POLITECHNIKA ŚWIĘTOKRZYSKA W KIELCACH		
Bazy danych 2		
Daniel Iwaniec **Artur Kałuża**	**Labolatorium**	**Grupa 311A**

1. Wstęp

Tematem labolatorium jest **hurtownia danych** składu materiałów budowlanych. Zakres zagadnień obejmuje stworzenie struktury hurtowni danych, przygotowanie danych oraz ich import i analizę.

2. Struktura

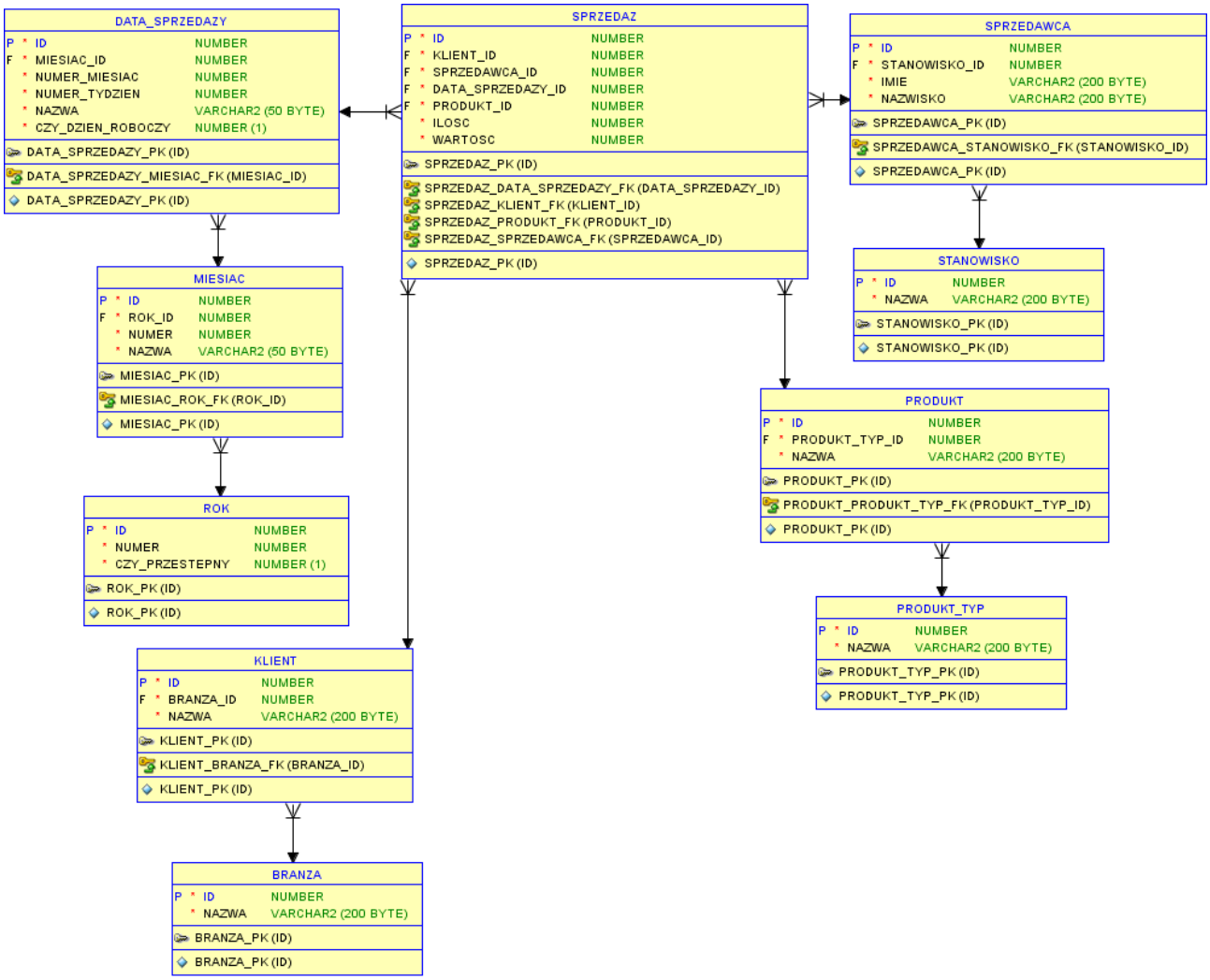

Diagram został wygenerowany w notacji bazującej na notacji **kruczej stopki** (ang. *crow's foot*) za pomocą programu Oracle SQL Data Modeler.

Struktura została stworzona zgodnie z konwencją **schematu płatka śniegu** (ang. *snowflake schema*). Składa się z **tabeli faktów** (sprzedaz) oraz 4 znormalizowanych **wymiarów** (produkt, sprzedawca, klient, data_sprzedazy).

