście mogłoby wystąpić, czy nie.
- Strategia **oczekiwania ostrożnego**
  Załóżmy, że transakcja T_i próbuje zablokować element danych, który jest już blokowany przez inną transakcję T_k. Jeśli T_k **nie czeka** na pewien inny zablokowany element, to transakcja T_i **będzie czekać**, wpp transakcja T_i jest **anulowana**.

Strategie Czekaj-kończ i Zakończ-czekaj nie powodują zagłodzenia.

9.2.3 Wykrywanie zakleszczeń

- Skonstruowanie **grafu oczekiwania**. Wierzchołki - wykonywane transakcje. Krawędzie $T_i \rightarrow T_k$ - T_i próbuje zablokować element danych, który jest blokowany przez T_k. Po zwolnieniu blokady krawędź jest usuwana.
  Cykl w grafie oznacza zakleszczenie.
  Wybór ofiary – na ofiarę można wybrać transakcję młodszą, lub tę, która mniej zmodyfikowała.
- Użycie **limitów czasu** - jeśli transakcja **czeka** na zasób **dłużej niż przyjęta wartość** progowa, to system przyjmuje, że uległa zakleszczeniu i **anuluje** ją.

9.3 Blokada U

Blokada U - gdy element danych **jest odczytywany i być może będzie potem aktualizowany**. (Podnoszenie S na X może prowadzić do zakleszczenia).

przyznana ↓ / przyznawana →	S	X	U
S	tak	nie	tak
X	nie	nie	nie
U	tak/nie	nie	nie

Blokada U jest zakładana **przy odczycie**, przed wykonaniem aktualizacji **jest konwertowana do X**.

9.4 Poziom izolacji Cursor Stability

Rozszerzenie sposobu blokowania w poziomie Locking READ COMMITTED. Dodaje się **operację rc (czytaj kursor, pobierz wiersz)** dla instrukcji FETCH, blokada (S lub nowy typ blokady do odczytu **scroll lock**) będzie utrzymywana **do chwili przejścia** do innego wiersza lub do **zamknięcia** kursora.

Aktualizacja wiersza przez kursor – operacja **wc** powoduje założenie na ten wiersz **długiej blokady X**.

Dla operacji na kursorze można zdefiniować odmianę problemu P4:
P4C: rc1[x]...w2[x]...c2...wc1[x]...c1

Poziom izolacji Cursor Stability **eliminuje P4C**, w2[x] będzie wstrzymane do zdjęcia blokady (S, scroll lock) przez przejście do innego wiersza lub zamknięcie kursora.
Uwaga: READ COMMITTED « Cursor Stability « REPEATABLE READ

9.5 Poziom izolacji Snapshot i podobne

Transakcja czyta dane (zatwierdzone) z **chwili swojego początku**, Start-Timestamp.

- **Snapshot isolation (MS SQL Server SNAPSHOT)**
  - Podobna do propozycji 1, ale są stosowane **blokady do zapisu**, ponadto przy każdym zapisie transakcja wykonuje podobne sprawdzenie jak wykonywane w propozycji 1 na końcu transakcji.
  - Przechowywane są **różne wersje danych**. Transakcja **odczytuje dane aktualne** w momencie rozpoczęcia transakcji.
  - **Nie ma blokad do odczytu**, operacja odczytu nie blokuje operacji zapisu ani innych operacji odczytu. Są jednak stosowane **długie blokady wyłączne do zapisu**.
  - Transakcja T_1 przy każdym zapisie sprawdza, czy istnieje inna transakcja T_2, która zmodyfikowała dane zapisywane i zakończyła się zatwierdzeniem. Jeśli istnieje, to T_1 jest wycofywana.
  - Stosowaną tu zasadę można określić jako **First-writer-wins**.
- **Read Committed Snapshot (Oralce READ COMMITTED)**
  - Podobna do propozycji 2, ale operacja odczytu **czyta ostatnią zatwierdzoną wartość** elementu danych (niekoniecznie sprzed początku transakcji).
  - Jednak w przyjętej implementacji wiersze kursora czytane są w momencie otwarcia kursora, a nie w momencie odczytu wiersza.

