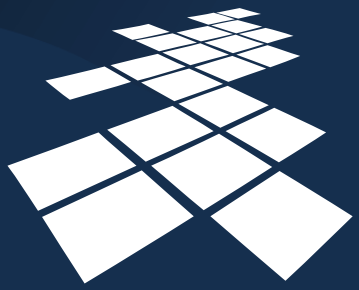
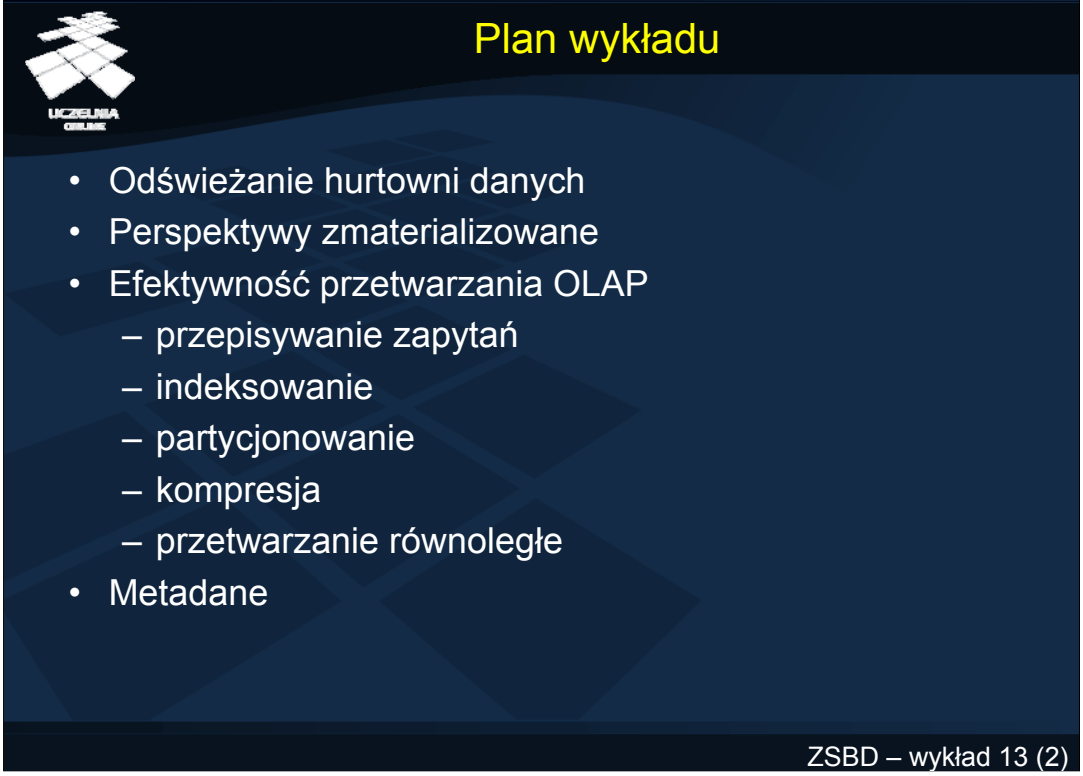
Hurtownie danych – 2

Zagadnienia implementacyjne i efektywność przetwarzania OLAP

Wykład przygotował:
Robert Wrembel



UCZELNIA
ONLINE



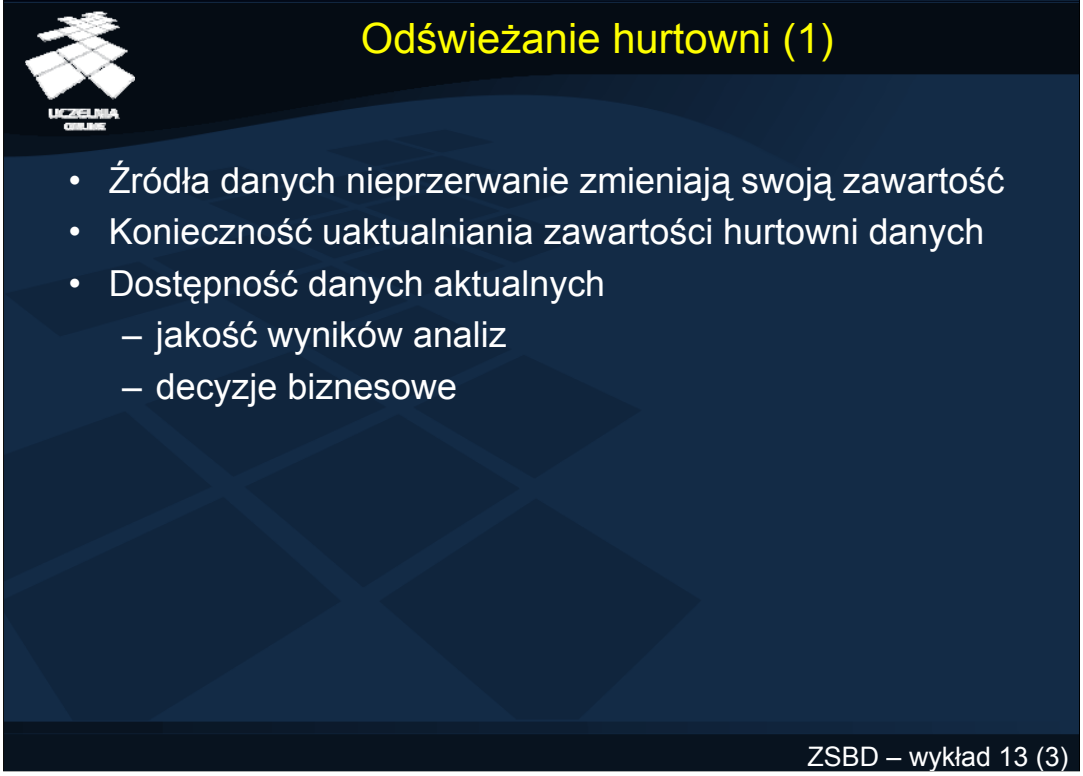
Celem wykładu jest omówienie podstawowych zagadnień związanych z implementacją hurtowni danych. W ramach wykładu zostanie przedstawiona następująca problematyka:

- odświeżania hurtowni danych w czasie jej pracy,

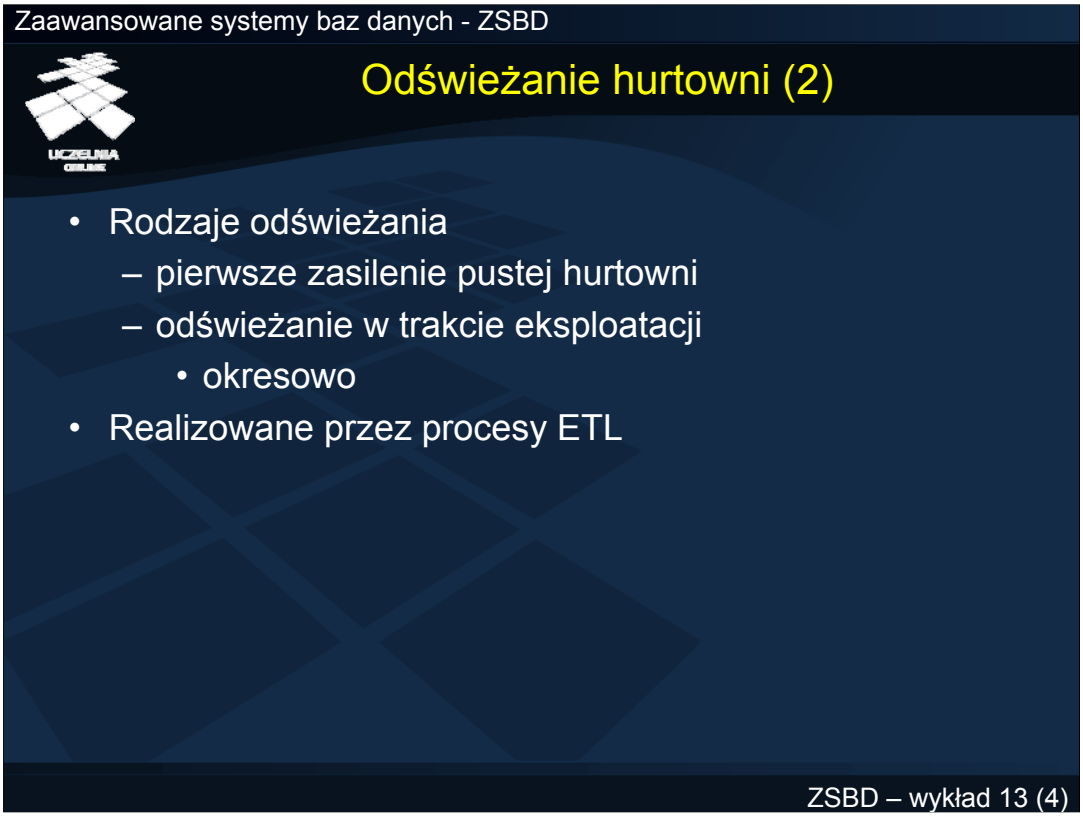
- wykorzystanie perspektyw zmaterializowanych do implementowania hurtowni,

- techniki zwiększające efektywność przetwarzania analitycznego, m.in. przepisywanie zapytań w oparciu o perspektywy zmaterializowane, indeksowanie danych przy użyciu różnych struktur, partycjonowanie danych i indeksów, kompresja danych i indeksów, przetwarzanie równoległe,

- metadane opisujące hurtownię.

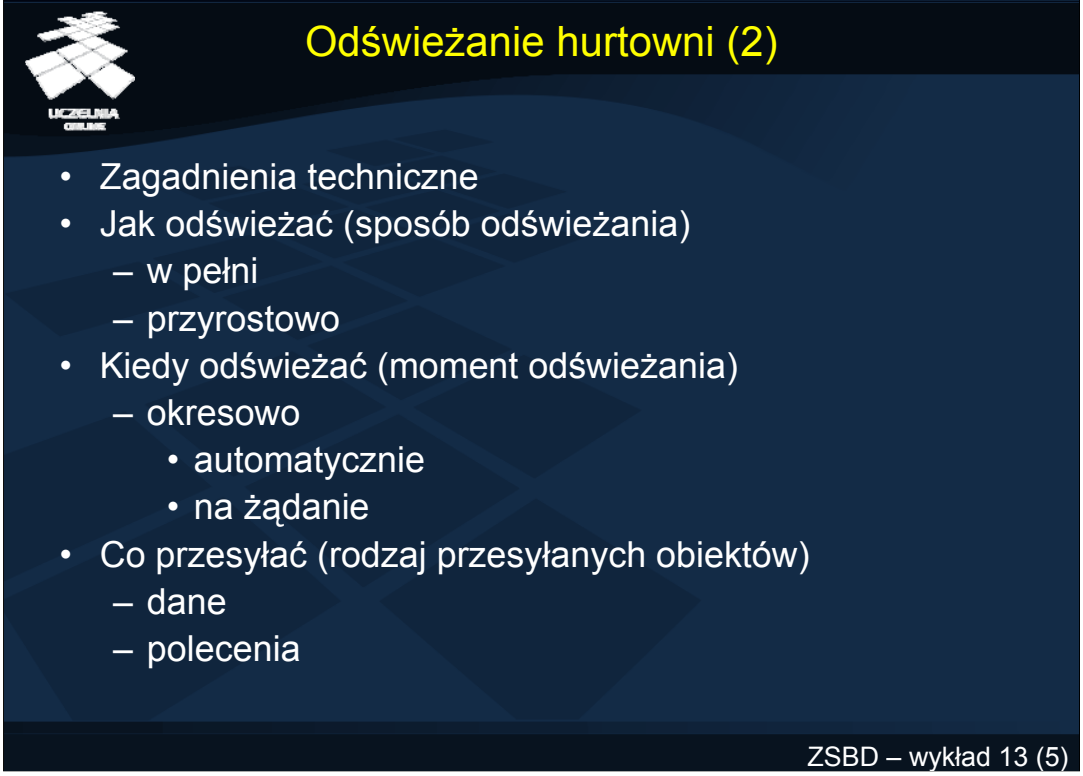


Hurtownia danych integruje dane ze źródeł, których zawartość podlega nieustannym zmianom (np. systemy obsługi bieżącej banku rejestrują nieprzerwanie nowe transakcje). Z tego względu, zachodzi konieczność dostarczania do hurtowni danych aktualnych. Dostępność danych aktualnych ma kluczowy wpływ na jakość wyników pracy aplikacji analitycznych, działających na zawartości hurtowni. Złe analizy, nietrafione prognozy trendów, fałszywe wzorce zachowań klientów mogą prowadzić decydentów do podjęcia złych decyzji inwestycyjnych, skutkujących poważnymi stratami finansowymi organizacji. Dlatego problem dostarczania aktualnych danych do hurtowni jest problemem równie ważnym, co jej właściwe zaprojektowanie i implementacja.



