



POZNAN UNIVERSITY OF TECHNOLOGY

Obiekt typu Dimension

Robert Wrembel

Politechnika Poznańska

Instytut Informatyki

Robert.Wrembel@cs.put.poznan.pl

www.cs.put.poznan.pl/rwrembel



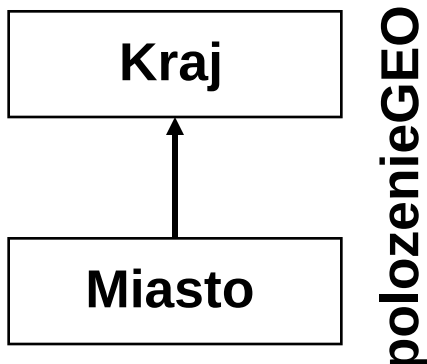
Modelowanie wymiarów

⇒ Wymiar

- **struktura umożliwiająca grupowanie danych z tabeli faktów**
- **implementowana jako obiekt bazy danych DIMENSION**
- **wykorzystanie DIMENSION**
 - **zaawansowane przepisywanie zapytań (ang. query rewrite)**
 - **wspomaganie projektowania zbioru zmaterializowanych perspektyw przez Summary Advisor**
 - **odświeżanie przyrostowe zmaterializowanych perspektyw zawierających agregaty**

CREATE DIMENSION (1)

⇒ znormalizowane tabele wymiarów



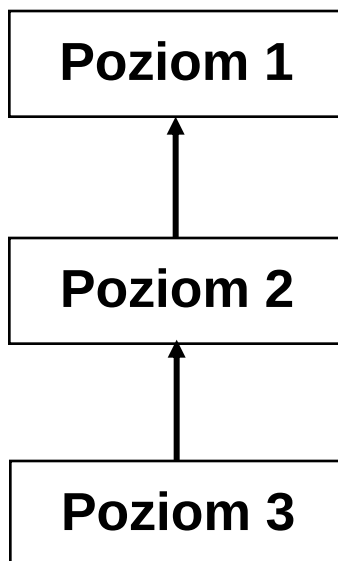
Nazwa	NULL?	Typ
-----	-----	-----
K_NAZWA	NOT NULL	VARCHAR2 (30)
L_MIESZK	NOT NULL	NUMBER (10)
Nazwa	NULL?	Typ
-----	-----	-----
M_NAZWA	NOT NULL	VARCHAR2 (30)
L_MIESZK	NOT NULL	NUMBER (10)
K_NAZWA		VARCHAR2 (30)

```

create dimension polozenieGEO
  level L_KRAJ is kraj.k_nazwa
  level L_MIASTO is miasto.m_nazwa
  hierarchy h_kraj_miasto
  (L_MIASTO child of L_KRAJ
   join key miasto.k_nazwa references L_KRAJ);
  
```

CREATE DIMENSION (2)

➔ znormalizowane tabele wymiarów

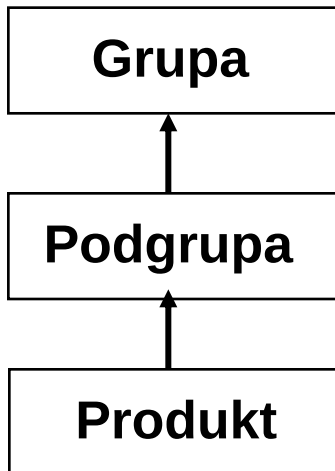


```
create dimension D
  level p3 is tabela3.PK
  level p2 is tabela2.PK
  level p1 is tabela1.PK
  hierarchy H
  (p3 child of
    p2 child of
      p1
    join key tabela3.FK references p2
    join key tabela2.FK references p1);
```

CREATE DIMENSION (3)

Name	Null?	Type
-----	-----	-----
ID_PROD	NOT NULL	NUMBER(6)
NAZWA	NOT NULL	VARCHAR2(30)
CENA_DET	NOT NULL	NUMBER(6,2)
CENA_HURT	NOT NULL	NUMBER(6,2)
PODGRUPA	NOT NULL	VARCHAR2(30)
PODGR_INF		VARCHAR2(50)
GRUPA	NOT NULL	VARCHAR2(30)
GRUPA_INF		VARCHAR2(50)

