

**Wielowymiarowy model danych:  
Oracle OLAP, Analytic Workspaces,  
OLAP DML**

# Plan rozdziału

- Architektura Oracle OLAP
- Porównanie architektury wielowymiarowej i relacyjnej
- Dostęp do danych OLAP
- Analityczne przestrzenie robocze
- OLAP DML

# Integracja relacyjno-wielowymiarowa

- Oracle 9i/10g jest przykładem relacyjno-wielowymiarowej bazy danych (RDBMS-MDDS). Zalety integracji modelu wielowymiarowego z relacyjną bazą danych:
  - uproszczenie zarządzania
  - efektywność
  - bezpieczeństwo
  - pojedyncze źródło danych
  - łatwość dostępu
  - uproszczenie ładowania
- Oracle OLAP jest następcą produktu Express Server (możliwość importu starych plików \*.eif)
- Język Express SPL jest w 99% kompatybilny z językiem OLAP DML (konieczne drobne modyfikacje)

# Architektura Oracle OLAP

- Technologie
  - relacyjna: przechowywanie danych, interfejs SQL
  - obiektowa: mapowanie między modelem relacyjnym i wielowymiarowym
  - wielowymiarowa: przechowywanie danych, efektywne przetwarzanie analityczne
- Interfejsy
  - Oracle Call Interface
  - JDBC
  - OLAP API
- Metadane
  - OLAP Catalog

