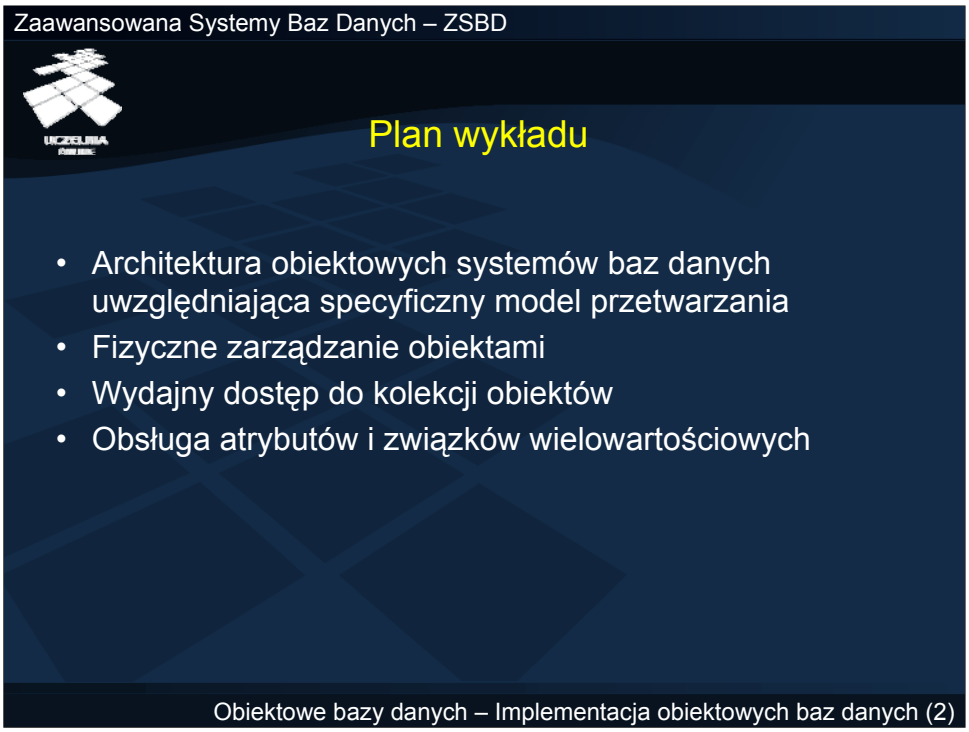
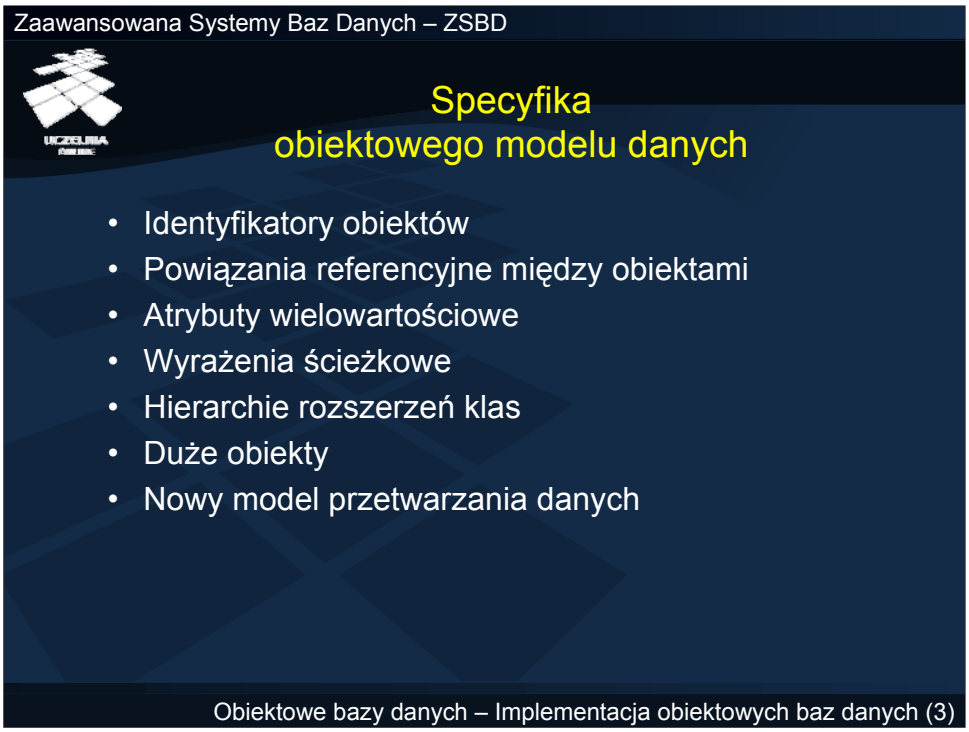

Niniejszy wykład jest poświecony problemom implementacji systemów obiektowych baz danych. Nowe własności modelu danych wymagają wydajnej implementacji uwzględniającej specyfikę obiektowego modelu danych i nowego modelu przetwarzania..



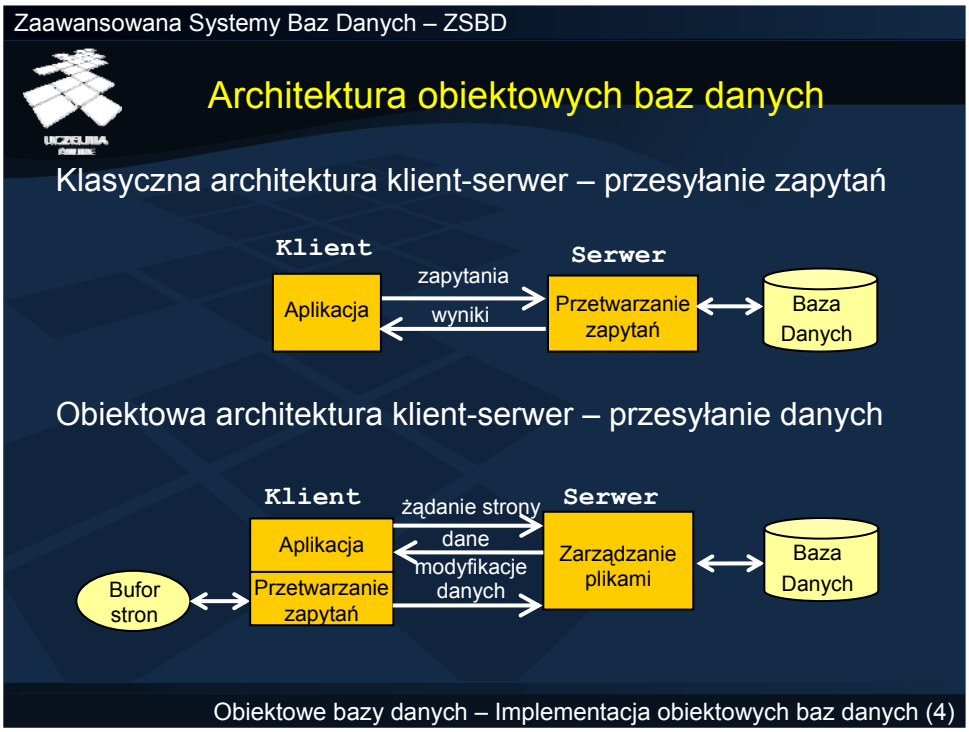

Celem wykładu jest poznanie rozwiązań implementacyjnych dedykowanych dla obiektowych i obiektowo-relacyjnych systemów baz danych.

W ramach wykładu zostaną przedstawione nowe architektury systemów zarządzania bazami danych uwzględniające charakterystyki przetwarzania danych właściwe dla nowych dziedzin zastosowań. Następnie poznamy rozwiązania dotyczące fizycznego zarządzania obiektami oraz nowe struktury danych przyśpieszające przeszukiwanie dużych powiązanych kolekcji obiektów. Na końcu zostaną przedstawione rozwiązania wydajnego przetwarzania atrybutów wielowartościowych i związków wielokrotnych.




Obiektowy model danych posiada wiele nowych unikalnych cech nieznanych w relacyjnym modelu danych. Należą do nich systemowe identyfikatory obiektów OID, referencyjne związki między obiektami, atrybuty wielowartościowe, wyrażenia ścieżkowe i hierarchie rozszerzeń klas. Te nowe elementy modelu wymagają nowych i wydajnych rozwiązań implementacyjnych.

Ponadto, dziedziny zastosowań obiektowych baz danych mają inne wymagania co własności systemów baz danych. Dotyczy to na przykład konieczności przechowywania w bazie danych bardzo dużych obiektów, do przechowywania na przykład projektów, animacji lub długich dokumentów. Kolejną cechą specyficzną nowych dziedzin zastosowań jest inny model przetwarzania danych, polegający na wielokrotnym i intensywnym przetwarzaniu dużych powiązanych kolekcji obiektów, na przykład w ramach systemy wspomagania projektowania. W systemach relacyjnych podstawowe problemy wydajności dotyczyły efektywnego wyszukiwania w bardzo dużych zbiorach danych niewielkich podzbiorów danych oraz efektywnej realizacji operacji łączenia relacji. W zastosowaniach obiektowych baz danych bardziej typowe są operacje nawigacji wzdłuż długich hierarchicznych ścieżek powiązań.