Wszystkie tabele (*encje*) spełniają wymogi pierwszej postaci normalnej (**1NF**), ponieważ każdy wiersz posiada klucz klucz główny (ang. *primary key* lub *PK*) identyfikujący jendoznacznie dane, które są atomowe i nie powielają się.

Wszystkie tabele spełniają również wymogi drugiej postaci normalnej (**2NF**), ponieważ dane nie zależące funkcyjnie od klucza głównego zostały przeniesione do osobnych encji i powiązane za pomocą kluczy obcych (ang. *foregin key* lub *FK*).

Wszystkie tabele spełniają również wymogi trzeciej postaci normalnej (**3NF**), ponieważ nie istnieją atrybuty wtórne nie zależące od klucza głównego.

```
CREATE OR REPLACE PROCEDURE DROP_ALL IS
  CURSOR table_cursor IS
    SELECT object_name FROM user_objects WHERE object_type = 'TABLE';

  BEGIN
    FOR table_item IN table_cursor LOOP
      EXECUTE IMMEDIATE ('DROP TABLE ' || table_item.object_name || ' CASCADE CONSTRAINTS');
    END LOOP;
  END;
/
EXECUTE DROP_ALL;
DROP PROCEDURE DROP_ALL;

CREATE TABLE branza (
  id    INTEGER       NOT NULL,
  nazwa VARCHAR2(200) NOT NULL,
  CONSTRAINT branza_pk PRIMARY KEY (id)
);

CREATE TABLE klient (
  id        INTEGER       NOT NULL,
  branza_id INTEGER       NOT NULL,
  nazwa     VARCHAR2(200) NOT NULL,
  CONSTRAINT klient_pk        PRIMARY KEY (id),
  CONSTRAINT klient_branza_fk FOREIGN KEY (branza_id) REFERENCES branza (id)
);

CREATE TABLE stanowisko (
  id    INTEGER       NOT NULL,
  nazwa VARCHAR2(200) NOT NULL,
  CONSTRAINT stanowisko_pk PRIMARY KEY (id)
);

CREATE TABLE sprzedawca (
  id            INTEGER       NOT NULL,
  stanowisko_id INTEGER       NOT NULL,
  imie          VARCHAR2(200) NOT NULL,
  nazwisko      VARCHAR2(200) NOT NULL,
  CONSTRAINT sprzedawca_pk            PRIMARY KEY (id),
  CONSTRAINT sprzedawca_stanowisko_fk FOREIGN KEY (stanowisko_id) REFERENCES stanowisko (id)
);

CREATE TABLE rok (
  id             INTEGER      NOT NULL,
  numer          INTEGER      NOT NULL,
  czy_przestepny NUMBER(1)    NOT NULL,
  CONSTRAINT rok_pk PRIMARY KEY (id)
);

CREATE TABLE miesiac (
  id     INTEGER      NOT NULL,
  rok_id INTEGER      NOT NULL,
  numer  INTEGER      NOT NULL,
  nazwa  VARCHAR2(50) NOT NULL,
  CONSTRAINT miesiac_pk     PRIMARY KEY (id),
  CONSTRAINT miesiac_rok_fk FOREIGN KEY (rok_id) REFERENCES rok (id)
);

CREATE TABLE data_sprzedazy (
  id                INTEGER      NOT NULL,
  miesiac_id        INTEGER      NOT NULL,
  numer_miesiac     INTEGER      NOT NULL,
  numer_tydzien     INTEGER      NOT NULL,
  nazwa             VARCHAR2(50) NOT NULL,
  czy_dzien_roboczy NUMBER(1)    NOT NULL,
  CONSTRAINT data_sprzedazy_pk         PRIMARY KEY (id),
  CONSTRAINT data_sprzedazy_miesiac_fk FOREIGN KEY (miesiac_id) REFERENCES miesiac (id)
);

CREATE TABLE produkt_typ (
  id    INTEGER       NOT NULL,
  nazwa VARCHAR2(200) NOT NULL,
  CONSTRAINT produkt_typ_pk PRIMARY KEY (id)
);
```

```
CREATE TABLE produkt (
  id             INTEGER       NOT NULL,
  produkt_typ_id INTEGER       NOT NULL,
  nazwa          VARCHAR2(200) NOT NULL,
  CONSTRAINT produkt_pk             PRIMARY KEY (id),
  CONSTRAINT produkt_produkt_typ_fk FOREIGN KEY (produkt_typ_id) REFERENCES produkt_typ (id)
);

CREATE TABLE sprzedaz (
  id                INTEGER           NOT NULL,
  klient_id         INTEGER           NOT NULL,
  sprzedawca_id     INTEGER           NOT NULL,
  data_sprzedazy_id INTEGER           NOT NULL,
  produkt_id        INTEGER           NOT NULL,
  ilosc             INTEGER           NOT NULL,
  wartosc           INTEGER           NOT NULL,
  CONSTRAINT sprzedaz_pk                PRIMARY KEY (id),
  CONSTRAINT sprzedaz_klient_fk         FOREIGN KEY (klient_id)         REFERENCES klient (id),
  CONSTRAINT sprzedaz_sprzedawca_fk     FOREIGN KEY (sprzedawca_id)     REFERENCES sprzedawca (id),
  CONSTRAINT sprzedaz_data_sprzedazy_fk FOREIGN KEY (data_sprzedazy_id) REFERENCES data_sprzedazy (id),
  CONSTRAINT sprzedaz_produkt_fk        FOREIGN KEY (produkt_id)        REFERENCES produkt (id)
);
```

