

## 1 Wprowadzenie

Tradycyjny sposób korzystania z baz danych sprowadza się, najczęściej, do realizacji zapytań poprzez aplikacje lub raporty. Sposób w jaki użytkownik korzysta z bazy danych (w jaki realizuje do niej dostęp) nazywamy *modelem przetwarzania*. Tradycyjny model przetwarzania – „*przetwarzanie transakcji w trybie on-line*” (ang. On Line Transaction Processing OLTP) jest w pełni satysfakcjonujący w przypadku bieżącej obsługi działalności danej firmy, dla dobrze zdefiniowanych procesów (obsługa klienta w banku, rejestracja zamówień, obsługa sprzedaży, itp.). Model ten charakteryzuje się krótkimi i prostymi transakcjami, które operują na niewielkiej części danych przechowywanych w bazie danych. Komercyjnie dostępne systemy OLTP (systemy zarządzania bazami danych SZBD) dostarczają efektywnych rozwiązań dla takich problemów jak: efektywne i bezpieczne przechowywanie danych, transakcyjne odtwarzanie danych, dostępność danych, optymalizacja dostępu do danych, zarządzanie współbieżnością. Podstawowym kryterium oceny efektywności działania systemu OLTP jest przepustowość transakcji, tj. liczba transakcji na sekundę. Niestety, ten klasyczny model przetwarzania danych nie wspomaga procesów analizy danych oraz aplikacji wspomagających podejmowanie decyzji, gdyż w znacznie mniejszym stopniu systemy te wspomagają operacje agregacji danych, wykonywania podsumowań czy też optymalizacji złożonych zapytań formułowanych ad hoc, które są charakterystyczne dla systemów wspomagania podejmowania decyzji.

Wspomaganie decyzji stanowi drugi bardzo ważny obszar zastosowania systemów baz danych. Rozwijające się gałęzie przemysłu, handlu, medycyny, nauki wymagają składowania i przetwarzania ogromnych ilości danych. Dane są zwykle przechowywane w systemach informatycznych posiadających różne struktury i wykorzystujących różne modele danych (np. hierarchiczne, relacyjne, obiektowe), w dokumentach tekstowych, czy arkuszach kalkulacyjnych. Ponieważ przedsiębiorstwa, instytucje, organizacje (producenci danych) są często geograficznie rozproszone, więc same dane mają również charakter rozproszony. Posiadanie danych opisujących działanie przedsiębiorstwa w dłuższym przedziale czasu pozwala na analizę trendów, anomalii, poszukiwania wzorców zachowań (wzorców zakupów, podobieństw między klientami, itp.) czy też odkrywania wiedzy. Przykładowe klasy aplikacji analitycznych to: bankowość (np. identyfikacja czynników ryzyka wskazujących, którzy klienci gwarantują bezpieczne spłacanie udzielonego kredytu), rynki finansowe (np. identyfikacja trendów w zakresie akcji spółek giełdowych), telekomunikacja (np. identyfikacja klientów zainteresowanych nowymi usługami i nowymi warunkami współpracy z firmą), medycyna (np. analiza efektywności procedur leczenia pacjentów), itd. Dane przechowywane w bazie danych zawierają olbrzymią ilość potencjalnie użytecznej wiedzy, która może zostać użyta w procesie podejmowania decyzji strategicznych dotyczących działalności przedsiębiorstwa. Dysponując danymi opisującymi działalność np. dużego supermarketu (sprzedaż produktów, zamówienia, stan rezerw) możemy postawić szereg pytań: W jaki sposób wykorzystać przechowywane dane do usprawnienia funkcjonowania firmy? Jakie czynniki kształtują taki a nie inny popyt na produkty? Czym różnią się klienci supermarketu w Poznaniu i Warszawie? Jakie oddziały supermarketu miały „anormalną” sprzedaż w pierwszym kwartale 2002 r? Jakie produkty miały największą dynamikę sprzedaży w roku 2001? Jakie produkty klienci supermarketu kupują najczęściej razem?

Dla potrzeb powyższej analizy danych opracowano nowy model przetwarzania, nazywany *przetwarzaniem analitycznym w trybie on-line* (ang. On-Line Analytical Processing OLAP) oraz stworzono nowy typ architektury bazy danych nazwany *magazynem danych* (ang. data warehouse). Magazyny danych są bardzo dużymi bazami danych, w których gromadzi się dane pochodzące z wielu heterogenicznych źródeł danych: scentralizowanych lub rozproszonych baz relacyjnych, relacyjno-obiektowych, obiektowych oraz ze źródeł innych niż bazy danych (np. arkusze kalkulacyjne, dokumenty XML lub pliki tekstowe).

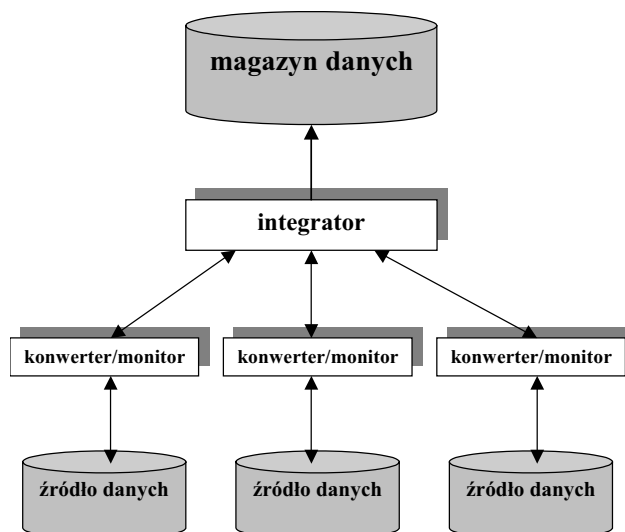
Celem tego artykułu jest przedstawienie podstawowych koncepcji i pojęć związanych z technologią magazynów (hurtowni) danych. W rozdziale 2 przedstawiono koncepcję i architekturę logiczną magazynu danych, oraz omówiono proces projektowania, budowy i ładowania magazynu danych. Rozdział 3 jest poświęcony omówieniu modelu przetwarzania danych OLAP. Przedstawiono w nim podstawowy model danych wykorzystywany w technologii OLAP, a mianowicie, wielowymiarowy model danych (strukturę wielowymiarowej kostki, zbiór podstawowych operatorów i zbiór ograniczeń integralnościowych). W rozdziale 4 krótko omówiono fizyczne architektury magazynów danych. Rozdział 5 jest poświęcony omówieniu podstawowych struktur schematów pojęciowych magazynów danych. W rozdziale 6 przedstawiono i omówiono typy serwerów OLAP (ROLAP, MOLAP, HOLAP). Rozdział 7 jest poświęcony wybranym zagadnieniom związanym z zapewnieniem odpowiedniej efektywności funkcjonowania magazynu danych: materializacji agregatów, partycjonowaniu danych, przetwarzaniu równoległemu, indeksom bitmapowym i połączeniowym. Rozdział 8 zawiera podsumowanie.

## 2 Magazyn danych

Magazyn danych jest „(...) zorientowaną tematycznie, zintegrowaną, zmienną w czasie i trwałą kolekcją (bazą) danych zaprojektowaną i zaimplementowaną dla potrzeb wspomagania podejmowania decyzji, w której dane odnoszą się do określonej chwili czasowej” (W. H. Inmon, *Building the Data Warehouse*, QED Tech. Pub. Group, 1992).” Innymi słowy, jest to zbiór technologii, których celem jest wspieranie decydentów (menedżerów, analityków) i umożliwianie im podejmowania lepszych i szybszych decyzji.