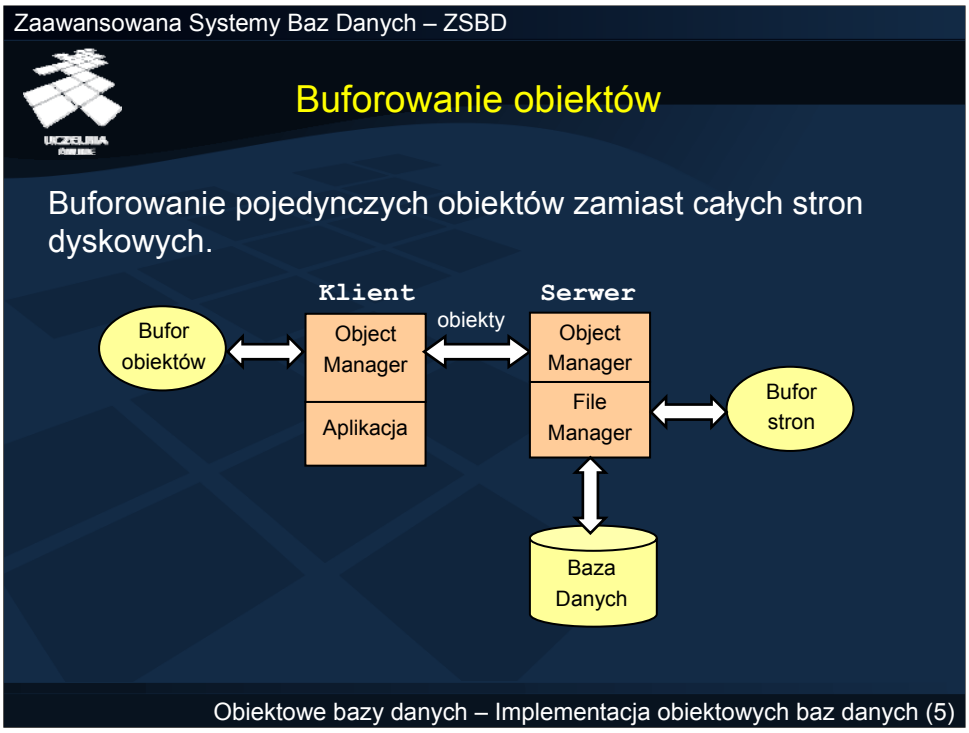



Model przetwarzania danych dla dziedzin zastosowań typowych dla okresu dominacji relacyjnego modelu danych polega na jednokrotnym wykonywaniu przez daną transakcję małej liczby operacji na bardzo niewielkim podzbiorze danych bardzo dużej bazy danych. Typowymi wykonywanymi operacjami w tym modelu są operacje selekcji i łączenia. W bankowym systemie informatycznym typowym przykładem jest transakcja wypłaty pieniędzy z konta obejmująca wyszukanie pojedynczej krotki reprezentującej dane konto w zbiorze setek tysięcy kont, następnie dodanie pojedynczej krotki do liczącego miliony krotek zbioru operacji bankowych i na koniec zmodyfikowanie salda w pojedynczej krotce reprezentującej konto.

Dla takiego wzorca przetwarzania właściwy model przepływu danych w architekturze klient-serwer minimalizujący wolumen transferu danych, polega na przesyłaniu w jedną stronę żądań wykonania operacji na bazie danych i w drugą stronę niewielkich przetwarzanych zbiorów danych. Przetwarzanie zapytań odbywa się w całości na serwerze bazy danych. Do klientów przez sieć komputerową docierają tylko wyniki tego przetwarzania, czyli najczęściej niewielkie podzbiory danych.

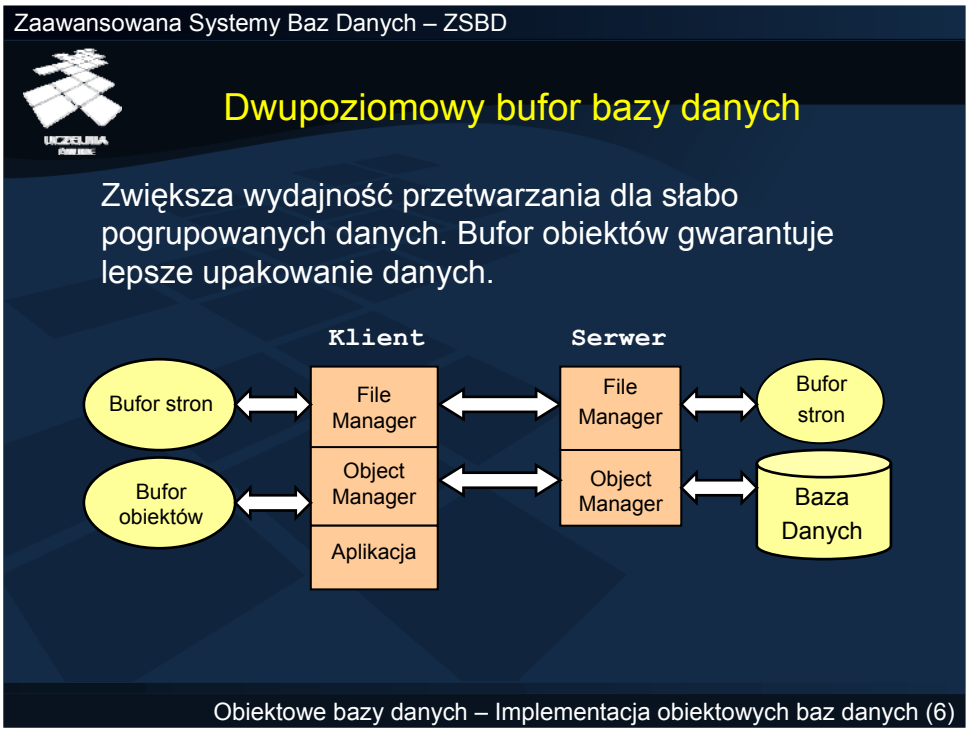
Model przetwarzania właściwy dla zastosowań typowych dla obiektowych baz danych polega intensywnym przetwarzaniu stosunkowo dużego zbioru powiązanych danych. W ramach pojedynczej sesji mają miejsce wielokrotne odwołania do tych samych obiektów. Typowymi operacjami są nawigacje między powiązanymi obiektami lub w głąb atrybutów złożonych. W systemie wspomagania projektowania przykładem są operacje projektowe polegające na cyklicznych modyfikacjach niewielkiego zbioru danych i ich weryfikacji w szerszym kontekście wszystkich powiązanych obiektów.

Dla takiego wzorca przetwarzania przedstawiony model przepływu danych nie jest optymalny. Dla pojedynczej sesji projektowej wiązałby się on z bardzo dużą liczbą żądań wykonania operacji przez serwer bazy i dodatkowo duża część tych żądań powtarzałaby się. Bardziej wydajnym rozwiązaniem jest przeniesienie funkcjonalności przetwarzania zapytań na stronę klienta serwera bazy danych. W tej sytuacji serwer bazy danych będzie przesyłał do klienta całe strony dyskowe, które następnie będą utrzymywane w lokalnym buforze stron dyskowych. Teraz wielokrotnie powtarzające się odwołania do tych samych obiektów będą obsługiwane lokalnie.



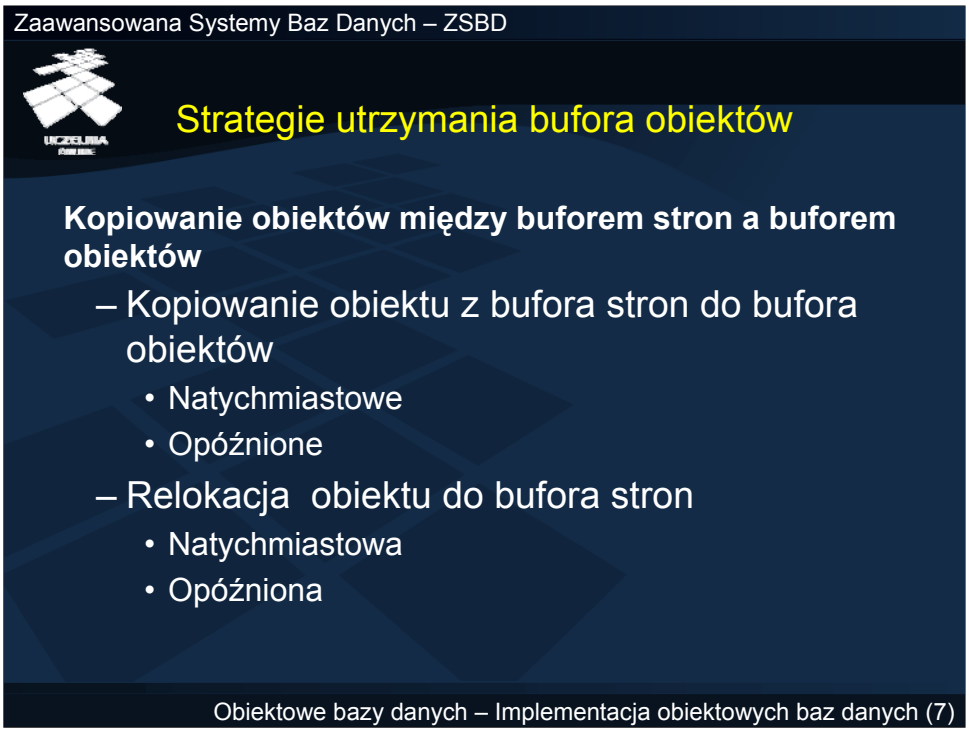

Dla słabo pogrupowanych danych wydajność bufora stron klientów serwera bazy danych będzie niska. Jedynie niewielki procent transferowanych i przechowanych w buforze stron danych jest użyteczny. Reszta nie przetwarzanych w ramach danej sesji obiektów, a znajdująca się na tych samych stronach dyskowych stanowi tylko zbędny balast. Nawigacja do kolejnych obiektów przy przepełnionym niepotrzebnymi danymi buforze stron będzie wymagała częstej i wielokrotnej wymiany stron w buforze.