Procedura **DROP_ALL** pobiera z predefiniowanej tabeli **user_object** wszystkie utworzone tabele do kursora, które następnie za pomocą pętli są kolejno usuwane. Wykonywanie **DDL** wewnątrz procedury wymaga użycia **EXECUTE IMMEDIATE**. Na końcu procedura jest usuwana. Powodem użycia tej procedury jest możliwość jej poprawnego wykonania w przypadku gdy tabele nie istnieją.

Wszystkie zdefiniowane tabele posiadają jawnie zdefiniowane *klucze główne* oraz *klucze obce* (jeżeli te są potrzebne). Wszystkie zdefiniowane kolumny nie mogą być puste (**NOT NULL**). Wartości liczbowe są zdefiniowane jako **INTEGER**, tekstowe jako **VARCHAR2**, a logiczne jako **NUMER(1)** (gdzie 0 oznacza fałsz, a 1 prawdę).

3. Import danych

```
Import danych za pomocą SQL*Loader'a do tabeli BRANZA

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 4
Commit point reached - logical record count 5
```

```
Import danych za pomocą SQL*Loader'a do tabeli KLIENT

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 64
Commit point reached - logical record count 99
Commit point reached - logical record count 100
```

```
Import danych za pomocą SQL*Loader'a do tabeli ROK

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 4
Commit point reached - logical record count 5
```

```
Import danych za pomocą SQL*Loader'a do tabeli MIESIAC

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 59
Commit point reached - logical record count 60
```

```
Import danych za pomocą SQL*Loader'a do tabeli DATA_SPRZEDAZY

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 64
Commit point reached - logical record count 128
Commit point reached - logical record count 192
Commit point reached - logical record count 256
Commit point reached - logical record count 320
Commit point reached - logical record count 384
Commit point reached - logical record count 448
Commit point reached - logical record count 512
Commit point reached - logical record count 576
Commit point reached - logical record count 640
Commit point reached - logical record count 704
Commit point reached - logical record count 768
Commit point reached - logical record count 832
Commit point reached - logical record count 896
Commit point reached - logical record count 960
Commit point reached - logical record count 1024
Commit point reached - logical record count 1088
Commit point reached - logical record count 1152
Commit point reached - logical record count 1216
Commit point reached - logical record count 1280
Commit point reached - logical record count 1344
Commit point reached - logical record count 1408
Commit point reached - logical record count 1472
Commit point reached - logical record count 1536
Commit point reached - logical record count 1600
Commit point reached - logical record count 1664
Commit point reached - logical record count 1728
Commit point reached - logical record count 1792
Commit point reached - logical record count 1825
Commit point reached - logical record count 1826
```

```
Import danych za pomocą SQL*Loader'a do tabeli STANOWISKO

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 4
Commit point reached - logical record count 5
```

```
Import danych za pomocą SQL*Loader'a do tabeli SPRZEDAWCA

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 49
Commit point reached - logical record count 50
```

```
Import danych za pomocą SQL*Loader'a do tabeli PRODUKT_TYP

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 23
Commit point reached - logical record count 24
```

```
Import danych za pomocą SQL*Loader'a do tabeli PRODUKT

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 64
Commit point reached - logical record count 128
Commit point reached - logical record count 192
Commit point reached - logical record count 256
Commit point reached - logical record count 320
Commit point reached - logical record count 384
Commit point reached - logical record count 448
Commit point reached - logical record count 512
Commit point reached - logical record count 576
Commit point reached - logical record count 640
Commit point reached - logical record count 704
Commit point reached - logical record count 768
Commit point reached - logical record count 832
Commit point reached - logical record count 896
Commit point reached - logical record count 960
Commit point reached - logical record count 1024
Commit point reached - logical record count 1088
Commit point reached - logical record count 1152
Commit point reached - logical record count 1189
Commit point reached - logical record count 1190
```

```
Import danych za pomocą SQL*Loader'a do tabeli SPRZEDAZ

SQL*Loader: Release 11.2.0.2.0 - Production on Pn Gru 29 13:39:14 2014

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 64
Commit point reached - logical record count 128
Commit point reached - logical record count 192
Commit point reached - logical record count 256
Commit point reached - logical record count 320
Commit point reached - logical record count 384
Commit point reached - logical record count 448