⇒ **zdenormalizowane
tabele wymiarów**



```
create dimension dim_produkty
  level produkt is produkty.produkt_id
  level podgrupa is produkty.podgrupa
  level grupa is produkty.grupa
hierarchy hier_produkty
  (produkt child of podgrupa
   podgrupa child of grupa);
```

Uwagi

- ➔ **W ramach tej samej hierarchii związek 1:n między poziomem nadrzędnym i podrzędnym**
- ➔ **Dla znormalizowanych tabel wymiarów definiować klucze obce**
- ➔ **Cykle w hierarchii są niedozwolone**

Zależności funkcyjne (1)

➔ **zdenormalizowane tabele wymiarów**

➔ **zależności funkcyjne między atrybutami wymiarów**

- **id_prod** → {nazwa, cena_det, cena_hurt}
- **podgrupa** → {podgr_inf}
- **grupa** → {grupa_inf}

Name	Null?	Type
-----	-----	-----
ID_PROD	NOT NULL	NUMBER(6)
NAZWA	NOT NULL	VARCHAR2(30)
CENA_DET	NOT NULL	NUMBER(6,2)
CENA_HURT	NOT NULL	NUMBER(6,2)
PODGRUPA	NOT NULL	VARCHAR2(30)
PODGR_INF		VARCHAR2(50)
GRUPA	NOT NULL	VARCHAR2(30)
GRUPA_INF		VARCHAR2(50)

Zależności funkcyjne (2)

⇒ wykorzystanie - przepisywanie zapytań

```
create table sprzedaz
(id_sprzed number(6) not null primary key,
 id_prod number(6) not null references produkty(id_prod),
 id_sklep number(6) not null references sklepy(id_sklep),
 data date not null,
 ilosc number(5) not null,
 kwota_calk number(9,2) not null);
```

id_prod → nazwa

```
create materialized view mv_sprzedaz
...
as select id_prod, id_sklep, sum(kwota_calk)
   from sprzedaz group by id_prod, id_sklep;
```

```
select pr.nazwa, sk.id_sklep, sum(sp.kwota_calk)
from produkty pr, sklepy sk, sprzedaz sp
where sp.id_prod=pr.id_prod
and sp.id_sklep=sk.id_sklep
group by pr.nazwa, sk.id_sklep;
```

Zależności funkcyjne (3)

⇒ jeżeli zdefiniowano

- $\text{id_prod} \rightarrow \{\text{nazwa}, \dots\}$

⇒ wyniki poniższego zapytania można wyznaczyć na podstawie **mv_sprzedaz**

```
create materialized view mv_sprzedaz
...
as select id_prod, id_sklep, sum(kwota_calk)
from sprzedaz group by id_prod, id_sklep;
```

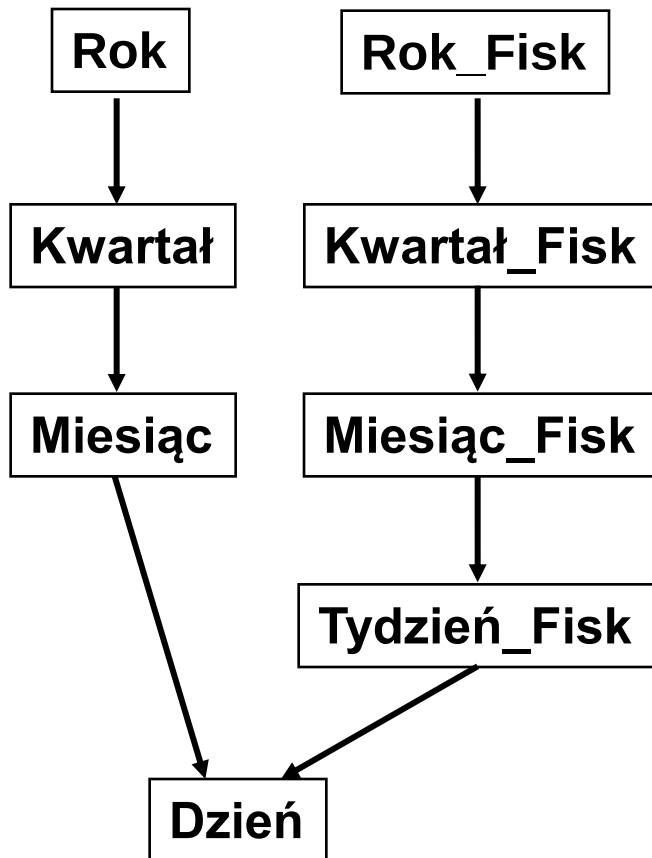
```
select pr.nazwa, sk.id_sklep, sum(sp.kwota_calk)
from produkty pr, sklepy sk, sprzedaz sp
where sp.id_prod=pr.id_prod
and sp.id_sklep=sk.id_sklep
group by pr.nazwa, sk.id_sklep;
```