Rozwiązaniem jest zastąpienie słabo wypełnionego użytecznymi z punktu widzenia danej sesji danymi bufora stron lepiej upakowanym buforem obiektów. Serwer bazy danych będzie dla minimalizacji transferu zamiast całych stron przesyłał do klienta pojedyncze obiekty, a bufor obiektów klienta będzie służył do przechowywania jedynie przetwarzanych w danym momencie obiektów. Dzięki lepszemu upakowaniu danych w buforze znacznie rzadziej potrzebna będzie czasochłonna wymiana zawartości bufora.




Rozwiązanie polegające na zamianie bufora stron na bufor obiektów jest bardzo wydajne dla słabo zgrupowanych danych. Tymczasem jednym z rozwiązań stosowanych w systemach zarządzania obiektowymi bazami danych jest fizyczne grupowanie danych polegające na przechowywaniu powiązanych i często przetwarzanych w ramach jednej sesji danych w tych samych lokalizacjach pamięci dyskowej. Obsługa dobrze pogrupowanych danych jest w wypadku zastawania buforów obiektów mało wydajna. Pobranie z serwera bazy danych na przykład stu powiązanych ze sobą obiektów znajdujących się na jednej stronie dyskowej będzie wymagało stukrotnego wywołania żądania pobrania z serwera pojedynczego obiektu.

Żeby uczynić przedstawioną na poprzednim slajdzie architekturę systemów obiektowych baz danych bardziej elastyczną wprowadzono po stronie klienta dwupoziomowy bufor składający się z bufora stron i bufora obiektów. Między serwerem a klientem są przesyłane całe strony dyskowe. Dla dobrze pogrupowanych danych minimalizuje to komunikację między klientem a serwerem. Jednak przetwarzane przez aplikację bazy danych obiekty mogą być kopiowane z bufora stron do bufora obiektów. Optymalizuje to wykorzystanie buforów dla źle pogrupowanych danych. Usuwanie stron dyskowych z przepełnionego bufora stron nie oznacza braku dostępności znajdujących się na nich obiektów. Przed usunięciem strony dyskowej z bufora stron aktualnie przetwarzane obiekty będą skopiowane do bufora obiektów. Dzięki temu kolejne odwołania do tych obiektów nie będą wymagały ponownego załadowania zawierających je stron. Aplikacja będzie przetwarzała kopie tych obiektów w buforze obiektów.



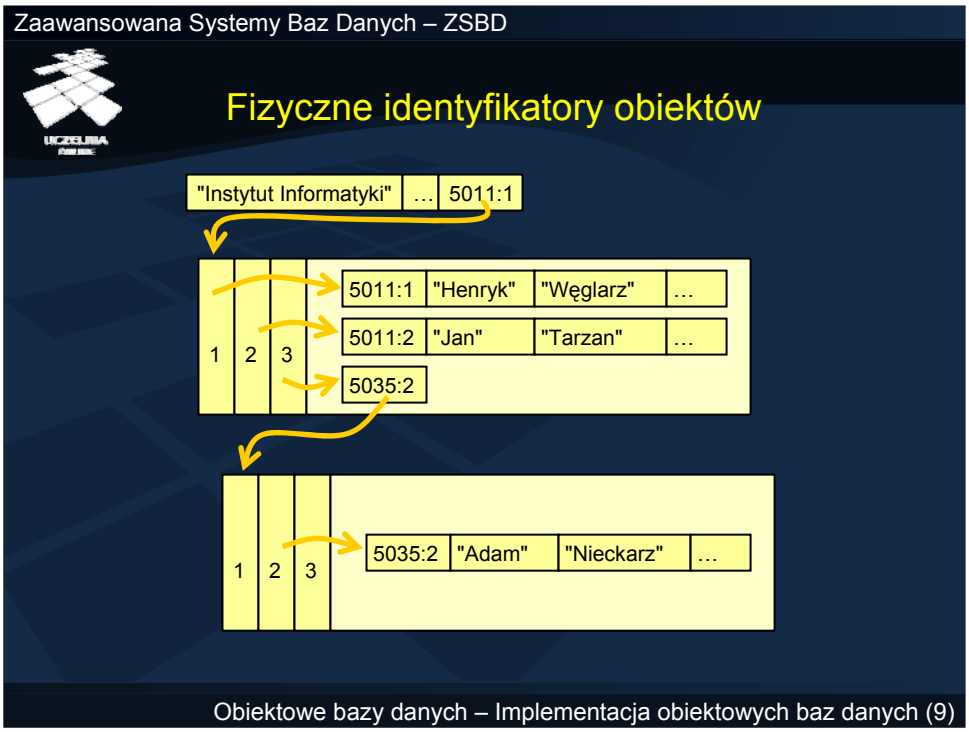

Proponowane są różne strategie utrzymywania buforów obiektów właściwe dla różnych modeli przetwarzania. Strategie te różnią się momentem kopiowania obiektu z bufora stron do bufora obiektów oraz momentem realokacji zmodyfikowanego obiektu z bufora obiektów z powrotem do bufora stron.

Można wyróżnić dwie opcje momentu kopiowania obiektu z bufora stron do bufora obiektów. Pierwsza opcja zakłada, że aplikacja może korzystać tylko i wyłącznie z obiektów znajdujących się w buforze obiektów. Wymaga to natychmiastowego kopiowania obiektów z bufora stron do bufora obiektów, przy pierwszym odwołaniu do danego obiektu. Bufor obiektów powinien być w tej sytuacji dużo większy niż bufor stron. Lepsze upakowanie obiektów zmniejsza znacznie liczbę błędów stron. Opcja ta będzie optymalna dla długiego i intensywnego przetwarzania większego zbioru obiektów.

Druga opcja zakłada, że aplikacja może odwoływać się do kopii obiektów ulokowanych w buforze obiektów, jak i do oryginałów obiektów znajdujących się buforze stron. Pozwala to na utrzymywanie tylko jednej kopii obiektu albo w buforze obiektów, albo w buforze stron. W tej sytuacji obiekty są kopiowane do bufora obiektów jedynie w momencie usuwania zawierającej je strony z bufora stron, po to by następne odwołanie do danego obiektu nie spowodowało błędu strony, czyli konieczności jej powtórnego załadowania z serwera bazy danych. W tej sytuacji rozmiary buforów stron i buforów obiektów powinny być zrównoważone. Opcja ta będzie optymalna dla krótkiego przetwarzania dużych zbiorów obiektów.

Następne dwie opcje dotyczą momentu realokacji obiektu zmodyfikowanego w buforze obiektów z powrotem na właściwe miejsce na stronie w buforze stron. Pierwsza opcja zakłada, że zmodyfikowany obiekt jest umieszczany z powrotem na właściwej stronie w momencie powtórnego załadowania tej strony do bufora stron. Druga opcja polega na przesunięciu momentu realokacji, aż do czasu gdy obiekt przestanie być przetwarzany. W wypadku błędu strony, będzie to wymagać ściągnięcia tej strony z serwera bazy danych do bufora stron.

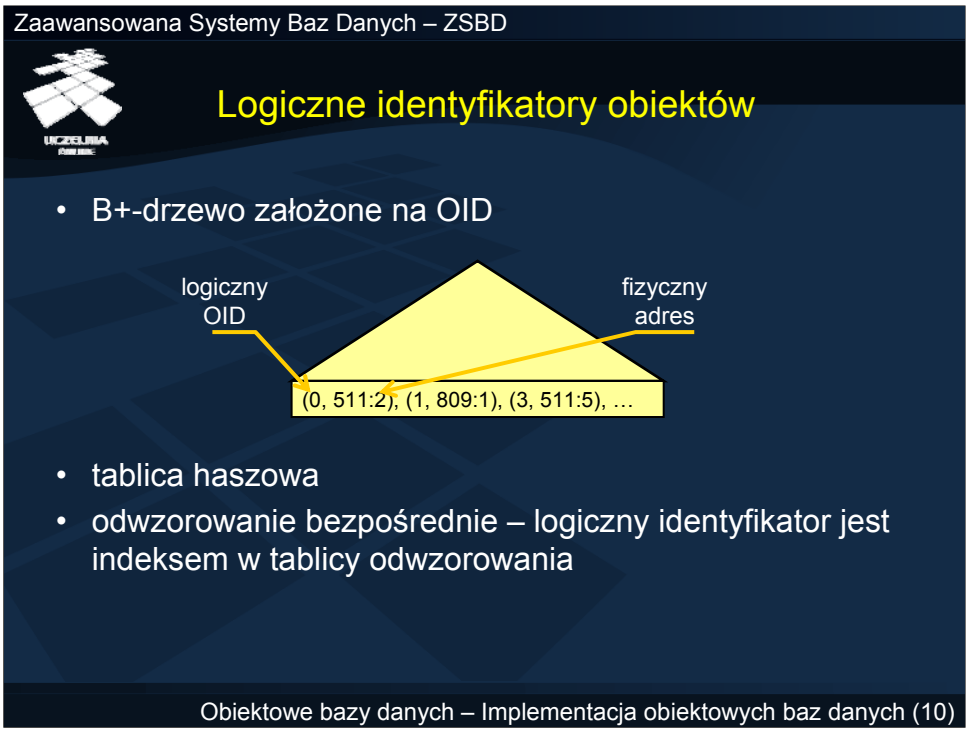
Kombinacje wymienionych opcji dają w sumie cztery potencjalne strategie utrzymania buforów obiektów.  Wybór określonej strategii zależy od charakterystyki przetwarzania danej dziedziny zastosowania.




Stosowane są dwa rodzaje identyfikatorów obiektów: identyfikatory fizyczne i identyfikatory logiczne.