Poziom izolacji Snapshot **nie gwarantuje szeregowalności konfliktowej** harmonogramów.

9.6 A5 Data Item Constraint Violation

Załóżmy, że na elementy danych x oraz y narzucono pewne **ograniczenie C()**. Każda transakcja z osobna dba o spełnienie C().

A5A Skrzywiony odczyt (Read Skew)
T_1 odczytuje x, potem inna transakcja T_2 aktualizuje x oraz y do nowych wartości i zostaje zatwierdzona. Jeśli następnie T_1 odczyta y, to będzie miała niespójny obraz danych.
A5A: r1[x]...w2[x]...w2[y]...c2...r1[y]...(c1 or a1)

A5B Skrzywiony zapis (Write Skew)
T_1 odczytuje x (ew. odczytuje też y). Następnie inna transakcja T_2 odczytuje y (ew. odczytuje też x). Następnie T_1 zapisuje y a T_2 zapisuje x i obydwie zostają zatwierdzone. Ostatnie cztery operacje mogą być zrealizowane w dowolnej (sensownej) kolejności. Każda transakcja przy zapisie dba o spełnienie ograniczenia C(), jednak w wyniku przeplatanego wykonania ograniczenie C() może nie być spełnione po zatwierdzeniu obydwu transakcji.
A5B: r1[x]...r2[y]...(w1[y] w2[x] c1 i c2 w dowolnej sensownej kolejności)

A5A oraz A5B nie wystąpią w harmonogramach, w których wykluczony jest P2.

	P0	P1	A3	P3	A5A	A5B
Snapshot Isolation	nie	nie	nie	tak	nie	tak
Read Commited					tak	
Locking Repeatable Read				tak		nie

REPEATABLE READ »« Snapshot Isolation

10 Techniki wielowersyjne sterowania współbieżnością

10.1 Technika wielowersyjna oparta na znacznikach czasu

Dla każdej wersji X_i elementu X przechowywane są dwa znaczniki czasu TS_odczytu(X_i), TS_zapisu(X_i).

- Jeśli transakcja T **może wykonać operację** zapisz_element(X), tworzona jest nowa wersja X_j elementu X i TS_odczytu(X_j) oraz TS_zapisu(X_j) ustawia się na TS(T).
- Jeśli transakcja T czyta X i **odczytuje wartość** wersji X_i, wartość $TS\_odczytu(X_i) = max\{TS\_odczytu(X_i), TS(T)\}$.

- Jeżeli transakcja T wykonuje operację **zapisz_element(X)** i dla wersji i-tej elementu X o maksymalnym znaczniku takim że $TS\_zapisu(X_i) <= TS(T)$, zachodzi:
  - $TS\_odczytu(X_i) > TS(T)$, wówczas transakcja T zostaje **wycofana** i potem będzie uruchomiona z **nowym znacznikiem czasowym**.
  - $TS\_zapisu(X_i) = TS(T)$, to wersja X_i jest **modyfikowana**.
  - $TS\_zapisu(X_i) < TS(T)$, wówczas tworzona jest **nowa wersja** X_j elementu X z wartościami $TS\_odczytu(X_j) = TS\_zapisu(X_j) = TS(T)$.
- Jeżeli transakcja T wykonuje operację **odczytaj_element(X)**, wówczas odnajdywana jest wersja i-ta elementu X o maksymalnym znaczniku takim, że $TS_zapisu(X_i) <= TS(T)$. Wartość X_i jest przekazywana do transakcji T i wartość $TS\_odczytu(X_i) = max\{TS(T), TS\_odczytu(X_i)\}$.

10.2 Technika odczytu spójnych wersji czasowych

Tzw. **technika wielowersyjna z użyciem blokad do zapisu typu X lub U**.