Magazyn danych jest **zorientowany tematycznie**. Oznacza to, że struktura danych w magazynie danych jest zorganizowana odpowiednio do podstawowego obszaru działalności danego przedsiębiorstwa. Operacyjne bazy danych firmy ubezpieczeniowej mogą być zorientowane na poszczególne segmenty działalności firmy, np. ubezpieczenia na życie, ubezpieczenia samochodów, domów, itp. Magazyn danych tej firmy będzie zorientowany tematycznie na: klientów, typy ubezpieczeń, polisy, konta, żądania wypłat, itp. Magazyn danych jest **zintegrowany**. Aby efektywnie wspierać proces wspomagania podejmowania decyzji, magazyn danych musi zawierać możliwie pełny zbiór danych opisujących działalność danego przedsiębiorstwa. Ponieważ dane opisujące działalność przedsiębiorstwa są najczęściej rozproszone niezbędna staje się **integracja danych** z wielu heterogenicznych źródeł. Dane operacyjne są regularnie aktualizowane i zmieniane. Transakcje w systemach OLTP dokonują dostępu do pojedynczych krotek, zmieniają ich wartości, usuwają bądź wstawiają krotki. Magazyny danych są natomiast **trwale**. Po załadowaniu danych do magazynu dane nie są z magazynu usuwane. Po dezaktualizacji danych, dane są archiwizowane. Aktualizacja danych przechowywanych w magazynie danych, czyli *odświeżenie danych* (ang. refresh) odbywa się w trybie wsadowym, w określonych odstępach czasu (raz na tydzień, raz na miesiąc). Istotną cechą magazynów danych jest ich **zmiennosc w czasie**. Horyzont czasowy magazynu danych jest znacząco większy niż horyzont czasowy operacyjnych baz danych. Bazy operacyjne przechowują aktualne wartości danych i nie zawsze zawierają element czasu. W przeciwieństwie do nich, magazyny danych przechowują całą historię danych (czyli zbiór migawek zrobionych w pewnych odstępach czasowych) i czas stanowi zawsze jeden z podstawowych elementów składowych magazynu danych.

Architektura magazynu danych została przedstawiona na Rys. 1 [11]. Obiekty oznaczone jako *źródło danych* reprezentują wymienione wyżej heterogeniczne źródła informacji. Z każdym z takich źródeł jest związana warstwa oprogramowania –*konwerter/monitor*.



Rys.1. Architektura logiczna magazynu danych

Zadaniem modułu *konwertera* jest transformowanie danych z formatu wykorzystywanego w źródle, do formatu wykorzystywanego w magazynie. Dlatego, dla każdego modelu danych źródłowych konieczne jest zastosowanie specyficznego modułu *konwertera*. Przykładowo, jeśli źródło przechowuje dane w dokumentach tekstowych, a magazyn został zaprojektowany z wykorzystaniem modelu relacyjnego, to *konwerter* musi zapewnić poprawne odwzorowanie danych z plików w struktury modelu relacyjnego. Konwersja danych ze źródeł do magazynu rozpoczyna się od procesu *ekstrakcji danych* (ang. data extraction). Ekstrakcja danych ze źródeł danych odbywa się poprzez *bramki* (ang. gateways) lub standardowe interfejsy (Information Builders EDA/SQL, ODBC, JDBC, Oracle Open Connect, Sybase Enterprise Connect, Informix Enterprise Gateway, itp.). Niestety, w przypadku niestandardowych źródeł danych zachodzi konieczność zaimplementowania wyspecjalizowanych procedur ekstrakcji danych. Kolejnym krokiem jest *czyszczenie danych* (ang. data cleaning, data cleansing, data scrubbing). Proces ten ma na celu zapewnienie jakości i poprawności danych w magazynie. Ma on szczególne znaczenie w przypadku, gdy źródła danych są wysoce heterogeniczne. Istnieje bowiem duże prawdopodobieństwo, że dane z wielu źródeł będą zawierały błędy i anomalie: niespójne długości pól, niespójne opisy atrybutów, różne formaty danych, wartości puste, naruszone ograniczenia integralnościowe. Źródłem niespójności są często pola opcjonalne. Dane przed transformacją często nazywane są danymi „brudnymi”. Przykładami zabrudzenia danych są [4]: różne formaty danych dla tego samego pola (np. informacja o województwie może być przedstawiona jako pełna nazwa, skrót nazwy lub specjalny identyfikator), niespójne wartości tej samej danej spowodowane błędami przy wprowadzaniu, niezgodności między wartością atrybutu i jego nazwą (np. pole *Nazwa* może zawierać nazwę firmy lub nazwisko indywidualnego klienta), brakujące wartości, które zgodnie ze schematem magazynu danych powinny być wypełnione, czy też redundantne informacje na temat jakiegoś obiektu świata rzeczywistego (mogą pojawiać się zarówno w ramach jednego źródła informacji, jak i w dwóch różnych źródłach). Podstawowymi metodami czyszczenia danych są: konwersja i normalizacja (transformacja i standaryzacja heterogenicznych formatów danych), czyszczenie specjalne (uspójnianie wartości pola na podstawie słownika synonimów), czyszczenie niezależne od domeny (porównywanie pól pochodzących z różnych źródeł w celu określenia stopnia ich podobieństwa), czyszczenie oparte na regułach (określanie podobieństwa między polami na podstawie zespołu predefiniowanych reguł)

Po wyczyszczeniu danych następuje etap *ładowania danych* (ang. data loading) do magazynu danych. Procesem ładowania zarządza zaznaczony moduł *integratora*. Ładowanie danych pociąga za sobą dodatkowe przetwarzanie, np. sprawdzanie ograniczeń integralnościowych, sortowanie, podsumowywanie, budowanie indeksów, itp. Ładowanie danych odbywa się najczęściej w trybie wsadowym. Aplikacja dokonująca załadowania danych musi pozwalać administratorowi na monitorowanie procesu ładowania, zawieszanie i odwieszanie ładowania, restartowanie po awarii (np. po naruszeniu ograniczeń integralnościowych). Proces ładowania danych może być bardzo czasochłonny i trwać wiele godzin bądź dni. W rzeczywistości taki proces może być traktowany jako jedna transakcja konstruująca nową bazę danych. Po załadowaniu danych wszystkie następujące zmiany w danych źródłowych są propagowane do magazynu danych podczas procesu odświeżania. *Odświeżanie danych* (ang. data refresh) to proces propagowania zmian zachodzących w źródłach danych do magazynu. Z odświeżaniem wiążą się dwa istotne problemy: w jaki sposób odświeżać oraz kiedy odświeżać. Odświeżanie magazynu danych może zachodzić okresowo (np. pod koniec dnia, kiedy magazyn danych nie jest używany), natychmiastowo (zmiany propagowane są do magazynu natychmiast po wystąpieniu w danych źródłowych) lub w sposób określony przez charakter źródła (np. po wystąpieniu określonej ilości zmian). Jeśli chodzi o metody odświeżania można wyróżnić zasadniczo dwie metody: (1) transfer danych (ang. data shipping) (relacja w magazynie danych jest traktowana jak zdalna migawka źródła danych; wyzwalacze działające w źródle są odpowiedzialne za aktualizację logu migawki i propagację danych do magazynu), (2) transfer transakcji (ang. transaction shipping) (log transakcji w źródle danych jest okresowo przeglądany celem wykrycia zaistniałych zmian; wszelkie zmiany są przesyłane do serwera replikacji, który przesyła nowe transakcje do magazynu danych).

Zadaniem modułu *monitora* jest wykrywanie zmian w danych źródłowych i ich przekazywanie do warstwy oprogramowania *integratora* (po uprzedniej konwersji do modelu danych magazynu). Sposób wykrywania zmian w danych źródłowych zależy od własności samych źródeł. Wyróżnia się cztery następujące rodzaje źródeł danych [11]: (1) **aktywne** (ang. active sources), tzn. posiadające zaimplementowane mechanizmy wyzwalaczy, które informują *monitor* o zmianach zachodzących w danych źródłowych, (2) **utrzymujące dzienniki operacji wykonywanych na danych źródłowych** (ang. logged sources), tj. zmiany są wykrywane przez analizę zawartości dziennika przez moduł *monitora*, (3) **przepytywalne** (ang. queryable sources), tj. w celu wykrycia zmian w danych źródłowych *monitor* okresowo wydaje zapytania do źródła, oraz (4) **wspierające mechanizm migawek** (ang. snapshot sources), tj. w tego rodzaju źródłach zmiany wykrywa się przez porównanie zawartości kolejnych migawek.

W przypadku gdy problem wykrywania i propagowania zmian w danych źródłowych jest trudny do rozwiązania, stosuje się okresowe uaktualnianie całego magazynu w trybie wsadowym. Wówczas magazyn jest niedostępny dla użytkowników. Ta technika jest wykorzystywana również w przypadku gdy aktualność danych nie jest konieczna dla poprawnego działania firmy, instytucji czy organizacji, oraz gdy dopuszczalna jest czasowa niedostępność danych.