# Architektura wielowymiarowa

- Obraz logiczny

- miary
- wymiary
- hierarchie
- poziomy
- atrybuty

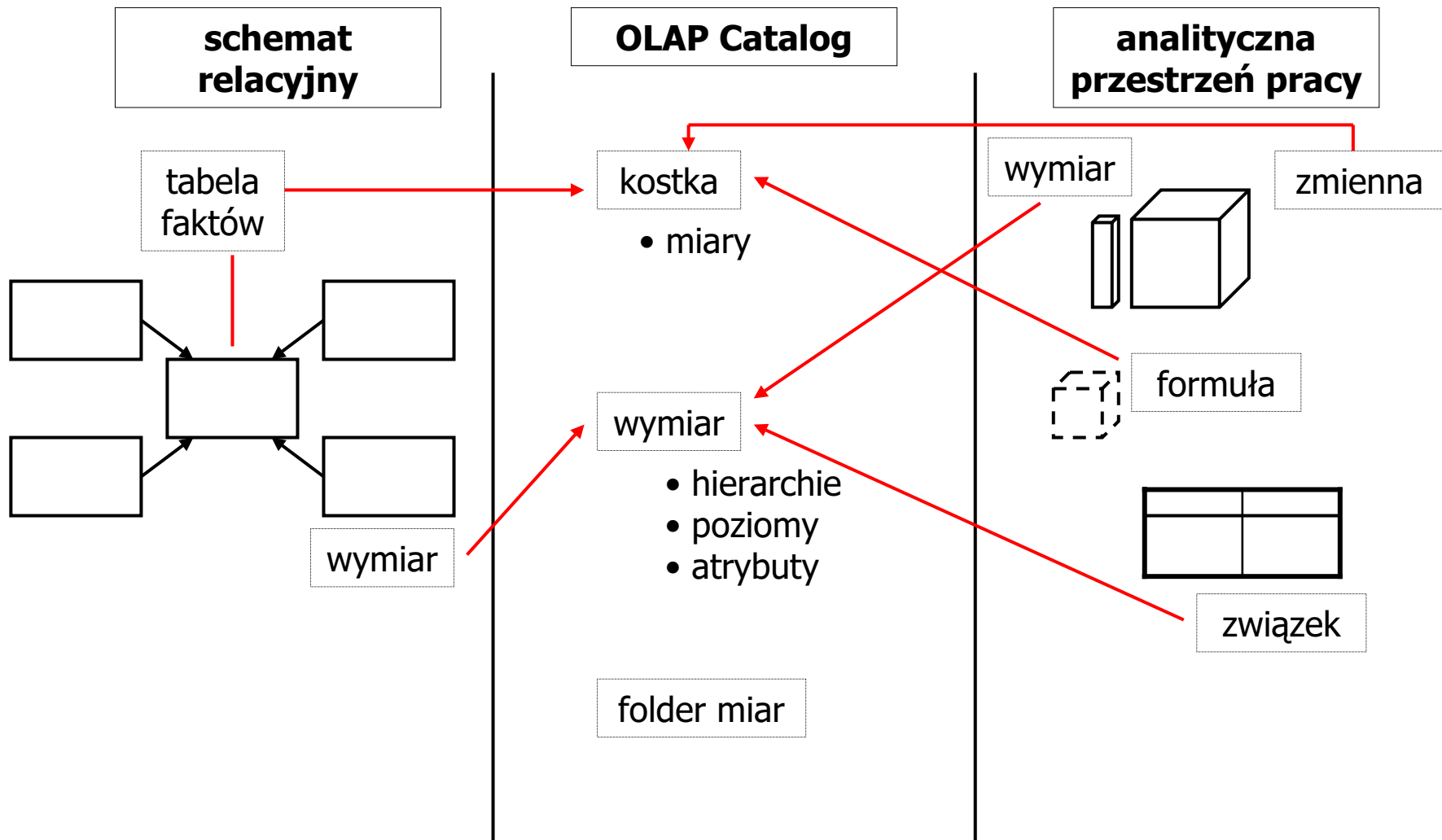
- Obraz fizyczny

- wymiary
- relacje
- zmienne
- formuły
- procedury składowane

- Elementy Oracle OLAP

- silnik obliczeń i agregacji: wykonywanie kalkulacji wewnątrz jądra BD
- analityczne przestrzenie pracy: magazyny obiektów wielowymiarowych
- OLAP Catalog: słownik wielowymiarowej bazy danych
- OLAP API: biblioteki Java
- OLAP DML: język definiowania i manipulowania danymi wielowym.
- funkcje tablicowe PL/SQL: mapowanie danych wielowym. do relacyjnych

# Model metadanych OLAP



# Narzędzia Oracle OLAP

- Narzędzia graficzne
  - Oracle Enterprise Manager
  - Oracle Warehouse Builder
  - Analytic Workspace Manager
- Interfejsy programistyczne
  - CMW1 i CMW2
  - pakiet DBMS\_AWM
- Metadane
  - OLAP Catalog

# Tworzenie obiektów OLAP

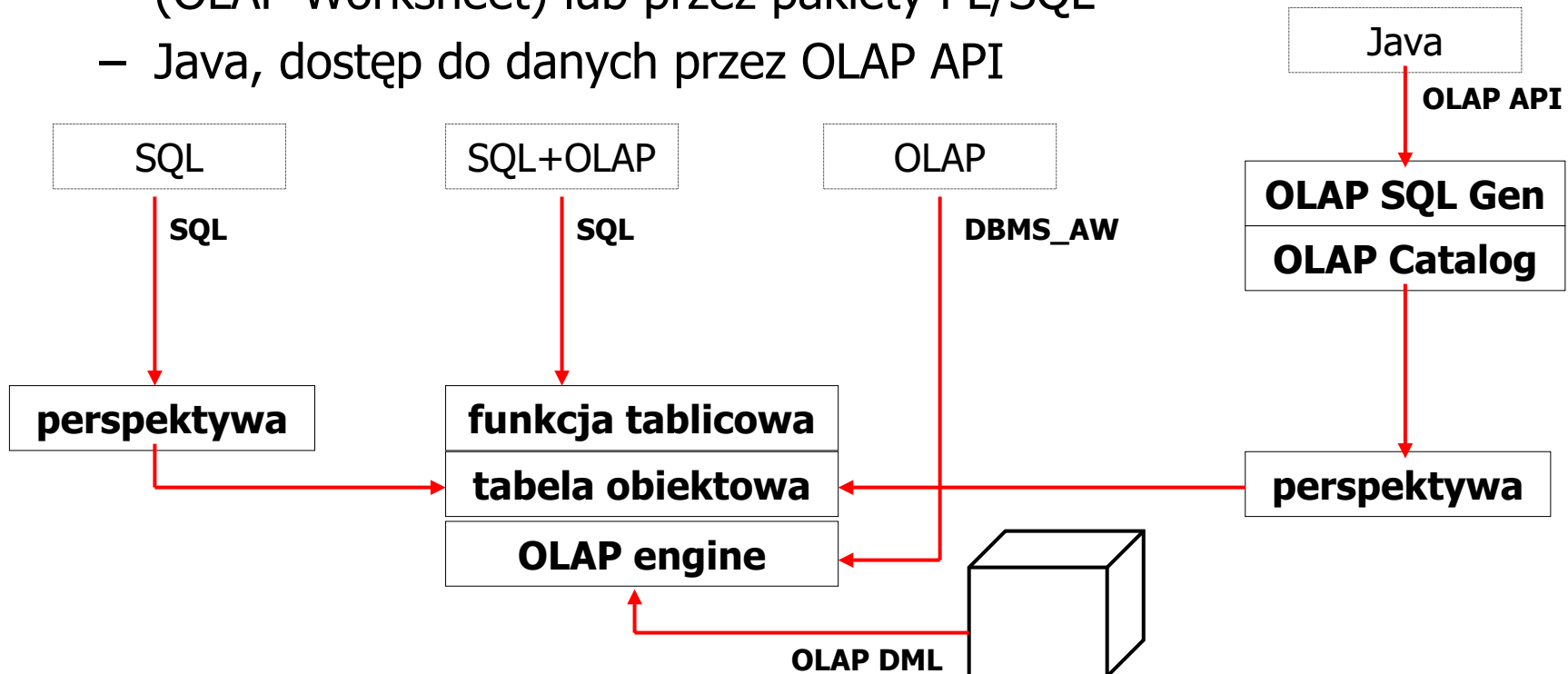
- **Możliwe scenariusze**
  - OWB tworzy wszystkie obiekty
  - OWB tworzy relacyjny schemat gwiazdy, AWM tworzy obiekty wielowymiarowe i wypełnia je danymi
  - OEM tworzy relacyjny schemat gwiazdy, AWM tworzy obiekty wielowymiarowe i wypełnia je danymi
  - CWM2 i DBMS\_AWM tworzą obiekty wielowymiarowe
- **Przechowywanie danych**
  - tabele relacyjne: tabele faktów i wymiary
  - analityczne przestrzenie pracy: agregowane dane, kostki danych, hierarchie wymiarów