Po wdrożeniu hurtownia wymaga odświeżania jej zawartości. W praktyce, wyróżnia się dwa rodzaje odświeżania hurtowni, tj. pierwsze zasilenie, gdy hurtownia jest pusta bezpośrednio po jej zaprojektowaniu i okresowe odświeżanie w trakcie eksploatacji hurtowni.

Odświeżanie pierwszego jak i drugiego rodzaju jest realizowane przez procesy ETL.

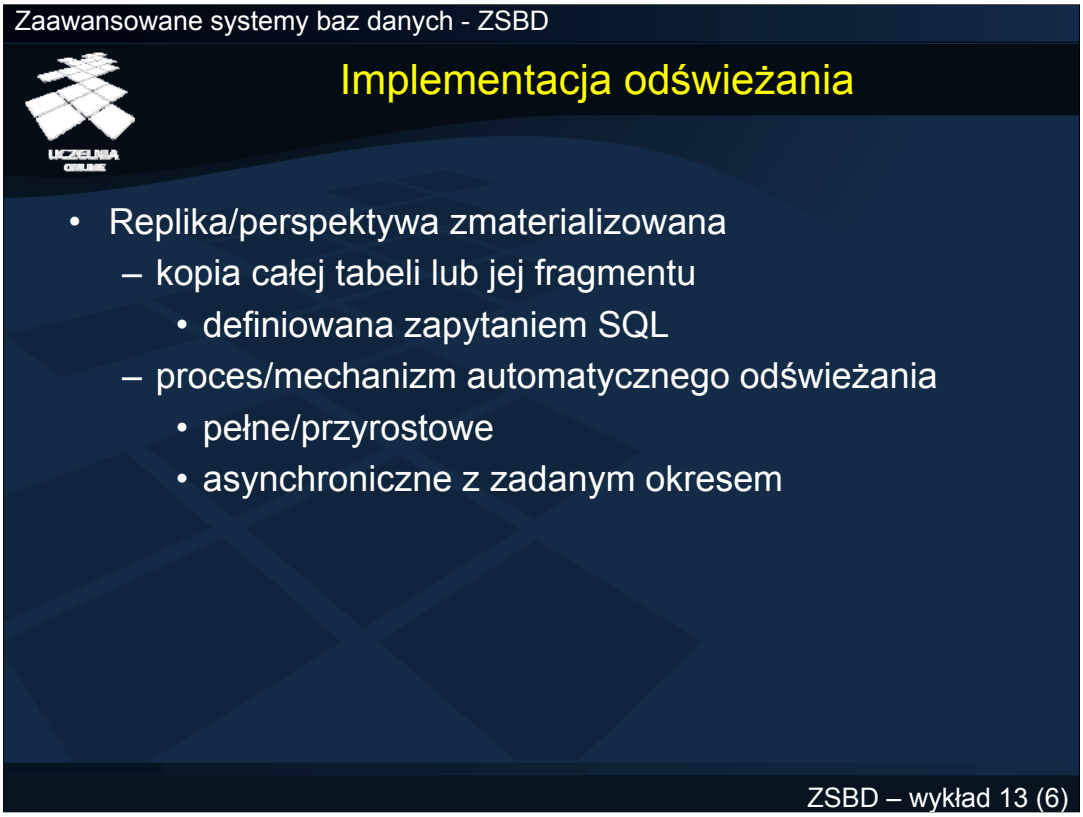


Z odświeżaniem w trakcie eksploatacji hurtowni wiążą się trzy podstawowe zagadnienia techniczne, tj. sposób odświeżania, moment odświeżania, rodzaj przesyłanych obiektów ze źródła do hurtowni.

Jeśli chodzi o pierwsze zagadnienie, to wyróżnia się odświeżanie pełne i odświeżanie przyrostowe. W odświeżaniu pełnym, ze źródła do hurtowni są przesyłane wszystkie dane wymagane do wypełnienia hurtowni. W odświeżaniu przyrostowym, ze źródła do hurtowni są przesyłane tylko dane nowe lub zmodyfikowane od czasu ostatniego odświeżenia.

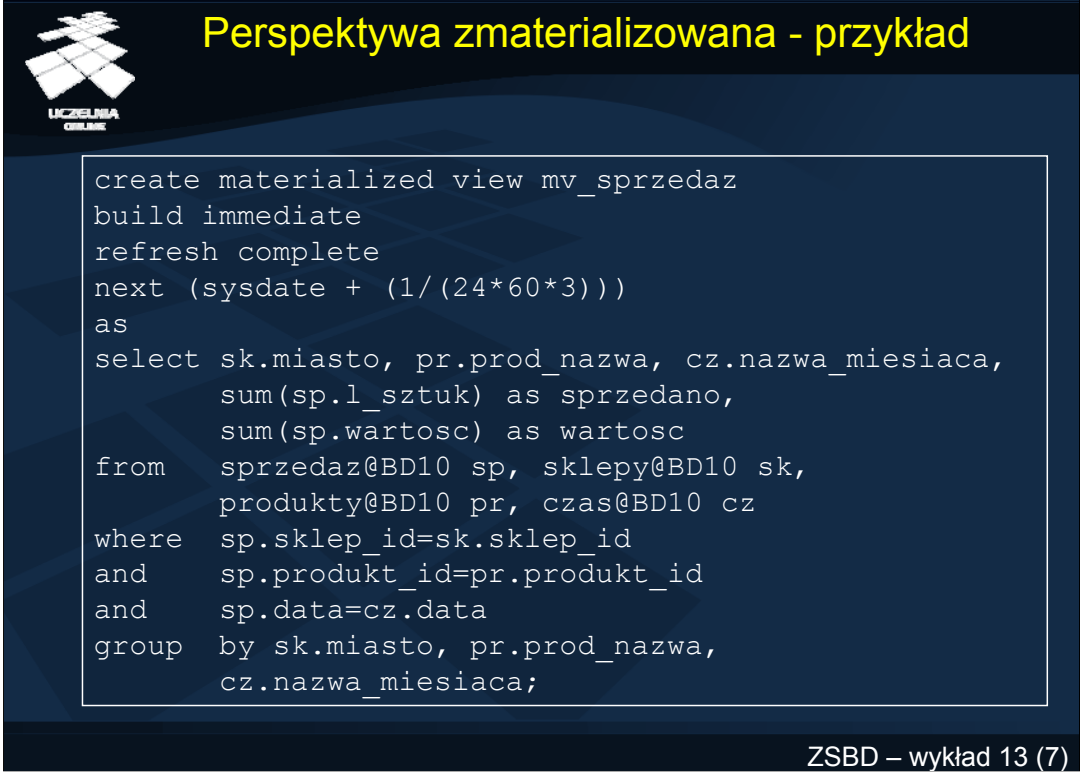
Jeśli chodzi o moment odświeżania, to w praktyce wykorzystuje się odświeżanie okresowe, albo inicjowane automatycznie przez procesy systemowe, albo inicjowane na żądanie użytkownika.

Jeśli chodzi o rodzaj przesyłanych obiektów, to w praktyce przesyła się ze źródła do hurtowni albo dane albo polecenia modyfikujące zawartość hurtowni.

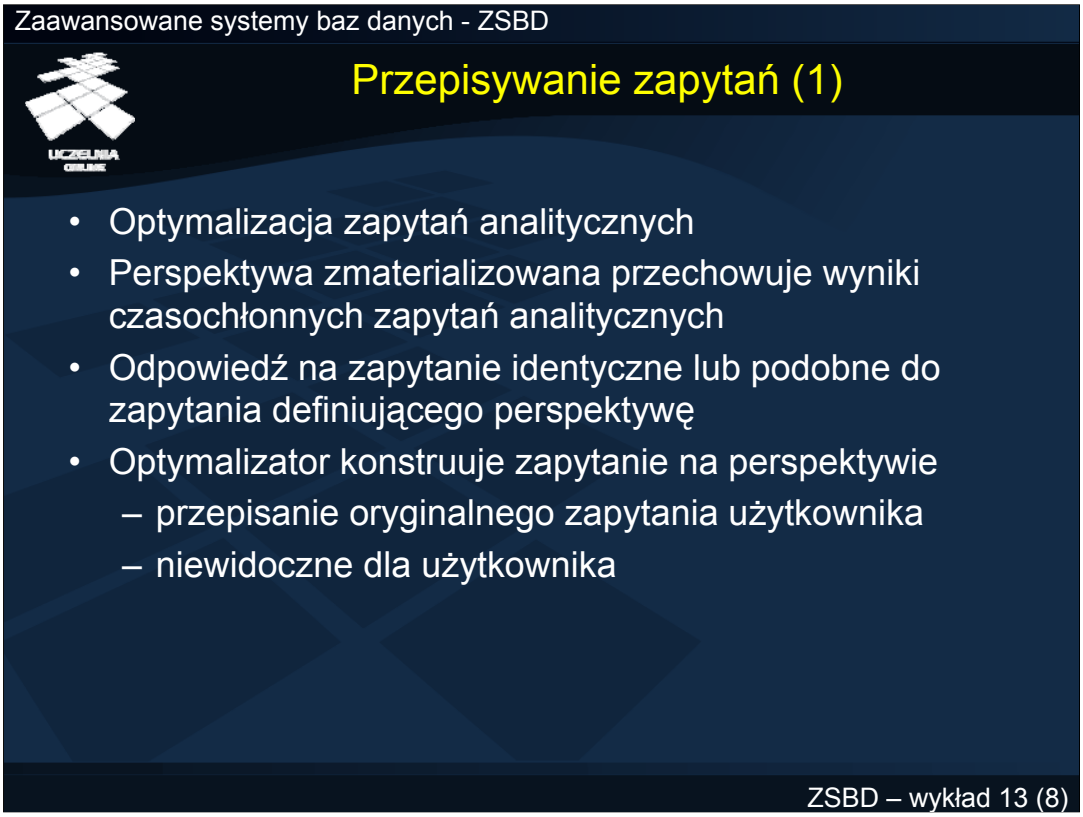


Implementacyjnie, odświeżanie hurtowni danych jest realizowane za pomocą tzw. replik, zwanych również perspektywami zmaterializowanymi (ang. materialized views). Perspektywa taka jest kopią albo całej tabeli znajdującej się w źródle danych, albo jej fragmentu, będącego podzbiorem atrybutów i/lub rekordów tabeli źródłowej. Struktura i zawartość perspektywy zmaterializowanej jest definiowana za pomocą zapytania SQL.

Z perspektywą jest związany systemowy proces realizujący automatyczne odświeżanie. W praktyce, perspektywy zmaterializowane są odświeżane przyrostowo lub w pełni, asynchronicznie, to znaczy z zadanym okresem odświeżania.

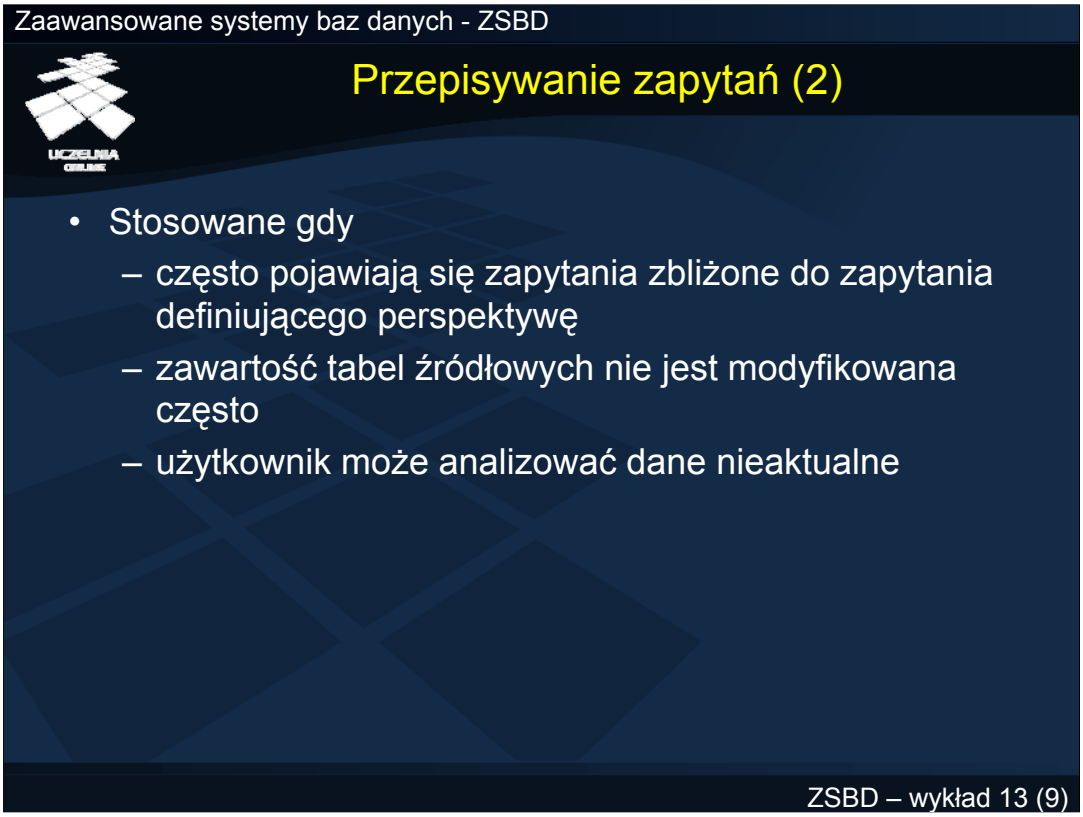


Na slajdzie przedstawiono przykładowe polecenie definiujące perspektywę zmaterializowaną o nazwie MV_SPRZEDAZ. Klauzula REFRESH COMPLETE zapewnia pełne odświeżanie perspektywy, a klauzula NEXT definiuje okres jej odświeżania (w przykładzie co 20 sekund). W klauzuli AS podano zapytanie określające strukturę perspektywy i jej zawartość. Perspektywa ta udostępnia atrybuty miasto , prod_nazwa i nazwa_miesiaca oraz agregaty sum(l_sztuk) i sum(wartosc) pochodzące z tabel źródłowych sprzedaz , sklepy , produkty i czas znajdujących się w zdalnym źródle o nazwie BD10 .