Istotnym składnikiem magazynu danych jest *repozytorium metadanych* (ang. metadata repository), w którym przechowywane są informacje wspomagające zarządzanie magazynem. Metadane wspomagają też efektywny dostęp do zasobów magazynu poprzez przechowywanie informacji o zawartości magazynu, zależnościach między poszczególnymi komponentami systemu lub ich fizycznej lokalizacji w obrębie magazynu. Repozytorium zazwyczaj zawiera następujące informacje: **metadane fizyczne** (listę źródłowych baz danych i opis ich zawartości, opisy i charakterystyki bramek między bazami źródłowymi a magazynem, schemat magazynu danych, definicję perspektyw i danych wyliczalnych, przechowywanych w magazynie, opisy wymiarów i hierarchii (patrz dalej), zbiór predefiniowanych zapytań i raportów, lokalizację tematycznych hurtowni danych, indeksy i reguły partycjonowania danych), **metadane logiczne** (reguły biznesowe, podstawowe pojęcia i definicje, procedury postępowania, logiczne definicje tablic i atrybutów magazynu danych, odwzorowanie danych operacyjnych na struktury magazynu danych), **metadane operacyjne** (historia integracji danych, reguły ekstrakcji, czyszczenia, transformacji i

korekcji danych źródłowych, zasady odświeżania danych, aktualność danych – dane szczegółowe i dane wyprowadzalne), **metadane historyczne** (dane reprezentujące zmiany zachodzące w środowisku magazynu danych, informacja dotycząca aliasów), **metadane administracyjne** (dane dotyczące bezpieczeństwa magazynu, autoryzacja użytkowników, prawa dostępu do poszczególnych komponentów magazynu, profile użytkowników i profile grup użytkowników), oraz **metadane personalizacyjne** (reguły obliczania pewnych agregatów dla określonych użytkowników końcowych lub grup użytkowników). Szereg producentów oferuje specjalne narzędzia do zarządzania metadanymi – Platinum repository, IBM DataGuide, Prism Directory Manager. O ważności metadanych świadczy fakt powołania specjalnej grupy standaryzacyjnej (o nazwie Metadata Council), której zadaniem jest specyfikacja standardów metadanych (tj. formatów i struktur metadanych), aby zagwarantować możliwość wymiany metadanych pomiędzy produktami różnych wytwórców oprogramowania.

### 3 Model przetwarzania analitycznego w trybie on-line (OLAP)

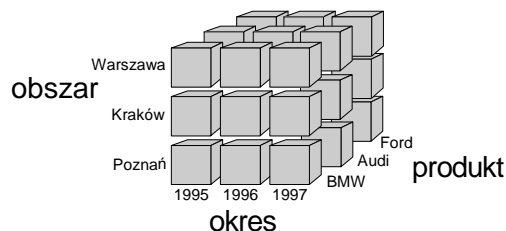
Nowy model przetwarzania danych, nazwany *przetwarzaniem analitycznym on-line* (ang. On-Line Analytical Processing OLAP), ma za zadanie wspieranie procesów analizy magazynów danych dostarczając narzędzi umożliwiających analizę magazynu w wielu „wymiarach” definiowanych przez użytkowników (czas, miejsce, klasyfikacja produktów, itp.). Analiza magazynu polega na obliczaniu agregatów dla zadanych „wymiarów” magazynu. Należy podkreślić, że proces analizy jest całkowicie sterowany przez użytkownika. Mówimy czasami o *analizie danych sterowanej zapytaniami* (ang. query-driven exploration). Typowym przykładem takiej analizy jest zapytanie o sprzedaż produktów w supermarkecie w kolejnych kwartałach, miesiącach, tygodniach, itp., zapytanie o sprzedaż produktów z podziałem na rodzaje produktów (AGD, produkty spożywcze, kosmetyki, itp.), czy wreszcie zapytanie o sprzedaż produktów z podziałem na oddziały supermarketu. Odpowiedzi na powyższe zapytania umożliwiają decydującym określić wąskie gardła sprzedaży, produktów przynoszących deficyt, oraz podjęcie odpowiednich działań poprawiających sytuację. Model OLAP adresowany jest przede wszystkim do decydentów, menadżerów, analityków systemowych. Model ten charakteryzuje się m.in. małą liczbą złożonych i kosztownych zapytań. Znakomita większość transakcji w modelu OLAP składa się z operacji odczytu dużych wolumenów danych (pojedyncze zapytanie operuje na tysiącach bądź milionach rekordów). Podstawową miarą wydajności systemów nie jest ilość transakcji na jednostkę czasu (jak w przypadku systemów OLTP), lecz czas odpowiedzi na pojedyncze zapytanie.

Każdy logiczny model danych, a takim jest również model OLAP, składa się z trzech zasadniczych elementów:

- struktury danych, która opisują logiczną organizację danych i sposób, w jaki dane są postrzegane przez użytkowników,
- zbioru operatorów umożliwiających wyszukiwanie i modyfikowanie danych, oraz
- ograniczeń integralnościowych, specyfikujących poprawność danych.

Podstawowym modelem logicznym dla OLAP jest **wielowymiarowy model danych** (ang. multidimensional data model – MDD model). W modelu tym, z logicznego punktu widzenia, dane są postrzegane przez użytkowników w postaci **wielowymiarowej perspektywy** (tzw. kostki OLAP). Obiektem analizy w modelu MDD jest zbiór **miar numerycznych** nazywanych **faktami**. **Fakt** opisuje pojedyncze zdarzenie, o którym chcemy przechowywać informację w magazynie danych. Fakt jest daną ilościową (numeryczną) reprezentującą jednostkę aktywności biznesowej przedsiębiorstwa, np. sprzedaż produktów, średnia ocena studenta, ilość gości hotelowych, zysk, wartość produktu krajowego, itp. Wartość każdej miary zależy od zbioru wymiarów. Zbiór wymiarów określa kontekst miary. Przykładowo, zbiór wymiarów związanych z miarą sprzedaży może zawierać: miasto, nazwę produktu, datę sprzedaży. W wielowymiarowym modelu danych,

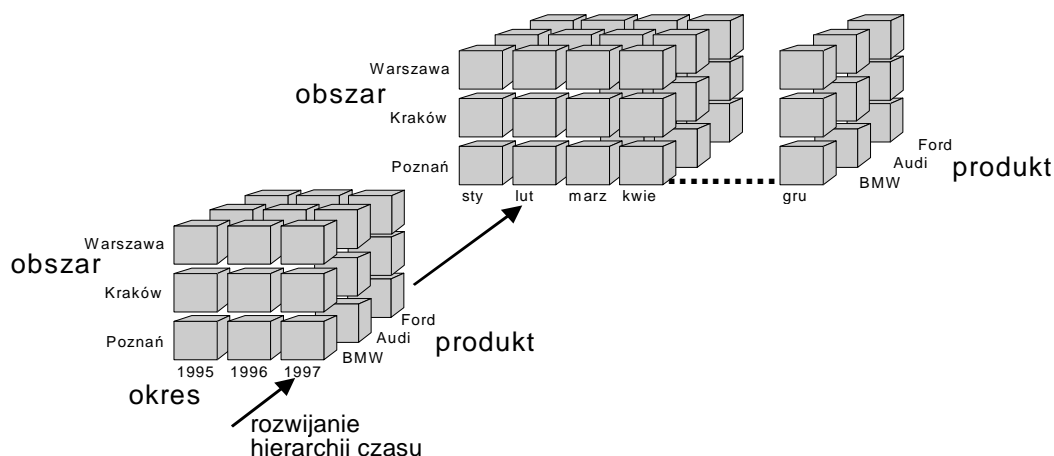
miara jest reprezentowana jako punkt w wielowymiarowej przestrzeni wymiarów. Każdy wymiar jest opisany **zbiorem atrybutów**. Przykładowo, wymiar *produkt* może być opisany 4 atrybutami: kategoria produktu, branża, rok wprowadzenia, średni zysk. Atrybuty wymiaru mogą tworzyć **hierarchię atrybutu**. Przykładowo, dla wymiaru Produkt, hierarchia może mieć postać: produkt → kategoria\_produkту → branża. Rysunek 2 przedstawia przykładową kostkę trójwymiarową z wymiarami *obszar*, *okres* i *produkt*.