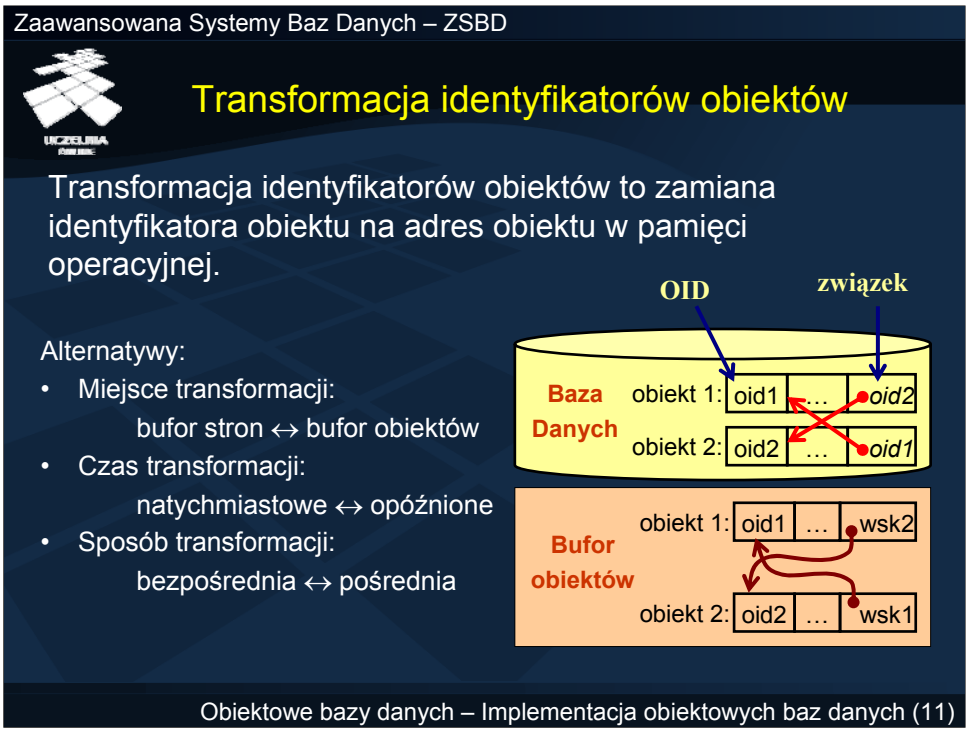
Jako identyfikator obiektu może służyć fizyczna lokalizacja obiektu w bazie danych. Bazy danych odwzorowują zajmowaną przez nie przestrzeń dyskową na uporządkowany i ponumerowany zbiór stron dyskowych, które są minimalnymi jednostkami transferu danych z pamięci dyskowej do pomięci operacyjnej. Na pojedynczej stronie dyskowej alokowanych jest wiele obiektów. Zazwyczaj na identyfikator obiektu składa się numer strony dyskowej, na której jest ulokowany obiekt oraz numer lokalizacji obiektu na tej stronie nazywany „*slotem*”.

Podstawową zaletą identyfikatorów fizycznych jest szybkość dostępu do obiektu. Znajomość identyfikatora umożliwia natychmiastowy dostęp do obiektu. Jest to bardzo istotne dla wydajności operacji nawigacji. Wadą jest przywiązanie identyfikatorów do fizycznej lokalizacji, co utrudnia fizyczną reorganizację bazy danych. Na przykład w wypadku, gdy rozmiar obiektu zlokalizowanego na danej stronie w wyniku modyfikacji przekroczy rozmiar wolnego miejsca na stronie, obiekt musi być przesunięty na inną stronę. Żeby uniknąć zmiany identyfikatora przesuwanego obiektu obiekt zwalnia miejsce na stronie, ale nie zwalnia przydzielonego numeru slotu. Ilustruje to przykład. Identyfikator obiektu reprezentującego osobę Adama Nieckarza, który przesunięto ze strony dyskowej o numerze 5011 na stronę o numerze 5035, pozostał niezmieniony. Na stronie 5011 w slocie 3 wstawiony został wskaźnik na nową lokalizację obiektu. Jednak pozostawienie tego samego identyfikatora wydłuży czas dostępu do tego obiektu. Zamiast pojedynczego dostępu do dysku teraz potrzebne będą dwa dostępy. Najpierw do oryginalnej strony, na którą wskazuje identyfikator obiektu, a potem do strony, na której faktycznie znajduje się przesunięty obiekt.




Alternatywą dla fizycznych identyfikatorów obiektów są identyfikatory logiczne. Identyfikatory logiczne są sztucznie generowanymi wartościami, całkowicie niezależnymi od fizycznej lokalizacji obiektu. Zaletą tego rozwiązania jest swoboda w reorganizacji fizycznej struktury bazy danych, nie wpływającej w żaden sposób na wartości identyfikatorów, czy pogorszenie warunków dostępu do obiektów. Wadą logicznych identyfikatorów jest dłuższy czas dostępu do obiektów na podstawie znajomości identyfikatora obiektu.

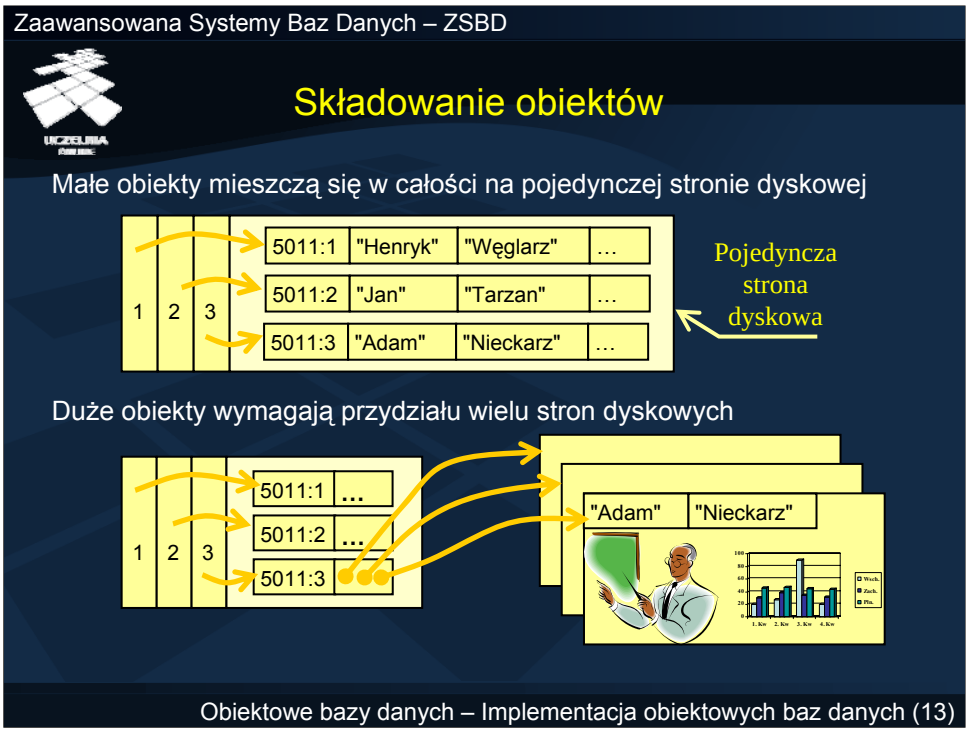
Stosowane są trzy metody odwzorowania logicznego identyfikatora obiektu na jego fizyczny adres. Pierwszą jest założenie indeksu na logicznych identyfikatorach obiektu. Przykład takiego indeksu jest zilustrowany na slajdzie. Drugą metodą odwzorowania jest zastosowanie funkcji haszowej. Argumentem funkcji haszowej jest logiczny identyfikator obiektu. Trzecią metodą jest bezpośrednie odwzorowanie w tablicy, w której identyfikator służy jako indeks tablicy, a komórki tablicy zawierają fizyczne adresy obiektów.




Transformacja identyfikatorów obiektów (ang. pointer swizzling) polega na zamianie w obiektach składowanych w pamięci operacyjnej identyfikatorów obiektów reprezentujących związki między obiektami na adresy tych obiektów w pamięci operacyjnej. Taka transformacja może mieć miejsce, jeżeli zarówno obiekt zawierający referencje, jak i obiekt przez nią wskazywany znajdują się jednocześnie w pamięci operacyjnej, to jest w buforze obiektów lub w buforze stron. Celem transformacji identyfikatorów jest przyśpieszenie operacji nawigacji. Nawigacja wykonywana przez adres obiektu w pamięci operacyjnej jest znacznie szybsza, niż nawigacja wymagająca przejścia przez scentralizowane struktury dostępu, na przykład B+-drzewo założone na identyfikatorach obiektów, a następnie ustalenia lokalizacji obiektu w buforach. Zysk z zastosowania takiej transformacji jest wprost proporcjonalny do liczby powtórzeń operacji nawigacji. Jednak jak ustaliliśmy wielokrotne przetwarzanie tego samego zbioru obiektów jest typowe dla modelu przetwarzania obiektowych baz danych.

Idea transformacji identyfikatorów obiektów została przedstawiona na rysunku. Baza danych zawiera dwa obiekty o identyfikatorach oid1 i oid2. Obiekty te są wzajemnie powiązane. Obiekt o identyfikatorze oid1 zawiera referencję na obiekt o identyfikatorze oid2 i na odwrót, obiekt oid2 zawiera referencję na obiekt oid1. Te dwa obiekty załadowane do bufora w pamięci operacyjnej mają przetransformowane referencje z globalnych identyfikatorów obiektów na wskaźniki w pamięci operacyjnej.

