Dane wielowymiarowe i język MDX w systemach OLAP

Tadeusz Pankowski
Politechnika Poznańska, Katedra Automatyki, Robotyki i Informatyki
e–mail: pankowski@put.poznan.pl

Abstrakt. Systemy OLAP są względnie nową a jednocześnie coraz popularniejszą i coraz ważniejszą klasą systemów informatycznych zajmującą się interakcyjnym tworzeniem, zarządzaniem i analizą danych postrzeganych jako struktury wielowymiarowe. Nowym językiem wspomagającym te funkcje jest język MDX (*MultiDimensional Expressions*). W pracy omówiono aktualne i proponowane cechy języka SQL zorientowane na funkcje OLAP, przedstawiono sformalizowane spojrzenie na dane wielowymiarowe proponując model MDW (*Model Danych Wielowymiarowych*) oraz opisano podstawowe idee i konstrukcje języka MDX.

1. Wprowadzenie

Termin OLAP (ang. *OnLine Analitical Processing*) został po raz pierwszy użyty w [2], chociaż już wcześniej istniały systemy przetwarzania danych realizujące idee technologii OLAP. Pojęcie to można zdefiniować jako "interakcyjny proces tworzenia, zarządzania i analizy danych postrzeganych jako wielowymiarowe tablice, a także sporządzania raportów w oparciu o te dane" [3]. Na obecnym etapie rozwoju systemów informatycznych systemy OLAP można traktować z jednej strony jako samodzielną klasę systemów ułatwiających dokonywanie wszechstronnych analiz danych pobieranych z operacyjnych baz danych lub częściej z hurtowni danych [3, 11, 17], a z drugiej – jako element pośredni pomiędzy hurtowniami danych, a systemami odkrywania wiedzy [5, 8].

Od kilku lat obserwujemy duże zainteresowanie systemami OLAP zarówno ze strony użytkowników, jak i ze strony producentów komercyjnych systemów zarządzania bazami danych. W systemach komercyjnych uwzględnianych jest coraz więcej narzędzi realizujących funkcje systemu OLAP. Na przykład firma Microsoft włączyła do serwera MS SQL Server 7.0 system o nazwie OLAP Services, gdzie zaimplementowano między innymi język o nazwie MDX (ang. *MultiDimensional Expressions*) wspomagający realizację funkcji systemu OLAP [9, 10, 14, 15, 18]. Firma Oracle oferuje produkty przeznaczone do wspomagania analizy poszczególnych grup zagadnień analitycznych od strony klienta (Oracle Express Analyzer, Oracle Financial Analyzer, Oracle Sales Analyzer). W nowej wersji standardu języka SQL proponuje się uwzględnienie środków zorientowanych na funkcji OLAP [4, 7].

Głównym celem niniejszej pracy jest omówienie zasad projektowania i operowania na danych wielowymiarowych. Rozpoczynamy od analizy cech języka SQL przeznaczonych do wspomagania OLAP – cech już zaimplementowanych oraz nowych propozycji wprowadzanych do standardu (rozdział 3). W rozdziale 4 proponujemy sformalizowane spojrzenie na dane wielowymiarowe przedstawiając model MDW (*Model Danych Wielowymiarowych*). Następnie (rozdział 5) omawiamy podstawowych idei i konstrukcji języka MDX. Przykłady ilustrujące rozważania odnoszą się do przykładowej bazy danych podanej w rozdziale 2.

2. Przykładowa baza danych

W dalszym ciągu odwoływać się będziemy do przykładowej bazy danych Salon, zawierającej dane o sprzedaży samochodów, przedstawionej na rys. 1. W bazie danych są cztery tabele, przy czym Sprzedaz zawiera dane o zrealizowanych transakcjach: IdTow (identyfikator towaru, samochodu), IdKli (identyfikator klienta), Data (data sprzedaży), Wartosc (przychód związany ze sprzedażą), Koszt (koszt związany ze sprzedażą). Zawartość pozostałych tabel nie wymaga objaśnień. Zauważmy, że tabele Towary, Dni i Klienci nie są w trzeciej postaci normalnej [3, 13]. Wynika to z dwóch powodów, po pierwsze ułatwi nam prezentację, a po drugie – takie odstępstwa od

normalizacji są dopuszczalne, a często wręcz zalecane ze względu na efektywność przetwarzania, co ma szczególne znaczenia przy tworzeniu hurtowni danych.

Sprzedaz

	IdTow	IdKli	Data	Wartosc	Koszt
1	11	101	10.01.1998	30000.0	25000.0
2	11	201	23.04.1999	32000.0	28000.0
3	12	102	15.02.1998	20000.0	18000.0
4	12	202	10.01.1998	22000.0	20000.0
5	21	103	20.04.1998	35000.0	30000.0
6	21	203	22.02.1999	36000.0	31000.0
7	22	101	02.05.1999	60000.0	45000.0
8	22	201	20.04.1998	58000.0	44000.0
9	22	204	23.04.1999	60000.0	48000.0

Towary

	IdTow	Producent	Marka
1	11	Fiat	Punto
2	12	Fiat	Uno
3	21	Ford	Escord
4	22	Ford	Fokus

Klienci

	IdKLi	Miasto	Wojewodztwo	StanCywilny	Plec
1	101	Leszno	wielkopolskie	M	M
2	102	Leszno	wielkopolskie	W	K
3	103	Gniezno	wielkopolskie	M	K
4	104	Gniezno	wielkopolskie	W	M
5	201	Sopot	pomorskie	M	M
6	202	Sopot	pomorskie	M	K
7	203	Gdynia	pomorskie	M	K
8	204	Gdynia	pomorskie	W	K

Dni

	Data	NrDnia	NrTygodnia	NrMca	Kwartal	Rok
1	10.01.1998	7	2	1	KW1	1998
2	15.02.1998	1	8	2	KW1	1998
3	20.04.1998	2	17	4	KW2	1998
4	22.02.1999	2	9	2	KW1	1999
5	23.04.1999	6	17	4	KW2	1999
6	02.05.1999	1	19	5	KW2	1999

Rys. 1. Stan bazy danych salonu sprzedaży samochodów Salon

Prezentowane dane mają znaczenie wybitnie ewidencyjne – gromadzenie tak szczegółowych danych ewidencyjnych jest niezbędne do bieżącej działalności firmy. Jednak nagromadzenie dużej liczby danych opisujących działalność firmy w dłuższym okresie czasu umożliwia znacznie więcej – umożliwia dokonanie analizy tej działalności, wykrycia pewnych zależności, wyciągnięcie wniosków co do dalszego postępowania. W szczególności kierownictwo firmy może być zainteresowane realizacją następujących poleceń:

- Podaj ogólną wartość sprzedaży poszczególnych samochodów według wzrastających poziomów agregacji: od miasta, przez województwo do kraju i od marki do producenta, dla lat 1998 i 1999.
- Przeanalizuj zysk ze sprzedaży dla poszczególnych marek w poszczególnych miesiącach. Uporządkuj miesiące według wzrastającej wartości zysku.
- Czy odchylenia liczby zawartych transakcji od średniej w poszczególnych miesiącach są istotne?