Oprócz replikowania danych, inną bardzo ważną i często stosowaną dziedziną zastosowań perspektyw zmaterializowanych jest optymalizacja zapytań analitycznych. Dla tych zastosowań perspektywy zmaterializowane służą do przechowywania wyliczonych danych (najczęściej zagregowanych), których wyznaczenie jest czasochłonne. Jeżeli w systemie pojawi się zapytanie, które może zostać wykonane z wykorzystaniem zmaterializowanych perspektyw, zamiast korzystania z tabel źródłowych, wówczas optymalizator zapytań skonstruuje odpowiednie zapytanie do tych perspektyw. Jest to tzw. przepisanie zapytania (ang. query rewriting). Proces ten jest niewidoczny dla użytkownika.

Materializowanie danych ma w tym przypadku sens jeżeli w systemie często pojawiają się zapytania identyczne lub podobne do tego, które występuje w definicji perspektywy. Dodatkowo, dane w tabelach źródłowych takiej perspektywy nie powinny ulegać częstemu modyfikowaniu.

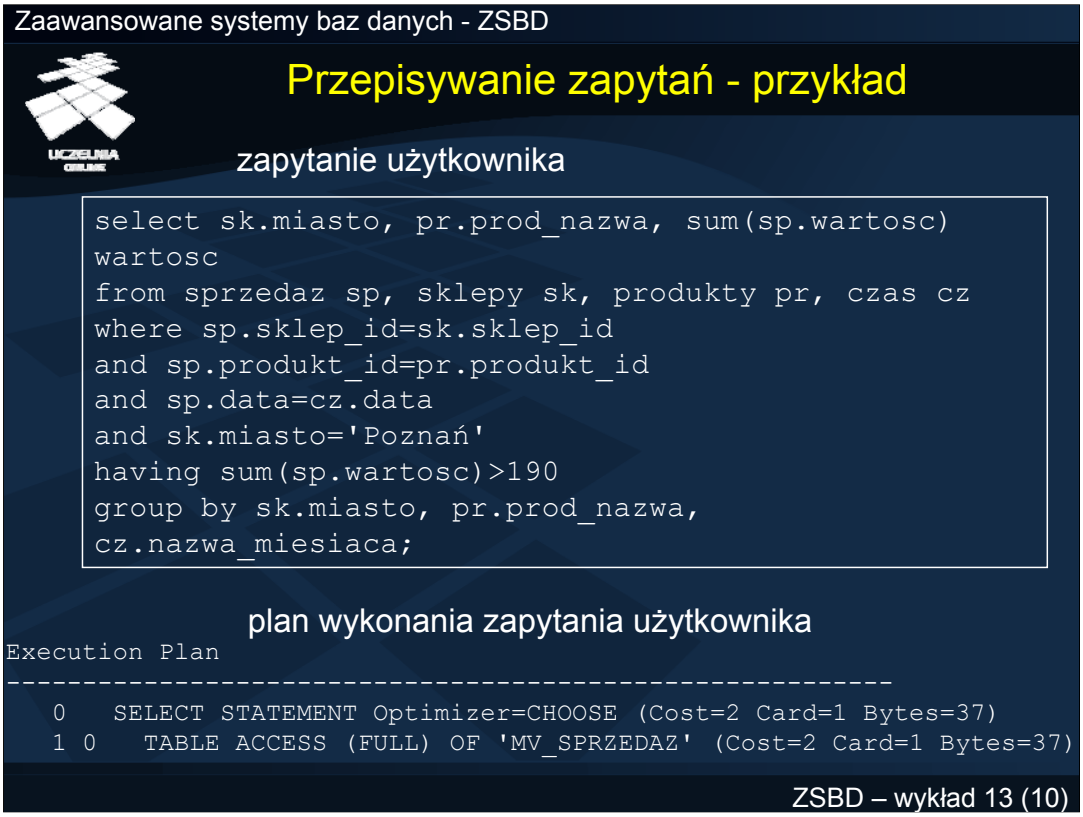


Materializowanie danych z wykorzystaniem perspektyw zmaterializowanych jest stosowane gdy:

1. w systemie często pojawiają się zapytania identyczne lub podobne do tego, które występuje w definicji perspektywy,

2. zawartość tabel źródłowych perspektywy zmaterializowanej jest modyfikowana rzadko,

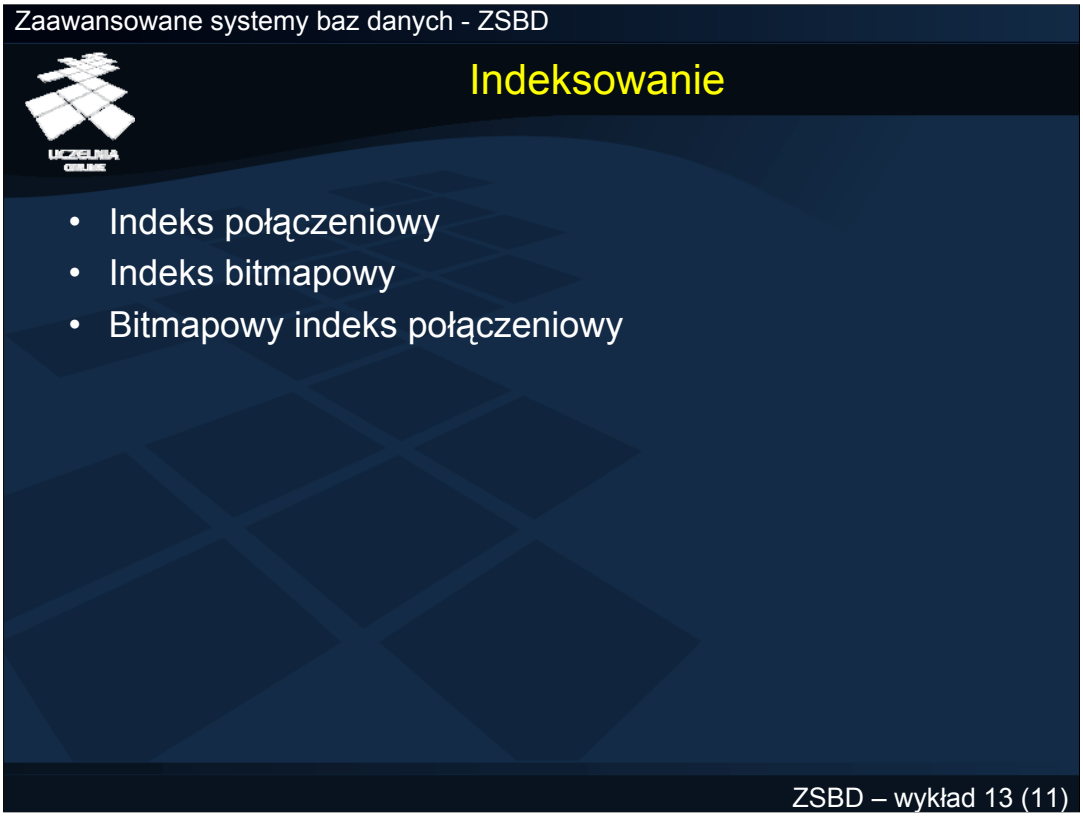
3. użytkownik zgadza się na przetwarzanie danych, które nie zawsze muszą być aktualne.



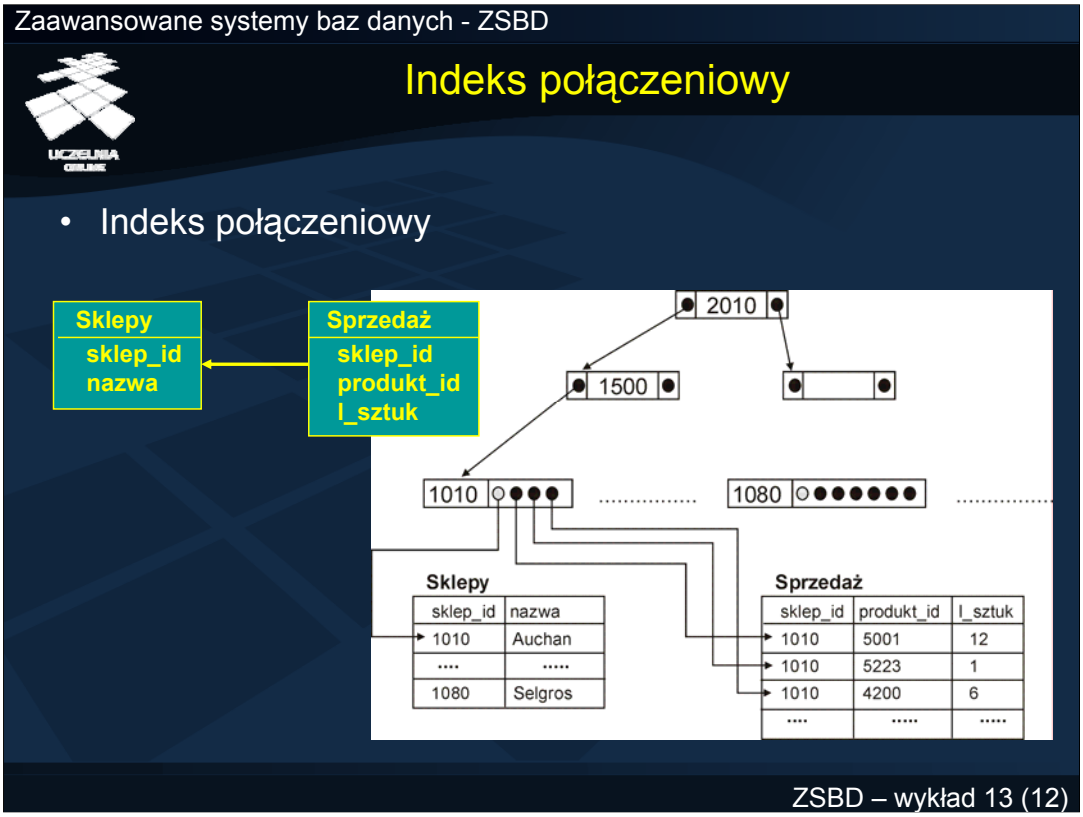
Jako przykład przepisania zapytania, rozważmy omówioną wcześniej perspektywę zmaterializowaną MV_SPRZEDAZ. Przyjmijmy, że użytkownik wyspecyfikował zapytanie, jak na slajdzie.

Zapytanie to oblicza sumę wartości sprzedaży poszczególnych produktów w poszczególnych miastach. Przy czym użytkownika interesuje sumaryczna sprzedaż tylko powyżej wartości 190.

Wyniki tego zapytania zostaną wyznaczone w oparciu o perspektywę MV_SPRZEDAZ, co potwierdza plan wykonania zapytania. W planie tym, pozycja TABLE ACCESS (FULL) OF 'MV_SPRZEDAZ' mówi, że oryginalne zapytanie użytkownika zostało zastąpione (przepisane) zapytaniem na perspektywie zmaterializowanej.



Oprócz materializowanych perspektyw i przepisywania zapytań, do optymalizacji zapytań analitycznych stosuje się różnego rodzaju specjalizowane struktury indeksowe. Najczęściej stosowanymi w praktyce są: indeksy połączeniowe, indeksy bitmapowe i bitmapowe indeksy połączeniowe.



Indeks połączeniowy (ang. join index) łączy z sobą rekordy z różnych tabel posiadające tę samą wartość atrybutu połączeniowego, jest więc strukturą zawierającą zmaterializowane połączenie wielu tabel. Indeks taki posiada strukturę B–drzewa zbudowanego na atrybucie połączeniowym tabeli (bądź na wielu takich atrybutach). Liście indeksu zawierają wspólne wartości atrybutu połączeniowego tabel wraz z listami adresów rekordów w każdej z łączonych tabel.