- Przechowywane są **różne wersje** danych. Transakcja odczytuje dane **aktualne w momencie rozpoczęcia** transakcji.
- **Nie ma blokad do odczytu** (S).
- **Długotrwałe** blokady **wyłączne** do **zapisu**.
- Transakcja T_1 **przy każdym zapisie** sprawdza czy istnieje inna transakcja T_2, która **zmodyfikowała** dane zapisywane i **zakończyła się** zatwierdzeniem. Jeśli istnieje taka transakcja T_2, transakcja T_1 jest **wycofywana**. Stosowaną tu zasadę można określić jako **First-writer-wins**.

Ta technika jest stosowana w systemie Oracle w poziomie izolacji transakcji **Serializable** oraz w systemie Microsoft SQL Server w poziomie izolacji **Snapshot**.

10.3 Techniki optymistycznego sterowania współbieżnością

Techniki optymistyczne zakładają, że **na ogół transakcje operują na innych danych** i w związku z tym można **na bieżąco nie stosować mechanizmów zapobiegających problemom**, natomiast **na końcu** transakcji następuje **sprawdzenie**, czy realizowany harmonogram nie spowoduje problemów. Jeśli tak, to transakcja jest wycofywana, jeśli nie, to jest zatwierdzana.
Wyróżniamy **trzy fazy**:

- **Faza odczytu** – transakcja może **odczytywać** wartości zatwierdzonych elementów danych z bazy, **aktualizacje** są stosowane tylko względem **kopii lokalnych** elementów danych w **obszarze roboczym** transakcji.
- **Faza walidacji** – wykonywane jest **sprawdzenie**, mające na celu **zapobieżenie naruszeniu szeregowalności** w przypadku, gdy aktualizacje transakcji zostaną zastosowane.
- **Faza zapisu** – w razie powodzenia fazy walidacji, **aktualizacje transakcji** są **zastosowane w bazie**. W przeciwnym razie aktualizacje są **odrzucane** a transakcja później **ponawiana**.

10.4 Blokady zapobiegające, blokowanie na wielu poziomach ziarnistości

Ziarnistość (granularity) blokowania
Dotychczasowo zakładaliśmy blokady tylko **na poziomie wierszy. Można blokować całe tabele, pewne zbiory** wierszy, bloki danych, kolumny, itd. W kontekście blokowania można rozważać **drzewo (hierarchę)** elementów danych, którego **korzeniem** jest cała **baza danych**, natomiast **liśćmi** są **najmniejsze elementy**, na które może być nałożona **blokada**.

Blokowanie na tylko jednym **poziomie najmniejszych elementów** (np. wierszy) powoduje **duży narzut** (duża liczba blokad obsługiwanych przez zarządcę blokad). Blokowanie tylko **dużych elementów** powoduje **zmniejszenie wydajności współbieżnych** transakcji.

Blokady na wielu poziomach (struktura drzewa). W celu ograniczenia przeszukiwania koniecznego do odnalezienia zablokowanych następników węzła można zastosować **blokady zapobiegające** i **protokół blokowania zapobiegającego**.

Nowe blokady na poziomie tabeli (pewnego węzła macierzystego)

- **IS** – **wspólna** blokada zapobiegająca – transakcja zamierza założyć **blokadę S** na poszczególnych **wierszach** tabeli (ogólnie: węzłach potomnych)
- **IX** – **wyłączna** blokada zapobiegająca – zamiar założenia **blokady X** na **wierszach**
- **SIX** – **wspólna wyłączna** blokada zapobiegająca – na **tabeli** (węźle macierzystym) **jest** już blokada **S**, **zamierza** się założyć blokadę **X na wiersze (połączenie S oraz IX)**.

Macierz kompatybilności blokad

	X	**SIX**	**IX**	**S**	**IS**	**bez blokady**
X	Nie	Nie	Nie	Nie	Nie	Tak
SIX	Nie	Nie	Nie	Nie	Tak	Tak
IX	Nie	Nie	Tak	Nie	Tak	Tak
S	Nie	Nie	Nie	Tak	Tak	Tak
IS	Nie	Tak	Tak	Tak	Tak	Tak
bez blokady	Tak	Tak	Tak	Tak	Tak	Tak

Protokół blokowania zapobiegającego
Transakcja nie może założyć blokady na wierszu (generalnie: węźle drzewa), zanim nie otrzyma odpowiedniej blokady na tabeli zawierającej ten wiersz (generalnie: na węźle – rodzicu).