Rys.2. Kostka trójwymiarowa

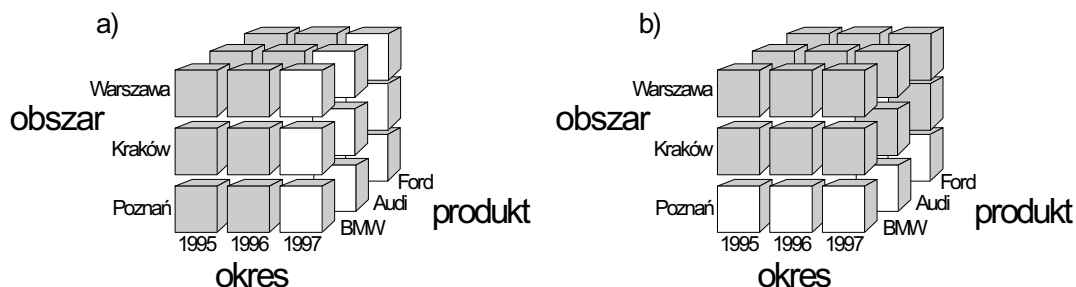
Drugim elementem modelu logicznego jest zbiór operatorów (operacji), służących do przetwarzania struktur danych. Podstawowy zbiór operatorów wielowymiarowego modelu danych jest następujący:

- pivoting – wyznaczanie punktu centralnego: wskazanie miary, która nas interesuje i wybranie 2 wymiarów, w których ma ona być reprezentowana (np. miara: *sprzedaż*, wymiary: *miasto* i *rok*). Przykładowo, w wymiarze produktu reprezentującego samochód marki „BMW” i wymiarze obszaru reprezentującego sklepy województwa poznańskiego może być prezentowana liczba sprzedanych samochodów.
- drill-down – rozwijanie: nawigacja wzdłuż hierarchii danego wymiaru w celu rozbicia agregatu na agregaty składowe (np. sprzedaż w poszczególnych branżach; sprzedaż w poszczególnych kategoriach produktów). Jako przykład rozważmy informacje o sprzedaży samochodów marek BMW, Audi i Ford, w latach 1995, 1996 i 1997, w poszczególnych miastach (por. Rys. 3). W celu dokonania analizy sprzedaży w poszczególnych miesiącach roku 1997 należy rozwinąć hierarchię reprezentującą czas, tj. rok 1997. Analiza sprzedaży w poszczególnych dniach wybranego miesiąca będzie możliwa po rozwinięciu hierarchii reprezentującej ten miesiąc.



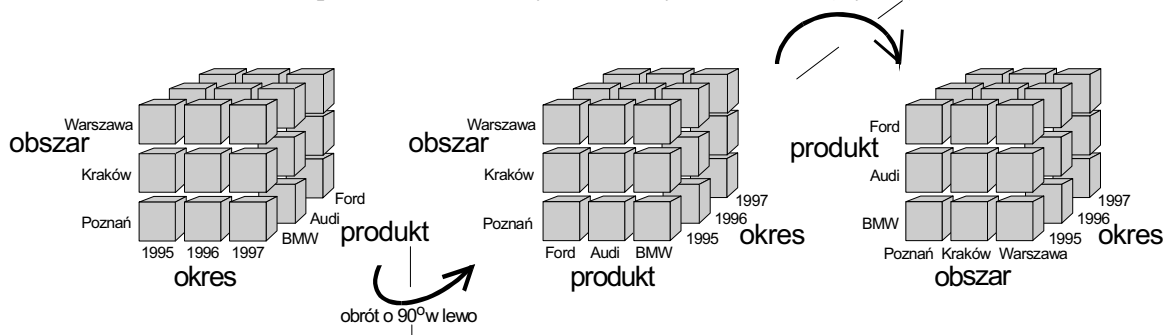
Rys.3. Operacja rozwijania hierarchii wymiaru

- roll-up – zwijanie: dla wskazanego wymiaru następuje nawigacja w górę hierarchii wymiaru w celu prezentacji większych agregatów.
- slice\_and\_dice – wycinanie: odpowiada operacji redukcji liczby wymiarów, tj. następuje projekcja danych na wybranym podzbiorze wymiarów dla wybranych wartości innych wymiarów. Przykładowo, dyrektor ds marketingu będzie zainteresowany wielkością sprzedaży wszystkich produktów, we wszystkich miastach kraju, w roku bieżącym (zob. Rys. 4a); natomiast kierownika oddziału firmy w Poznaniu będzie interesowała wielkość sprzedaży wszystkich produktów, w ciągu całego okresu działalności (zob. Rys. 4b).



Rys.4. Wycinanie danych w różnych wymiarach

- obracanie –umożliwia prezentowanie danych w różnych układach (Rys 5.)



Rys.5. Operacja obracania

- ranking – wybór pierwszych n elementów (np. wybierz 5 najlepiej sprzedających się produktów w miesiącu czerwcu).
- inne (selekcja, procedury składowane, itp.).

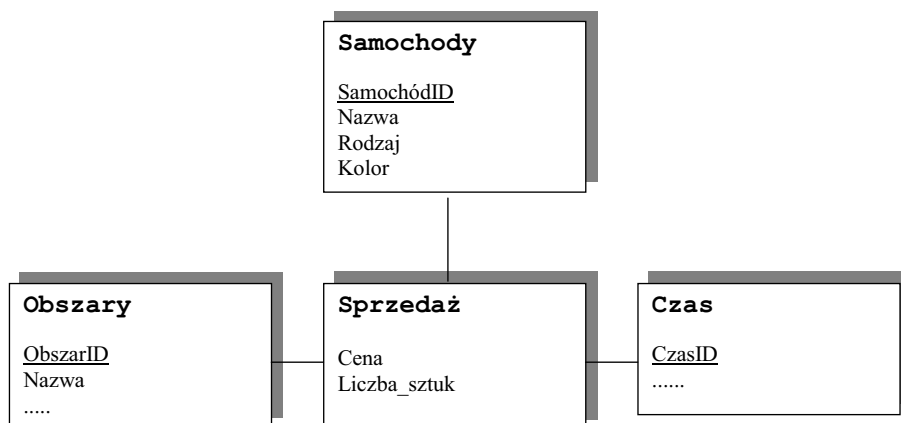
Trzecim elementem modelu danych jest zbiór ograniczeń integralnościowych definiujący poprawność przechowywanych danych. W klasycznym modelu relacyjnym wyróżniamy kilka klas ograniczeń integralnościowych, takich jak: ograniczenia klucza podstawowego, ograniczenia referencyjne, ograniczenia wartości pustych, itp. Ograniczenia integralnościowe dla modelu wielowymiarowego można podzielić na dwie zasadnicze klasy: **ograniczenia integralnościowe pojedynczej kostki danych** (ang. intra cube constraints) oraz **ograniczenia integralnościowe pomiędzy kostkami danych** (ang. inter cube constraints). Te pierwsze są związane z definicjami zależności pomiędzy atrybutami wymiarów, wymiarami, wymiarami a miarami, oraz hierarchiami wymiarów. Te drugie określają związki pomiędzy dwoma lub więcej kostkami danych. W szczególności, określają zależności pomiędzy miarami kostek, wymiarami kostek, miarą jednej kostki a wymiarami innych kostek, związkami pomiędzy kostkami.

Wielowymiarowy model danych jest podstawowym modelem logicznym magazynu danych, ale nie jedynym. Drugim, często spotykanym modelem logicznym dla OLAP, jest klasyczny model relacyjny, w którym dane dotyczące faktów oraz wymiarów są widziane przez użytkowników w postaci relacji.

## 4 Projektowanie schematu pojęciowego magazynu danych

Podobnie jak w przypadku systemów baz danych, pierwszym etapem budowy magazynu danych jest zaprojektowanie schematu pojęciowego magazynu. Do zaprojektowania takiego schematu można wykorzystać dowolny z modeli pojęciowych wykorzystywanych do projektowania schematów pojęciowych baz danych (model encji-związków, UML, itd.). Zasadnicza różnica pomiędzy projektowaniem schematu pojęciowego magazynu danych a bazy danych polega na metodyce projektowania. Znane z systemów baz danych metodyki projektowania są nieodpowiednie dla potrzeb analizy danych. Schemat pojęciowy magazynu danych powinien koncentrować się, z jednej strony, na podstawowych pojęciach i dziedzinach aktywności danego przedsiębiorstwa, z drugiej, powinien być łatwo transformowalny do wielowymiarowego modelu danych. W rzeczywistości struktury schematów pojęciowych magazynów danych są skodyfikowane i wyróżnia się trzy struktury schematów pojęciowych magazynów danych: schemat gwiazdy, schemat płatka śniegu, oraz schemat konstelacji faktów (schemat konstelacyjny).