Slajd przedstawia strukturę indeksu połączeniowego zdefiniowanego na atrybucie sklep_id tabeli Sklepy . W tym przypadku, liście indeksu zawierają:

- wskaźniki do rekordów opisujących każdy ze sklepów

- adresy wszystkich rekordów z tabeli Sprzedaż opisujących sprzedaż danego sklepu.

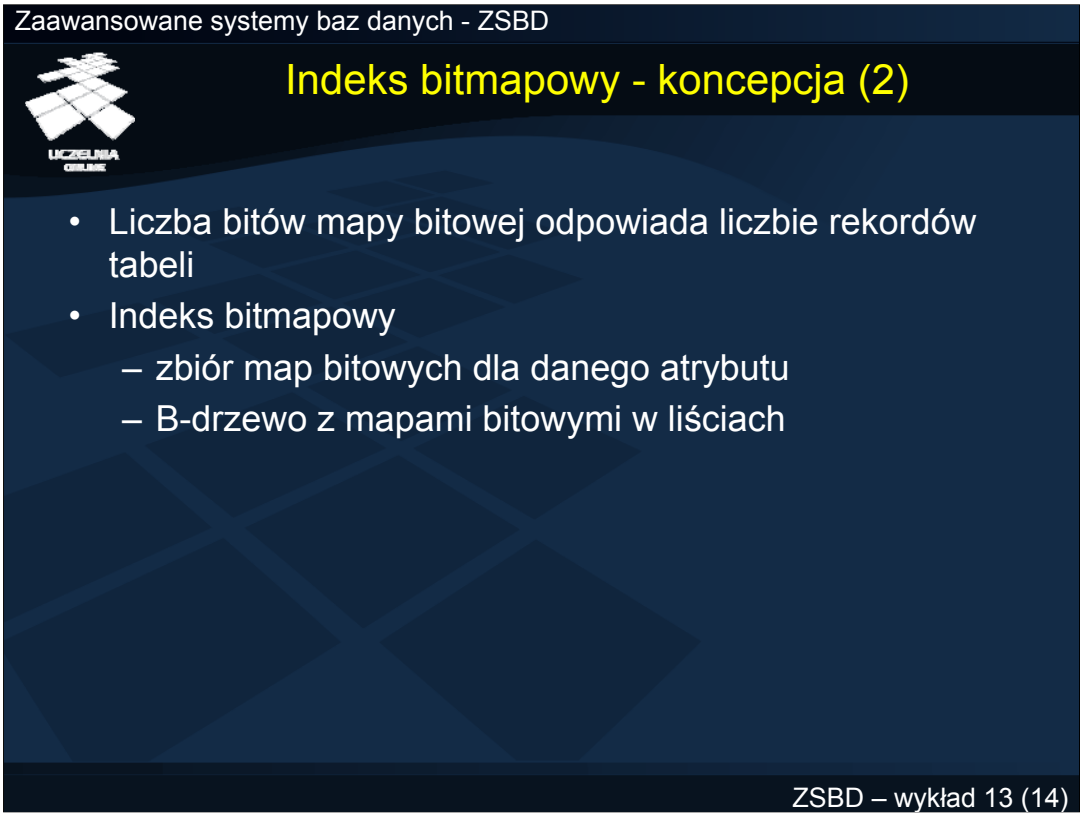
W przykładzie liść z wartością indeksowanego atrybutu sklep_id równą 1010 zawiera wskaźnik do rekordu w tabeli Sklepy opisującego sklep o tym numerze i listę wskaźników do rekordów tabeli Sprzedaż opisujących sprzedaż w sklepie o numerze 1010.

Ten przykładowy indeks przyspiesza wyszukiwanie danych na temat sprzedaży wskazanego sklepu.

Indeks bitmapowy - koncepcja (1)

- Mapa bitowa - budowana dla każdej wartości z dziedziny indeksowanego atrybutu
- Konkretny bit mapy odpowiada konkretnemu rekordowi tabeli
  - bit 1 - pierwszemu rekordowi, bit 2 - drugiemu itp.
- Mapa A='zielony'
  - bit n przyjmuje wartość 1 jeśli wartością atrybutu A n-tego rekordu jest 'zielony'
  - w przeciwnym przypadku bit n przyjmuje wartość 0

Ideą indeksu bitmapowego (ang. bitmap index) jest wykorzystanie pojedynczych bitów do zapamiętania informacji o tym, że dana wartość atrybutu występuje w określonym rekordzie tabeli. Dla każdej unikalnej wartości atrybutu jest przechowywana tablica bitów, zwana mapą bitową . Każdy bit mapy odpowiada jednemu rekordowy w tabeli R – bit pierwszy odpowiada pierwszemu rekordowi w tabeli R , bit drugi – drugiemu rekordowi itp. Dla mapy A = ’zielony ’ bit n przyjmuje wartość jeden, jeśli wartością atrybutu A rekordu o numerze n jest ‘zielony ’. W przeciwnym przypadku bit n przyjmuje wartość zero.

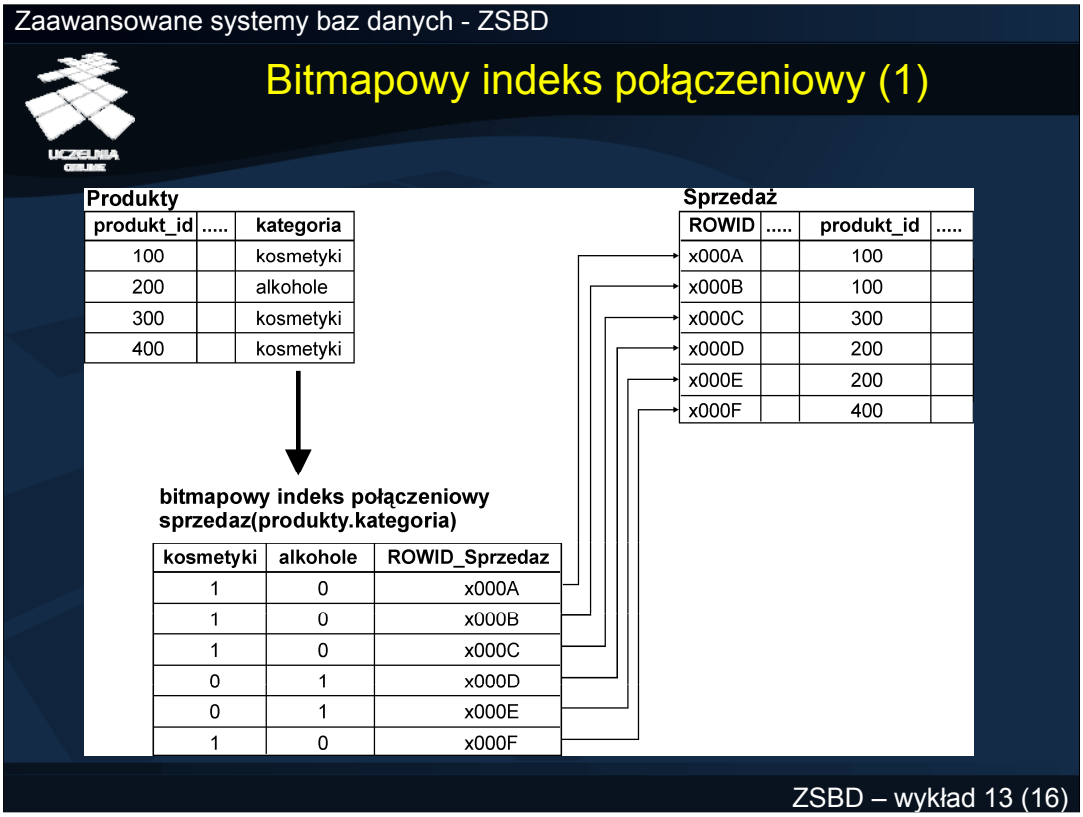


Liczba bitów mapy bitowej odpowiada liczbie rekordów tabeli R . Indeks bitmapowy jest zbiorem map bitowych dla wszystkich unikalnych wartości danego atrybutu. Indeks tego typu (w zależności od implementacji) może również posiadać strukturę B–drzewa, w którego liściach zamiast adresów rekordów są przechowywane mapy bitowe.

Indeks bitmapowy - przykład

typ	typ			
	coupe	*limuzyna*	*sedan*	*sport*
sedan	0	0	1	0
coupe	1	0	0	0
sport	0	0	0	1
limuzyna	0	1	0	0
sport	0	0	0	1
sport	0	0	0	1
sport	0	0	0	1
sport	0	0	0	1
sport	0	0	0	1
limuzyna	0	1	0	0
limuzyna	0	1	0	0
sport	0	0	0	1
sedan	0	0	1	0
sport	0	0	0	1

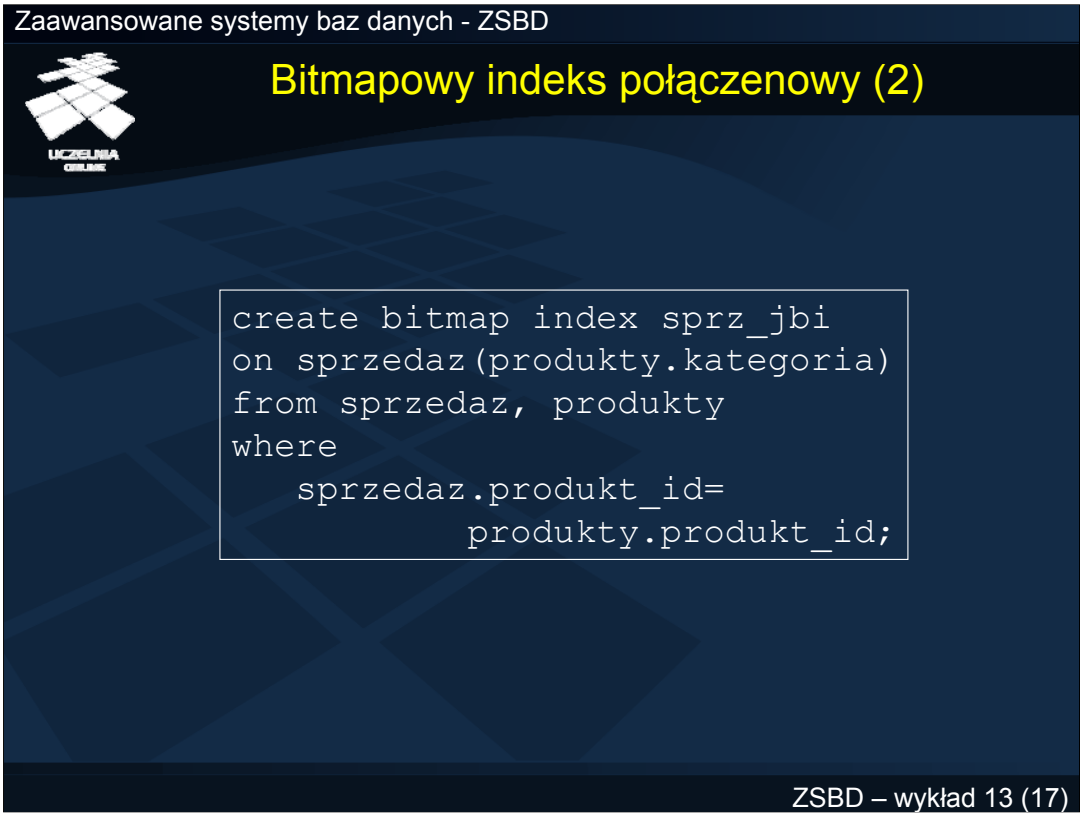
Przykład indeksu bitmapowego dla atrybutu typ przedstawiono na slajdzie. Ponieważ atrybut typ może przyjąć jedną z czterech wartości, tj. 'coupe', 'limuzyna', 'sedan', 'sport', więc indeks bitmapowy składa się z czterech map - po jednej mapie dla każdej wartości. Przykładowo, pierwszy bit mapy bitowej opisującej samochody 'coupe' przyjmuje wartość 0. Oznacza to, że wartością atrybutu typ pierwszego rekordu nie jest 'coupe'. Drugi bit tej mapy przyjmuje wartość 1, co oznacza, że wartością atrybutu typ drugiego rekordu jest 'coupe'.



Bitmapowy indeks połączeniowy (ang. bitmap join index) łączy w sobie koncepcję indeksu połączeniowego i bitmapowego.