- Należy ściśle trzymać się **zgodności blokad**.
- **Korzeń** drzewa **musi** zostać **zablokowany** jako **pierwszy** w dowolnym trybie.
- Wierzchołek **może zostać zablokowany** przez transakcję T w trybie **S** lub **IS** tylko wtedy, gdy **wierzchołek nadrzędny został już zablokowany** przez transakcję T w trybie **IS** lub **IX**.
- Wierzchołek **może zostać zablokowany** przez transakcję T w trybie **X** lub **IX** lub **SIX** tylko wtedy, gdy wierzchołek nadrzędny został już zablokowany przez transakcję T w trybie **IX** lub **SIX**.
- Transakcja **może zablokować** wierzchołek tylko wtedy, gdy **nie zwolniła blokady** żadnego wierzchołka (wymuszenie stosowania protokołu **blokowania dwufazowego**)
- Transakcja **może odblokować** wierzchołek tylko wtedy, gdy **żaden z potomków** tego wierzchołka **nie jest zablokowany** przez tę transakcję.

Eskalacja blokad – **łączenie** blokad **drobnoziarnistych** w **jedną gruboziarnistą**.

Inne typy blokad - blokady modyfikacji schematu (dostępu do tabeli której struktura jest modyfikowana), blokady zakresów w kluczu, blokady w indeksach.

Zatrzaski - blokady zakładane na **krótki czas**, nie według reguły 2PL, stosowane np. w celu zagwarantowania **fizycznej spójności** bloku (strony) danych w czasie **zapisywania** na dysk.

11 Transakcje rozproszone

Transakcje dotyczące **operacji na wielu bazach danych**, a nawet na wielu **serwerach**.

Dwufazowy protokół (2PC, 2 Phase Commit) zatwierdzania transakcji rozproszonych z wykorzystaniem **zarządcy transakcji** (jeden globalny) oraz **zarządców zasobów** (węzłów uczestników):

- **Faza przygotowania** (prepare)
  Zarządca transakcji otrzymuje **żądanie** wykonania operacji **COMMIT**, po czym wysyła **żądanie prepare** do **zarządców zasobów** przygotowania zakończenia.
  Zarządcy zasobów zgłaszają do zarządcy transakcji **wynik operacji prepare** (czy transakcja może być w **lokalnym** węźle zatwierdzona).
- **Faza zatwierdzenia** (commit)
  Jeśli **wszyscy** zarządcy zasobów **zgłosili powodzenie**, to zarządca transakcji **wysyła** do nich **polecenie commit**.
  Wpp przesyła polecenie **wycofania** (rollback).

12 Systemy RAID

- **Programowe**
  Programowe systemy RAID są tańsze, ale powodują większe obciążenie procesora.

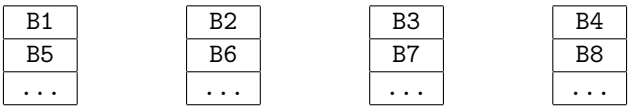
- **Sprzętowe**
  W wielu sprzętowych systemach RAID można **wymienić uszkodzony dysk** w trakcie normalnej **pracy** systemu.

12.1 RAID 0

Również znany jako **system z przeplotem** (striped set). Dane umieszczane są równomiernie na dwóch lub więcej dyskach.

- **Nie ma nadmiarowości!** (redundancy) – system **nie jest odporny** na **błędy**.
- **Niezawodność** jest **odwrotnie proporcjonalna** do **liczby dysków** w systemie RAID 0.
- RAID 0 **nie jest polecany** w środowiskach wymagających **podwyższonej odporności** na błędy (mission-critical environments).
- **Szybszy odczyt i zapis** w porównaniu z pojedynczym dyskiem dzięki operacjom równoległym.
- Czasami był używany do **połączenia** dwóch lub więcej **mniejszych dysków** w jeden **większy logiczny dysk**.