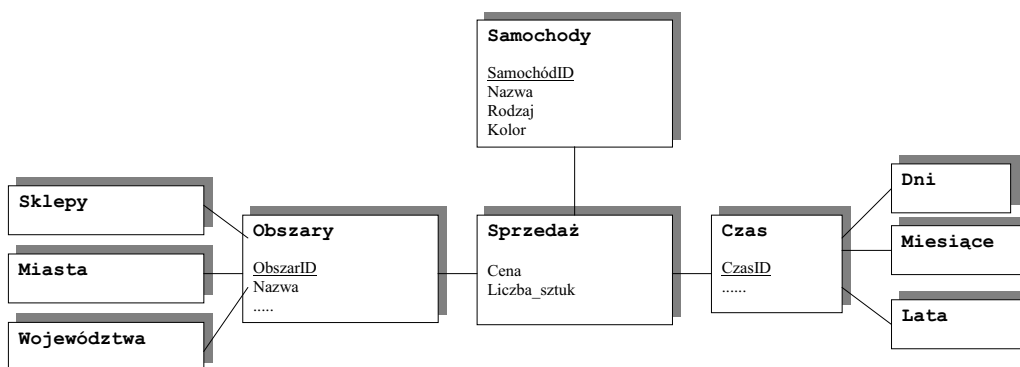
Podstawową strukturą schematu pojęciowego magazynu danych jest *struktura gwiazdy* (ang. star schema). W strukturze tej centralna encja opisuje podstawową miarę (zbiór miar), która jest powiązana z encjami wymiarów (Rys. 6).



Rys 6. Schemat gwiazdy

Centralna encja *Sprzedaż* zawiera informacje o sprzedaży samochodów, w pewnych obszarach geograficznych, w określonym czasie. Encje *Samochody*, *Obszary* i *Czas* są encjami *wymiarów*, natomiast encja centralna jest nazywana *encją faktów*. Pomiedzy encją faktów a encjami wymiarów występują związki typu „jeden-do-wiele”. W modelu tym nie są uwzględnione explicite hierarchie wymiarów.

Drugą popularną strukturą schematów pojęciowych magazynów danych jest *struktura płatka śniegu* (ang. snowflake schema). Jest to zmodyfikowana wersja struktury gwiazdy, w której explicite zamodelowane są hierarchie wymiarów (Rys. 7).



Rys 7. Schemat płatka śniegu

Podobnie jak w przypadku struktury gwiazdy centralna encja faktów jest powiązana związkami „jeden-do-wiele” z encjami wymiarów, które tworzą hierarchie wymiarów. Struktura płatka śniegi jest czytelniejsza i semantycznie bogatsza aniżeli struktura gwiazdy.

Trzecią strukturą schematów pojęciowych, strukturą najogólniejszą, jest **struktura konstelacji faktów** (ang. fact constellation). W strukturze tej zbiór encji faktów współdzieli zbiór encji wymiarów, choć niekoniecznie na tym samym poziomie hierarchii tych wymiarów. Przykładowo, dwie różne miary *zysk* i *podatek VAT* są powiązane z wymiarami *obszar*, *produkt* i *czas*. Jednakże związek pomiędzy miarą *zysk* a miarą *czas* jest definiowany na poziomie „dnia”, natomiast w przypadku miary *podatek VAT* ten związek występuje na poziomie „kwartału” lub „roku”, które są zdefiniowane na wyższym poziomie hierarchii wymiaru *czas*. Co więcej, *zysk* jest związany z *obszarem*, *produktem* i *czasem*, natomiast *podatek VAT* zależy wyłącznie od czasu i produktu (*podatek VAT* jest taki sam niezależnie od sklepu, powiatu czy województwa).

Łatwo zauważyć, że proces projektowania schematów pojęciowych magazynów danych, w stosunku do projektowania schematów pojęciowych baz danych, jest generalnie prostszy. Zasadniczą trudność projektowania schematu polega na identyfikacji podstawowych miar i wymiarów przetwarzania.

## 5 Typy magazynów danych

Magazyny danych można podzielić na dwa zasadnicze typy ze względu na wykorzystywane przez nie modele danych. Pierwszy, to magazyny danych wykorzystujące *model relacyjny*, nazywane również **ROLAP** (ang. Relational OLAP), drugi to magazyny danych wykorzystujące wielowymiarowy model danych, nazywane również **MOLAP** (ang. Multidimensional OLAP). Popularnym typem magazynu danych są magazyny łączące zalety obu wspomnianych wyżej magazynów nazywane **HOLAP** (ang. Hybrid OLAP).

Magazyn danych typu ROLAP jest zbudowany w oparciu o system zarządzania relacyjną bazą danych posiadający mechanizmy efektywnego przetwarzania zapytań typu OLAP. Zwykle schemat logiczny takiego magazynu posiada strukturę *gwiazdy* lub strukturę *płatka śniegu*. Struktury te odpowiadają odpowiednim strukturom schematów pojęciowych. W schemacie gwiazdy mamy centralną relację faktów powiązaną kluczami obcymi z odpowiednimi relacjami wymiarów. W przypadku schematu płatka śniegu, relacje wymiarów są znormalizowane, co w istocie prowadzi do wyodrębnienia hierarchii wymiaru. Zarówno relacja faktów jak i relacje wymiarów są pamiętane w specjalizowanej relacyjnej bazie danych. Magazyn typu ROLAP charakteryzuje się dużą skalowalnością i elastycznością, jednakże w stosunku do magazynów typu MOLAP cechują się niższą efektywnością przetwarzania danych. Często, w celu skrócenia czasu potrzebnego na wyznaczenie wyników zapytania relacje bazy danych są denormalizowane, np. zawierają wartości zagregowane, są wynikiem połączenia wielu innych relacji [1].

Magazyn danych zaprojektowany w technologii MOLAP do przechowywania danych wykorzystuje specjalizowane *wielowymiarowe tablice* (ang. multidimensional arrays) zwane też *kostkami danych* (ang. data cubes). Tablice te zawierają również pewne wstępnie przetworzone, tj. zagregowane, dane. Pozycja komórki wielowymiarowej tablicy jest wyznaczona przez kombinację wartości odpowiednich wymiarów. Pewne komórki mogą zawierać wartości puste. Kostki danych są tworzone przed rozpoczęciem przetwarzania i mają charakter statyczny, to znaczy, dodanie/usunięcie wymiaru, modyfikacja hierarchii wymiaru wymagają usunięcia i utworzenia kostki od nowa. Magazyn typu MOLAP charakteryzuje się wysoką efektywnością wielowymiarowego przetwarzania danych, jednakże, w stosunku do magazynów typu ROLAP, cechują się gorszą skalowalnością i elastycznością. Magazyn typu MOLAP jest odpowiedni do przetwarzania niewielkich i średnich wolumenów danych.

Magazyn typu HOLAP jest połączeniem obu podejść. Najczęściej, zasadniczy, scentralizowany, magazyn danych jest typu ROLAP, tj. dane są przechowywane w strukturze relacyjnej, natomiast złożone przetwarzanie danych jest realizowane na serwerze typu MOLAP, do którego są eksportowane niezbędne dane z magazynu ROLAP.

## 6 Fizyczna architektura magazynów danych

W praktyce spotyka się trzy podstawowe architektury fizyczne magazynów danych: architekturę scentralizowaną, architekturę sfederowaną, oraz architekturę wielowarstwową. W architekturze scentralizowanej występuje jeden scentralizowany fizyczny magazyn danych. Zaletą magazynu scentralizowanego jest łatwość dostępu do danych i łatwość administrowania magazynem danych. Alternatywą dla architektury scentralizowanej jest architektura rozproszona. Podstawowymi powodami, dla których stosuje się rozproszone architektury magazynów danych, są: skalowalność, zwiększenie dostępności i odporności na awarie oraz zmniejszenie czasu odpowiedzi systemu (dane są zlokalizowane bliżej aplikacji klientów, w tematycznych magazynach danych, wolumeny przeszukiwanych danych są mniejsze). W przypadku decentralizacji magazynu danych każdy z węzłów rozproszonego magazynu danych posiada własne repozytorium metadanych. W architekturze sfederowanej dane są zintegrowane logicznie, lecz fizycznie są przechowywane w osobnych bazach danych. Ważną cechą architektury sfederowanej jest więc fakt, że magazyn danych jest czysto wirtualny. *Tematyczne magazyny danych* (ang. data marts) przechowują szczegółowe dane, które są wykorzystywane do wykonywania zapytań.

W przypadku architektury wielowarstwowej dane są przechowywane fizycznie w centralnym (lub rozproszonym magazynie danych). W kolejnej warstwie magazynu danych są przechowywane coraz bardziej zagregowane dane. W lokalnych magazynach danych przechowywane są dane charakterystyczne dla danego oddziału. Są to dane wstępnie przetworzone i wzbogacone o agregaty, statystyki, itp. W kolejnej warstwie, np. klienta, są przechowywane już tylko wyniki raportów lub wyspecyfikowane agregaty.