Udzielanie odpowiedzi na tego typu pytania realizowane jest za pomocą środków określanych jako systemy (narzędzia, usługi) OLAP. Przy realizacji tych systemów przyjmuje się że:

1. Analizowane dane znajdują się w bazach danych (w bazach operacyjnych, hurtowniach danych lub składach danych), których wielkość może być bardzo zróżnicowana i sięgać od kilku megado wielu terabajtów. W utrzymywaniu i podstawowym przetwarzaniu tych danych wykorzystywane są możliwości konwencjonalnych serwerów baz danych.
2. Przetwarzanie analityczne, w tym wykonywanie analiz wielowymiarowych, wymaga przedstawiania danych za pomocą modeli wielowymiarowych, w których wyróżnia się takie pojęcia jak: *tabela faktów* (podstawowa tabela, w której zawarte są te wielkości, *pomiary*, analizą których jesteśmy zainteresowani), *funkcje agregujące* (określają, w jaki sposób z danych elementarnych tworzone są dane zagregowane), *wymiary* (określają przestrzeń, w której analizowane są pomiary), *człony wymiarów* (służą do określenia *punktów* w przestrzeni wielowymiarowej), *komórki* (zawierają wartości pomiarów w określonym punkcie), *zbiory komórek*, *kostka* (zbiór wszystkich możliwych komórek).
3. Formułowanie zapytań wielowymiarowych, tj. zapytań zgodnych z wielowymiarowym modelem danych, wymaga języków zapytań zorientowanych na dane wielowymiarowe. Dane wielowymiarowe, a także wyniki analizy wielowymiarowej najwygodniej jest odczytywać, gdy w i-zualizowane są w postaci graficznej.

3. Cechy języka SQL wspomagające funkcje OLAP

Producenci komercyjnych serwerów baz danych wł ączają do implementowanych wersji j ęzyka SQL pewne cechy zorientowane na OLAP, które jeszcze formalnie nie zostały wprowadzone do standardu [9, 12]. Cechy te omawiamy w podrozdziale 3.1. Jednocze śnie ciągle rozwijany standard języka SQL uwzgl ędnia te i jeszcze inne rozszerzenia wspomagaj ące funkcje OLAP (podrozdz. 3.2)

3.1. Operacje CUBE i ROLLUP

Pierwsze rozszerzenie j ęzyka SQL związane z funkcjami OLAP dotyczyło rozszerzenia frazy `GROUP BY` wyrażenia `SELECT` o operacje `CUBE` i `ROLLUP` do postaci:

`select ... group by <lista_kolumn> with {cube | rollup}`, w SQL Serverze

`select ... group by {cube | rollup}(<lista_kolumn>)`, w Oraclu.

W wyrażeniu `SELECT` można ponadto używać operację `TOP-n` oraz o funkcję `GROUPING`.

Operacja CUBE

`CUBE` określa, że do zbioru wynikowego zwracanego standardowo przez `GROUP BY`, dołączane są wiersze agreguj ące dla wszystkich pozostałych mo żliwych kombinacji kolumn (wymiarów) pod anych w `<lista_kolumn>`. Jeżeli podano *n* kolumn, to liczba mo żliwych kombinacji wynosi 2^n. Dla trzech kolumn R(ok), P(roducent), W(ojewództwo) mamy nast ępujące osiem kombinacji:

```
(R, P, W), (R, P, All), (R, All, W), (All, P, W),
(R, All, All), (All, P, All), (All, All, W), (All, All, All),
```

gdzie `(R, P, W)` oznacza, że wiersze agreguj ące tworzone są dla każdej różnej trójki warto ści (*rok, producent, województwo*), a `(R, P, All)` oznacza, że wiersze agreguj ące tworzone są dla każdej różnej pary warto ści (*rok, producent*), a w wierszach tych trzecia kolumna wypełniana jest wart o-ścią `NULL`. Podobnie dla pozostałych kombinacji. W tabeli wynikowej mamy wówczas du żą liczbę warto ści `NULL`. Dla rozróżnienia tych warto ści `NULL`, które oznaczają warto ści nieznane (puste) od tych, które generowane s ą podczas agregowania za pomoc ą operacji `CUBE` i `ROLLUP`, wykorzystuje się funkcję `GROUPING`: w pierwszym przypadku zwraca ona warto ść 0, a w drugim 1. Do wyboru wierszy o najwi ększej lub najmniejszej warto ści podanego zestawu kolumn słu ży operacja `TOP-n`. Przykład wyra żenia `SELECT` z frazą `GROUP BY ... WITH CUBE` i tabelę wynikową po jego wykona-niu na tabelach z rys.1, przedstawi ono na rys 2.

```
select D.Rok,T.Producent,K.Wojewodztwo, sum(S.Wartosc) as Wartosc
from Sprzedaz S, Towary T, Klienci K, Dni D
where S.IdTow=T.IdTow AND S.IdKli=K.IdKli AND S.Data=D.Data
group by D.Rok,T.Producent,K.Wojewodztwo with cube
```

Rok	Producent	Wojewodztwo	Wartosc
1998	Fiat	pomorskie	22000.0
1998	Fiat	wielkopolskie	50000.0
1998	Fiat	NULL	72000.0
1998	Ford	pomorskie	58000.0
1998	Ford	wielkopolskie	35000.0
1998	Ford	NULL	93000.0
1998	NULL	NULL	165000.0
1999	Fiat	pomorskie	32000.0
1999	Fiat	NULL	32000.0
1999	Ford	pomorskie	96000.0
1999	Ford	wielkopolskie	60000.0
1999	Ford	NULL	156000.0
1999	NULL	NULL	188000.0
NULL	NULL	NULL	353000.0
NULL	Fiat	pomorskie	54000.0
NULL	Fiat	wielkopolskie	50000.0

```
NULL   Fiat       NULL               104000.0
NULL   Ford       pomorskie          154000.0
NULL   Ford       wielkopolskie       95000.0
NULL   Ford       NULL               249000.0
1998   NULL       pomorskie           80000.0
1999   NULL       pomorskie          128000.0
NULL   NULL       pomorskie          208000.0
1998   NULL       wielkopolskie       85000.0
1999   NULL       wielkopolskie       60000.0
NULL   NULL       wielkopolskie      145000.0

(26 row(s) affected)
```

Rys. 2. Przykład wykorzystania operacji CUBE

Operacja ROLLUP

ROLLUP do zbioru wynikowego zwracanego standardowo przez GROUP BY, dołącza wiersze agregujące dla hierarchicznej kombinacji kolumn podanych w <*lista_kolumn*>, przy czym kolejność kolumn określa poziomy hierarchii: pierwsza kolumna oznacza poziom najwyższy, druga o jeden niższy itd. Jeżeli podano *n* kolumn, to liczba takich kombinacji wynosi $n + 1$. Agregowanie następuje zgodnie z zasadą *zwijania hierarchii* z dołu do góry – od najniższego, najbardziej szczegółowego poziomu do najwyższego. Na przykład dla trzech kolumn (poziomów hierarchii) R(ok), K(wartał), M(iesiąc), tworzone będą wiersze zagregowane według następujących schematów:

```
(R, K, M), (R, K, All), (R, All, All), (All, All, All).
```

Zastosowanie ROLLUP dla takich kolumn jak Rok, Producent, Województwo, a więc dla niezależnych wymiarów, jest teoretycznie możliwe jednak mało sensowne. Jeszcze mniej sensowne jest stosowanie operacji CUBE dla kolumn należących do jednej hierarchii, a więc na przykład dla kolumn Rok, Kwartał i Miesiąc. Przykład wyrażenia SELECT z frazą GROUP BY ... WITH ROLLUP i tabelę wynikową po jego wykonaniu na tabelach z rys.1, przedstawiono na rys 3.

```
select D.Rok,D.Kwartal,D.NrMca, sum(S.Wartosc) as Wartosc
from Sprzedaz S, Dni D
where S.Data=D.Data
group by D.Rok,D.Kwartal,D.NrMca with rollup

Rok    Kwartal NrMca  Wartosc
------ ------- ------ -------
1998   KW1     1      52000.0
1998   KW1     2      20000.0
1998   KW1     NULL   72000.0
1998   KW2     4      93000.0
1998   KW2     NULL   93000.0
1998   NULL    NULL  165000.0
1999   KW1     2      36000.0
1999   KW1     NULL   36000.0
1999   KW2     4      92000.0
1999   KW2     5      60000.0
1999   KW2     NULL  152000.0
1999   NULL    NULL  188000.0
NULL   NULL    NULL  353000.0

(13 row(s) affected)
```