Cztery dyski (RAID 0) – przykład.



Żądanie bloku B1 będzie obsłużone przez dysk 1 (pierwszy z lewej). Jednoczesne żądanie bloku B5 będzie wstrzymane, ale żądanie B2 może być obsłużone równolegle.

12.2 RAID 1

- **Mirroring** lub **duplexing** (gdy każdy dysk ma osobny kontroler).
- Na ogół zawiera **dwa** dyski, czasem więcej. **Te same dane zapisywane są w obu dyskach**.
- Na ogół dane mogą być **odczytywane** z obu dysków **niezależnie** (równolegle).
- **Odczyt** RAID 1 jest prawie **dwukrotnie szybszy** od odczytu pojedynczego dysku.
- **Wolniejszy zapis** w porównaniu z pojedynczym dyskiem. Dane muszą być zapisane na obydwu dyskach, na początku operacji głowice są na ogół nad innymi ścieżkami i sektorami.
- **Pamięć podręczna** (cache) powinna być **włączona** (o ile jest podtrzymywanie bateryjne) w celu przyspieszenia operacji zapisu.
- **Używa** efektywnie jedynie **50%** całkowitej **pojemności**.
- RAID 1 jest **odporny na awarie** – system, który zawiera N dysków może przetrwać jednoczesną awarię N-1 dysków.

Typowe zastosowanie: dziennik transakcji.

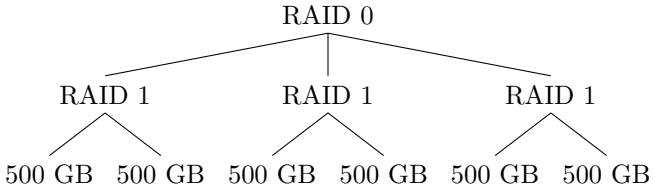
12.3 RAID 5

- Zawiera **trzy lub więcej** dysków.
- **Zapisy** są wykonywane **blokami** na **wielu** dyskach z wykorzystaniem **bloków parzystości**.
- **Wolne** operacje **zapisu**. Zapis = dwa odczyty i dwa zapisy.
- **Szybki odczyt**, względnie **efektywne wykorzystanie przestrzeni** dyskowej.
- **Odporność na błędy** – system może przetrwać awarię dysku.
  Bloki parzystości są odczytywane, jeśli przy odczycie wykryty jest **błąd sumy kontrolnej**. W takim przypadku pozostałe bloki z paska są **automatycznie użyte** do **odtworzenia** informacji w bloku uszkodzonym. Podobnie dzieje się przy uszkodzeniu całego dysku.
- Należy **włączyć pamięć podręczną** (cache) jeśli jest podtrzymywanie bateryjne. Operacje zapisu mogą być wówczas wykonywane znacząco szybciej.
- Wykorzystuje **1/n pojemności** dysku **na bloki parzystości**, gdzie oznacza liczbę dysków w systemie.

Typowe zastosowanie: systemy, w których **większość operacji to operacje odczytu**, dla tablic lub indeksów, które sa tylko do odczytu lub są rzadko modyfikowane.

12.4 RAID 10 (1+0)

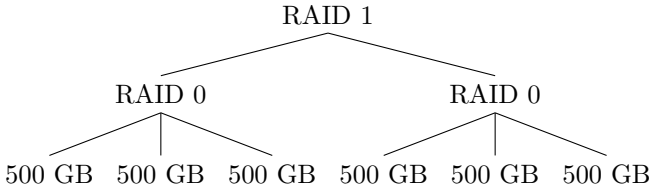
System z przeplotem z systemami RAID 1 jako elementami składowymi. **Jeden dysk z każdego systemu RAID 1 może ulec awarii bez całkowitej utraty danych**. Jeśli uszkodzeniu ulegnie jeden dysk, tylko **jeden** staje się **krytycznym punktem** układu.



- **Najszybszy zapis i odczyt**.
- **Odporność na błędy**.
- **Wykorzystuje** efektywnie **50%** pojemności całkowitej.
- Najlepszy, ale najdroższy.