Opracowując koncepcję systemu wspomagania podejmowania decyzji należy odpowiedzieć na dwa zasadnicze pytania odnośnie architektury takiego systemu i modelu przetwarzania. Pierwsze pytanie brzmi: czy analiza powinna mieć charakter rozproszony czy scentralizowany, innymi słowy, czy dane należy zgromadzić i przetwarzać w jednym miejscu w sposób scentralizowany, czy też korzystając z mechanizmu transakcji rozproszonych można przetwarzać dane w sposób rozproszony. Drugie pytanie dotyczy koegzystencji dwóch systemów – systemu bieżącej obsługi działania przedsiębiorstwa oraz systemu wspomagania podejmowania decyzji. Oba systemy operują na tych samych danych, stąd pytanie, czy oba modele OLAP i OLTP mogą współistnieć w tym samym systemie bazy danych, czy też powinny funkcjonować niezależnie.

W chwili obecnej odpowiedź na powyższe pytania brzmi: analiza powinna mieć charakter scentralizowany, a modele OLAP i OLTP powinny funkcjonować niezależnie. Oczywiście,

odpowiedź na pytania o architekturę i model przetwarzania jest uzależniona od aktualnego stanu rozwoju technologii informatycznej. Ze względu na charakter i pracochłonność obliczeń, częściowo również ze względu na problem autoryzacji dostępu do danych, analiza danych jest aktualnie prowadzona w sposób scentralizowany. Wraz z rozwojem sieci komputerowych, wzrostem prędkości transmisji danych, należy się jednak spodziewać przechodzenia od modelu przetwarzania analitycznego scentralizowanego do modelu przetwarzania analitycznego rozproszonego.

## 7 Efektywność magazynów danych

Istotne znaczenie dla funkcjonalności magazynu danych ma efektywność przetwarzania danych, w szczególności, efektywność przetwarzania złożonych i czasochłonnych zapytań, które przetwarzają duże ilości danych. W celu poprawy efektywności działania magazynów danych stosuje się wiele technik: materializowanie agregatów, przetwarzanie równoległe, partycjonowanie danych oraz indeksowanie danych.

### Materializowanie agregatów

Jedną z podstawowych technik gwarantujących znaczną poprawę efektywności analizy magazynu danych jest wstępne przeprowadzenie obliczeń i zmaterializowanie otrzymanych wyników w magazynie danych w celu ich późniejszego wykorzystania. Niestety, obliczenia agregatów są bardzo czasochłonne, a liczba możliwych agregatów jest wykładniczo zależna od liczby „wymiarów”. Stąd, rodzą się dwa zasadnicze pytania: (1) które z agregatów materializować, a które agregaty pozostawić do obliczeń w trybie on-line, oraz (2) w jaki sposób pielęgnować zmaterializowane agregaty (ponowne obliczanie agregatów, inkrementalna pielęgnacja agregatów)? Pojawia się szereg dodatkowych interesujących pytań: w jaki sposób reprezentować i realizować dostęp do agregatów? Czy nie należy również materializować pośrednich wyników obliczeń (nie tylko samych agregatów), np. wyników niektórych operacji połączeń, które są wspólne dla wielu agregatów? W jaki sposób definiować i pielęgnować indeksy na zmaterializowanych perspektywach?

Pierwszy z wymienionych problemów jest znany jako problem *selekcji zmaterializowanych perspektyw* (ang. view selection problem). Wybór agregatów, które należy zmaterializować, jest problemem obliczeniowo trudnym i zależy od charakterystyki obciążenia, częstości określonych zapytań, kosztu przechowywania i aktualizacji agregatów. Zaproponowano w literaturze szereg heurystyk [3, 9, 11] umożliwiających wybór zbioru agregatów do materializacji. Drugi z wymienionych problemów jest znany jako problem *pielęgnacji zmaterializowanych perspektyw* (ang. view maintenance problem). Każdorazowa zmiana wartości danych, które są argumentami funkcji agregacji pociąga za sobą konieczność aktualizacji zagregowanych danych. Najprostszym, ale nieefektywnym, rozwiązaniem tego problemu jest obliczanie wartości agregatu od początku. Rozwiązaniem znacznie bardziej efektywnym, ale również znacznie trudniejszym, jest inkrementalna pielęgnacja agregatów. W niektórych przypadkach, np. gdy zmaterializowana perspektywa (agregat) jest wynikiem wykonania operacji połączenia i grupowania, inkrementalna pielęgnacja zmaterializowanej perspektywy jest niemożliwa lub obliczeniowo nieefektywna. Materializacji powinny zatem podlegać agregaty, których aktualizacja ma miejsce bardzo rzadko.

### Przetwarzanie równoległe

*Przetwarzanie równoległe* (ang. parallel processing) polega na rozbiciu złożonych operacji na mniejsze, które następnie są wykonywane równoległe, np. na wielu procesorach lub komputerach. W efekcie, czas wykonania całej operacji jest krótszy. W przypadku hurtowni danych, najczęściej równoległe przetwarza się zapytania, sortuje dane, wykonuje operacje odczytu i zapisu na dysk, buduje relacje i indeksy, oraz wczytuje dane do magazynu danych.

Przetwarzanie równoległe wspierają m.in. systemy zarządzania bazami danych: *Oracle* (Oracle Corporation) , *DB2* (IBM), *OnLine Extended Parallel Server*, *OnLine Dynamic Server* (Informix), *Red Brick Warehouse* (Red Brick), *Sybase IQ* (Sybase) [3].

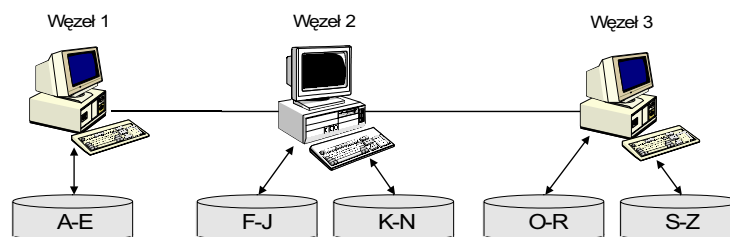
## Partycjonowanie danych

*Partycjonowanie danych* (ang. data partitioning) umożliwia automatyczne rozpraszanie danych (pochodzących z jednej lub wielu relacji) na wiele dyskach, znajdujących się w tym samym lub wielu węzłach (komputerach) sieci. Dzięki podziałowi dużej relacji na mniejsze: (1) bardzo kosztowne operacje wejścia/wyjścia, tj. dostępu do dysków, mogą być wykonywane równoległe, (2) równoważone jest obciążenie dysków, (3) polecenia SQL mogą być wykonywane równoległe, np. tworzenie relacji i indeksów, wykonywanie zapytań, (4) wzrasta bezpieczeństwo danych w przypadku awarii sprzętu, oraz (5) wzrasta szybkość tworzenia kopii zapasowych magazynu danych i szybkość odtwarzania danych po awarii.

W komercyjnych SZBD stosuje się cztery podstawowe techniki partycjonowania danych: *round-robin* (ang. round-robin partitioning), *partycjonowanie bazujące na wartościach* (ang. range partitioning), *partycjonowanie haszowe* (ang. hash partitioning), oraz *partycjonowanie hybrydowe* (ang. hybrid partitioning).

Technika *round-robin* umożliwia równomierne rozproszenie danych w węzłach sieci. Przykładowo, jeśli w sieci znajdują się trzy węzły, to pierwsza krotka relacji zostanie umieszczona w węźle pierwszym, druga – w węźle drugim, trzecia krotka – w węźle trzecim, czwarta – znów w węźle pierwszym itd. Rozwiązanie to ma jednak wadę. Ponieważ dane są rozproszone w sposób przypadkowy, więc odnalezienie żądanych informacji wymaga przeszukania wszystkich węzłów.

W przypadku *partycjonowania bazującego na wartościach* rozmieszczenie danych w sieci zależy od wartości samych danych. Przykładowo, relacja zawierająca informacje o klientach sieci supermarketów może być podzielona zgodnie z wartością pierwszej litery nazwiska, jak na Rys.8. Węzeł pierwszy posiada jeden dysk, na którym umieszczono klientów o nazwiskach rozpoczynających się od liter A do E. Węzły drugi i trzeci posiadają po dwa dyski, na których umieszczono klientów o nazwiskach odpowiednio z zakresów F–J, K–N, O–R, S–Z.



Rys. 8. Parcelacja danych bazująca na wartości

Ten sposób rozpraszania danych jest efektywny dla zapytań wykorzystujących zakresy wartości w predykatkach selekcji, ponieważ umożliwia szybki dostęp do danych z żadanego zakresu, bez potrzeby przeszukiwania wszystkich węzłów.