Rys. 3. Przykład wykorzystania operacji ROLLUP

3.2. SQL/OLAP – rozszerzenia języka SQL w standardzie SQL:1999

Długo oczekiwana trzecia generacja standardu SQL, znana obecnie pod nazwą SQL:1999, a wcześniej określana jako SQL3, zawiera rozszerzenie określane jako SQL/OLAP. Oficjalna robocza wersja poprawki do standardu zapowiadana jest na koniec roku 2000. Aktualna wersja tego dokumentu [4, 7] wprowadza wiele mechanizmów przeznaczonych do analizy danych. Są to:

1. Nowe funkcje numeryczne (LN, EXP, POWER, SQRT, FLOOR, CAIL, WIDTH_BUCKED); nowe funkcje agregujące 1-argumentowe (logiczne EVERY i SOME; statystyczne: VAR_POP, VAR_SAMP, STDEV_POP i STDDEV_SAMP – wariancja i odchylenie standardowe odpowiednio z całej populacji i z próby); funkcje agregujące 2-argumentowe, z których jeden argument jest zmienną zależną, a drugi zmienną niezależną: COVAR_POP i COVAR_SAMP (kowariancja z całej populacji i z próby), CORR (współczynnik korelacji), REGR_SLOPE (regresja liniowa wyznaczona metodą najmniejszych kwadratów); odwrotne funkcje do funkcji rozkładu: PERCENTILE_CONT i PERCENTILE_DISC (do wyznaczania percentyli).
2. Funkcje "what-if": odnoszą się do takich funkcji OLAP jak RANK, DENSE_RANK, PERCENT_RANK i CUME_DIST. Argumentami ich są wartość A i wyrażenie porządkujące stosowane do wszystkich wierszy w grupie. Do uporządkowanego wielozbioru wartości dołączane jest A i następnie wyznaczana jest odpowiednia charakterystyka położenia A w wielozbiorze.

Mimo, że omówione funkcje są ważne z punktu widzenia wspomagania zadań OLAP, to bardziej istotne jest wprowadzenie do SQL-a zupełnie nowego pojęcia, jakim jest pojęcie *okna* (ang. *window*). Okno oznacza wybór wierszy tabeli zdefiniowany przez grupowanie wszystkich jej wierszy na podstawie:

- jednakowych wartości w podanej kolumnie lub zbiorze kolumn,
- bliskości do określonego (identyfikowanego) wiersza określonej jako liczba wierszy poprzedzających zidentyfikowany wiersz lub następujących po nim, bądź określonej jako liczba grup wierszy powiązanych poprzez wartości w kolumnie.

Istnieje również mechanizm wykluczania z okna wiersza identyfikowanego lub wszystkich wierszy o równych mu wartościach w podanych kolumnach.

Ogólna postać wyrażenia tabelowego jest według propozycji SQL/OLAP następująca:

```
<wyrażenie tabelowe> ::= <fraza from> [<fraza where>][<fraza group by>]
                         [<fraza having>][<fraza window>]
```

Wynikiem wyrażenia tabelowego jest wówczas zbiór wierszy łącznie z deskryptorami struktur okien. Oprócz więc pojęcia *tabela grupowana* (ang. *grouped table*) , związanego z działaniem frazy `GROUP BY`, wprowadzone zostaje pojęcie *tabela okienkowana* (ang. *windowed table*), związane z działaniem frazy `WINDOW`.

Kolejnym rozszerzeniem jest włączenie możliwości sterowania sposobem postępowania z wartościami `NULL` podczas sortowania definiowanego we frazie `ORDERED BY`. Wprowadzono opcje `NULLS FIRST` i `NULLS LAST` polecające umieszczać krotki z pustymi wartościami argumentów sortowania na początku bądź końcu posortowanego zbioru wynikowego.

3.3. Podsumowanie

Omówione nowe mechanizmy języka SQL, które bądź już zostały zaimplementowane w niektórych serwerach baz danych, bądź są proponowane w opracowywanym ciągle standardzie SQL:1999, mają na celu wspomaganie funkcji przetwarzania danych w relacyjnej bazie danych dla potrzeb systemów OLAP. Mechanizmy te jednak ciągle odwołują się do relacyjnego postrzegania danych jako znormalizowanych (płaskich) tabel. Nie proponuje się żadnych pojęć nawiązujących do wielowymiarowości danych, do rozróżnienia między danymi odnoszącymi się do wymiarów, a danymi odnoszącymi się do faktów. Nie uwzględnia się hierarchicznych zależności między danymi będącymi członami wymiarów. Oczywiście w żaden sposób nie odnosi się również do sposobu wizualizacji danych. Wynika z tego, że środowisko serwera SQL-owej bazy danych może realizować jedynie funkcje związane z pamiętaniem danych i wykonywaniem podstawowych operacji na tych danych – również operacji bezpośrednio związanych z analizą wielowymiarową. Obecny stan rzeczy jest więc taki, że producenci komercyjnych systemów zarządzania bazami danych, a także producenci niezależni, proponują wyspecjalizowane pakiety przeznaczone do organizacji hurtowni

danych i/lub do analizy danych w technologii OLAP, przy czym pakiety te korzystają z serwera baz danych. Powstaje więc pytanie, do jakiego stopnia funkcje związane z integracją i eksploracją danych zostaną włączone do podstawowego zestawu funkcji serwera baz danych, a w jakim zakresie realizowane będą w oddzielnych systemach współpracujących z tymi serwerami.

4. Algebraiczny model danych wielowymiarowych

W tym rozdziale przedstawiamy propozycję sformalizowanego modelu danych wielowymiarowych MDW. Model ten uwzględnia najważniejsze cechy występujące podczas tworzenia danych wielowymiarowych w systemach OLAP [1, 6, 16]. Ściśle zdefiniowane pojęcia modelu MDW pozwolą nam lepiej zrozumieć i objaśnić pojęcia związane z modelowaniem i operowaniem za pomocą języka MDX. Odwoływać się będziemy do przykładowej bazy danych przedstawionej na rys. 1.

4.1. Model schematu danych wielowymiarowych

W modelu MDW dane wielowymiarowe na poziomie schematu definiowane są następująco:

1. Niech $R(U)$, gdzie R jest nazwą tabeli, a U jest zbiorem atrybutów, będzie zadanym schematem *tabeli faktów*.
2. Niech M będzie zbiorem *(nazw) pomiarów*, a AGR = {Sum, Max, Min, Count, CountDistinct} niech będzie zbiorem *funkcji agregujących*. Wówczas parę o jednej z następujących postaci:

   $M = (m, agg(R.A))$,

   $M = (m, f(m_1, ..., m_k)$,

   gdzie $m, m_i \in$ M, $agg \in$ AGG, $R.A$ jest kolumną z tabeli faktów, a f jest *funkcją k*-argumentową, nazywamy *definicją pomiaru m*. Pomiary zdefiniowane z zastosowaniem funkcji agregujących nazywamy *pomiarami agregowanymi*, a pomiary zdefiniowane za pomocą funkcji odwołującej się do innych pomiarów – *pomiarami obliczanymi*.
3. Niech D będzie zbiorem *(nazw) wymiarów*, a E niech będzie zbiorem symboli zwanych *członami*. Wówczas parę o postaci

   $D = (d, \{e_1, ..., e_r\})$,

   gdzie $d \in$ D i $\{e_1, ..., e_r\} \subseteq$ E, nazywamy *definicją wymiaru d*; zbiór mem(d) $=\{e_1, ..., e_r\}$ nazywamy zbiorem *członów wymiaru d*. Przyjmujemy ponadto, że:

   - wymiary są rozłączne, czyli każdy człon $e \in$ E może należeć co najwyżej do jednego wymiaru,
   - zbiór członów wymiaru może być podzielony na rozłączne podzbiory wiązane z *nazwami poziomów* – poziomy uporządkowane są hierarchicznie zgodnie ze stopniem ogólności ich znaczeń;
   - z każdym członem $e \in$ E może być związany zbiór *parametrów*, co zapisujemy $e(P_1 : p_1, ..., P_k : p_k)$, gdzie P_i jest *nazwą parametru*, a p_i *wartością parametru* o nazwie P_i.
4. Niech R będzie schematem tabeli faktów, $M_1, ..., M_n$ – definicjami pomiarów, a $D_1, ..., D_m$ – definicjami wymiarów. Wówczas

   $$C = (R, \{M_1, ..., M_n\}, \{D_1, ..., D_m\})$$

   nazywamy *schematem kostki*.