12.5 RAID 01 (0+1)

RAID 1 złożony z RAIDów 0 (zamiast dysków). RAID 0+1 jest uważany za **gorszy** niż RAID 10. RAID 0+1 **nie przetrwa dwóch równoczesnych awarii** jeśli **nie** są to dyski **z tego samego układu** RAID. Po awarii jednego dysku, **wszystkie dyski** w drugim pasku stanowią **krytyczne punkty**. **Połowa** dysków **przestaje być wykorzystywana**.



12.6 Zastosowanie systemów RAID

- **Dziennik transakcji** na oddzielnych systemach **RAID 1** lub **RAID 10**
- **Dane** powinny być umieszczone w systemach **RAID 5** (jeśli co najwyżej 10% liczby wszystkich operacji to operacje zapisu) lub **RAID 10** (w przeciwnym przypadku).
- Należy dodać odpowiednią liczbę dysków aby **zapewnić co najwyżej 125 losowych op/s** na dysk. Należy systematycznie **monitorować liczbę operacji** na sekundę dla **każdego** dysku.
- Użycie systemu RAID **nie powinno zastąpić** odpowiedniej strategii robienia **kopii zapasowych**.

13 Indeksy

- Indeksy odgrywają **kluczową rolę dla wydajności** w systemach baz danych.
- Cel stosowania: **zmniejszenie czasu szukania**, **zminimalizowanie liczby odczytów** z dysku.
- Mogą **zmniejszyć czas wykonania kwerend**.
- Mogą **wydłużać** czas wykonania operacji **zmiany** danych, jeśli zmiany te dotyczą **kolumny indeksowanej** co może powodować **konieczność** wykonania odpowiedniej **zmiany w indeksie**.
- Informacje na temat indeksów wraz z informacjami o wartościach przechowywanych w kolumnach (**statystyki**) są **analizowane** przez **optymalizatory kwerend** w celu wyznaczenia **najlepszego planu wykonania kwerendy**.
- Indeks **może ulegać fragmentacji**, zatem może wymagać okresowej **defragmentacji** lub **przebudowy**. Należy zadbać o **aktualizację statystyk**.
- **Obciążenie serwera** powinno być okresowo **monitorowane** m.in. w celu **dobrego indeksowania**.

13.1 Typy indeksów dla plików sekwencyjnych

Indeksy dla plików sekwencyjnych:

- **Główne** (Primary) - na identyfikatorach wierszy.

- **Drugorzędne** (Secondary) - pomocnicze.

- **Proste**.
- **Złożone**.

- **Gęste** (zagęszczone, dense) – każdy wiersz danych ma swój wpis w indeksie.
- **Rzadkie** (niezagęszczone, sparse) – np. każdy blok danych ma jeden wpis w indeksie. Dane muszą być **posortowane według klucza indeksu** (może to być klucz główny w tabeli lub inne pola).

- **Grupujące** (clustering) – ten termin jest wieloznaczny. W odniesieniu do plików sekwencyjnych oznacza grupowanie w jednym bloku **wierszy z taką samą wartością klucza indeksu**; taki **indeks** budowany jest **na polu niekluczowym** i wymaga **posortowania wierszy** według **pola indeksowanego**.
- **Wielopoziomowe**.

Indeksy dla standardowych typów danych:

- Drzewa B+
- Tablice haszujące
- Indeksy binarne (mapy binarne)

Indeksy dla danych przestrzennych, wielowymiarowych:

- Drzewa ćwiartek (quad trees),
- R-drzewa,
- k-d drzewa,
- indeksy binarne,
- uogólnione haszowanie: pliki siatkowe, podzielone funkcje haszujące