# Wybór modelu

- Całość danych w schemacie relacyjnym
  - operacje na danych za pomocą SQL, skalowalność i bezpieczeństwo, brak redundancji, konieczność materializacji agregatów
- Całość danych w strukturach wielowymiarowych
  - operacje na danych za pomocą OLAP DML, wysoka efektywność zapytań analitycznych, wsparcie modelowania, predykcji, obliczeń statystycznych
- Model hybrydowy
  - część danych (najczęściej bardziej szczegółowa) przechowywana w modelu relacyjnym, dane zagregowane w modelu wielowymiarowym, konieczność uspoźniania danych

# Dostęp do danych OLAP

- Możliwe scenariusze:
  - SQL, odczyt danych relacyjnych
  - SQL, odczyt danych wielowymiarowych za pomocą funkcji tablicowych i tabel obiektowych
  - OLAP DML, wykonywanie poleceń przez specjalizowane narzędzie (OLAP Worksheet) lub przez pakiety PL/SQL
  - Java, dostęp do danych przez OLAP API



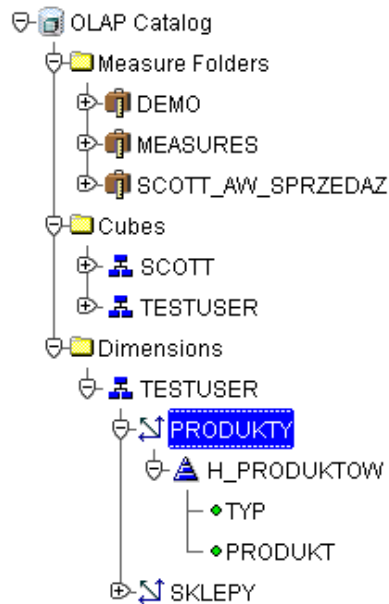
# OLAP Catalog

- Mapowanie struktur logicznych schematu na struktury fizyczne (zarówno relacyjne jak i wielowymiarowe)
- Dwa repozytoria
  - CWM: Oracle 9i R1
  - CWM2: Oracle 9i R2 / 10g (rozszerzenie wcześniejszego API)
- Zawartość repozytorium
  - tabele zawierające metadane modelu
  - API do odczytu i zapisu (pakiety PL/SQL)
  - preprocesory (tylko CWM2)
- Narzędzia
  - Oracle Warehouse Builder
  - Oracle Enterprise Manager
  - CWM2 API (`CWM2_OLAP_CUBE`, `CWM2_OLAP_DIMENSION`, `CWM2_OLAP_LEVEL`, `CWM2_OLAP_MEASURE`, ...)

# Analityczna przestrzeń robocza

- Analityczna przestrzeń robocza (ang. analytical workspace, AW) to obiekt bazy danych typu LOB (ang. Large Object) przechowujący obiekty modelu wielowymiarowego
- AW optymalizują przetwarzanie analityczne na obiektach wielowymiarowych i ułatwiają zarządzanie agregowanymi danymi
- Dane przechowywane w AW mogą być udostępniane aplikacjom relacyjnym za pomocą perspektyw
- AW mogą być tworzone:
  - za pomocą Oracle Warehouse Builder
  - za pomocą Analytic Workspace Manager
  - programistycznie (`CWM2_OLAP_AW_CREATE`)

# Tworzenie i edycja AW



Analytic Workspace Manager - Connected as system to Database miner.cs.put.poznan.pl:1521:miner10g

File View Tools Help

Basic Program Properties

```

variable _detach boolean
variable _aw text
variable _aws text
variable _pos integer
variable _after_aw text
variable _sch text
variable _i integer

trap on CLEANUP noprint

if aw(attached 'sys.awxml') ne YES
then do
  aw attach sys.awxml ro last
  _detach = YES
doend

aw = aw(fullname 'this_aw')
if isvalue(__xml_attached_aws_aws) eq YES
then mnt __xml_attached_aws dlt _aw

aws = aw(list)
sch = sysinfo(user)
i=1
while _i le numlines(_aws)
do
  if findchars(extlines(_aws _i 1) '.') eq 0 and extlines(_aws _i 1) ne

```

Compile Apply Revert

miner.cs.put.poznan.pl:1521:miner10g.