Są trzy podstawowe alternatywy dla implementacji mechanizmu transformacji identyfikatorów. Pierwsza dotyczy miejsca wykonywania transformacji w dwupoziomowym systemie buforów. Polega ona na wyborze miejsca wykonywania transformacji: w buforze stron albo w buforze obiektów. Druga alternatywa dotyczy czasu transformacji. Możliwe opcje to: natychmiast po załadowaniu obiektów do pamięci operacyjnej albo dopiero przed pierwszą próbą dostępu przez aplikacje bazy danych do danego obiektu. Ostatnia alternatywa dotyczy sposobu transformacji i obejmuje dwie opcje. Pierwszą opcją jest transformacja referencji zawierających identyfikatory obiektów na adresy tych obiektów w pamięci operacyjnej. Druga opcja to transformacja na adres pośredniczącej struktury w pamięci operacyjnej. To drugie rozwiązanie umożliwia przesuwanie obiektów w buforach bez konieczności modyfikacji wartości przetransformowanych wskaźników.




W obiektowych bazach danych ważna jest wydajna implementacja składowania dużych obiektów. Rozmiary obiektów zawierających długie animacje, duże rysunki lub długie dokumenty mogą osiągać wielkości mierzone w gigabajtach. Rozróżnienie między dużymi a małymi obiektami jest całkowicie jednoznaczne. Małe obiekty to takie, które mieszczą się całości na pojedynczej stronie dyskowej. Podczas gdy rozmiar dużych obiektów przekracza rozmiar strony dyskowej.

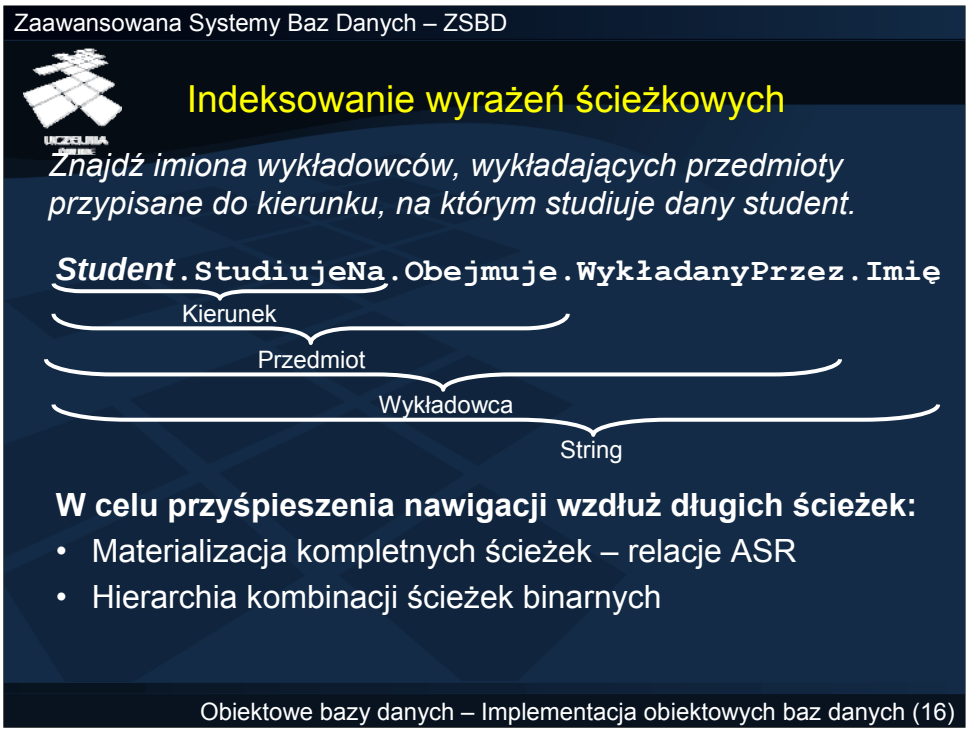
Górny rysunek na slajdzie ilustruje sposób organizacji małych obiektów. Na pojedynczej stronie dyskowej znajdują się w całości trzy małe obiekty. Dolny rysunek ilustruje zmianę organizacji obiektu reprezentującego Adama Nieckarza. Do obiektu dodano: animacje, wykresy i obrazy, w wyniku czego rozmiar obiektu przekroczył rozmiar całej strony dyskowej, a nie tylko rozmiar wolnego miejsce na stronie, którą współdzielił z trzema innymi obiektami. Do przechowywania wartości tego obiektu będą potrzebne trzy nowe strony dyskowe. Będę to prywatne strony dużego obiektu, nie będą one współdzielone z innymi obiektami. Nagłówek obiektu po jego transformacji do struktury dużego obiektu pozostanie na źródłowej stronie o numerze 5011.


Rozmiar dużych obiektów może być na tyle duży, że istotne staje się wydajne zarządzanie pojedynczymi dużymi obiektami. Dla dużych obiektów należy zagwarantować wydajną realizację operacji wyszukiwania, wstawiania, modyfikacji i usuwania sekwencji bajtów o określonej długości i lokalizacji w strukturze obiektu rozumianej jako ciąg bajtów.

Wydajna implementacja dużych obiektów opiera się na dwóch istotnych założeniach. Po pierwsze zapewnienia wydajnej metody wyszukiwania w ciągu bajtów rozrzuconych po wszystkich stronach dyskowych dużego obiektu bajtu o numerze określającym jego pozycję w ciągu bajtów. Po drugie zagwarantowania, że lokalne modyfikacje obiektu będą wymagały jedynie lokalnej rekonstrukcji jego struktury.

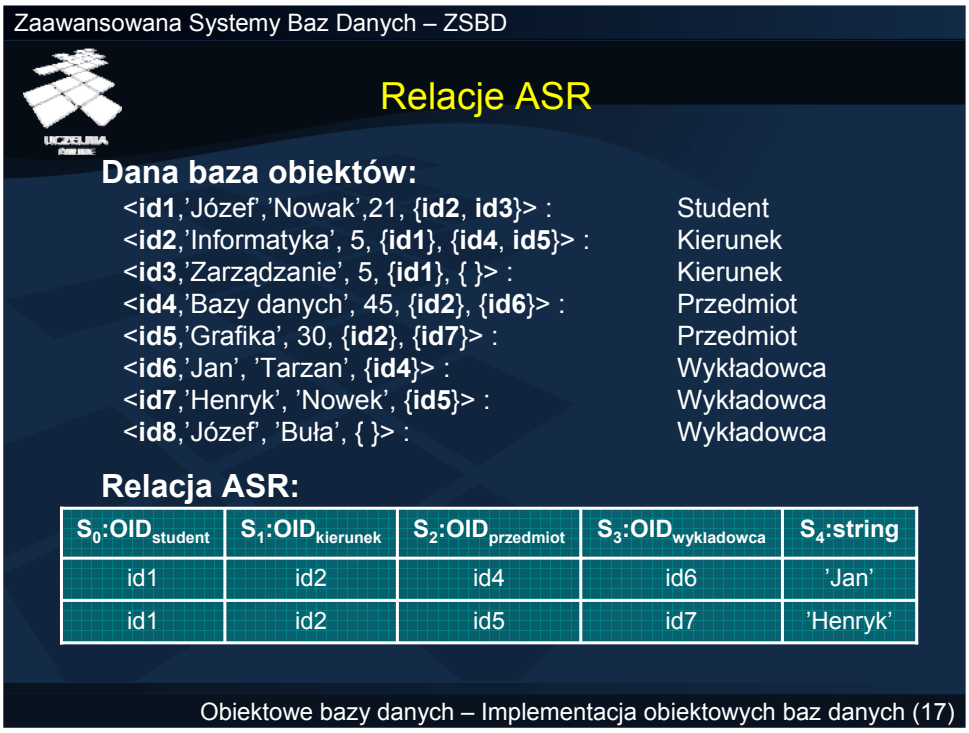
Jednym z rozwiązań wydajnej implementacji dużych obiektów jest ich organizacja w postaci rzadkiego indeksu o strukturze B-drzewa. Atrybutem na którym jest zakładany taki indeks jest numer pozycji danego bajtu w ciągu wszystkich bajtów składających się na duży obiekt. Liśćmi indeksu są strony dyskowe składające się na duży obiekt. Węzły wewnętrzne indeksu nie przechowują zawartości obiektu, lecz służą do wyszukiwanie bajtów o określonej lokalizacji. Dla zagwarantowania wydajności operacji modyfikacji wypełnienie stron dyskowych nie jest maksymalizowane. Tak jak w zwykłych B-drzewach algorytmy utrzymania indeksu gwarantują wypełnienie liści w granicach od 50 do 100%. Dzięki temu operacje powiększania dużego obiektu przez wstawienie w określonej pozycji nowego podciągu bajtów lub usunięcie z obiektu podciągu o określonej pozycji i długości, wymagają jedynie lokalnych rekonstrukcji ograniczonej liczby stron dyskowych.

Opisywana struktura została zilustrowana na slajdzie. Przedstawiony duży obiekt składa się z sześciu stron dyskowych. Maksymalny rozmiar stron to 512 bajtów. Strony te zawierają kolejne fragmenty całego obiektu o długościach odpowiednio: 450, 500, 379, 400, 510 i 496 bajtów. Dwa wyższe poziomy indeksu składają się z trzech węzłów wewnętrznych. Na slajdzie pokazano operację wyszukiwania pojedynczego bajtu, który jest 1800 bajtem obiektu. W korzeniu indeksu pozycja ta jest przeliczona na pozycję w prawym poddrzewie indeksu. Lewe poddrzewo indeksuje bajty począwszy od 1 do 1329. Stąd względna pozycja szukanego bajtu w prawym poddrzewie jest różnicą bezwzględnej pozycji szukanego bajtu: n=1800 i liczby bajtów znajdujących się w lewym poddrzewie. Przeliczone względna pozycja w prawym poddrzewie jest równa n'=461. Z kolejnego przeliczenia w prawym węźle wewnętrznym drugiego poziomu wynika, że poszukiwany bajt jest 61 bajtem piątej strony dyskowej całego obiektu.