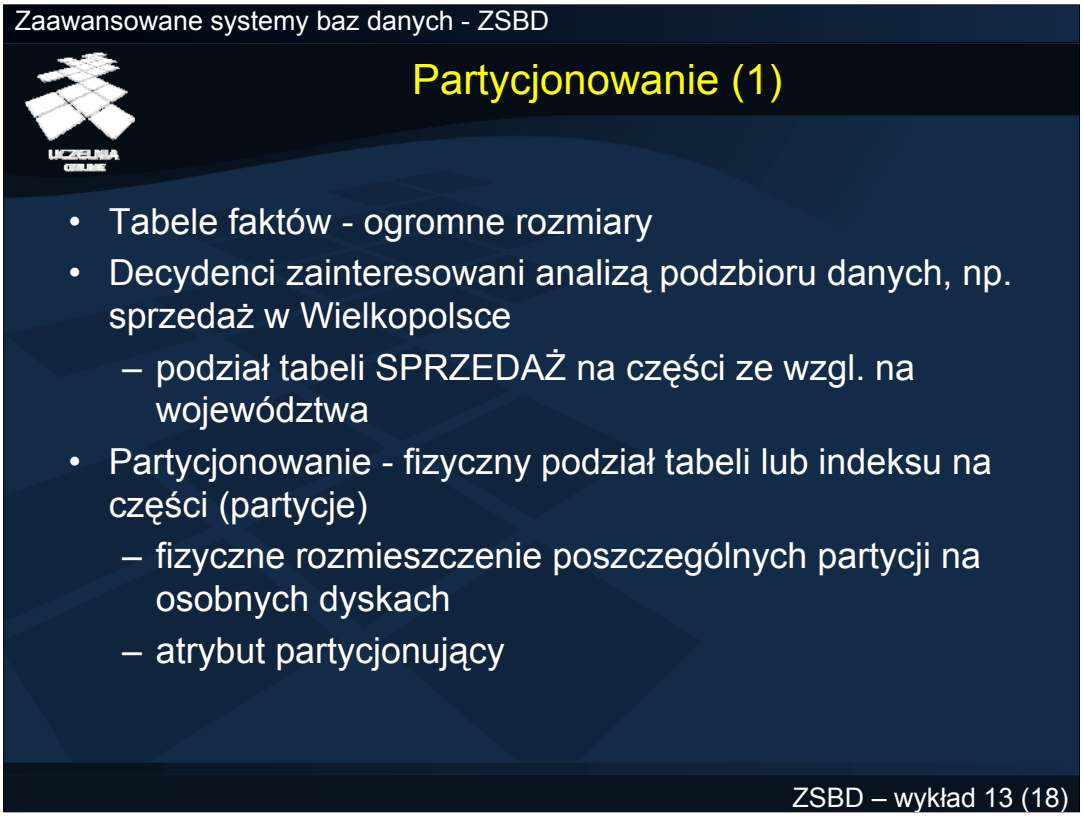
Na slajdzie przedstawiono przykład koncepcji bitmapowego indeksu połączeniowego zdefiniowanego na atrybucie kategoria tabeli Produkty . Ponieważ atrybut ten przyjmuje dwie różne wartości (kosmetyki, alkohole), więc indeks będzie się składał z dwóch map bitowych. Każda z map będzie opisywała rekordy z tabeli Sprzedaż . Mapa o nazwie 'kosmetyki' będzie opisywała sprzedaż kosmetyków, a mapa 'alkohole' - sprzedaż alkoholi. Pierwszy bit mapy 'kosmetyki' przyjmuje wartość 1, co oznacza, że pierwszy rekord w tabeli Sprzedaż dotyczy kosmetyku. Drugi bit tej mapy przyjmuje również wartość 1, co również oznacza, że drugi rekord tabeli Sprzedaż dotyczy kosmetyku. Podobnie jest w przypadku bitu 3 i 6 mapy 'kosmetyki'.

Implementacyjnie, bitmapowy indeks połączeniowy posiada strukturę B-drzewa, w którego liściach znajdują się mapy bitowe opisujące łączone rekordy.



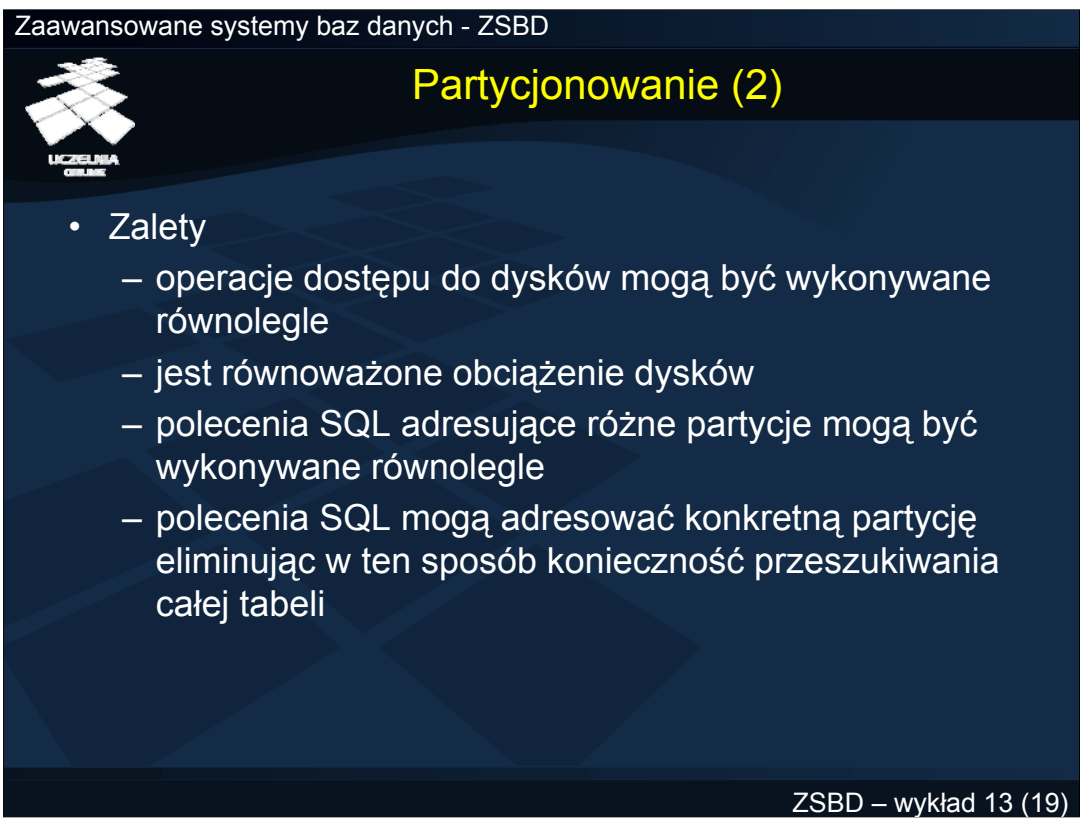
Na slajdzie przedstawiono przykład polecenia SQL tworzącego bitmapowy indeks połączeniowy dla bazy Oracle9i/10g.

Należy zwrócić uwagę na klauzulę ON, w której specyfikuje się łączone tabele. Indeks jest definiowany na atrybucie kategoria tabeli Produkty , ale zawartość indeksu opisuje również rekordy tabeli Sprzedaż . Ważna jest także klauzula WHERE, w której określa się sposób łączenia tabel, w sposób identyczny do łączenia tabel w standarowych zapytaniach.



W hurtowni danych największe rozmiary osiągają tabele faktów. Przeszukiwanie dużych tabel jest czasochłonne, nawet z wykorzystaniem indeksów. Często decydenci są zainteresowani analizą tylko podzbioru rekordów tabeli, np. ilości sprzedanych produktów z grupy kosmetyki w Wielkopolsce. Dla takiego zapytania, podział dużej tabeli Sprzedaż na mniejsze, np. ze względu na województwa, w których dokonano sprzedaży znacznie skróciłby czas dostępu do wybranego podzbioru danych.

Fizyczny podział tabeli (lub indeksu) na części jest nazywany partycjonowaniem (ang. partitioning). Każda z części nazywa się partycją (ang. partition). Często jest ona fizycznie umieszczana na osobnym dysku, znajdującym się w tym samym lub wielu węzłach (komputerach) sieci. Rozmieszczanie danych w poszczególnych partycjach jest realizowane na podstawie wartości wskazanego atrybutu tabeli (indeksu), tzw. atrybutu partycjonującego . Fizycznie podzielona tabela (lub indeks) stanowi logiczną całość z punktu widzenia użytkownika.



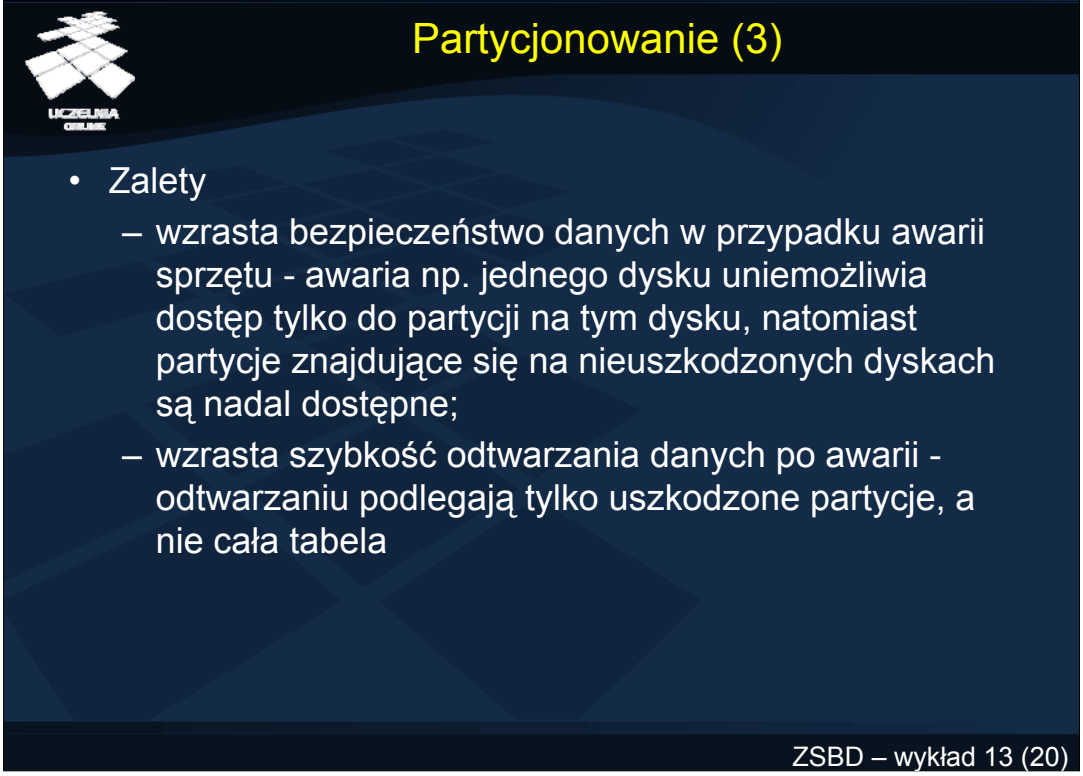
Podział dużej tabeli lub indeksu na mniejsze fragmenty zapewnia, że:

- bardzo kosztowne operacje wejścia/wyjścia, tj. dostępu do dysków mogą być wykonywane równolegle;

- jest równoważone obciążenie dysków;

- polecenia SQL adresujące różne partycje mogą być wykonywane równolegle;

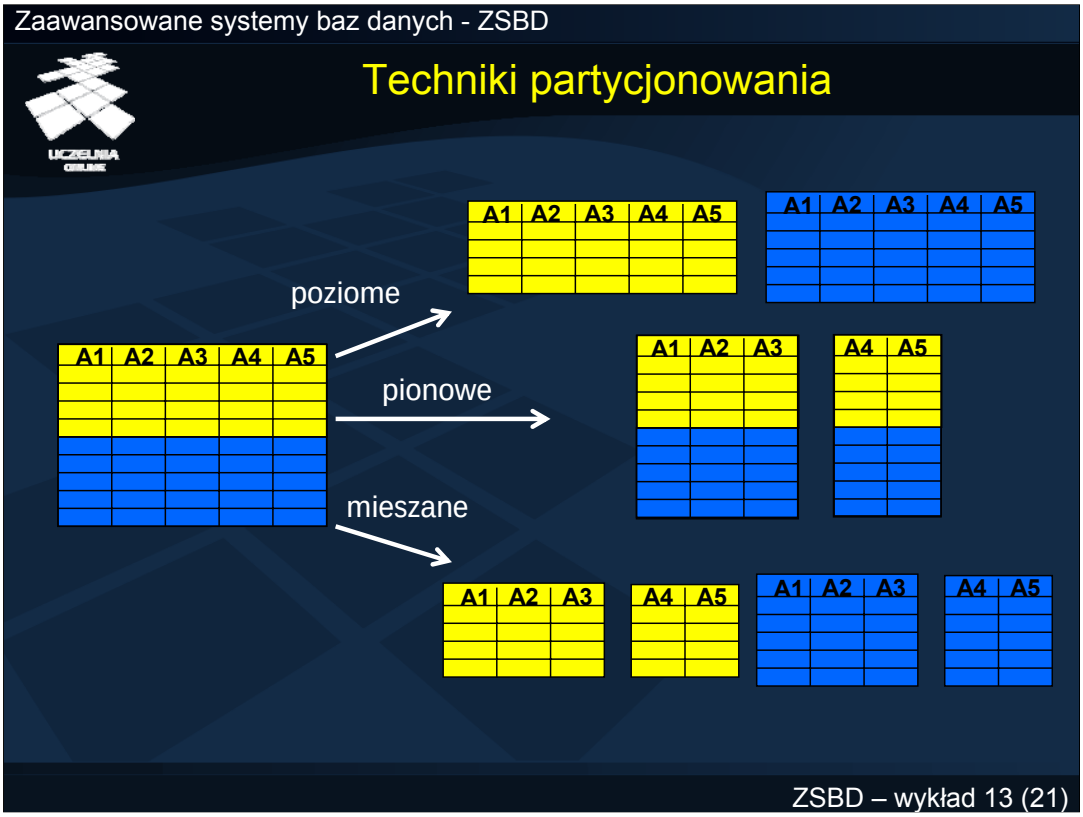
- polecenia SQL mogą adresować konkretną partycję, eliminując w ten sposób konieczność przeszukiwania całej tabeli lub indeksu;



Zalety partycjonowania są następujące:

- wzrasta bezpieczeństwo danych w przypadku awarii sprzętu ponieważ awaria np. jednego dysku uniemożliwia dostęp tylko do partycji na tym dysku, natomiast partycje znajdujące się na nieuszkodzonych dyskach są nadal dostępne;

- wzrasta szybkość odtwarzania danych po awarii ponieważ odtwarzaniu podlegają tylko uszkodzone partycje, a nie cała tabela.