Create Analytic Workspace Wizard: Processing status

Processing Defining Load for Cube: AW\_SPRZEDAZ

83%

Creating analytic workspace...

Preparing

Creating analytic workspace...

Processing cube AW\_SPRZEDAZ

Processing Creating Dimension : AW\_PRODUKTY

Processing Creating Dimension : AW\_SKLEPY

Processing Defining Load for Dimension: AW\_PRODUKTY

Processing Refreshing Dimension : AW\_PRODUKTY

Processing Defining Load for Dimension: AW\_SKLEPY

Processing Refreshing Dimension : AW\_SKLEPY

Processing Creating Cube : AW\_SPRZEDAZ

Processing Defining Load for Cube: AW\_SPRZEDAZ

Pomoc Wstecz Finish Close Abort

# Dostęp do AW z SQL

- AW musi zostać udostępniona (!)
- Podstawowym narzędziem dostępu do AW z poziomu SQL jest pakiet DBMS\_AW
- Standardowa funkcja tablicowa OLAP\_TABLE umożliwia odczyt zawartości AW w formie relacyjnej

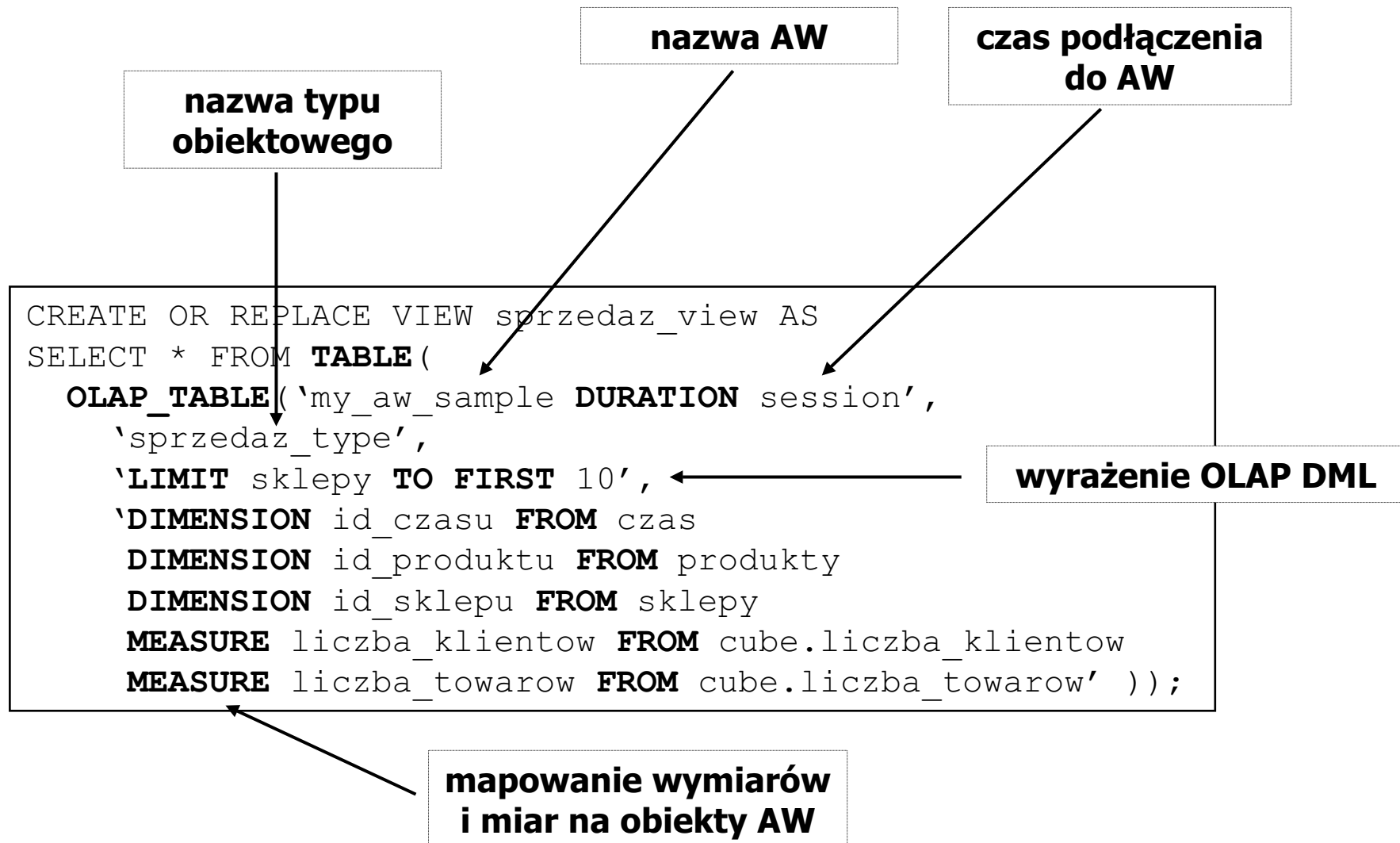
```
SET SERVEROUTPUT ON
EXECUTE DBMS_AW.EXECUTE('AW ATTACH my_aw_sample RW
    DEFINE podatek VARIABLE <sklepy produkty czas>');
EXECUTE DBMS_AW.PRINTLOG(DBMS_AW.INTERPCLOB('AW CREATE my_aw_test
    IMPORT ALL FROM EIF FILE ''export/database.eif'' DATA DFNS;
    DESCRIBE'));
```