Ścieżki są wieloczłonowymi wyrażeniami opisującymi nawigację wzdłuż powiązań między obiektami, w głąb atrybutów złożonych i do wyników metod bezparametrowych. Wartościami wyrażeń ścieżkowych są wartości proste lub obiekty, których typ jest określony przez ostatni element ścieżki. Na slajdzie pokazano przykład wyrażenia ścieżkowego, które dla uczelnianej bazy danych pozwala znaleźć imiona wykładowców, którzy wykładają przedmioty na kierunku, na którym studiuje dany student. Kolejne człony zdefiniowanej ścieżki opisują przejścia między kolejnymi zbiorami obiektów: kierunki, na których studiuje student, przedmioty, które są wykładane na tych kierunkach, wykładowcy, którzy wykładają te przedmioty i na końcu zbiór imion tych wykładowców.

Szybka nawigacja wzdłuż długich wyrażeń ścieżkowych wymaga wydajnej implementacji struktur dostępu do wartości wyrażeń ścieżkowych. Znane są dwa rozwiązania tego problemu. Pierwsze polegające na materializacji kompletnych ścieżek dostępu przez zapamiętanie wszystkich potencjalnych dróg nawigacji ze zbioru obiektów reprezentujących pierwszy człon ścieżki do zbioru obiektów osiągalnego za pomocą ścieżki i reprezentujących ostatni człon ścieżki. Zmaterializowane ścieżki są składowane w tak zwanych relacjach ASR (ang. Access Support Relations). Drugie rozwiązanie polega na materializacji hierarchii wszystkich kombinacji ścieżek binarnych w wyrażeniu ścieżkowym pamiętających tylko początki i końce ścieżek, bez członów pośrednich. Obydwa rozwiązania dopuszczają by definiowane ścieżki obejmowały nawigacje wzdłuż związków wielokrotnych i atrybutów wielowartościowych.



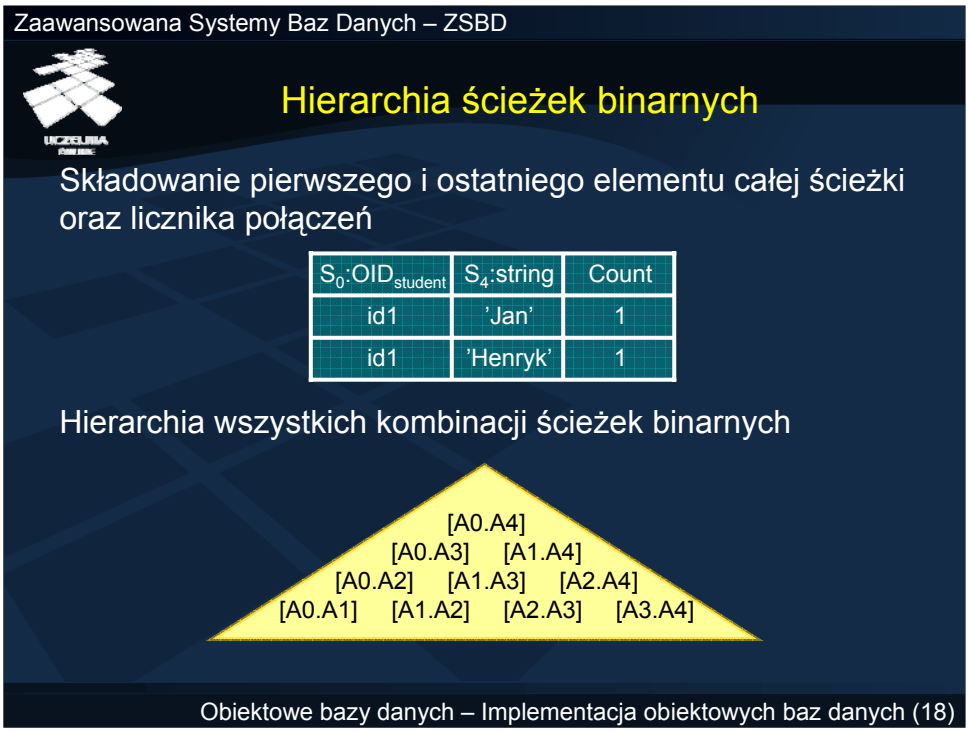

Na slajdzie pokazano przykładową bazę danych zawierającą cztery kolekcje obiektów w uczelnianej bazie danych: studentów, kierunki studiów, wykładane przedmioty i wykładowców. Z przedstawionego stanu bazy danych wynika, że student Józef Nowak studiuje równolegle na dwóch kierunkach studiów: Informatyce i Zarządzaniu. Na kierunku Informatyka są wykładane dwa przedmioty: Bazy danych i Grafika. Wykładowca Jan Tarzan wykłada Bazy danych, a wykładowca Henryk Nowek – grafikę.

Na dole slajdu pokazano przykładową strukturę relacji ASR materializującą ścieżki nawigujące od Studentów, przez Kierunki studiów, wykładane przedmioty i wykładowców do imion wykładowców. Są tylko dwie kompletne ścieżki. Pierwsza prowadzi od studenta Józefa Nowaka przez kierunek studiów Informatyka, przedmiot Bazy danych, wykładowcę Jana Tarzana do imienia Jan. Początkiem drugiej ścieżki jest również student Józef Nowak. Prowadzi ona przez kierunek studiów Informatyka, przedmiot Grafiką, wykładowcę Henryka Nowka, do imienia wykładowcy: Henryk.

Technologia relacji ASR przewiduje możliwość pamiętania również niekompletnych ścieżek. W podanym przykładzie niekompletną ścieżką jest ścieżka prowadząca od studenta Józefa Nowaka i kończąca się na kierunku Zarządzania. Kierunek ten nie przypisanych żadnych przedmiotów, więc ścieżka kończy się przedwcześnie.

Relacje ASR pozwalają na wydajną nawigację wzdłuż wszystkich ścieżek, które są fragmentami maksymalnej ścieżki zdefiniowanej relacji ASR. Na przykład, relacja ASR może być zastosowana dla przyśpieszenia nawigacji wzdłuż ścieżki wyszukującej imion wykładowców pracujących na kierunku Informatyka, która to ścieżka jest częścią maksymalnej ścieżki zdefiniowanej w relacji ASR.

Dla przyśpieszenia dostępu do relacji ASR na obydwu końcach relacji, to jest na pierwszym i ostatnim atrybucie relacji są zakładane dotykowe struktury fizyczne, na przykład indeksy typu B-drzewo.



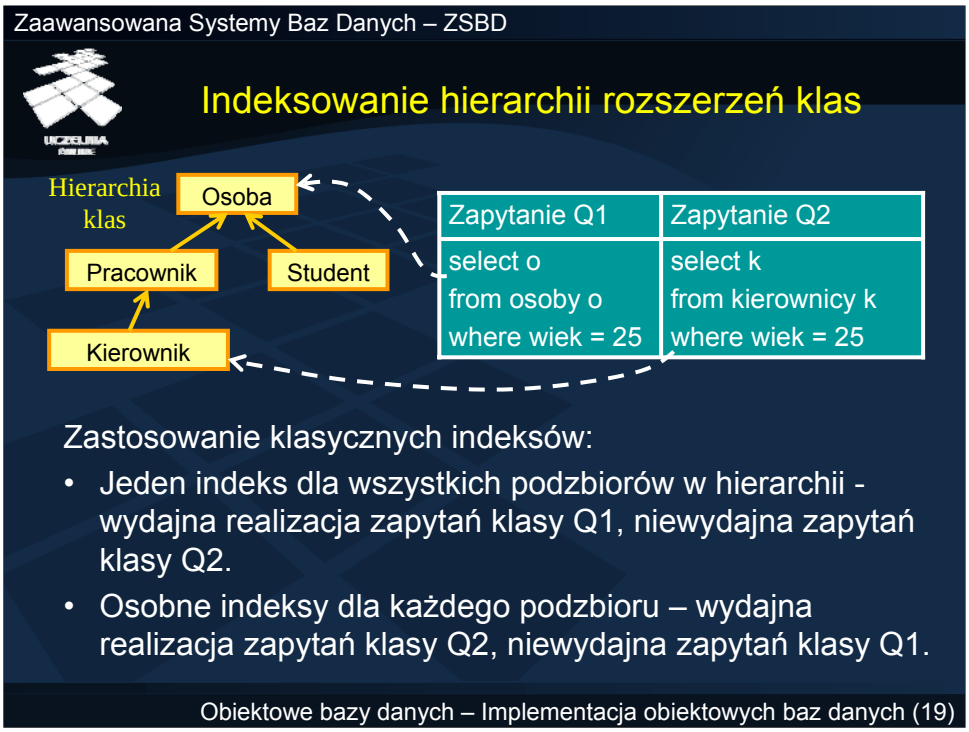

Alternatywą dla relacji ASR są hierarchie kombinacji ścieżek binarnych. W rozwiązaniu tym, w binarnej relacji ścieżek jest materializowany tylko pierwszy i ostatni element ścieżki. Ponieważ może być wiele ścieżek prowadzących od tego samego źródła do tego samego celu ścieżki, dodatkowo jest pamiętana liczba tych ścieżek.

Na slajdzie pokazano przykład relacji binarnej dla bazy danych z poprzedniego slajdu. Relacja ta będzie zawierała dwie krotki reprezentujące ścieżki prowadzące od studenta Józefa Nowaka do imion jego wykładowców. Obydwie ścieżki są jednokrotne.