   Zinterpretujemy te pojęcia odwołując się do przykładowej bazy danych z rys. 1.

Tabela faktów (ang. *fact table)*

Tabelą faktów nazywamy tabelę pamiętaną w bazie danych (hurtowni danych) i zawierającą informacje, które mają podlegać analizie w systemie OLAP. W naszym przypadku jest to tabela

Sprzedaz(IdTow, IdKli, Data, Wartosc, Koszt).

Dane w niej zawarte - a więc data sprzedaży, wartość sprzedaży, koszty związane ze sprzedawanym towarem, towary i klienci uczestniczący w transakcji sprzedaży - będą podlegały analizie w przestrzeni wielowymiarowej, na przykład wyznaczonej przez czas, klientów i sprzedawane towary.

Pomiary (ang. *measures*)

Pomiary (zwane niekiedy w literaturze polskoj ęzycznej niezbyt fortunnie *miarami*) definiowane są jako te cechy, których wartości są istotne dla użytkownika w procesie analizy. Wyznaczane są na podstawie tabeli faktów. Intuicyjnie rzecz biorąc, wartości te uzyskiwane są w wyniku pomiarów w przestrzeni wielowymiarowej. Wartości pomiarów rozważane są na różnych poziomach agregacji, na przykład na poziomie miesiąca, kwartału lub roku (w wymiarze czasowym), albo też na poziomie miasta, województwa czy kraju (w wymiarze terytorialnym). Uzyskanie zagregowanych wartości pomiarów odbywa się bądź za pomocą odpowiedniej funkcji agregującej stosowanej do wskazanej kolumny tabeli faktów, bądź też poprzez przeliczenie wartości innych pomiarów wcześniej zagregowanych.

W naszym przykładzie możemy zdefiniować następujące pomiary:

Przychód : Sum(Wartosc),
Koszt : Sum(Koszt),
Zysk : Przychod – Koszt,
DataOstSprz : Max(Data),
LiczbaTrans : Count(IdKli),
LiczbaKli : CountDistinct(IdKli).

W przypadku pomiarów Przychód i Koszt obliczanie wartości pomiarów odbywa się poprzez sumowanie wartości w kolumnach odpowiednio Wartosc i Koszt tabeli faktów. Pomiar Zysk jest pomiarem obliczanym i jego wartość na żądanym poziomie agregacji wyliczana jest z zagregowanych wartości pomiarów Przychod i Koszt. Pomiar DataOstSprz obliczany jest z zastosowaniem funkcji agregującej Max. Wartość pomiaru LiczbaTrans jest równa liczbie wartości w kolumnie IdKli tabeli Sprzedaz (a więc liczbie wierszy w tej tabeli). Natomiast wartość pomiaru LiczbaKli jest równa liczbie różnych wartości w kolumnie IdKli tabeli Sprzedaz.

Z punktu widzenia optymalizacji obliczania wartości zagregowanych istotne jest, w jakim stopniu wartości obliczone na niższym poziomie zagregowania (na przykład półrocza) mogą być wykorzystane do obliczenia wartości na wyższym poziomie zagregowania (na przykład roku). Istotne jest więc czy i kiedy zachodzi równość:

$agg(X \cup Y) = agg'(agg(X), agg(Y))$,

gdzie X i Y oznaczają zbiory wartości pewnego pomiaru, na przykład wartość sprzedaży lub liczbę klientów w pierwszym i drugim półroczu.

Zauważmy, że dla rozważanych funkcji agregujących możemy stosować następujące reguły (które w sposób naturalny możemy uogólnić na dowolną skończoną liczbę składników):

$\mathrm{Sum}(X \cup Y) = \mathrm{Sum}(\mathrm{Sum}(X), \mathrm{Sum}(Y))$,
$\mathrm{Max}(X \cup Y) = \mathrm{Max}(\mathrm{Max}(X), \mathrm{Max}(Y))$,
$\mathrm{Min}(X \cup Y) = \mathrm{Min}(\mathrm{Min}(X), \mathrm{Min}(Y))$,
$\mathrm{Count}(X \cup Y) = \mathrm{Sum}(\mathrm{Count}(X), \mathrm{Count}(Y))$, przy czym $\mathrm{Count}(X)$ może być stosowane tylko w przypadku, gdy X oznacza kolumnę tabeli faktów,
$\mathrm{CountDistinct}(X \cup Y)$ – musi być zawsze wyliczane z tabeli faktów.

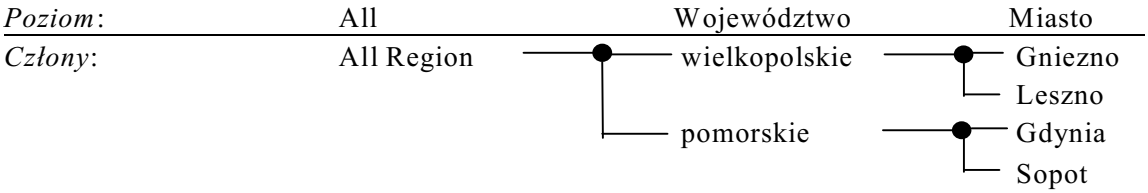
Wymiary (ang. *dimensions*)

Wymiary definiują przestrzeń, w której analizowane są pomiary. Każdy wymiar organizowany jest hierarchicznie poprzez określenie:

- hierarchii (opcjonalnie) (ang. *hierarchy*),
- poziomów (ang. *levels*),
- członów (ang. *members*).

W naszym przykładzie możemy wyodrębnić następujące wymiary:

Wymiar : Region
Hierarchia : – (w tym wymiarze określamy tylko jedną hierarchię)



Wymiar Region obejmuje więc następujący zbiór członów:

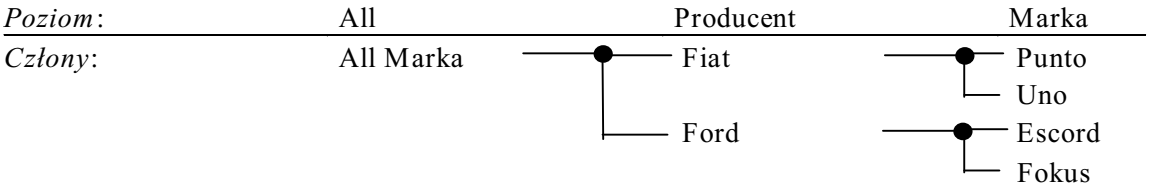
Region = { Region.[All Region],
Region.[All Region].wielkopolskie,
Region.[All Region].pomorskie,
Region.[All Region].wielkopolskie.Gniezno,
Region.[All Region].wielkopolskie.Leszno,
Region.[All Region].pomorskie.Gdynia,
Region.[All Region].pomorskie.Sopot }.

W przypadku, gdy nie ma problemu z jednoznacznością nazw, możemy pominąć przedrostki i wówczas zbiór członów wymiaru Region jest następujący:

{[All Region], wielkopolskie, pomorskie, Gniezno, Leszno, Gdynia, Sopot}.

Wyróżniony człon [All Region] traktowany jest jako pojedynczy człon wyróżnionego poziomu All i służy do agregowanie pomiarów na najwyższym poziomie.