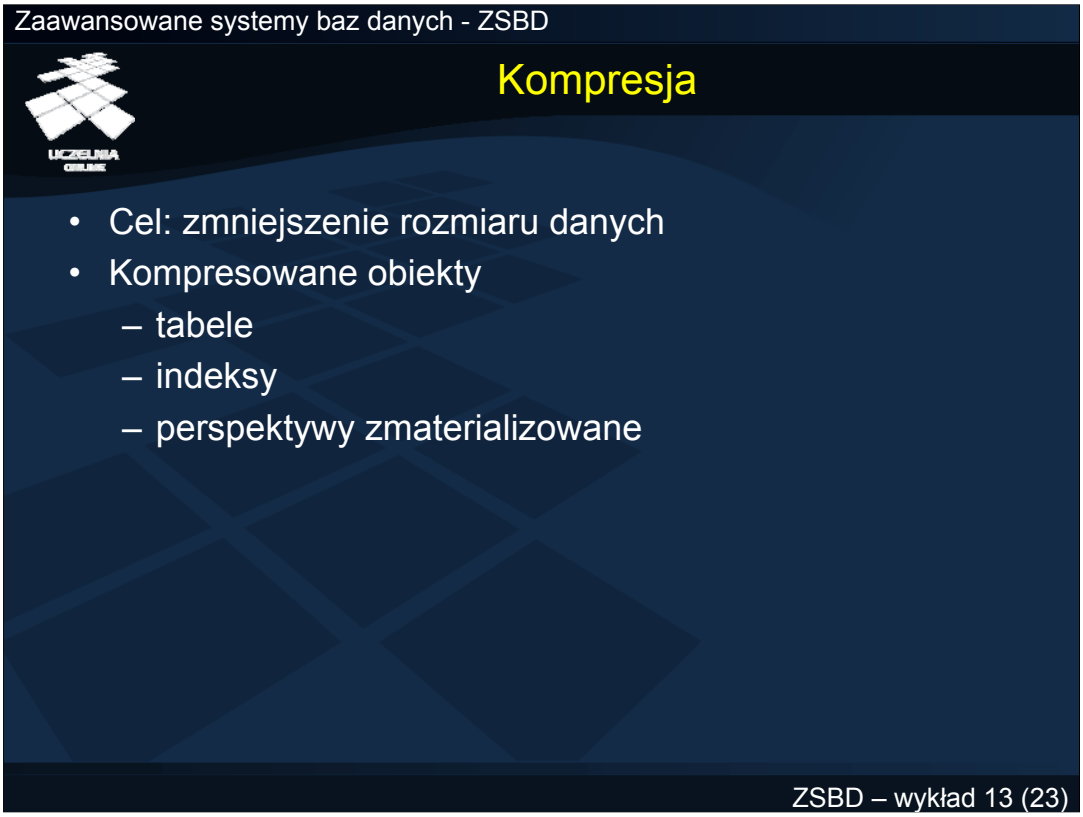
Ze względu na sposób podziału tabeli, wyróżnia się trzy rodzaje partycjonowania: poziome, pionowe i mieszane.

Partycjonowanie poziome (ang. horizontal partitioning) umożliwia podział zbioru rekordów tabeli na mniejsze podzbiory, z których każdy jest opisany identyczną liczbą atrybutów.

Partycjonowanie pionowe (ang. vertical partitioning) umożliwia podział tabeli w pionie, tj. jej rozbicie na partycje złożone z podzbiorów atrybutów tabeli pierwotnej. Każda partycja zawiera identyczną liczbę rekordów. Dany atrybut może się znaleźć tylko w jednej partycji. Nie dotyczy to atrybutów kluczowych, które są umieszczane w każdej partycji i służą one do łączenia partycji.

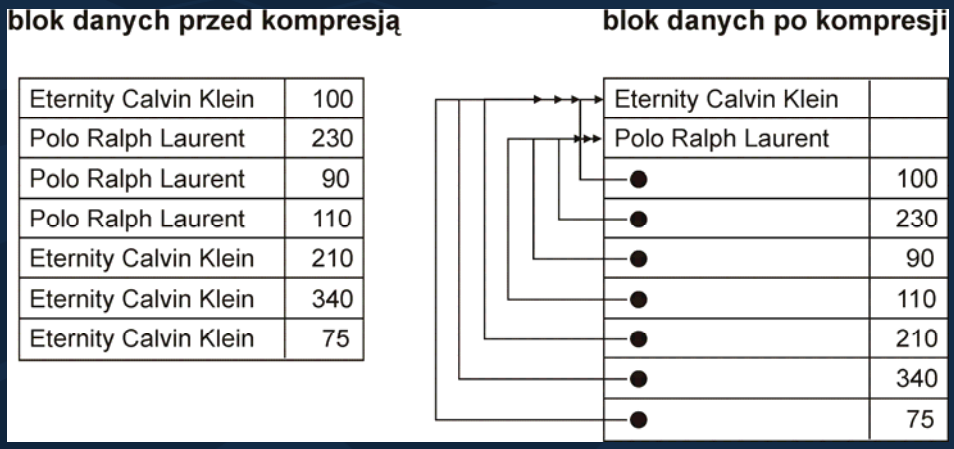
Partycjonowanie mieszane (ang. hybrid partitioning) stanowi połączenie partycjonowania poziomego i pionowego. W takim przypadku, tabela jest najpierw dzielona np. poziomo, a następnie wszystkie lub wybrane partycje są dalej dzielone pionowo.

Podstawowym i najprostszym algorytmem partycjonowania jest algorytm bazujący na wartości. W tym algorytmie rozmieszczenie danych w partycji zależy od wartości samych danych. Przykładowo, tabela zawierająca informacje o klientach sieci supermarketów może być podzielona zgodnie z wartością nazwy województwa, w którym klienci mieszkają, jak na slajdzie. Partycja 1 (umieszczona na dysku 1) przechowuje klientów z województwa Wielkopolskiego, partycja 2 (na dysku 2) - klientów z województwa Małopolskiego, a partycja 3 - klientów z województwa Mazowieckiego.



W celu zmniejszenia rozmiarów danych przechowywanych w magazynie stosuje się ich kompresję. Kompresji mogą podlegać m.in. następujące obiekty: tabele, perspektywy zmaterializowane, indeksy.

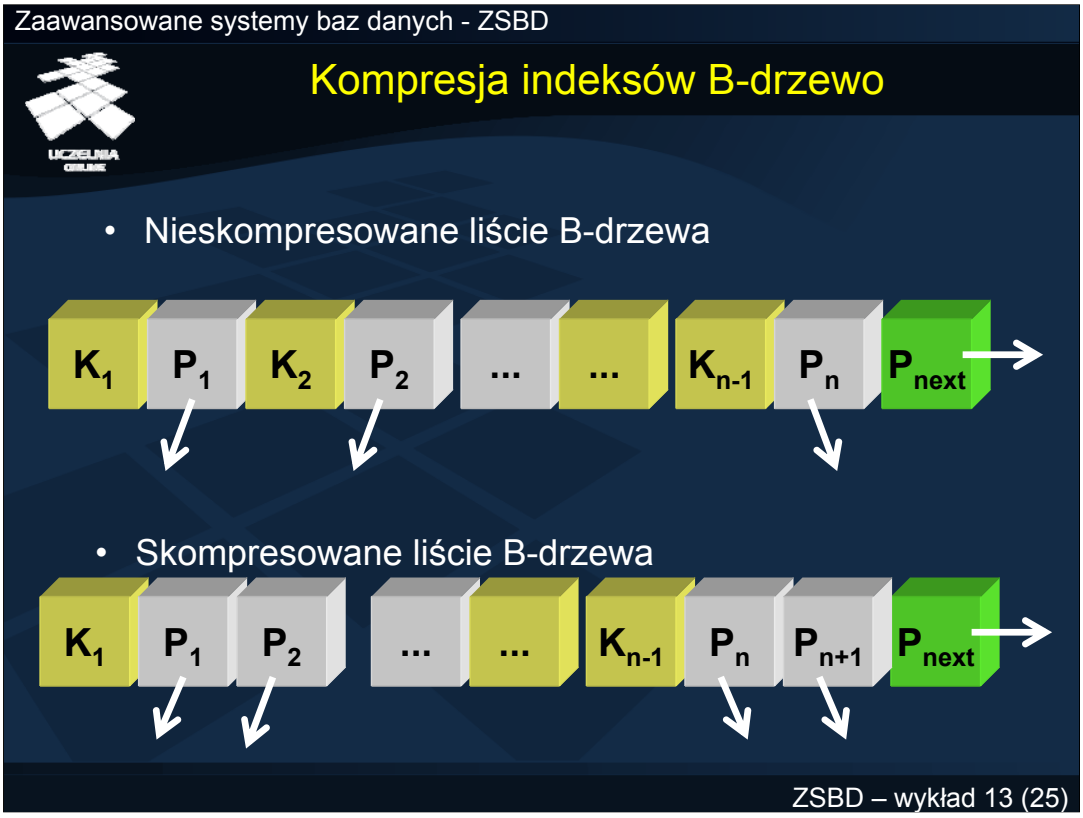
Kompresja bloków danych



Technika kompresji danych w blokach została zilustrowana na slajdzie. W bloku nieskompresowanym powtarzające się wartości atrybutów są przechowywane wielokrotnie. Natomiast w bloku skompresowanym, powtarzające się wartości są umieszczane na początku bloku, w przeznaczonym do tego celu obszarze/katalogu. W rekordach są umieszczane wskaźniki do odpowiedniej pozycji z katalogu. Ta technika została zaimplementowana m.in. w systemie Oracle.

Na slajdzie, po lewej stronie, przedstawiono nieskompresowany blok danych. Jak widać nazwa kosmetyku powtarza się wielokrotnie. Po prawej natomiast, przedstawiono skompresowany blok danych. Na początku bloku znajduje się katalog wartości, zawierający niepowtarzające się wartości. W dalszej części bloku znajdują się właściwe rekordy danych. W tym przypadku jednak zamiast wartości (nazwy kosmetyku) w rekordzie znajduje się wskaźnik do odpowiedniej pozycji z katalogu.

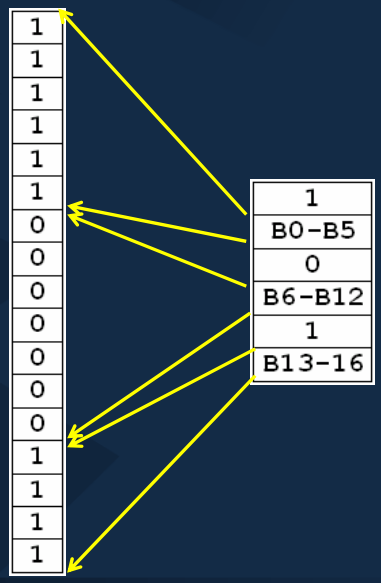
W typowym bloku o rozmiarze rzędu kB do zaadresowania pozycji z katalogu wystarczą wskaźniki 2B. Współczynnik kompresji zależy od oryginalnej długości atrybutów i szerokości dziedziny atrybutów znajdujących się w bloku. Dla wąskich dziedzin (np. kilka wartości) współczynniki kompresji są wyższe niż dla szerokich dziedzin.