13.2 Drzewa B+

- **Wszystkie wartości w liściach**
- Każdy **wewnętrzny węzeł** (nie liść) drzewa B+ ma postać:
  $< P_1, K_1, P_2, K_2, \ldots, P_{q-1}, K_{q-1}, P_q >$ gdzie $q <= p, P_i (i = 1, ..., q)$ jest wskaźnikiem do poddrzewa.
  $K_1 < K_2 < \cdots < K_{q-1}$.
- Dla każdej **wartości** X w poddrzewie wskazywanym przez P_i, mamy: $K_{i-1} < X <= K_i$ dla $1 < i < q$ oraz $X <= K_i$ dla $i = 1$ oraz $K_{i-1} < X$ dla $i = q$.
- Każdy **wewnętrzny węzeł** ma **co najwyżej p** wskaźników do poddrzew.
- Każdy **węzeł**, z wyjątkiem korzenia, ma **przynajmniej** $\lceil p/2 \rceil$ wskaźników do poddrzew.
- **Korzeń** ma **co najmniej dwa** wskaźniki do poddrzew, jeśli jest węzłem wewnętrznym.
- Węzeł wewnętrzny z q wskaźnikami, $q <= p$, ma $q - 1$ **wartości**.
- Każdy **liść** drzewa B+ ma postać:
  $<< K_1, P_{r_1} >, < K_2, P_{r_2} >, \ldots, < K_{q-1}, P_{r_{q-1}} >, P_{next} >>$, gdzie $q <= p$ i P_{r_i} jest wskaźnikiem do danych, P_{next} wskazuje następny węzeł liść (**liście tworzą listę**).
  $K_1 < K_2 < \cdots < K_{q-1}, q <= p$.
- Każdy P_{r_i} jest wskaźnikiem do danych, który wskazuje na:
  - **rekord**, którego wartość w polu indeksowanym jest równa K_i
    lub
  - **blok** zawierający rekord
    lub
  - **blok** wskaźników do rekordów z takimi samymi wartościami K_i, jeśli klucz indeksu nie jest unikalny

  W praktyce w SZBD unikalność może być zapewniona np. przez rozszerzenie klucza – dołączenie do klucza jakiejś liczby).

- Każdy węzeł liść przechowuje co najmniej $\lceil p/2 \rceil$ wartości.
- Wszystkie **węzły liście są na tym samym poziomie**.

13.2.1 Operacje na drzewach B+

- **Podział liścia**
  - Jeśli węzeł jest **pełny** i ma być do niego wstawiony nowy wpis, wówczas węzeł zostaje **podzielony**.
  - Pierwsze $j = \lceil ((p\_leaf + 1)/2) \rceil$ wpisów **zostaje w oryginalnym** węźle, pozostałe zostają **przesunięte** do nowego węzła liścia. p_leaf oznacza maksymalną liczbę wskaźników do danych w węzłach liściach.
  - j-ta wartość jest **kopiowana do węzła rodzica** i węźle tym tworzony jest **nowy wskaźnik** (do nowego węzła).
- **Podział węzła wewnętrznego**
  - Jeśli węzeł wewnętrzny jest **pełny** i ma być do niego wstawiony nowy wpis, wówczas węzeł **zostaje podzielony**.
  - Wpisy aż do P_j – j-tego wskaźnika do poddrzewa, gdzie $j = \lceil ((p + 1)/2) \rceil$ **są zatrzymane**, podczas gdy wartość j-ta jest **przesuwana** do **węzła macierzystego**, nie kopiowana.
  - **Nowy węzeł** wewnętrzny przechowuje wpisy od P_{j+1}.
  - Podział może **propagować w górę** aż do utworzenia **nowego węzła korzenia**.
- **Usuwanie wartości**
  - Usunięcie wartości jest **zawsze realizowane w liściu**, jeśli usunięta wartość występuje również w węźle wewnętrznym, musi być też stamtąd usunięta.
  - Przy usuwaniu K_i z **węzła wewnętrznego**, wartość K_{i-1} umieszczona bezpośrednio z lewej strony usuwanej wartości w **liściu** musi **zastąpić wartość** K_i w węźle wewnętrznym.
  - Usuwanie wpisów może doprowadzić do **zmniejszenia liczby wpisów w węźle poniżej wymaganego z definicji poziomu**. W tym przypadku **wpisy z sąsiadów** są rozmieszczane tak, by węzły były **wypełnione w odpowiednim stopniu**. Jeśli nie da się tego zrealizować, wówczas **sąsiadujące węzły są łączone** i **liczba liści zostaje zmniejszona**.