Wymiar : Marka
Hierarchia : – (w tym wymiarze określamy tylko jedną hierarchię)



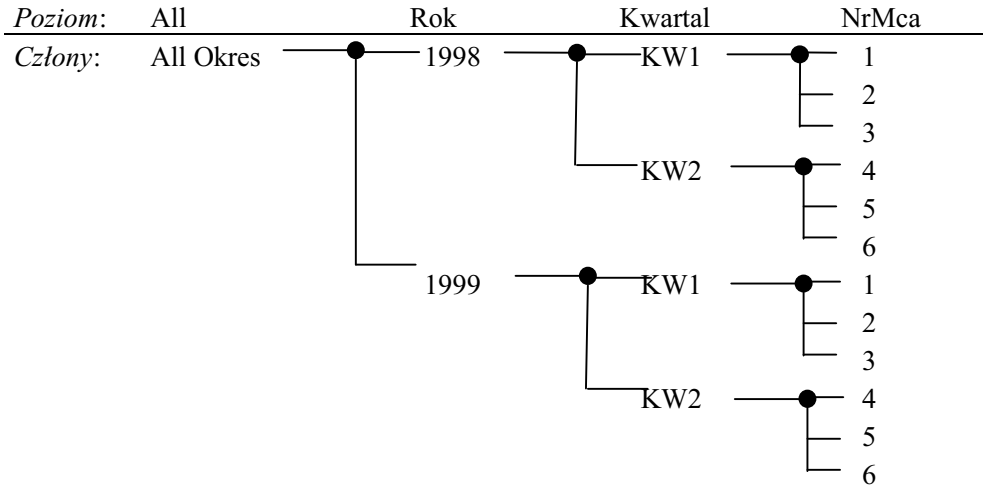
Wymiar Marka obejmuje więc następujący zbiór członów:

[All Marka], Fiat, Ford, Punto, Uno, Escord, Fokus.

Podobnie możemy zdefiniować wymiar czasowy Okres złożony z następującego zbioru członów (zauważmy, że tym razem dla zachowania jednoznaczności konieczne jest prefiksowanie):

Okres = {[All Okres], 1998, 1999, 1998.KW1, 1998.KW2, 1999.KW1, 1999.KW2,
1998.KW1.1, 1998.KW1.2, 1998.KW1.3, 1998.KW2.1, 1998.KW2.2, 1998.KW2.3,
1999.KW1.1, 1999.KW1.2, 1999.KW1.3, 1999.KW2.1, 1999.KW2.2, 1999.KW2.3}

Wymiar : Okres
Hierarchia : – (w tym wymiarze określamy tylko jedną hierarchię)



Niekiedy powstaje potrzeba definiowania kilku hierarchii w obrębie jednego wymiaru. Przypuśćmy, że chcemy analizować pomiary ze względu na dni i miesiące oraz na dni i tygodnie. Ponieważ jeden tydzień może być rozdzielony pomiędzy dwa sąsiednie miesiące, istnieje konieczność utworzenia dwóch hierarchii w wymiarze Okres - jedna, na przykład o nazwie MD, obejmowałaby poziomy: Miesiąc-Dzień, a druga, na przykład o nazwie TD, obejmowałaby poziomy Tydzien-Dzień. W takim przypadku prefiksowanie członów obejmuje też nazwę hierarchii: na przykład człon Okres.MD.3.5 oznacza piąty dzień trzeciego miesiąca w hierarchii MD wymiaru Okres, a Okres.TD.3.5 oznacza piąty dzień trzeciego tygodnia w hierarchii TD wymiaru Okres.

Dla każdej kolumny A tabeli faktów, używanej do zdefiniowania pomiarów, określamy *tabelę rzutów* kolumny A na poszczególne człony: TB_A(A, e_1, ..., e_n), gdzie e_1, ..., e_n są wszystkimi członami wszystkich wymiarów. W kolumnie TB_A.A pamiętane są wszystkie wartości z kolumny $R.A$ z powtórzeniami, a w pozostałych kolumnach znajdują się wartości binarne. Żądamy przy tym, aby spełniony był następujący warunek: dla każdego wiersza $t \in$ TB_A, $t.e_i = 1$ wtedy i tylko wtedy, gdy wartość $t.A$ w źródłowej bazie danych powiązana jest z symbolem (członem) e_i. Przykładową tabelę rzutów utworzoną dla przykładu z rys. 1, przedstawiono w tab. 1.

Tabela 1. Tabela rzutów kolumny Koszt na człony wymiarów

TB_Koszt

Koszt	All Region	wielkopolskie	...	Sopot	...	Fiat	...	[1998].[KW2]
25 000.00	1	1		0		1		0
28 000.00	1	0		1		1		0
18 000.00	1	1		0		1		0
20 000.00	1	0		1		1		0
30 000.00	1	1		0		0		1
31 000.00	1	0		0		0		0
45 000.00	1	1		0		0		0
44 000.00	1	0		1		0		1
48 000.00	1	0		0		0		0

Binarne kolumny tabeli rzutów można efektywnia reprezentować w postaci list binarnych, które następnie można kompresować (na przykład według metody kompresji list binarnych opisanej w [13]) uzyskując istotną poprawę zużycia pamięci. Listy binarne, co zobaczymy dalej, umożliwiają efektywne wyznaczanie danych zagregowanych (zbiorów komórek).

Z członami mogą być związane parametry. Na przykład każdy klient może mieć parametry dotyczące jego stanu cywilnego i płci (por. tabela Klienci na rys. 1). Wartościami tych parametrów dla członu 102 byłyby na przykład: W (wolny) i K (kobieta), co zapisujemy:

102(StanCywilny : 'W', Plec : 'K').

Wybór wymiarów, hierarchii, poziomów i członów zale ży od tego, z jakiego punktu widzenia ma być przeprowadzana analiza danych. Człony wymiarów musz ą oczywiście pozostawać w powiązaniach z pomiarami. Je śli ograniczamy si ę do relacyjnej bazy danych, to powi ązania te realizowane są za pomocą powiązań referencyjnych mi ędzy kluczami głównymi tabel wymiarów, a kluczami obcymi tabeli faktów. Zale żności te mogą być bezpośrednie, jak w naszym przykładzie, który odpowiada tworzeniu struktur wielowymiarowych według tzw. *schematu gwiazdy*, lub pośrednie, gdzie powi ązania realizowane s ą z udziałem tabel po średnich, co odpowiada modelowaniu według tzw. *schematy płatka śniegu* [6, 16, 17]. Z rozważanego tutaj punktu widzenia nie jest istotne, w jaki sposób otrzymujemy zbiór członów poszczególnych wymiarów, wa żne jest jedynie to, że zbiór taki istnieje. Podział członów na rozłączne i hierarchicznie uporz ądkowane podzbiory odpowiadające poziomom i hierarchiom ma przede wszystkim za zadanie ułatwienie definiowania i operowania na danych wielowymiarowych w j ęzyku MDX.

Schematy kostek (ang. *cube schema*)

Dla rozważanego przykładu definicja schematu kostki dla potrzeb analizy sprzeda ży może mieć następującą postać:

```
AnalizaSprz =
   Sprzedaz(IdTow, IdKli, Data, Wartosc, Koszt),
   Przychód      : Sum(Wartosc),
   Koszt         : Sum(Koszt),
   Zysk          : Przychod - Koszt,
   DataOstSprz   : Max(Data),
   LiczbaTrans   : Count(IdKli),
   LiczbaKli     : CountDistinct(IdKli)},
   Region  = {[AllRegion],wielkopolskie,pomorskie,Gniezno,Leszno,Gdynia,Sopot},
   Marka   = {[All Marka], Fiat, Ford, Punto, Uno, Escord, Focus},
   Okres   = {[All Okres], 1998, 1999, 1998.KW1, 1998.KW2, 1999.KW1, 1999.KW2,
    1998.KW1.1, 1998.KW1.2, 1998.KW2.4, 1999.KW1.2, 1999.KW2.4, 1999.KW2.5}}).
```

4.2. Wystąpienia danych wielowymiarowych

Stan wielowymiarowej bazy danych składa si ę z danych wielowymiarowych tworzonych zgodnie ze zdefiniowanym schematem. Wówczas ka żda jednostka danych wielowymiarowych jest nazywana *wystąpieniem* tego schematu, podobnie jak ma to miejsce w tradycyjnych modelach danych. Zaczniemy od poj ęcia *punktu* w przestrzeni wymiarowej i poj ęcia *typu* takiego punktu.