W przypadku *partycjonowania haszowego* dane są umieszczane w węzłach zgodnie z wartością funkcji haszowej. Argumentem wejściowym tej funkcji jest wartość atrybutu, a jej wynikiem – adres węzła, w którym zostanie umieszczona krotka. W celu odnalezienia żądanych informacji SZBD wykorzystują tę samą funkcję haszową. Zaletą tej metody jest możliwość automatycznego umieszczania w tym samym węźle krotek pochodzących z różnych, powiązanych z sobą relacji. W ten sposób zwiększa się efektywność wykonywania operacji łączenia krotek, gdyż łączone z sobą krotki znajdują się w tym samym węźle.

*Partycjonowanie hybrydowe* umożliwia dwustopniowe rozpraszanie danych. W kroku pierwszym dane są umieszczane w poszczególnych węzłach za pomocą parcelacji haszowej. W kroku drugim dane są umieszczane na poszczególnych dyskach danego węzła, za pomocą parcelacji bazującej na wartości. Dzięki tej technice wzrasta równomierność rozproszenia danych i obciążenia węzłów.

## Indeksowanie danych

Indeksowanie danych polega na łączeniu wartości indeksowanych atrybutów z adresami fizycznych bloków dyskowych w celu przyspieszenia dostępu do danych. Dla złożonych zapytań typu OLAP, operujących na ogromnej liczbie danych, standardowe indeksy typu B–drzew okazują się nieefektywne ponieważ: (1) nie zapewniają wystarczająco szybkiego dostępu do danych, (2) ich rozmiar jest zbyt duży, przez co wzrastają koszty ich przetwarzania, przechowywania i utrzymywania. Dla tej klasy zastosowań stosuje się więc nowe struktury indeksów, tj. *indeksy bitmapowe* (ang. bit–mapped indexes) i *indeksy połączeniowe* (ang. join indexes) [7], [8], [9].

Ideą *indeksów bitmapowych* jest wykorzystanie pojedynczych bitów do zapamiętania informacji o tym, że dana wartość atrybutu występuje w określonej krotce relacji. Dla każdej unikalnej wartości atrybutu jest przechowywana tablica bitów, zwana *mapą bitową*. Każdy bit mapy odpowiada jednej krotce relacji  $R$  – bit pierwszy odpowiada pierwszej krotce relacji  $R$ , bit drugi – drugiej krotce itp. Dla mapy  $A='w'$  bit  $n$  przyjmuje wartość jeden, jeśli atrybut  $A$  krotki o numerze  $n$  przyjmuje wartość 'w'. W przeciwnym przypadku bit  $n$  przyjmuje wartość zero. Liczba bitów mapy bitowej odpowiada liczbie krotek relacji  $R$ . Indeks bitmapowy jest zbiorem map bitowych dla wszystkich unikalnych wartości danego atrybutu. Indeks tego typu może również posiadać strukturę B–drzewa, w którego liściach zamiast adresów rekordów są przechowywane mapy bitowe.

Dla ilustracji idei indeksu bitmapowego rozważmy następujący przykład. Tabela 1 przedstawia fragment relacji *Sprzedaz* przechowującej informacje o sprzedaży samochodów. Dla atrybutu *kolor* zdefiniowano indeks bitmapowy składający się z dwóch map bitowych. Pierwsza z nich opisuje te krotki relacji, dla których atrybut *kolor* przyjmuje wartość 'zielony', a druga – te krotki, dla których *kolor*=*'niebieski'*. Ponieważ atrybut ten przyjmuje dwie różne wartości, więc indeks zawiera dwie mapy bitowe. Bit pierwszy mapy *kolor*=*'zielony'* przyjmuje wartość 1, co oznacza, że atrybut *kolor* pierwszej krotki przyjmuje wartość 'zielony'. Natomiast bit drugi mapy przyjmuje wartość 0 ponieważ wartością atrybutu *kolor* krotki drugiej nie jest 'zielony'.

Sprzedaż		
klientID	marka	Kolor
1010	Fiat	zielony
1020	BMW	niebieski
1030	Fiat	zielony
1040	Audi	zielony
1050	Volvo	zielony
1060	Fiat	niebieski
1070	Ford	niebieski
1080	Opel	zielony
1090	Opel	niebieski
1100	Ford	zielony

Tabela 1. Przykładowa relacja *Sprzedaz*

kolor	
zielony	niebieski
1	0
0	1
1	0
1	0
1	0
0	1
0	1
1	0
0	1
1	0

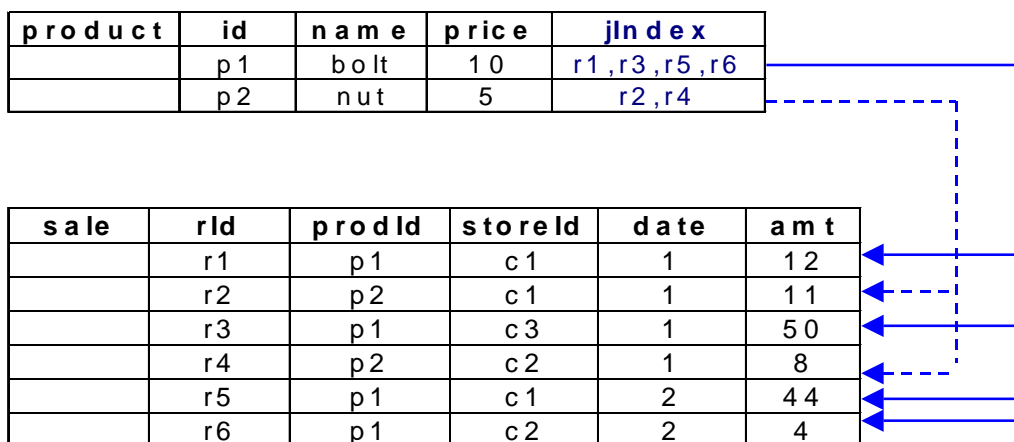
Tabela 2. Indeks bitmapowy dla atrybutu *kolor*

Podstawową zaletą indeksów bitmapowych jest ich mały rozmiar dla atrybutów o wąskiej dziedzinie wartości. Jako przykład rozważmy relację  $R$  zawierającą milion krotek. Przyjmijmy, że na atrybucie  $A$  przyjmującym cztery różne wartości utworzono indeks bitmapowy. Indeks ten składa się z czterech map bitowych o rozmiarze miliona bitów każda, co daje rozmiar każdej z map w przybliżeniu 125kB. Łączny rozmiar indeksu bitmapowego wynosi zatem w przybliżeniu 500kB ( $4 \times 125\text{kB}$ ). Zakładając, że adresy krotek są czterobajtowe, to odpowiedni indeks B–drzewa zbudowany na atrybucie  $A$ , ma rozmiar w przybliżeniu 4MB. Jest więc osiem razy większy od odpowiadającego mu indeksu bitmapowego. Rozważmy jednak tę samą relację  $R$ , której atrybut  $A$  przyjmuje 64 różne wartości. W tym przypadku indeks bitmapowy będzie miał rozmiar 8MB ( $64 \times 125\text{kB}$ ). Będzie więc większy niż odpowiadający mu indeks B–drzewa. W celu zmniejszenia rozmiarów indeksów bitmapowych stosuje się ich automatyczną kompresję.

Indeksy bitmapowe są bardziej efektywne od indeksów B–drzewa tylko dla określonej klasy zapytań adresowanych do bazy danych. Są to zapytania wykorzystujące dużą liczbę predykatów warunkowych oraz zapytania wykorzystujące funkcję COUNT. Większa efektywność tych indeksów wynika z: (1) dużej szybkości przetwarzania map bitowych za pomocą operatorów logicznych AND, OR i NOT (dla popularnych procesorów 64-bitowych, w jednym cyklu są przetwarzane 64 bity mapy), (2) małego rozmiaru indeksów (indeksy takie zdefiniowane na atrybutach o wąskiej dziedzinie są znacznie mniejsze od indeksów w postaci B–drzewa, dzięki czemu mogą być przechowywane w pamięci operacyjnej), oraz (3) możliwości wykonywania operacji logicznych i funkcji COUNT bezpośrednio na indeksach bitmapowych (znajdujących się w pamięci operacyjnej), a nie na samych krotkach; w rezultacie, system nie wykonuje kosztownych odczytów danych z dysku w czasie przetwarzania polecenia (dane są odczytywane dopiero po wykonaniu wszystkich operacji logicznych na indeksach; a ich wynikiem jest mapa bitowa opisująca tylko te krotki, które spełniają warunki selekcji). Indeksy bitmapowe wykazują jednak mniejszą efektywność dla zapytań wyszukujących dane z zadanych zakresów (operatory  $>$ ,  $<$  itp.), ponieważ dla takiej klasy zapytań należy wykonać wiele operacji sum logicznych na mapach bitowych dla wszystkich wartości z zadanego zakresu.