13.2.2 Indeksy typu drzewa B+

Wiele systemów umożliwia **wybór współczynnika wypełnienia węzłów** (PCTFREE w Oracle), a nawet można ustawiać współczynnik wypełnienia **osobno dla liści**, **osobno dla węzłów wewnętrznych** (FILLFACTOR i PADINDEX w systemie MS SQL Server).
Indeksy z odwróconym kluczem - wartość klucza jest zmieniana, np. **zapis binarny jest negowany**. W pewnych przypadkach **zapobiega** to **powstaniu wąskiego gardła**, jeśli więcej transakcji odwołuje się do **podobnych wartości klucza** (znajdujących się w jednym bloku).

Fragmentacja indeksów typu drzewo B+
Wstawianie nowych wartości klucza może spowodować **konieczność podziału węzła**. Przy podziale 50+50 i niekorzystnym układzie danych, po wykonaniu pewnej liczby wstawień **większość węzłów może być uzupełniona tylko w 50%**. **Nowo** przydzielone węzły (bloki) mogą być **rozrzucone po dysku** (nie są sąsiadujące).

Defragmentacja indeksów - zmiana kolejności bloków (stron), przebudowa z **przywróceniem oryginalnego** lub **ustawieniem nowego współczynnika wypełnienia** węzłów (może być oddzielnie dla węzłów liści i dla poziomów nieliściastych).

Charakteryzacja indeksów typu drzewa B+

- **selektywność** - liczba różnych wartości klucza (wartość względna, liczba różnych wartości podzielona przez liczbę wierszy),
- **dane** dotyczące **fragmentacji** bloków i ekstentów (liczba bloków i ekstentów ułożonych **nie po kolei**, **średnie zapełnienie** bloku)
- **współczynnik zgrupowania** (clustering factor) – liczba bloków, przez które należy przejść przy skanowaniu wierszy indeksu.
  Przy indeksie **zgodnym z posortowaniem** wierszy jest to **liczba bloków** danych; przy indeksie **niezgodnym** oraz przy **dużych wierszach** liczba ta może być **bliska liczbie wierszy**.

- **statystyki** (mogą zawierać histogramy), mogą być budowane **również dla kolumn nie indeksowanych**; tworzone i modyfikowane automatycznie lub ręcznie na podstawie pewnej próby.

13.3 Sposoby przechowywania danych w plikach

- **Sterty** (heap) – **nieuporządkowane**, zapis w blokach (stronach).
- **Tablice o strukturze indeksowej** (IOT Index Organized Tables, Oracle), **indeksy grupujące** (Clustered Indexes, MS SQL Serwer).
- **Klastry** (clusters) – przechowywanie **dwóch lub więcej** tabel w **jednym pliku**, wiersze są **posortowane** według pola (pól) łączącego.

W MS SQL Server
Można budować **indeksy grupujące** na polach **innych niż klucze**.
Indeksy niegrupujące (niezgrupowane, non-clustered) mają **inną budowę** węzłów liści w **przypadku**, gdy dla tabeli **istnieje już indeks grupujący**. Zamiast wskaźników do wierszy przechowywane są **wartości klucza indeksu grupującego**. **Zaleta**: w przypadku **zmiany położenia wiersza** (np. po podziale strony) **nie trzeba modyfikować** indeksów niegrupujących.

IOT w Oracle
Secondary indexes (odpowiednik indeksów niegrupujacych w SQL Serwerze): odnajdywanie wiersza na podstawie **wartości guess** tzn. wykorzystanie physical ROWID, a w przypadku gdy wiersz nie zostanie znaleziony wykorzystanie logical ROWID. W IOT **wiersze mogą być podzielone pionowo**: **część** kolumn jest przechowywana w **liściach** IOT, **część** może być przechowywana w oddzielnych **blokach**.