# OLAP\_TABLE (1)

- Funkcja tablicowa OLAP\_TABLE odczytuje dane wielowymiarowe z AW i prezentuje je w relacyjnej formie. Funkcja może być parametryzowana
- Sposób użycia
  - w definicji perspektywy
  - w zapytaniu

```
CREATE OR REPLACE TYPE sprzedaz_type AS OBJECT (  
    id_czasu NUMBER, id_produkту NUMBER, id_sklepu NUMBER,  
    liczba_klientow NUMBER, liczba_towarow NUMBER );  
/  
  
CREATE TABLE sprzedaz_table OF sprzedaz_type;
```

# OLAP\_TABLE (2)



# Administracja Oracle OLAP

- Zadania
  - konfiguracja bazy danych
  - bezpieczeństwo
  - pielęgnacja danych
  - optymalizacja
- O czym pamiętać?
  - parametry inicjalizacyjne bazy danych (`utl_file_dir`, `OLAP_PAGE_POOL_SIZE`, `QUERY_REWRITE_ENABLED`, `STAR_TRANSFORMATION_ENABLED`, ...)
  - parametry inicjalizacyjne OLAP API (`ALTER SESSION ...`)
  - użytkownik `OLAPSYS` (właściciel OLAP Catalog), rola `OLAP_DBA`
  - perspektywy wydajnościowe (`v$aw_calc`, `v$aw_olap`, ...)

# OLAP DML (1)

- Język dostępu i modyfikacji dla danych wielowymiarowych
- Kategorie poleceń
  - agregacja
  - alokacja
  - wybór danych
  - pobieranie danych z tabel relacyjnych
  - przewidywanie i regresja
  - operacje matematyczne i statystyczne
  - ranking
  - kontrola przepływu programu
  - obsługa wyjątków

# OLAP Worksheet Tool

The screenshot shows the Oracle OLAP Worksheet tool interface. The main window displays a list of surrogate keys and a list of commands. A help window is open, showing an alphabetical list of commands. Red arrows point from labels to specific parts of the interface.

**menu**: Points to the menu bar (File, Edit, Worksheet, Options, Help).

**przybornik**: Points to the toolbar on the left side of the main window.

**wyniki**: Points to the main text area displaying the list of surrogate keys and commands.

**polecenia**: Points to the help window displaying the alphabetical list of commands.

The main window displays the following surrogate keys:

```

WH_SPRZEDAZ_DA_PRT_RUNAGGMAP
WH_SPRZEDAZ_DA_PRT_TOPAGGMAP

11 SURROGATES

-----
WH_PRODUKTY_HIERLIST_SURR
WH_PRODUKTY_LEVLLIST_SURR
WH_PRODUKTY_PRODUKT_SURR
WH_PRODUKTY_TYP_SURR
WH_SKLEPY_HIERLIST_SURR
WH_SKLEPY_LEVLLIST_SURR
WH_SKLEPY_MIASTO_SURR
WH_SKLEPY_REGION_SURR
WH_SKLEPY_SKLEP_SURR
XML_GENERATED_1
XML_GENERATED_2
  
```

The help window displays the following alphabetical list of commands:

A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

The help window also displays the following list of commands:

- [= Command](#)
- [ABS Function](#)
- [ACROSS Command](#)
- [ADD\\_MONTHS Function](#)
- [AGGINDEX Command](#)
- [AGGMAP Command](#)
- [AGGMAPINFO Function](#)
- [AGGREGATE Command](#)
- [AGGREGATE Function](#)
- [AGGREGATION Function](#)
- [ALLCOMPILE Program](#)
- [ALLOCATE Command](#)
- [ALLOCERRLOGFORMAT Option](#)
- [ALLOCERRLOGHEADER Option](#)
- [ALLOCERRLOGMAP Command](#)

# Polecenia AW

## AW LIST

```
SCOTT.WH R/W UNCHANGED SCOTT.WH
EXPRESS R/O UNCHANGED
SYS.EXPRESS
(1 other user reading)
```

```
AW ATTACH SCOTT.WH RW
AW DETACH SCOTT.WH
```

```
UPDATE
COMMIT
```

```
AW CREATE NEW_AW
AW DELETE NEW_AW
```

```
REPORT AW (ATTACHED)
REPORT AW (CHANGED)
REPORT AW (DATE)
REPORT AW (FULLNAME)
REPORT AW (LISTNAMES)
REPORT AW (TIME)
```

```
->REPORT AW (ATTACHED)
AW (ATTACHED)
-----
yes
->REPORT AW (CHANGED)
AW (CHANGED)
-----
no
->REPORT AW (DATE)
AW (DATE)
-----
13FEB05
->REPORT AW (FULLNAME)
AW (FULLNAME)
-----
SCOTT.WH
```