Zależności funkcyjne (4)

➞ definiowanie

```
create dimension dim_produkty
  level produkt is produkty.id_prod
  level podgrupa is produkty.podgrupa
  level grupa is produkty.grupa
hierarchy hier_produkty
  (produkt child of
    podgrupa child of
      grupa)
attribute produkt determines
  (nazwa, cena_det, cena_hurt)
attribute podgrupa determines
  (podgr_inf)
attribute grupa determines
  (grupa_inf);
```

Wiele hierarchii



```
create dimension dim_czas
  level dzien is czas.id_dzien
  level miesiac is czas.miesiac
  level kwartal is czas.kwartal
  level rok is czas.rok
  level tydzien_fisk is czas.tydzien_fisk
  level miesiac_fisk is czas.miesiac_fisk
  level kwartal_fisk is czas.kwartal_fisk
  level rok_fisk is czas.rok_fisk
hierarchy hier_czas_kalendazowy
(dzien child of
 miesiac child of
 kwartal child of
 rok)
hierarchy hier_czas_fiskalny
(dzien child of
 tydzien_fisk child of
 miesiac_fisk child of
 kwartal_fisk child of
 rok_fisk);
```



Słownik bazy danych

- ➔ **USER_DIMENSIONS**
- ➔ **USER_DIM_ATTRIBUTES**
- ➔ **USER_DIM_CHILD_OF**
- ➔ **USER_DIM_HIERARCHIES**
- ➔ **USER_DIM_JOIN_KEY**
- ➔ **USER_DIM_LEVELS**
- ➔ **USER_DIM_LEVEL_KEY**
- ➔ **pakiet DEMO_DIM**
 - **PRINT_DIM('nazwa wymiaru')**
 - **PRINT_ALLDIMS**
 - **utworzyć pakiet w schemacie każdego użytkownika (plik %ORACLE_HOME%\rdbms\demo\smddim.sql)**

VALIDATE DIMENSION (1)

➤ Sprawdzenie poprawności struktury wymiaru określonej w poleceniu CREATE DIMENSION

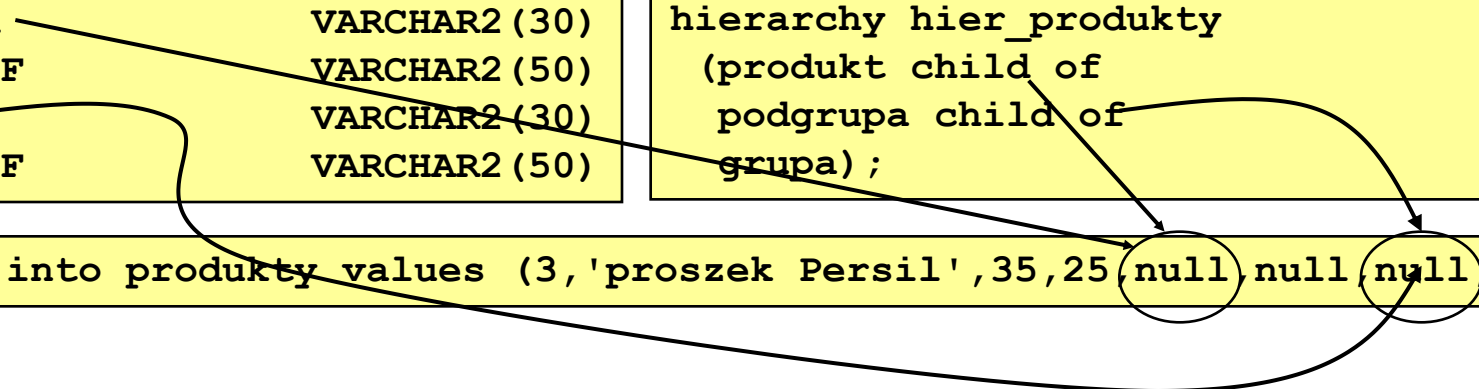
- sprawdzenie hierarchii wymiarów i zależności funkcyjnych