Zwróćmy uwagę, że pokazana relacja binarna pozwoli na przyspieszenie nawigacji jedynie wzdłuż kompletnych ścieżek. Przechowywane w niej dane będą nieprzydatne dla przyśpieszenia nawigacji w zapytaniach zawierających jedynie fragment maksymalnej ścieżki. Dlatego kompletne rozwiązanie obejmuje całą hierarchię relacji binarnych dla fragmentów ścieżek. W ogólności, hierarchia ta nie musi być kompletna. Wystarczy, że będzie ona zawierać tylko i wyłącznie relacje binarne, które będą przydatne w konkretnym zastosowaniu, dla określonego podzbioru wszystkich potencjalnych ścieżek zapytań.

Przykład na slajdzie pokazuje kompletną hierarchię relacji binarnych dla maksymalnej ścieżki składającej się z pięciu elementów: A0.A1.A2.A3.A4. Najniższy poziom hierarchii zawiera relacje binarne dla wszystkich ścieżek długości dwóch elementów, które są fragmentem ścieżki maksymalnej. Następny poziom zwiera relacje binarne dla wszystkich ścieżek o długości trzech elementów. Najwyższy poziom zawiera relację binarną dla maksymalnej ścieżki.

Tak jak dla relacji ASR również, w wypadku hierarchii relacji binarnych na każdej relacji zakłada się dwukierunkowe szybkie struktury dostępu.

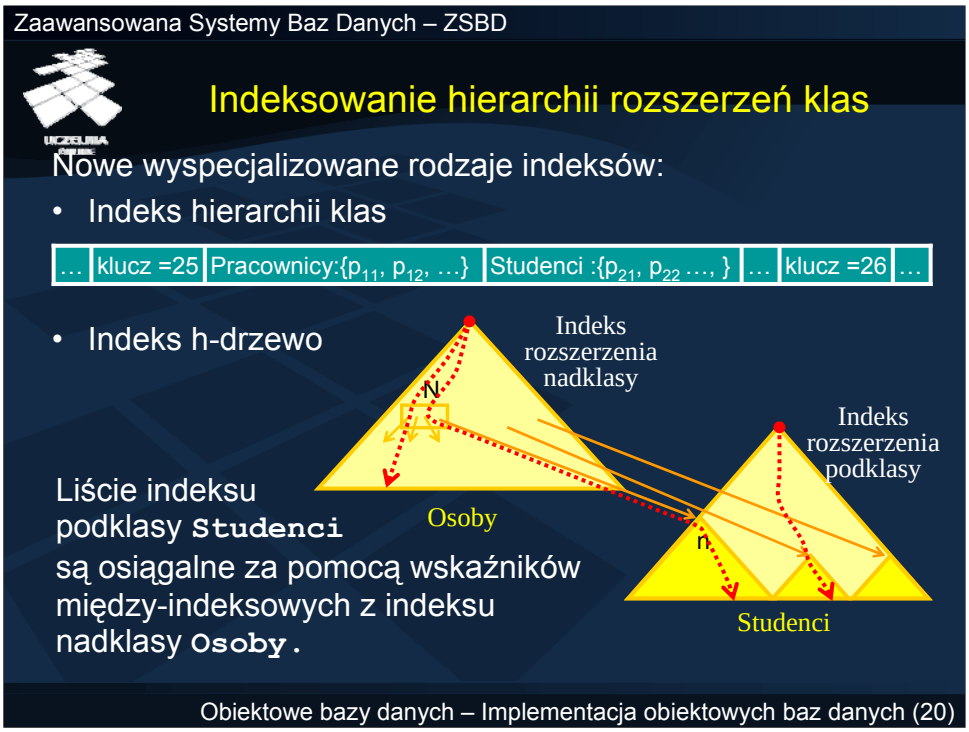



W obiektowych bazach danych typowe są zapytania wykonywane na heterogenicznych kolekcjach obiektów. Przykład takich zapytań przedstawiono na slajdzie. Zapytania adresują heterogeniczny zbiór osób obejmujący: kierowników, pracowników, którzy nie są kierownikami, studentów i pozostałe osoby, które nie są ani pracownikami, ani studentami. Zapytanie Q1 wyszukuje w tej heterogenicznej kolekcji, osób w wieku poniżej 30 lat. Wynikiem zapytania mogą być kierownicy, pracownicy, którzy nie są kierownikami, studenci oraz osoby nie będące ani pracownikami, ani studentami. Z kolei zapytanie Q2 dotyczy wyspecjalizowanego podzbioru osób, które pracują na stanowisku kierowniczym.

Dostępne są dwie alternatywy zastosowania klasycznych indeksów dla przyśpieszenia wykonywania przedstawionych zapytań. Pierwsze rozwiązanie polega na utworzeniu jednego wspólnego indeksu dla wszystkich podzbiorów w hierarchii. Taki indeks będzie przydatny dla efektywnej realizacji zapytań klasy Q1, natomiast nie będzie przydatny dla zapytań klasy Q2. W tym drugim przypadku duża część wskazywanych przez indeks obiektów spełniających zadane kryterium wiekowe, nie będzie spełniała kryterium typu danych. Ponieważ indeks wskazuje również osoby, które nie są kierownikami.

Drugie potencjalne rozwiązanie polega na utworzeniu osobnych indeksów dla każdego podzbioru obiektów. Przeciwnie niż dla poprzedniego rozwiązania, zbiór indeksów będzie dobrze wspierał wydajną realizację zapytań klasy Q2, a znacznie gorzej będzie wspierał zapytania klasy Q1. Znalezienie wszystkich osób w wieku 25 lat będzie wymagało przejrzenia wszystkich czterech indeksów.

Z powyższej analizy wynika, że potrzebne są nowe rozwiązania, które będą dedykowane dla heterogenicznych kolekcji danych i, które będą przydatne dla dowolnej klasy selektywnych zapytań na takiej kolekcji.

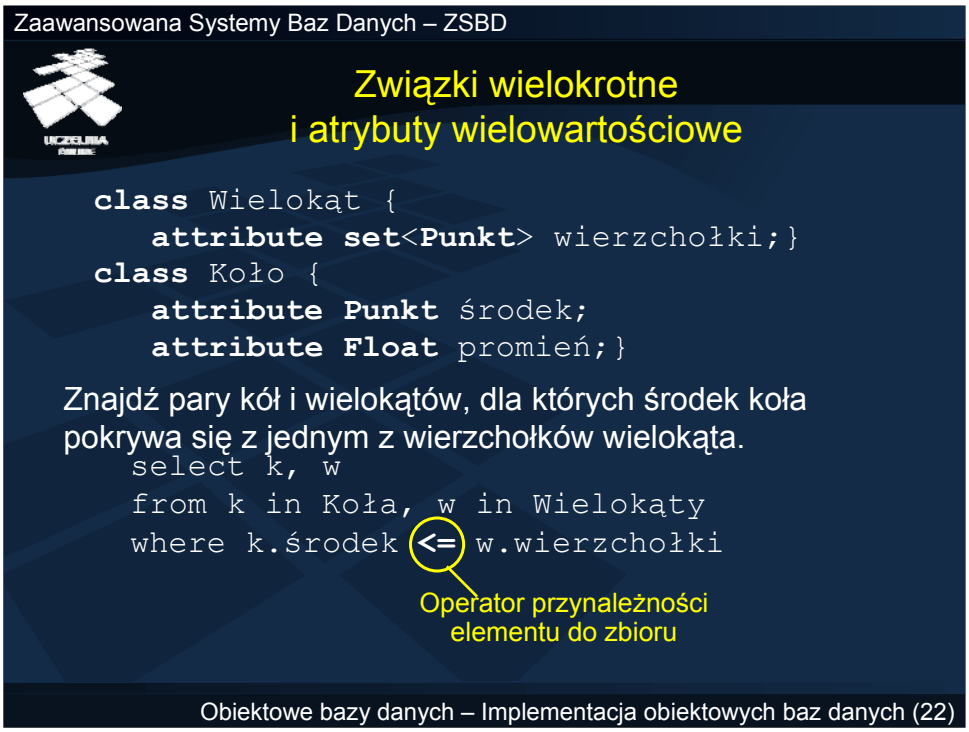



Na slajdzie przedstawiono dwie dedykowane metody wydajnego wyszukiwania obiektów w hierarchiach rozszerzeń klas. Obydwie polegają na modyfikacjach struktury indeksu typu B-drzewo.

Pierwsze rozwiązanie polega na rozszerzeniu struktury liści indeksu w celu przechowywania dodatkowej informacji o typie każdego indeksowanego obiektu. Ogólna struktura informacji przechowywanej w liściach indeksu jest następująca. Z każdym kluczem jest związana lista zbiorów wskaźników na obiekty o danej wartości klucza. Elementy listy odpowiadają poszczególnym podzbiorom danych. Na slajdzie pokazano kawałek liścia takiego indeksu założonego na atrybucie *„wiek”*. Z kluczem o wartości równej 25 lat jest związana lista zbiorów wskaźników *„p”*. Wskaźniki są pogrupowane w zależności od klasy obiektów których dotyczą. Dwie pierwsze grupy to wskaźniki na pracowników i studentów.

Drugim rozwiązaniem są tak zwiane h-drzewa. H-drzewo jest hierarchią B-drzew odpowiadającą hierarchii podzbiorów. Korzeniem tej hierarchii jest indeks odpowiadający klasie bazowej całej hierarchii klas, z którego są osiągalne wszystkie obiekty. Z indeksów ulokowanych niżej w hierarchii osiągalne są tylko obiekty należące do odpowiedniej pod-hierarchii klas. Indeksy ulokowane liściach hierarchii adresują obiekty będące wystąpieniami pojedynczych klas. Wynika stąd, że zakresy zasięgów indeksów z różnych poziomów hierarchii pokrywają się. Natomiast zakresy zasięgów indeksów znajdujących się na tym poziomie hierarchii są rozłączne. W pośrednich węzłach indeksów występują dwa rodzaje wskaźników: wskaźniki na węzły niższego poziomu w tym samym indeksie i wskaźniki na węzły w innym indeksie znajdującym się na niższym poziomie hierarchii indeksów.