# Dostępne typy danych

Data Type	Keyword	Description
Numeric	INTEGER	Whole number, range is: $(-2^{31}) - 1$ to $(2^{31}) - 1$
Numeric	SHORTINTEGER	Whole number, range is: $(-2^{15}) - 1$ to $(2^{15}) - 1$
Numeric	LONGINTEGER	Whole number, range is: $(-2^{63}) - 1$ to $(2^{63}) - 1$
Numeric	DECIMAL	A decimal with up to 15 significant digits
Numeric	SHORTDECIMAL	A decimal with up to 7 significant digits
Numeric	NUMBER	A decimal with up to 38 significant digits
Text	TEXT	Up to 4,000 bytes per line in the UTF-8 character encoding
Text	NTEXT	Up to 4,000 bytes per line in the database character set
Text	ID	Up to 8 characters per line
Date/Time	DATE	Between January 1, 4712 B.C. and December 31, 9999 A.D.
Boolean	BOOLEAN	Yes, True, On, No, False, Off

# Polecenia dotyczące wymiarów

- Rodzaje wymiarów
  - płaskie
  - hierarchiczne
  - złożone
  - skonkatenowane

```

MAINTAIN sklep ADD 'Sklep 1'
MAINTAIN sklep ADD 'Sklep 2'
MAINTAIN sklep ADD 'Sklep 4'
MAINTAIN sklep RENAME 'Sklep 4' 'Sklep 3'
REPORT sklep

```

```

DEFINE sklep DIMENSION TEXT
DEFINE rok DIMENSION INTEGER
LISTNAMES DIMENSION

```

```

OLAP_SYS_VIEWDIM
ROK
SKLEP
TIME_GLEVEL_DIMENSION

```

```

->REPORT sklep
SKLEP
-----
Sklep 1
Sklep 2
Sklep 3

```

# Tworzenie zmiennych

- Rodzaje zmiennych
  - skalarne
  - wielowymiarowe
  - tymczasowe
  - trwałe

```
DEFINE zmienna_tymczasowa VARIABLE NUMBER TEMP  
DEFINE liczba_klientow VARIABLE INTEGER <produkty sklepy>
```

kolejność wymiarów: od najbardziej zmiennego do najmniej zmiennego

# Przypisywanie wartości

- Metody przypisania wartości
  - odczyt z tabeli relacyjnej za pomocą komendy **SQL**
  - odczyt z pliku zewnętrznego za pomocą komendy **FILEREAD**
  - użycie operatora przypisania =

```
liczba_klientow = RANDOM(0,1000)
```

```

-----LICZBA_KLIENTOW-----
-----SKLEPY-----
          MIASTO_Atlanta MIASTO_Boston MIASTO_Chicago
PRODUKTY
-----
PRODUKT_1          533          741          659
PRODUKT_2          382          611          60
PRODUKT_3          849          849          211
PRODUKT_4          986          651          52

```

```
liczba_klientow(produkty 'PRODUKT_2' sklepy 'MIASTO_Boston') = 999
```



# Wczytywanie danych z tabel relacyjnych

```
ALLSTAT
```

```
SQL DECLARE my_cursor CURSOR FOR  
  SELECT id_sklepu, id_produktu, id_czasu, liczba_klientow  
  FROM sklepy NATURAL JOIN produkty  
    NATURAL JOIN czas  
    NATURAL JOIN sprzedaz  
  WHERE produkty.typ_produktu = 'GAME'  
    AND sklepy.region = 'EAST'
```

```
SQL OPEN my_cursor
```

```
SQL FETCH my_cursor LOOP INTO  
  :APPEND sklepy  
  :APPEND produkty  
  :APPEND czas  
  :liczba_klientow
```

```
SQL CLOSE my_cursor
```

```
SQL CLEANUP
```

# Status wymiaru

- Status wymiaru to tymczasowe ograniczenie zbioru aktywnych wartości wymiaru. Określenie statusu wymiaru umożliwia wykonywanie operacji slice&dice

**REPORT** sklepy

```
->report sklepy
SKLEPY
-----
MIASTO_Atlanta
MIASTO_Boston
MIASTO_Chicago
MIASTO_Cincinnati
MIASTO_Dallas
MIASTO_Denver
MIASTO_LosAngeles
MIASTO_Louisville
MIASTO_Miami
...
```

**LIMIT** sklepy **TO FIRST** 5  
**STATUS** sklepy

```
->status sklepy
The current status of WH_SKLEPY is:
MIASTO_Atlanta TO MIASTO_Dallas
```