Oprócz kompresji danych w blokach stosuje się kompresję indeksów B–drzewo i bitmapowych. Kompresja indeksu B–drzewo dotyczy jego liści. W liściu nieskopmpresowanym są przechowywane m.in. pary: wartość indeksowana Ki – adres rekordu posiadającego wartość Ki . Jeżeli indeks założono na atrybucie, którego wartość nie jest unikalna, wówczas wartość indeksowana Ki , pojawia się w liściach wielokrotnie – tyle razy ile jest rekordów z tą wartością. W przypadku liści skompresowanych jest budowana lista zawierająca: wartość indeksowaną Ki i adresy wszystkich rekordów posiadających wartość Ki . W ten sposób wartość indeksowana pojawia się w liściu jeden raz.

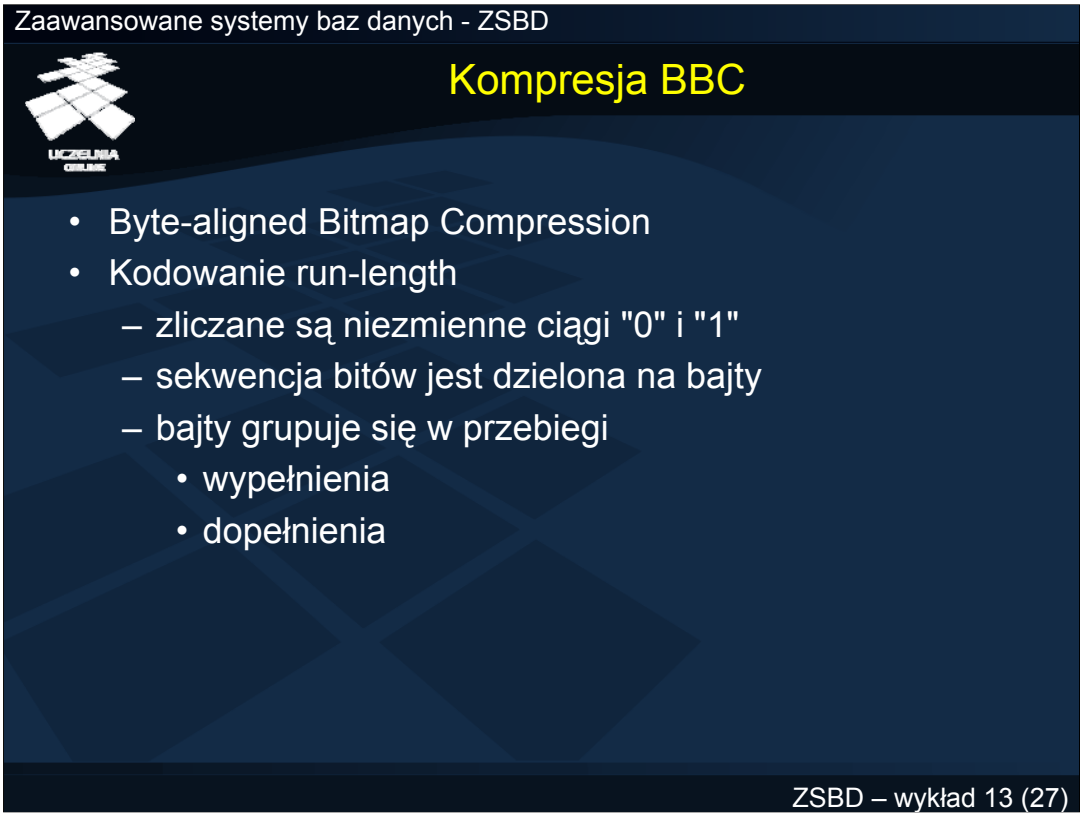
Kompresja indeksów bitmapowych

- Koncepcja schematu kompresji
- Algorytmy
  - BBC
  - WAH



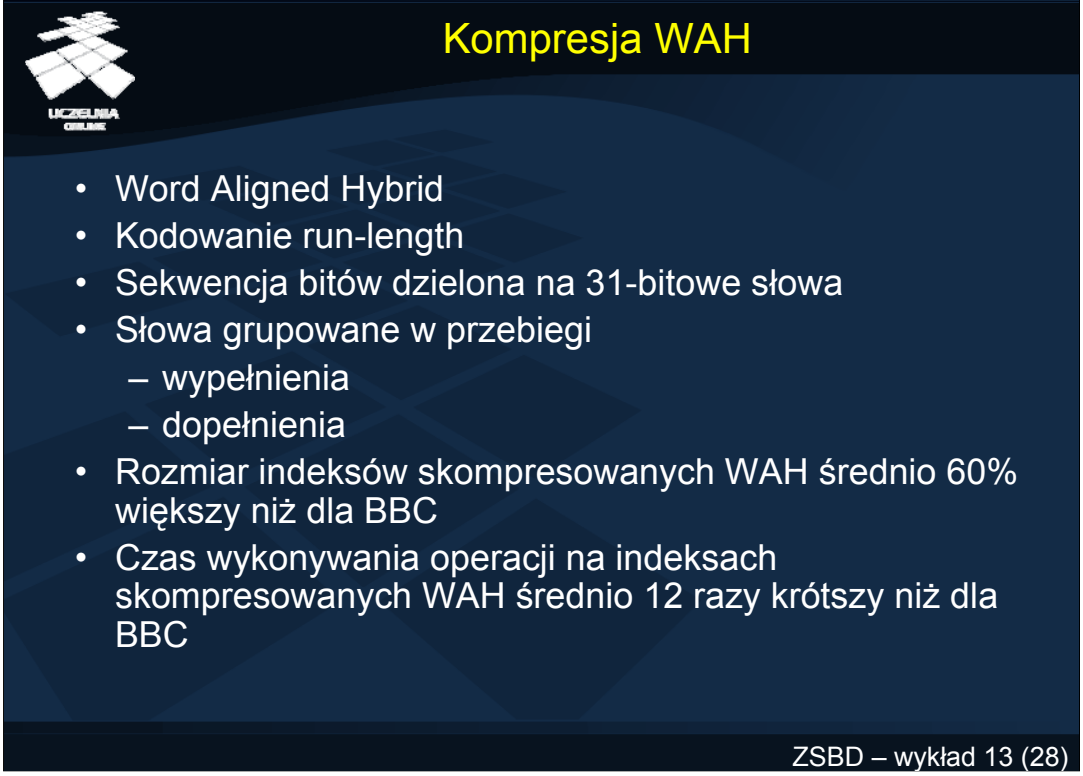
Indeksy bitmapowe ulegają kompresji wtedy, gdy liczba zer w mapach bitowych staje się zbyt duża w porównaniu do liczby jedynek. Jak pokazują eksperymenty, kompresja taka znacznie zmniejsza rozmiar indeksu bitmapowego, nie wpływając znacząco na efektywność przetwarzania samych map bitowych.

Na slajdzie przedstawiono koncepcję schematu kompresji pojedynczej mapy bitowej. Polega on na przechowywaniu symbolu - "0" lub "1" i liczby jego kolejnych wystąpień. Przykładowo, 6 pierwszych jedynek mapy jest przechowywanych jako wartość "1" i zakres kolejnych numerów pozycji w mapie, na których występuje wartość "1", tj. bity B0 do B5. Drugim ciągiem w mapie są zera, występujące od pozycji B6 do B12. Trzecim ciągiem są "1" występujące na pozycjach B13 do B16.



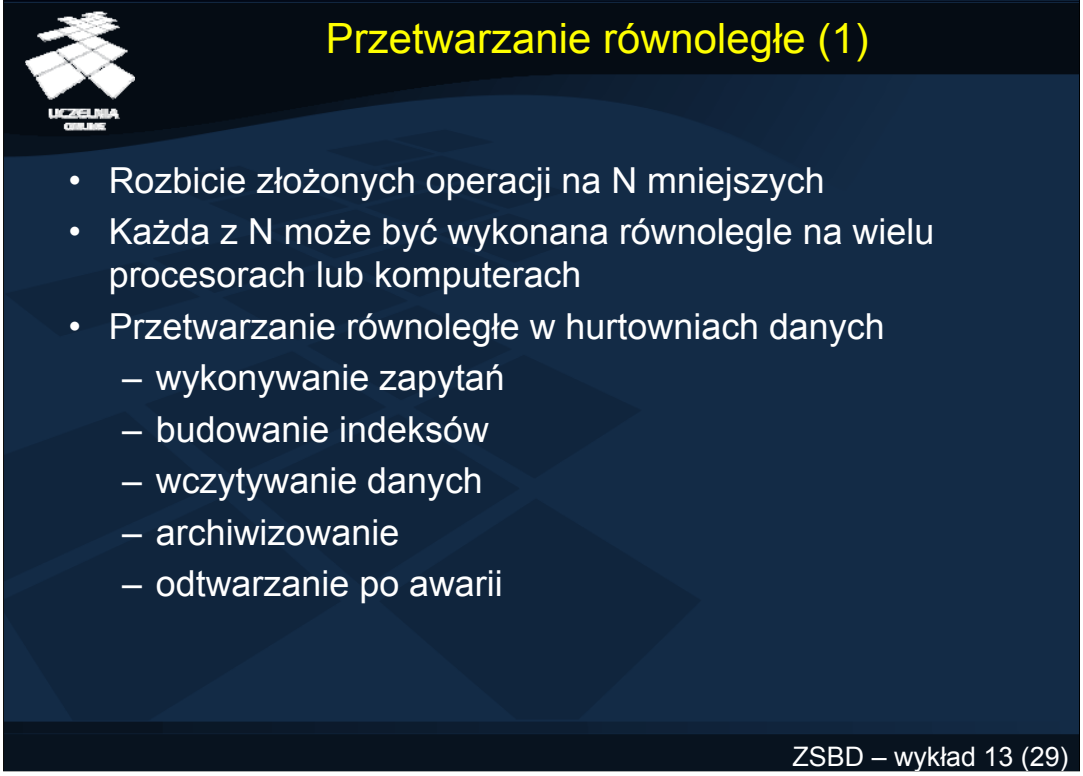
W praktyce stosuje się dwie techniki kompresji BBC i WAH.

Kompresja BBC (ang. Byte-aligned Bitmap Compression) stosowana jest między innymi w bazach danych Oracle. Opiera się na kodowaniu run-length, w którym zliczaniu podlegają niezmienne ciągi zer i jedynek. Sekwencja bitów dzielona jest na bajty, a następne bajty grupowane są w jeden z typów przebiegów (ang. runs) składających się z tzw. wypełnień - bajtów złożonych z samych zer lub samych jedynek (ang. fill) i dopełnień - bajtów zawierających i zera i jedynki (ang. tail).

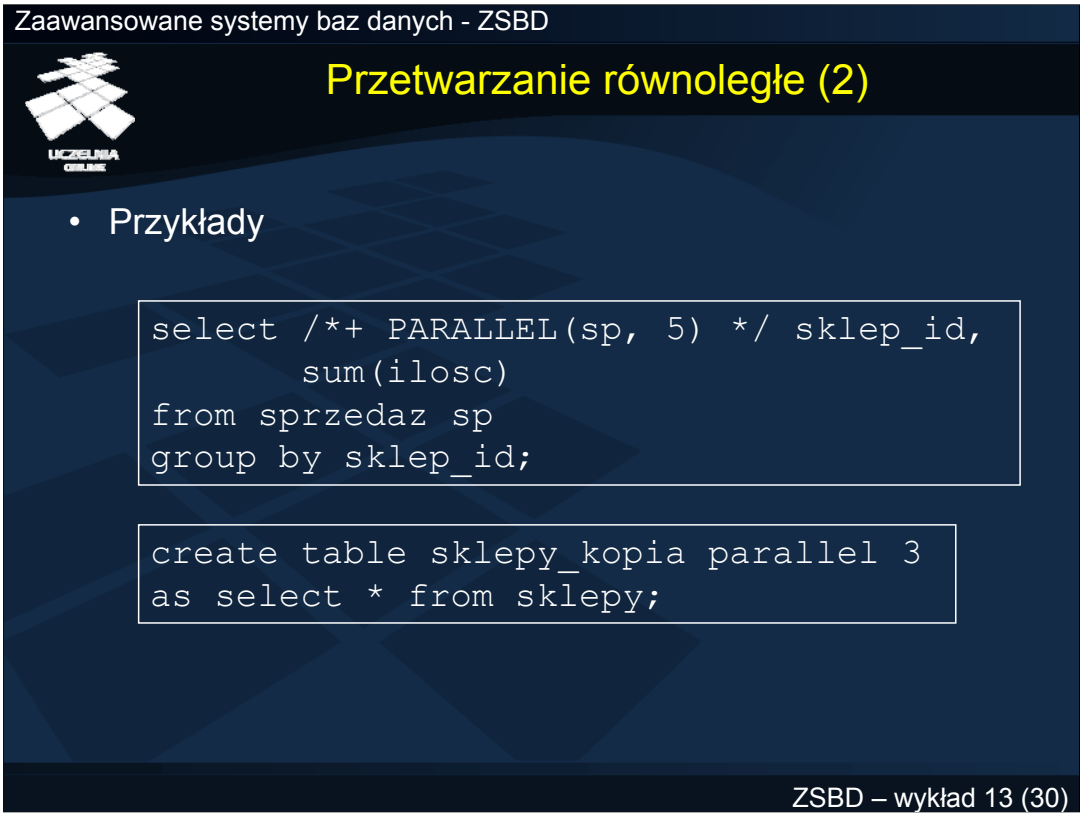


Kompresja WAH (ang. World Aligned Hybrid) stanowi rozwinięcie idei kompresji BBC. Również opiera się ona na kodowaniu run-length, ale sekwencja bitów dzielona jest na 31-bitowe słowa. Słowa te są następnie grupowane w dwa typy przebiegów: wypełnienia (ang. fills) - tak jak w BBC złożone z samych zer lub z samych jedynek i dopełnienia (ang. tails) - słowa zawierające i zera i jedynki.