Jeśli D jest zbiorem (nazw) wymiarów, to dowolny uporz ądkowany podzbiór D nazwiemy *typem punktu* w przestrzeni wielowymiarowej. Ci ąg członów $(e_1, ..., e_p)$ nazywamy punktem typu $T = (D_1, ..., D_p)$, jeśli e_i jest członem wymiaru D_i, dla każdego $i = 1, ..., n$. Na przykład:

- (Sopot, Fiat), (wielkopolskie, [All Marka]) s ą punktami typu (Region, Marka),
- ([All Okres], Escord, wielkopolskie) jest pun ktem typu (Okres, Marka, Region),
- każdy człon wymiaru Marka jest punktem typu Marka.

Komórka (ang. *cell*) jest to krotka oznaczaj ąca wartości wybranych pomiarów w pewnym punkcie przestrzeni wielowymiarowej. Na przykład komóka zawieraj ąca wartości pomiarów Koszt i LiczbaTrans w punkcie (Sopot, Fiat), ma postać przedstawion ą na rys. 4. Komórkę tę możemy zdefiniować za pomocą wyrażenia odwołującego się do żądanego schematu kostki:

		Koszt	LiczbaTrans
Sopot	Fiat	48,000.00	2

Rys. 4. Przykład pojedynczej komórki

```
AnalizaSprz([Koszt, LiczbaTrans], (Sopot, Fiat)),
```

Wartością tego wyrażenia jest krotka [Koszt: k, LiczbaTrans: l] (rys. 4), gdzie (patrz tab. 1):

$k = \text{SUM}\{t.\text{Koszt} \mid t \in \text{TB\_Koszt} \wedge t.\text{Sopot} = 1 \wedge t.\text{Fiat} = 1\}$,
$l = \text{COUNT}\{t \mid t \in \text{TB\_IdKli} \wedge t.\text{Sopot} = 1 \wedge t.\text{Fiat} = 1\}$.

Zbiór komórek (ang. *cellset*) określony jest przez wymienienie ciągu pomiarów (tak jak w definicji komórki) oraz zbioru punktów, przy czym wszystkie punkty muszą być tego samego typu. Na przykład

		Koszt	LiczbaTrans
Sopot	Fiat	48,000.00	2
	Ford	44,000.00	1
Gniezno	Escord	30,000.00	1

Rys. 5. Przykład zbioru komórek

```
AnalizaSprz([Koszt, LiczbaTrans], {(Sopot,fiat),(Sopot,ford),(Gniezno, Escord)}),
```

definiuje zbiór komórek przedstawiony na rys 5.

Kostką (ang. *cube*) nazywamy zbiór komórek wyznaczony dla wszystkich pomiarów i dla wszystkich możliwych punktów przestrzeni wielowymiarowej zdefiniowanej w schemacie kostki.

Obliczmy rozmiar kostki oschemacie `AnalizaSprz`. Liczba możliwych punktów jest równa liczbie elementów w iloczynie kartezjańskim zbiorów członów wymiarów Region, Marka i Okres, $7 * 7 * 19 = 931$. Każda komórka zawiera pięć wartości, cała kostka zawiera więc $931 * 5 = 4655$ jednostek danych. Dodatkowo, każda z 931 pozycji opisana jest trzema współrzędnymi punktu, co daje $931 * 3 = 2793$ jednostek danych. Rozważana kostka obejmuje więc $4655 + 2793 = 7448$ jednostek danych. Jest to więc wielokrotnie więcej niż zawierała cała źródłowa baza danych – stąd efektywne pamiętanie i przetwarzanie kostek jest problemem niezwykle istotnym.

5. Wprowadzenie do MDX

Język MDX (ang. ***M**ulti**d**imensional E**x**pressions*) jest językiem służącym do definiowania danych wielowymiarowych, do operowania na tych danych oraz do określania sposobu ich prezentacji [9, 10, 14, 15]. Zapytanie w MDX, podobnie jak w SQL, zawiera frazę SELECT (definiującą dane wynikowe), frazę FROM (definiującą kostki wejściowe) oraz frazę WHERE (określającą filtrowanie danych). Dostępnych jest także wiele funkcji służących do wyszukiwania danych i do operowania danymi. Możliwe jest rozszerzanie funkcjonalności języka poprzez definiowania własnych funkcji. Fraza WITH służy do zdefiniowania pomiarów obliczanych oraz do przypisania nazwy zbiorom wymiarów (umożliwia to odwoływanie się do zbioru przez nazwę zamiast wymieniania jego elementów).

Wyrażenie MDX ma następującą składnię:

```
[WITH   <specifikacja_formuły> [, <specifikacja_formuły> ...]]
SELECT  [<specifikacja_osi> [, <specifikacja_osi>...]]
FROM    [<specifikacja_kostki>]
[WHERE  [<specifikacja_plastra>]]
```

W MDX przyjmuje się, że zbiór pomiarów tworzy wyróżniony wymiar o nazwie Measures, stąd też wszystkie dalsze rozważania dotyczące wymiarów odnoszą się również do wymiaru Measures.

5.1. Fraza SELECT

Fraza `SELECT` określa wynikowy zbiór komórek oraz sposób jego prezentacji. Prezentacja zbioru określona jest poprzez przyporządkowanie *osiom* (ang. *axis*) punktów przestrzeni wielowymiarowej. Istnieje 128 osi, którym przypisane są numery, a ponadto pięć pierwszych osi ma przypisane nazwy: 0 – oś x (COLUMNS), 1 – oś y (ROWS), 2 – oś z (PAGES), 3 – SECTIONS, 4 – CHAPTERS. W każdym przypadku oś może być identyfikowana za pomocą wyrażenia AXIS(*indeks*), gdzie *indeks* jest liczbą całkowitą z przedziału [0, 127]. Zbiory punktów mogą być podawane jawnie lub specyfikowane z wykorzystaniem szeregu funkcji.

Specyfikacja osi ma następującą składnię:

```
<specifikacja_osi> ::= <zbiór_punktów> ON <nazwa_osi>
<nazwa_osi> ::= COLUMNS | ROWS | PAGES | SECTIONS | CHAPTERS | AXIS(<indeks>)
```

		- Producent	Marka			
		All Marka	- Fiat			+ Ford
- Wojewodztwo	Miasto	All Marka Total	Fiat Total	Punto	Uno	Ford Total
All Region	All Region Total	289,000.00	91,000.00	53,000.00	38,000.00	198,000.00
- pomorskie	pomorskie Total	171,000.00	48,000.00	28,000.00	20,000.00	123,000.00
	Gdynia	79,000.00				79,000.00
	Sopot	92,000.00	48,000.00	28,000.00	20,000.00	44,000.00
- wielkopolskie	wielkopolskie Total	118,000.00	43,000.00	25,000.00	18,000.00	75,000.00
	Gniezno	30,000.00				30,000.00
	Leszno	88,000.00	43,000.00	25,000.00	18,000.00	45,000.00

Rys. 6.

Osie zbioru komórek z rys. 6 mają następującą specyfikację:

```
SELECT  {[All Marka], Fiat, Punto, Uno, Ford} ON COLUMNS,
   {[All Region],pomorskie,Gdynia,Sopot,wielkopolskie,Gniezno,Leszno} on ROWS
```

Do zdefiniowania zbioru punktów można również wykorzystać funkcję `Members`, która zwraca wszystkie człony z wymiaru, hierarchii lub poziomu, do których jest stosowana:

```
SELECT  {[All Marka], Fiat, Punto, Uno, Ford} ON COLUMNS,
          Region.Members on ROWS
```

		+ Wojewodztwo		
+ Producent	+ Rok	All Region	+ pomorskie	+ wielkopolskie
All Marka	All Okres	9	5	4
	+ 1998	5	2	3
	+ 1999	4	3	1
+ Fiat	All Okres	4	2	2
	+ 1998	3	1	2
	+ 1999	1	1	
+ Ford	All Okres	5	3	2
	+ 1998	2	1	1
	+ 1999	3	2	1

Rys. 7.