**LIMIT** sklepy **TO**  
**TOP** 20 PERCENTOF liczba\_klientow

**LIMIT** czas **TO** 'Sty04' **TO** 'Lip04'

**LIMIT** dim **TO** |**ADD**|**KEEP**|**INSERT**|**REMOVE**|**SORT**

# Zbiory wartości

- Zbiór wartości przechowuje nazwany zbiór wartości wymiaru. Zbiory wartości umożliwiają łatwe wykonywanie powtarzalnych analiz

```
DEFINE top_20_sklepy VALUESET sklepy [TEMP] [SESSION]  
LIMIT top_20_sklepy TO TOP 20 PERCENTOF liczba_klientow  
SHOW VALUES(top_20_sklepy)  
LIMIT sklepy TO top_20_sklepy
```

```
->show values(top_20_sklepy)  
SKLEP_15  
MIASTO_Cincinnati  
REGION_East  
SKLEP_19  
SKLEP_20  
MIASTO_Philadelphia
```

# Programy OLAP DML

- Program OLAP DML to obiekt przechowywany w przestrzeni analitycznej i zawierający sekwencję poleceń OLAP DML
- Programy umożliwiają
  - definiowanie zmiennych lokalnych i tymczasowych
  - parametryzację poprzez argumenty wywołania
  - wykorzystanie zmiennych substytucyjnych
  - wykorzystanie instrukcji sterowania
    - instrukcje warunkowe
    - pętle
    - podprogramy
    - wykorzystanie kontekstu
    - obsługę błędów

# Definiowane i kompilowanie programów

```

Program my_prog
File Edit
LIMIT sklepy TO top_20_sklepy
LIMIT produkty TO FIRST 10

REPORT DOWN sklepy ACROSS |produkty: liczba_klientow

ALLSTAT

3/26 | Changed | Loaded (6 lines).

```

**DEFINE** my\_prog PROGRAM

**EDIT** my\_prog

**COMPILE** my\_prog

**CALL** my\_prog

->call my\_prog

WH_SKLEPY	--LICZBA_KLIENTOW--									
	PRODUKT_1	PRODUKT_2	PRODUKT_3	PRODUKT_4	PRODUKT_5	PRODUKT_6	PRODUKT_7	PRODUKT_8	PRODUKT_9	PRODUKT_10
MIASTO_Cincinnati	946	821	986	204	799	842	906	820	886	153
REGION_East	934	975	357	750	279	761	891	541	388	571
SKLEP_19	893	996	295	873	7	842	339	640	542	502
SKLEP_20	886	221	678	182	596	552	926	891	865	439
MIASTO_Philadelphia	878	901	908	898	88	408	714	328	495	456

# Zmienne w programach

- Zmienne
  - lokalne: skalarne zmienne lokalne, istnieją tylko podczas wykonania programu, nie związane z wymiarami, definiowane za pomocą polecenia **VARIABLE**
  - tymczasowe: posiadają definicję w AW, mogą być związane z wymiarami, zachowują wartość w trakcie trwania sesji, definiowane za pomocą polecenia **DEFINE TEMP**

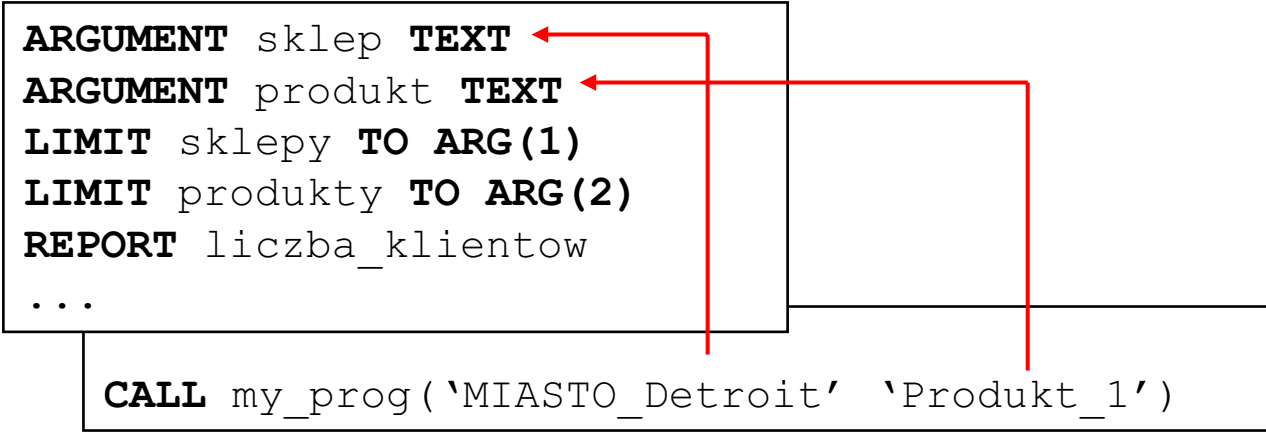
```
VARIABLE liczba_miast INTEGER
SQL DECLARE c_liczba_miast CURSOR FOR
  SELECT COUNT(miejscowosc) FROM sklepy
SQL OPEN c_liczba_miast
SQL FETCH c_liczba_miast INTO liczba_miast
...
```