Zasadniczym problemem jest pielęgnacja indeksu bitmapowego. Pielęgnacja indeksu bitmapowego w czasie pracy systemu jest kosztowne, ponieważ: (1) wstawienie pojedynczej krotki do relacji wymaga uaktualnienia wszystkich map bitowych zdefiniowanych dla tej relacji (de facto wymaga zmiany rozmiaru mapy bitowej), oraz (2) w przypadku usuwania krotek należy utrzymywać dodatkową mapę bitową dla każdej relacji, której krotki są usuwane.

Innym rodzajem indeksu, który nie znalazł zastosowania w systemach baz danych, natomiast z powodzeniem jest stosowany w magazynach danych, jest tzw. *indeks połączeniowy* (ang. join index). Indeks tego typu łączy z sobą krotki z różnych relacji posiadające tę samą wartość atrybutu połączeniowego - jest więc strukturą zawierającą zmaterializowane połączenie wielu relacji. Indeks taki posiada strukturę B–drzewa zbudowanego na atrybucie połączeniowym relacji (bądź na wielu takich atrybutach). W przypadku magazynu danych o strukturze gwiazdy indeks połączeniowy wiąże krotki relacji wymiaru (lub wymiarów) z krotkami w relacji faktów. Przykładowo, dla magazynu danych zawierającego relację faktów o nazwie *sale* i relację wymiaru *product*, indeks połączeniowy zdefiniowany na atrybucie id (identyfikator produktu) dla każdej wartości tego atrybutu utrzymuje listę adresów rekordów relacji faktów zawierającą daną wartość atrybutu połączeniowego (Rys 8.)



Rys. 9. Indeks połączeniowy

Odmianą tego indeksu jest tzw. *bitmapowy indeks połączeniowy* (ang. bit-mapped join index), który różni się od powyższego tym, że w liściach zamiast adresów krotek znajdują się mapy bitowe opisujące krotki łączonych relacji.

## 8 Podsumowanie

Magazyn danych jest nie jest produktem ani też aplikacją. Jest to architektura przetwarzania danych opracowana z myślą o budowie systemów wspomaganie podejmowania decyzji. Magazyn danych łączy w sobie szereg produktów, z których każdy jest (lub może być) wykorzystany poza architekturą magazynu danych (narzędzia do integracji, czyszczenia danych, ładowania danych, itd.).

W architekturze magazynów danych dane pochodzące z wielu heterogenicznych źródeł są integrowane do globalnego schematu magazynu danych. Dane są przechowywane w magazynie, który z punktu widzenia użytkownika jest bazą danych. Po załadowaniu danych do magazynu użytkownicy mogą wydawać zapytania do magazynu (analogicznie do zwyczajnej bazy danych), ale, generalnie, nie wolno im modyfikować danych znajdujących się w magazynie, ponieważ takie modyfikacje nie znajdowałyby odzwierciedlenia w zmianach danych źródłowych i mogłyby prowadzić do niespójności magazynu. Istnieją trzy główne podejścia do zagadnienia aktualności magazynu danych [11]:

- Magazyn jest okresowo rekonstruowany z danych źródłowych. Rekonstrukcja przebiega najczęściej w nocy (gdy z magazynu nie korzystają żadne aplikacje) i wymaga wyłączenia magazynu. Wadą tej metody jest czasochłonność oraz fakt, że dane w magazynie nie odzwierciedlają aktualnego stanu danych źródłowych.
- Magazyn danych jest okresowo aktualizowany i zmiany, które zaszły w źródłach, są propagowane do magazynu. Takie podejście wymaga uaktualnienia znacznie mniejszej ilości danych. Z drugiej strony inkrementalna aktualizacja jest znacznie trudniejsza a algorytmy wykrywające zmiany w danych źródłowych są bardziej skomplikowane.
- Magazyn danych jest aktualizowany natychmiast po wystąpieniu zmiany w źródle. Ze względu na ilość komunikacji i przetwarzania związanego z takim rozwiązaniem, znajduje

ono zastosowanie tylko w małych magazynach danych, w których dane źródłowe zmieniają się stosunkowo powoli.

Magazyny danych odgrywają szczególną rolę w aplikacjach OLAP. Po pierwsze, magazyn danych jest potrzebny do scentralizowania i zorganizowania danych w taki sposób, by efektywnie wspierać aplikacje OLAP. Dane potrzebne do analizy mogą być rozproszone w wielu bazach danych przedsiębiorstwa. Poza tym zapytania OLAP są złożone i wykorzystują duże wolumeny danych, stąd ich wykonanie w systemach zorientowanych na przetwarzanie transakcji (czyli systemach OLTP) jest z reguły nieefektywne.

Istotne znaczenie z punktu widzenia funkcjonalności magazynu danych ma efektywność przetwarzania złożonych i czasochłonnych zapytań, które przetwarzają duże ilości danych. W celu poprawy efektywności działania magazynów danych stosuje się wiele technik, których celem jest poprawa tej efektywności. Należą do nich m.in. przetwarzanie równoległe, partycjonowanie danych, materializowanie agregatów, oraz nowe metody indeksowania, np. omówione w niniejszym artykule indeksy bitmapowe i połączeniowe.

Jakie problemy pozostają nadal nierozwiązane lub wymagają nowych rozwiązań w zakresie technologii magazynów danych? Najtrudniejszym problemem wymagającym nowych rozwiązań jest problem aktualizacji wymiarów i ewolucji schematu magazynu danych. Wymiary analizy zmieniają się z upływem czasu. Zmiany te prowadzą do istotnych problemów związanych z poprawnością wykonywanych analiz. Magazyn danych jest w dużym stopniu statyczny i problem zapewnienia poprawności ewolucji schematu magazynu danych jest problemem otwartym. Dalszych prac wymagają narzędzia i techniki akwizycji danych (czyszczenie danych, rozwiązywanie niespójności danych), tak aby zapewnić odpowiednią jakość i czystość danych w magazynie danych. Szereg problemów związanych z efektywną implementacją wielowymiarowego danych nadal pozostaje otwartych: kodowanie i kompresja danych, przetwarzania rzadkich kostek z dużą ilością wartości pustych, fragmentacja wielowymiarowych wielowymiarowego magazynu danych. Nadal trwają prace nad poprawą efektywności przetwarzania zapytań, selekcją i pielęgnacją materializowanych perspektyw, efektywnym wykorzystaniem materializowanych perspektyw w procesie optymalizacji zapytań. Trwają intensywne prace badawcze nad opracowaniem nowych narzędzi do zarządzania metadanymi, technikami odtwarzania magazynu danych po awarii w czasie procesu ładowania i odświeżania danych, technikami automatycznego archiwizowania danych w momencie ich dezaktualizacji. Technologia magazynów danych jest ciągle jeszcze technologią na etapie rozwoju.

## Bibliografia

- [1] Colliat G., *OLAP, Relational, and Multidimensional Database Systems*, SIGMOD Record, Vol.25, No.3, wrzesień 1996
- [2] Data Management Strategies, Cutter Information Corp., <http://www.cutter.com/itgroup/>
- [3] Garcia-Molina, H., Ullman, J.D., Widom, J., *Database System Implementation*, Prentice Hall, 2000
- [4] Hurwicz, M., *Take Your Data To The Cleaners*, BYTE, styczeń 1997
- [5] *Designing the Data Warehouse on Relational Databases*, <http://www.informix.com/informix/corpinfo/zines/whitpprs/stg/metacube.htm>
- [6] Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P., *Fundamentals of Data Warehouses*, Springer-Verlag Berlin Heidelberg, 2000
- [7] O'Neil P., Graefe G., *Multi-Table Joins Through Bitmaped Join Indices*, SIGMOD Record, Vol.24, No.3, wrzesień 1995
- [8] O'Neil P., Quass D., *Improved Performance with Variant Indexes*, materiały konferencyjne SIGMOD, Tuscon, Arizona, USA, 1997
- [9] Sarawagi S., *Indexing OLAP data*, Data Engineering Bulletin 20(1), 1997

- [10] *Optimizing Interactive Performace for the Data Warehouse*,  
<http://www.novasys.com.cy/sysprod06.thm>
- [11] Widom J., *Research Problems in Data Warehousing*, materiały konferencyjne, 4–th  
International Conference on Information and Knowledge Management (DIKM), 1995