Przykładowe H-drzewo zostało pokazane na slajdzie. Korzeniem hierarchii indeksów jest indeks odpowiadający klasie „*Osoba”*. Z tego indeksu są osiągalne zarówno wystąpienia klasy „*Osoba”*, jak i wystąpienia klasy „*Student”*. Indeksem niższego poziomu jest indeks odpowiadający klasie „*Student”*, która jest podklasą klasy „*Osoba”*. Z tego indeksu są osiągalne jedynie obiekty, które są wystąpieniami klasy „*Student”*. W indeksie odpowiadającym klasie „*Osoba”* znajduje się węzeł oznaczony przez „*N”*, który zawiera wskaźniki na inne węzły tego indeksu i dodatkowo zawiera wskaźnik na węzeł oznaczony przez „*n”*, który znajduje się wewnątrz indeksu odpowiadającego klasie „*Student”*. Na rysunku pokazano przykładowe ścieżki wyszukiwania. Dwie z nich zamykają się wewnątrz indeksów składowych H-drzewa. Środkowa ścieżka pokazuje nawigację między różnymi indeksami hierarchii.




Obiektowy model danych pozwala bezpośrednio modelować struktury zbiorowe, takie jak związki wielokrotne i atrybuty wielowartościowe. Ze strukturami zbiorowymi skojarzone są operacje logiczne na zbiorach, na przykład niepustego przecięcia, podzbioru lub przynależności elementu do zbioru.

Na slajdzie pokazano przykład operacji na zbiorach dotyczącej atrybutów wielowartościowych. Definicja klasy Wielokąt zawiera atrybut wielowartościowy będący zbiorem punktów, który reprezentuje zbiór wierzchołków wielokąta. Definicja klasy Koło obejmuje z kolei jednowartościowy atrybut typu Punkt, który reprezentuje środek koła.

Pokazane na dole slajdu zapytanie wyszukuje w bazie danych pary Koło i Wielokąt takie, że środek koła pokrywa się z jednym z wierzchołków wielokąta. Zapytanie to zawiera operator zbiorowy, testowania przynależności elementu do zbioru.

Fizyczna implementacja zapytań zawierających operację łączenie kolekcji obiektów, w której warunek połączeniowy zawiera operacje porównania zbiorów jest bardzo czasochłonna z dwóch podstawowych powodów. Po pierwsze, realizacja takich operacji wymaga porównania wszystkich elementów dwóch łączonych kolekcji obiektów, i dodatkowo dla każdej pary porównywanych obiektów porównania wszystkich elementów zbiorów, to jest atrybutów wielowartościowych lub związków wielokrotnych. Dla dwóch kolekcji obiektów o rozmiarach M i N, i dla średniej liczebności atrybutów zbiorowych lub związków wielokrotnych równej O i P, złożoność kombinatoryczna operacji połączenia wynosi: M∗N∗O∗P. Drugą przyczyną czasochłonności takich operacji połączeń jest niemożność zastosowania w tym wypadku szybszych metod wykonywania połączeń, takich jak metody sort-merge lub hash-join. Metoda sort-merge nie może być zastosowana w tym wypadku ze względu na brak relacji porządkującej zbiór zbiorów wartości lub związków. Natomiast metoda hash-join nie może być zastosowana ze względu na jej ograniczenie do warunków połączenia zawierających operator równości.

## Sygnatury zbiorów

- Każdy element zbioru jest odwzorowany przez unikalną sygnaturę
- Za pomocą sumowania logicznego sygnatury wszystkich elementów zbioru są składane w pojedynczą sygnaturę aproksymującą cały zbiór
- Dla operatora podzbioru sygnatury zbiorów muszą spełniać następującą zależność:

$$s \subseteq t \Rightarrow \text{sig}(s) \,\&\, \sim\text{sig}(t) = 0$$

Sig(Punkt A (4.35, 11.26)) $\rightarrow$ 00010001
Sig(Punkt B (7.10, 12.05)) $\rightarrow$ 10001000
Sig({Punkt A, Punkt B}) $\rightarrow$ 10011001



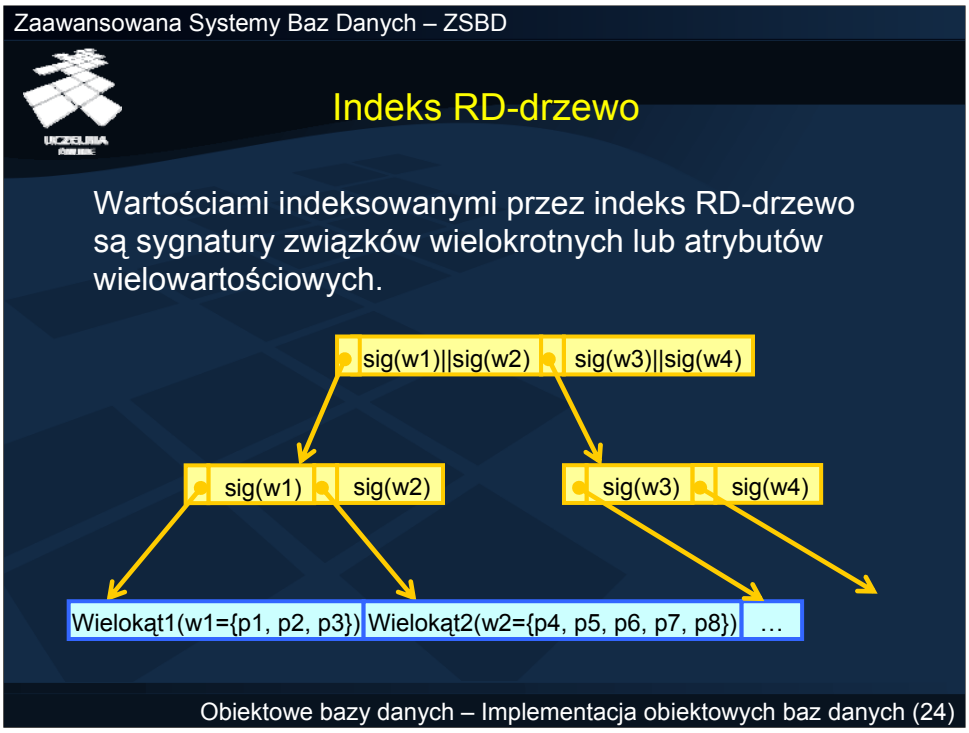
Metoda, która pozwala na wydajną realizację operacji porównania zbiorów zastępuje zbiory wartości lub zawiązków specjalnymi wartościami numerycznymi nazywanymi sygnaturami.

Dla każdego elementu zbiorów składowanych w obiektach, czyli pojedynczej wartości atrybutu wielowartościowego lub pojedynczej referencji związku wielokrotnego, jest wyznaczana unikalna sygnatura. Równość elementów oznacza równość ich sygnatur.

Następnie za pomocą logicznego sumowania sygnatur wszystkich elementów zbiorów są tworzone zbiorcze sygnatury dla całych zbiorów. Sumaryczne sygnatury są prostymi wartościami liczbowymi, dzięki czemu można je łatwo porównywać i porządkować, co pozwala na zastosowanie wydajnych metod łączenia, takich jak wspomniane metody sort-merge lub hash-join.

Specyficzne zależności zachodzące między wartościami sygnatur zbiorów oznaczają określone zależności zachodzące między samymi zbiorami. Na slajdzie pokazano zależność, która musi zachodzić między sygnaturami zbiorów, by między tymi zbiorami zachodziła relacja podzbioru. W tym konkretnym wypadku zależność, w której iloczyn logiczny sygnatury zbioru „*s*”, z negacją sygnatury zbioru „*t*” jest równa zero, oznacza, że zbiór „*t*” jest podzbiorem zbioru „*s*”.

Na dole slajdu pokazano przykład utworzenia sygnatury dla zbioru punktów A i B, którym przypisano dwie dowolnie ustalone sygnatury. Dla uzyskanej sygnatury zbioru punktów A i B można zweryfikować na podstawie zdefiniowanej wyżej zależności, zastosowanej dla sygnatury zbioru punktów A i B oraz sygnatury punktu A, że zbiór punktów A i B zawiera w sobie punkt A.




Dla wydajnej realizacji zapytań odwołujących się do atrybutów wielowartościowych lub związków wielokrotnych stosuje się dodatkowe struktury danych bazujące na sygnaturach zbiorów. Jedną z takich struktur jest indeks nazywany RD-drzewem od angielskiego Russian Doll. Indeks ten jest zakładany na sygnaturach atrybutów wielowartościowych lub związków wielokrotnych. Indeks ten ma strukturę hierarchiczną i jest wzorowany na B-drzewie. Wartościami kluczy przechowywanymi w węzłach wewnętrznych są sygnatury utworzone przez operację sumy logicznej wszystkich sygnatur przechowywanych w węźle niższego poziomu wskazywanym przez wskaźnik skojarzony z daną zbiorczą sygnaturą.

Na rysunku pokazano fragment dwupoziomowego RD-drzewa założonego na atrybucie Wierzchołki obiektów z kolekcji Wielokąty. W liściach indeksu znajdują się sygnatury utworzone dla zbiorów wierzchołków poszczególnych wielokątów. Na przykład sygnatura sig(w2) reprezentuje zbiór pięciu wierzchołków: p4, p5, p6, p7 i p8 wielokąta o nazwie Wielokąt2. Sygnatury w korzeniu drzewa są utworzone z sygnatur przechowywanych w liściach drzewa. Na przykład pierwsza sygnatura składowana w korzeniu jest sumą logiczną sygnatur dwóch zbiorów wierzchołków w1 i w2.