# Parametry programów

- Do definiowania prostych i złożonych parametrów służy komenda **ARGUMENT**. Pobranie wartości w programie następuje przez wywołanie funkcji **ARG**

```
ARGUMENT sklep TEXT  
ARGUMENT produkt TEXT  
LIMIT sklepy TO ARG(1)  
LIMIT produkty TO ARG(2)  
REPORT liczba_klientow  
...
```

```
CALL my_prog('MIASTO_Detroit' 'Produkt_1')
```

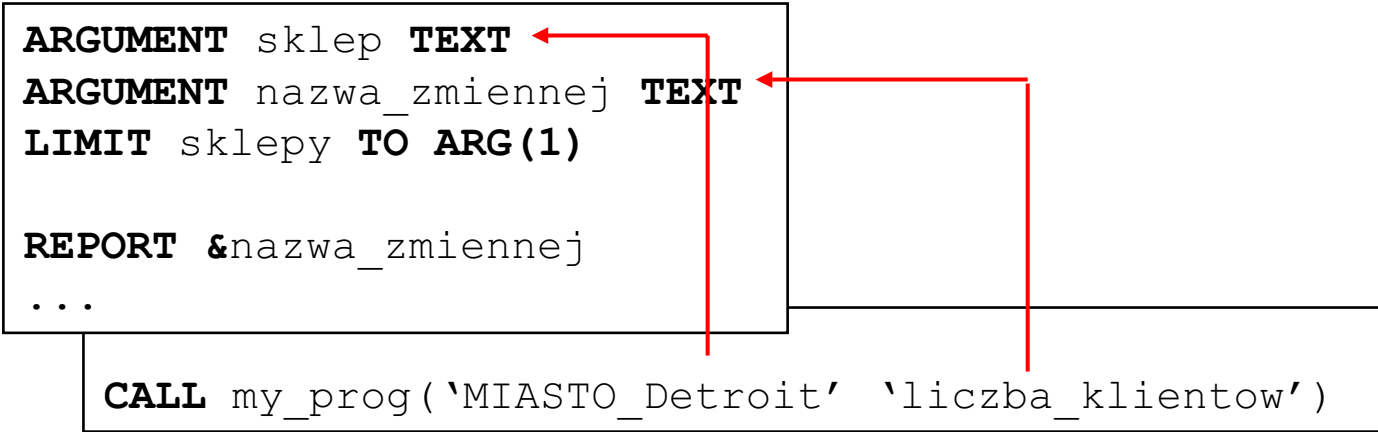
A diagram illustrating argument passing. A box on the left contains program code with two ARGUMENT lines: 'ARGUMENT sklep TEXT' and 'ARGUMENT produkt TEXT'. A box on the right contains a CALL statement: 'CALL my\_prog('MIASTO\_Detroit' 'Produkt\_1')'. Red arrows show the mapping: one arrow points from the first argument 'MIASTO\_Detroit' to the parameter 'sklep', and another arrow points from the second argument 'Produkt\_1' to the parameter 'produkt'.

# Substytucja

- Do substytucji należy wykorzystać znak `&`, wówczas do programu zostaje przekazana nazwa obiektu, a nie wartość
- Wyrażenia poprzedzane znakiem `&` są przed uruchomieniem programu zastępowane bieżącą wartością wyrażenia

```
ARGUMENT sklep TEXT  
ARGUMENT nazwa_zmiennej TEXT  
LIMIT sklepy TO ARG(1)  
  
REPORT &nazwa_zmiennej  
...
```

```
CALL my_prog('MIASTO_Detroit' 'liczba_klientow')
```



# Instrukcje sterujące

- Programy DML oferują następujące kategorie instrukcji
  - pętle: **FOR**, **WHILE**
  - podprogramy: **CALL**, **RETURN**
  - instrukcje warunkowe: **IF...THEN...ELSE**, **SWITCH...CASE**
  - instrukcje skoku: **GOTO**, **LIMIT...IFNONE**
  - operatory relacyjne: **GT**, **EQ**, **NE**, **LT**, **LE**, **GE**
  - obsługa kontekstu: **PUSH**, **POP**, **PUSHLEVEL**, **POPLEVEL**, **CONTEXT**
  - przekierowanie wyjścia: **OUTFILE**
  - obsługa błędów: **TRAP**

```
->call my_prog  
MIASTO_Atlanta - 533  
MIASTO_Boston - 741  
MIASTO_Chicago - 659  
MIASTO_Cincinnati - 946  
MIASTO_Dallas - 833  
MIASTO_Denver - 833  
MIASTO_Los Angeles - 709
```

```
LIMIT sklepy TO FIRST 20  
  
FOR sklepy  
DO  
    SHOW JOINCHARS(sklepy '-' liczba_klientow)  
DOEND  
...
```