W przypadku zbioru komórek na rys. 7, mamy:

- oś x – nagłówki kolumn tworzą punkty `[All Region], pomorskie, wielkopolskie`,
- oś y – nagłówki wierszy tworzone są z punktów dwuelementowych należących do produktu kartezjańskiego zbiorów `{[All Marka],Fiat,Ford}` i `{[All Okres],[1998],[1999]}`.

Do specyfikacji tych osi można wykorzystać funkcję `Crossjoin(X,Y)`, która tworzy iloczyn kartezjański zbiorów *X* i *Y*:

```
SELECT  {[All Region], pomorskie, wielkopolskie} ON COLUMNS,
   Crossjoin({[All Marka],Fiat,Ford},{[All Okres],[1998],[1999]}) on ROWS
```

Wariant tej specyfikacji z zastosowaniem funkcji `Members`:

```
SELECT  {[All Region], Region.Wojewodztwo.Members} ON COLUMNS,
   Crossjoin({[All Marka],Fiat,Ford},{[All Okres],Okres.Rok.Members}) on ROWS
```

Kolejną funkcją na członach wymiarów jest funkcja `Children`, która zwraca zbiór członów będących dziećmi członu, do którego jest stosowana, np.: `Fiat.Children = {Punto, Uno}`.

Wówczas specyfikacja

```
SELECT  {[All Region], Region.Wojewodztwo.Members} ON COLUMNS,
         Crossjoin(Fiat.Children, [1998].Children) on ROWS
```

oznacza, że osi y będą przyporządkowywane punkty utworzone z iloczynu kartezjańskiego zbiorów: `Fiat.Children = {Punto, Uno}` oraz `[1998].Children = {KW1, KW2}`.

Specyfikacja osi obejmująca wszystkie pomiary i wymiary występujące w kostce `AnalizaSprz` może wyglądać następująco:

```
SELECT {Przychod, Koszt, Zysk, DataOstSprz, LiczbaTrans, LiczbaKli} ON COLUMNS,
   Crossjoin(Crossjoin(Region.Members, Okres.Members), Marka.Members) ON ROWS
```

Zauważmy, że kostkę trójwymiarową jesteśmy w stanie reprezentować z wykorzystaniem dwóch osi (a więc na płaszczyźnie).

5.2. Fraza FROM

Fraza FROM specyfikuje źródłowe zbiory komórek, na których wykonywane są operacje. W obecnej wersji języka MDX możliwe jest jedynie podawanie nazwy jednej istniejącej kostki:

```
<specifikacja_kostki> ::= <nazwa_kostki>
```

5.3. Fraza WHERE

We frazie WHERE określane jest wyrażenie będące swego rodzaju filtrem ograniczającym zwracany zbiór komórek do pewnego *plastra* (ang. *slicer*). Domyślnie przyjmuje się, że każdy wymiar, który nie został przyporządkowany do żadnej osi reprezentowany jest we frazie WHERE poprzez swój człon `[All wymiar]`. Jawne określenie innego członu powoduje, że tylko komórki budowane z wykorzystaniem tego członu należą do wynikowego zbioru komórek.

Składnia frazy WHERE jest następująca:

```
[WHERE  [<specifikacja_plastra>]]
<specifikacja_plastra> ::= <punkt>
```

Na przykład

```
WHERE ([All Okres], Marka.Fiat, Koszt)
```

ograniczy zbiór komórek do tych, które dotyczyły fiatów i dowolnego okresu. Umieszczenie pomiaru `Koszt` we frazie WHERE oznacza, że tylko ten pomiar jest dla nas interesujący.

Następujące wyrażenie zwraca zbiór komórek z odpowiednio zagregowanymi wartościami pomiaru `LiczbaKli` odnoszącymi się do roku 1998:

```
SELECT {[All Marka], Fiat, Fiat.Children, Ford} ON COLUMNS,
        Region.Members ON ROWS
FROM AnalizaSprz
WHERE ([1998], LiczbaKli)
```

Wynik zapytania przedstawiono na rys. 8:

	All Marka	Fiat	Punto	Uno	Ford
All Region	5	3	1	2	2
pomorskie	2	1		1	1
Gdynia					
Sopot	2	1		1	1
wielkopolski	3	2	1	1	1
Gniezno	1				1
Leszno	2	2	1	1	

Rys. 8.

5.4. Fraza WITH

We frazie WITH można definiować pomiary obliczane oraz przypisywać nazwy zbiorom. Składnia tej frazy jest następująca:

```
WITH <specifikacja_formuły> [, <specifikacja_formuły> ...]
<specifikacja_formuły> ::= <specyfikacja_członu> | <specyfikacja_zbioru>
<specyfikacja_członu> ::= MEMBER <rodzic_członu>.<nazwa_członu>AS'<wyrażenie>'
<specyfikacja_zbioru> ::= SET <nazwa_zbioru> AS '<zbiór>'
```

Poniższe wyrażenie definiuje pomiar obliczany `ZyskProc` i włącza go do zbioru wynikowego:

```
WITH MEMBER Measures.ZyskProc AS '(Przychod - Koszt)/Koszt * 100'
SELECT  {Przychod, Koszt, ZyskProc} ON COLUMNS,
         Region.Members ON ROWS
FROM AnalizaSprz
```

Podsumowanie przychodu i kosztów w Gnieźnie i w Sopocie uzyskujemy tworząc wyrażenie:

```
WITH MEMBER [All Region].Gniezno_I_Sopot AS 'Sum({Gniezno, Sopot})'
SELECT  {Przychod, Koszt} ON COLUMNS,
        {Region.Members, Gniezno_I_Sopot} ON ROWS
FROM AnalizaSprz
```

Natomiast w następującym wyrażeniu należy użyć funkcji `Aggregate` aby agregowanie poszczególnych pomiarów było realizowane zgodnie z ich funkcjami agregującymi:

```
WITH MEMBER [All Region].Gniezno_I_Sopot AS 'Aggregate({Gniezno, Sopot})'
SELECT  {Przychod, DataOstSprz, LiczbaTrans} ON COLUMNS,
        {Region.Members, Gniezno_I_Sopot} ON ROWS
FROM AnalizaSprz
```

Zastosowanie frazy WITH do nadawania nazwy zbiorom ilustruje poniższy przykład:

```
WITH SET Miasta AS '{Gniezno, Sopot, Leszno}'
SELECT {Przychod, DataOstSprz, LiczbaTrans} ON COLUMNS,
        Miasta ON ROWS
FROM AnalizaSprz
```

5.5. Funkcje na zbiorach komórek

Następujące zapytanie zwraca zbiór komórek uporządkowany według wartości przychodu. Uporządkowanie odbywa się rosnąco oddzielnie w obrębie każdego członu wymiaru `Marka`. Sterowane jest to funkcją `Order(`*`zbiór`*`, `*`pomiar`*`, DESC)`, a do zbioru wynikowego należą tylko komórki niepuste, co jest efektem użycia operatora `NON EMPTY`:

```
SELECT {Przychod} ON COLUMNS,
        NON EMPTY Order(Crossjoin({Fiat.Children, Ford.Children},
                        Okres.Rok.Members), Przychod, DESC) on ROWS
FROM AnalizaSprz
```

Aby uporządkowanie było globalne, a nie ograniczone do poszczególnych członów wymiaru `Marka`, należy w funkcji `Order` użyć parametru `BDESC` zamiast `DESC`.