➤ Pakiet DBMS_DIMENSION

ID_PROD	NOT NULL NUMBER(6)
NAZWA	NOT NULL VARCHAR2(30)
CENA_DET	NOT NULL NUMBER(6,2)
CENA_HURT	NOT NULL NUMBER(6,2)
PODGRUPA	VARCHAR2(30)
PODGR_INF	VARCHAR2(50)
GRUPA	VARCHAR2(30)
GRUPA_INF	VARCHAR2(50)

```
create dimension dim_produkty
  level produkt is produkty.id_prod
  level podgrupa is produkty.podgrupa
  level grupa is produkty.grupa
hierarchy hier_produkty
  (produkt child of
   podgrupa child of
   grupa);
```

```
insert into produkty values (3,'proszek Persil',35,25,null,null,null,null);
```



VALIDATE DIMENSION (2)

➔ od Oracle10g (wcześniej pakiet DBMS_OLAP)

```
EXECUTE DBMS_DIMENSION.DESCRIBE_DIMENSION('produkty')
```

```
EXECUTE DBMS_DIMENSION.VALIDATE_DIMENSION('produkty',  
FALSE, TRUE, 'id walidacji');
```

TRUE - sprawdzane
tylko nowe rekordy

TRUE - sprawdzane czy wszystkie klucze obce wiążące wymiary
przyjmują wartości NOT NULL

➔ wyniki w tabeli **DIMENSION_EXCEPTIONS** tworzonej
w schemacie użytkownika walidującego

- skrypt **UTLDIM.SQL**

Modyfikowanie wymiarów (1)

```
SQL> exec demo_dim.print_dim('DIM_PRODUKTY')
DIMENSION SCOTT.DIM_PRODUKTY
LEVEL GRUPA IS SCOTT.PRODUKTY.GRUPA
LEVEL PODGRUPA IS SCOTT.PRODUKTY.PODGRUPA
LEVEL PRODUKT IS SCOTT.PRODUKTY.ID_PROD
HIERARCHY HIER_PRODUKTY (
PRODUKT
CHILD OF PODGRUPA
CHILD OF GRUPA
)
ATTRIBUTE GRUPA DETERMINES SCOTT.PRODUKTY.GRUPA_INF
ATTRIBUTE PODGRUPA DETERMINES SCOTT.PRODUKTY.PODGR_INF
ATTRIBUTE PRODUKT DETERMINES SCOTT.PRODUKTY.CENA_DET
ATTRIBUTE PRODUKT DETERMINES SCOTT.PRODUKTY.CENA_HURT
ATTRIBUTE PRODUKT DETERMINES SCOTT.PRODUKTY.NAZWA
```

```
alter dimension dim_produkty drop attribute produkt;
```

Modyfikowanie wymiarów (2)

```
SQL> exec demo_dim.print_dim('DIM_PRODUKTY')  
DIMENSION SCOTT.DIM_PRODUKTY  
LEVEL GRUPA IS SCOTT.PRODUKTY.GRUPA  
LEVEL PODGRUPA IS SCOTT.PRODUKTY.PODGRUPA  
LEVEL PRODUKT IS SCOTT.PRODUKTY.ID_PROD  
HIERARCHY HIER_PRODUKTY (  
  PRODUKT  
  CHILD OF PODGRUPA  
  CHILD OF GRUPA)
```

```
alter dimension dim_produkty  
drop hierarchy hier_produkty;
```

```
alter dimension dim_produkty drop level produkt;
```

Modyfikowanie wymiarów (3)

```
alter dimension dim_produkty add level  
    produkt is produkty.id_prod;
```

```
alter dimension dim_produkty add hierarchy  
    hier_produkty  
    (produkt child of  
      podgrupa child of  
        grupa) ;
```

```
alter dimension dim_produkty add attribute  
    podgrupa determines produkty.grupa_inf;
```

```
alter dimension dim_produkty compile;
```

```
drop dimension dim_produkty;
```