WAH przy większym rozmiarze skompresowanych map bitowych średnio o 60%, zapewnia jednocześnie średnio dwunastokrotnie szybszy czas wykonywania operacji logicznych na skompresowanych mapach bitowych niż BBC



Kolejną techniką zwiększającą efektywność zapytań analitycznych jest przetwarzanie równoległe (ang. parallel processing). Polega ono na rozbiciu złożonych operacji na mniejsze, które następnie są wykonywane równolegle, np. na wielu procesorach lub komputerach. W efekcie, czas wykonania całej operacji jest krótszy. W przypadku hurtowni danych, najczęściej równolegle przetwarza się zapytania, buduje indeksy, wczytuje dane to hurtowni, archiwizuje dane, odtwarza bazę danych po awarii systemu.



Przykładowo, pierwsze polecenie ze slajdu odczytuje zawartość tabeli Sprzedaż z wykorzystaniem 5 równocześnie działających procesów. Jest to zapytanie wykonane w systemie Oracle.

select /*+ PARALLEL(sp , 5) */ sklep_id , sum(ilosc)

from sprzedaz sp

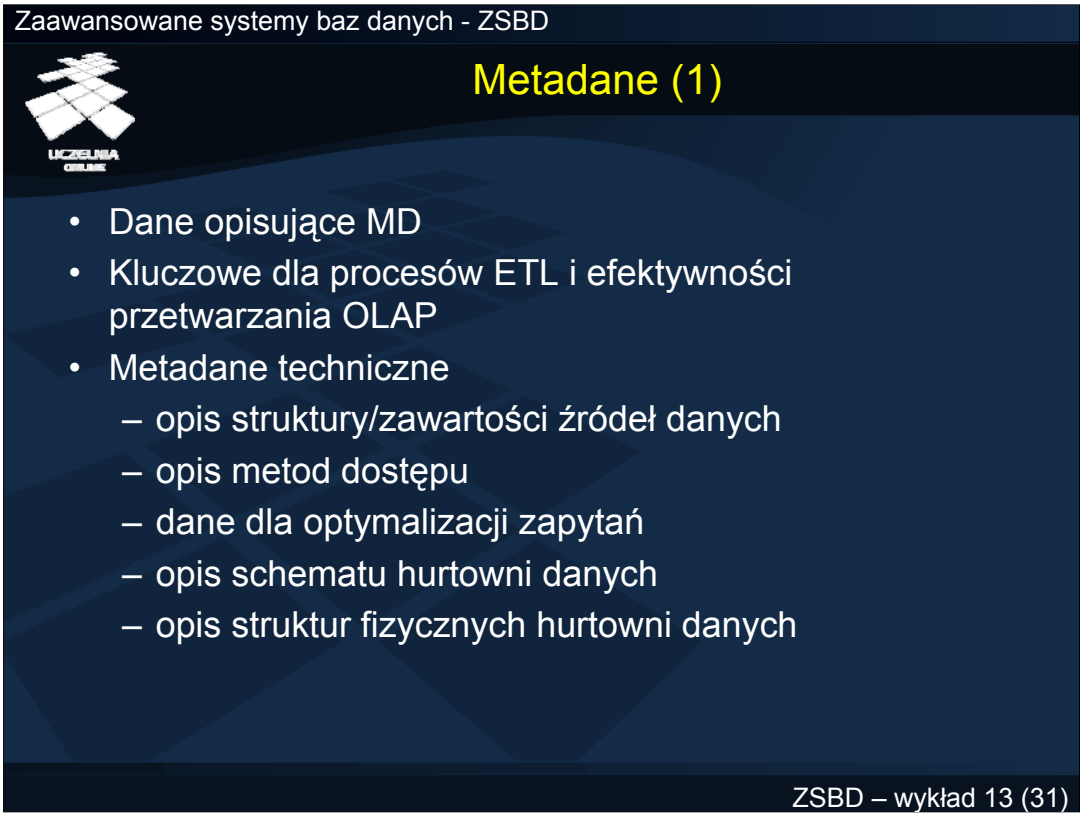
group by sklep_id ;

Liczbę procesów podaje się w zapytaniu za pomocą wskazówki PARALLEL. Wymaga ona dwóch argumentów wywołania. Pierwszym z nich jest alias do odczytywanej tabeli, a drugim - liczba procesów realizujących zapytanie.

Drugie polecenie ze slajdu tworzy tabelę sklepy_kopia z wykorzystaniem 3 równocześnie działających procesów.

create table sklepy_kopia parallel 3 as select * from sklepy ;

W tym przypadku liczbę procesów określa się w klauzuli PARALLEL.



Niezwykle ważnym komponentem każdej hurtowni danych są tzw. metadane.

Metadane można najprościej zdefiniować jako dane opisujące system i przechowywane w nim dane. Zarządzanie metadanymi i ich udostępnianie jest szczególnie ważnym problemem w zakresie integracji źródeł danych w ramach procesów ETL i zwiększania efektywności przetwarzania OLAP.

Wyróżnia się dwie podstawowe kategorie metadanych, tj. metadane techniczne i metadane administracyjne.

Metadane techniczne opisują m.in.:

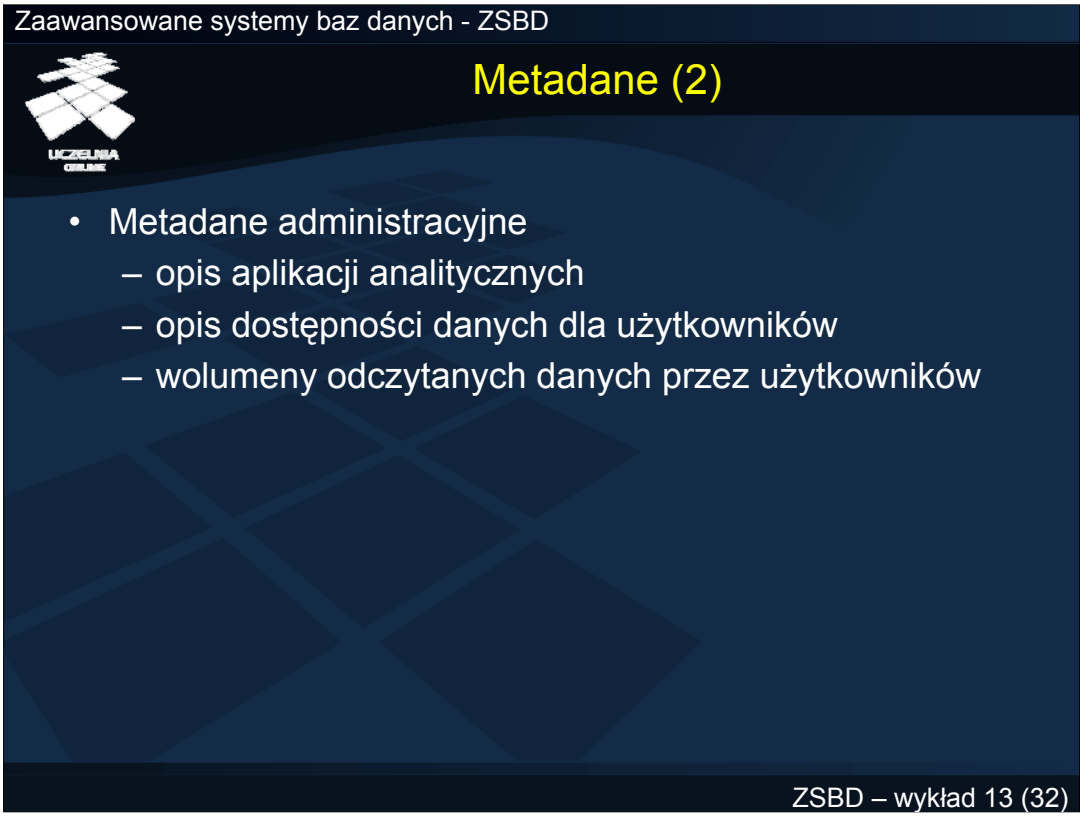
- struktury i zawartość źródeł danych,

- metody dostępu do źródeł danych,

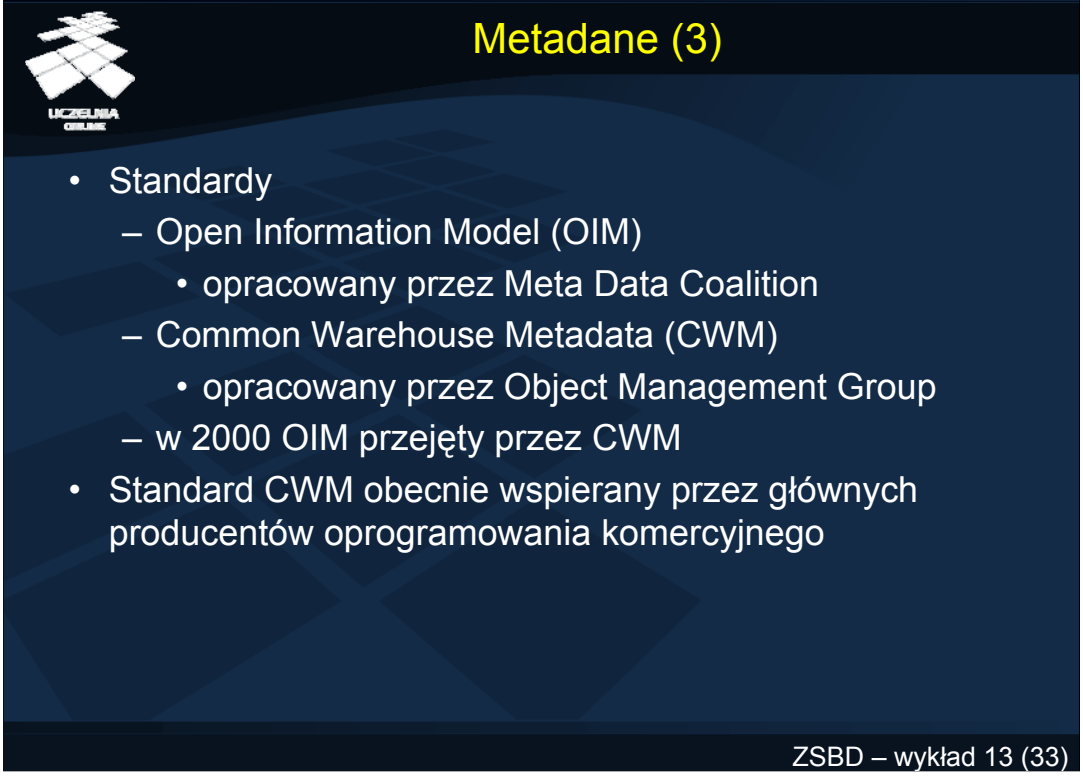
- własności danych wykorzystywane przez optymalizatory zapytań,

- opis schematu hurtowni danych (tabele, wymiary, ograniczenia integralnościowe),

- opis struktur fizycznych hurtowni danych (indeksy, partycje).



Metadane administracyjne opisują m.in.:
- aplikacje analityczne działające na hurtowni,
- dostępność danych dla użytkowników,
- wolumeny odczytanych danych przez poszczególnych użytkowników.



W początkowej fazie rozwoju systemów magazynów danych istniały dwa standardy opisu medatadnych, tj. Open Information Model (OIM) opracowany przez Meta Data Coalition i Common Warehouse Metadata (CWM) opracowany przez Object Management Group (Vetterli, Vaduva, Staudt, 2000; OMG, 2003). W roku 2000 standard OIM został zintegrowany do standardu CWM. Ten ostatni jest obecnie wspierany przez głównych producentów komercyjnego oprogramowania hurtowni danych.