Następujące wyrażenie wypisuje 5 pierwszych komórek o najwyższej wartości pomiaru obliczanego `ZyskProc`. Sterowane jest to funkcją `TopCount(`*`zbiór`*`, `*`liczba`*`, `*`pomiar`*`)`:

```
WITH MEMBER Measures.ZyskProc AS '(Przychod - Koszt)/Koszt * 100'
SELECT {Przychod, ZyskProc} ON COLUMNS,
        TopCount(Crossjoin({Fiat.Children, Ford.Children},
                 Okres.Rok.Members), 5, ZyskProc) on ROWS
FROM AnalizaSprz
```

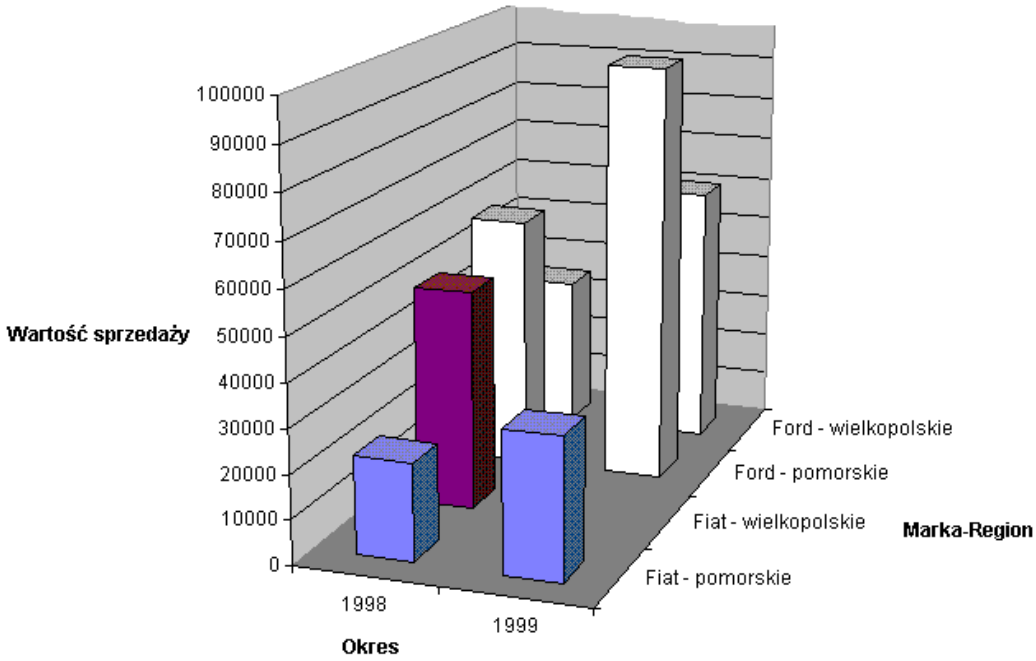
5.6. Wizualizacja graficzna

Wizualizacja graficzna zbioru komórek z kostek wielowymiarowych jest mo żliwa dzięki zintegrowaniu OLAP Services z Excelem 2000. Na rys. 9 pokazano tabelaryczn ą postać zbioru komórek prezentowaną w Excelu 2000,

Przychod	Producent	Wojewodztwo					
	Fiat		Fiat - Suma *	Ford		Ford - Suma *	Suma całkowita *
Rok	pomorskie	wielkopolskie		pomorskie	wielkopolskie		
1998	22000	50000	72000	58000	35000	93000	165000
1999	32000		32000	96000	60000	156000	188000
Suma całkowita *	54000	50000	104000	154000	95000	249000	353000

Rys. 9. Tabelaryczne przedstawienie zbioru komórek w systemie Excel 2000

a na rys. 10 jego graficzn ą prezentację w postaci wykresu słupkowego:



Rys. 10. Przedstawienie zbioru komórek z rys. 9 w postaci wykresu słupkowego

6. Podsumowanie i wnioski

W pracy omówiono problem tworzenia i przetwarzania danych wielowymiarowych w systemach OLAP. Opisano mo żliwości języka SQL z punktu widzenia środków wspomagaj ących funkcje OLAP – istniej ące w niektórych komercyjnych systemach baz danych oraz proponowane w nowym standardzie SQL:1999. Zaproponowano sformalizowan ą próbę interpretacji danych wielowymiar owych za pomoc ą modelu MDW (Model Danych Wielowymiarowych). Model ten w sposób prec yzyjny definiuj ą takie pojęcia jak: tabela faktów, pomiary i ich agregowanie, wymiary i człony w ymiarów, schematy i wyst ąpienia kostek, punkty, komórki i zbiory komórek. Nast ępnie omówiono podstawy j ęzyk MDX (MultiDimensional Expressions), który został zaimplementowany w systemie MS SQL Server OLAP Services. Zamieszczono wiele przykładów ilustruj ących tworzenie i przetwarzanie danych wielowymiarowych. J ęzyk MDX dost ępny jest w różnych produktach Microsoftu poprzez OLE DB i/lub ADO i mo że skutecznie wspomagać zintegrowan ą analizę danych gromadzonych w bazach i hurtowniach danych, dzi ęki korzystaniu z takich narz ędzi jak MS SQL Server, OLAP Services i Excel 2000.

MDX jest pierwszą propozycją języka zorientowanego na dane wielowymiarowe. Można przypuszczać, że koncepcja tego typu języków będzie rozwijana i wydaje się, że należy oczekiwać dalszych interesujących propozycji w tym zakresie.

Bibliografia

1. R. Agrawal, A. Gupta, S. Sarawagi, Modeling Multidimensional Databases, Proc. 13th ICDE 1997, s. 232-243.
2. E.F. Codd, S.B. Codd, C.T. Salley, Providing OLAP (Online Analytical Processing) to User-Analysts: An IT Mandate, Arbor Software Corp, 1993.
3. C.J. Date, An Introduction to Database Systems, Seventh Edition, Addison Wesley Longman, Reading Massachusetts, 2000.
4. A. Eisenberg, J. Melton, SQL Standardization: The Next Steps, ACM SIGMOD Record 29(1), 2000.
5. U.M. Fayyad i.in. Advances in Knowledge Discovery and Data Mining, AAAI Press/The MIT Press, Menlo Park, 1996.
6. B. Huesemann, J. Lechtenboerger, G. Vossen, Conceptual Data Warehouse Design, Proc. International Workshop on Design and Management of Data Warehouses DMDW'2000, Stokholm, Sweden, June 5-6 2000.
7. ISO/IEC 9075-1:1999. On-Line Analitical Processing (SQL/OLAP). Amendment 1, November 1999 ftp://jerry.ece.umassd.edu pub/SC32/WG3/Progression_Documents/FPDAM/ fpdam-olap-1999-11.pdf
8. T. Morzy, Eksploracja danych: problemy i rozwiązania, V Konferencja PLOUG, Zakopane, 1999.
9. MS SQL Server 2000 Analysis Services, Book Online.
10. Microsoft Corp. OLE DB for OLAP, February 1998,http://www.microsoft.com/data/oledb/olap/
11. OLAP Council. OLAP and OLAP Server Definitions, 1997. http://www.olapcouncil.org/research/ glossaryly.htm
12. Oracle8*i* Application Developer's Guide - Fundamentals. Release 8.1.5, Oracle Corporation, 1999.
13. T. Pankowski, Podstawy baz danych, Wydawnictwo Naukowe PWN, Warszawa, 1992.
14. J. Sturm, Hurtownie danych. Microsoft SQL Server 7.0. Przewodnik Techniczny, Microsoft Press/APN PROMISE, Warszawa, 2000.
15. D. Sussman, ADO 2.1. Programmer's Reference, Wrox Press Ltd., 1999.
16. P. Vassiliadis, T. Sellis, A Survey of Logical Models for OLAP Databases, ACM SIGMOD Record 28(4) 1999.
17. R. Wrembel, Z. Królikowski, M. Morzy, Magazyny danych – stan obecny i kierunki rozwoju, ProDialog 10 Wydawnictwo NAKOM, Poznań, 2000, s. 75-93.
18. S. Youness, Professional Data Warehousing with SQL Server 7.0 and OLAP Services, Wrox Press Arden House, 2000.