Commit point reached - logical record count 512
Commit point reached - logical record count 576
Commit point reached - logical record count 640
Commit point reached - logical record count 704
Commit point reached - logical record count 768
Commit point reached - logical record count 832
Commit point reached - logical record count 896
Commit point reached - logical record count 960
Commit point reached - logical record count 1024
Commit point reached - logical record count 1088
Commit point reached - logical record count 1152
Commit point reached - logical record count 1216
Commit point reached - logical record count 1280
Commit point reached - logical record count 1344
Commit point reached - logical record count 1408
Commit point reached - logical record count 1472
Commit point reached - logical record count 1536
Commit point reached - logical record count 1600
Commit point reached - logical record count 1664
Commit point reached - logical record count 1728
Commit point reached - logical record count 1792
Commit point reached - logical record count 1856
Commit point reached - logical record count 1920
Commit point reached - logical record count 1984
Commit point reached - logical record count 2048
Commit point reached - logical record count 2112
Commit point reached - logical record count 2176
Commit point reached - logical record count 2240
Commit point reached - logical record count 2304
Commit point reached - logical record count 2368
Commit point reached - logical record count 2432
Commit point reached - logical record count 2496
Commit point reached - logical record count 2560
Commit point reached - logical record count 2624
Commit point reached - logical record count 2688
Commit point reached - logical record count 2752
Commit point reached - logical record count 2816
Commit point reached - logical record count 2880
Commit point reached - logical record count 2944
Commit point reached - logical record count 3008
Commit point reached - logical record count 3072
Commit point reached - logical record count 3136
Commit point reached - logical record count 3200
Commit point reached - logical record count 3264
Commit point reached - logical record count 3328
Commit point reached - logical record count 3392
Commit point reached - logical record count 3456
Commit point reached - logical record count 3520
Commit point reached - logical record count 3584
Commit point reached - logical record count 3648
Commit point reached - logical record count 3712
Commit point reached - logical record count 3776
Commit point reached - logical record count 3840
Commit point reached - logical record count 3904
Commit point reached - logical record count 3968
Commit point reached - logical record count 4032
Commit point reached - logical record count 4096
Commit point reached - logical record count 4160
Commit point reached - logical record count 4224
Commit point reached - logical record count 4288
Commit point reached - logical record count 4352
Commit point reached - logical record count 4416
Commit point reached - logical record count 4480
Commit point reached - logical record count 4544
Commit point reached - logical record count 4608
Commit point reached - logical record count 4672
Commit point reached - logical record count 4736
Commit point reached - logical record count 4800
Commit point reached - logical record count 4864
Commit point reached - logical record count 4928
Commit point reached - logical record count 4992
Commit point reached - logical record count 4999
Commit point reached - logical record count 5000
```

```
@echo off
chcp 65001>nul

set /P login=Podaj login || set login=
if [%login%] == [] goto :error

set /P password=Podaj hasło || set password=
if [%password%] == [] goto :error

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli ROK %ESC%[0m
sqlldr userid=%login%/%password% control=rok.ctl log=rok.log

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli MIESIAC %ESC%[0m
sqlldr userid=%login%/%password% control=miesiac.ctl log=miesiac.log

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli DATA_SPRZEDAZY %ESC%[0m
sqlldr userid=%login%/%password% control=data_sprzedazy.ctl log=data_sprzedazy.log

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli BRANZA %ESC%[0m
sqlldr userid=%login%/%password% control=branza.ctl log=branza.log

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli KLIENT %ESC%[0m
sqlldr userid=%login%/%password% control=klient.ctl log=klient.log

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli STANOWISKO %ESC%[0m
sqlldr userid=%login%/%password% control=stanowisko.ctl log=stanowisko.log

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli SPRZEDAWCA %ESC%[0m
sqlldr userid=%login%/%password% control=sprzedawca.ctl log=sprzedawca.log

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli PRODUKT_TYP %ESC%[0m
sqlldr userid=%login%/%password% control=produkt_typ.ctl log=produkt_typ.log

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli PRODUKT %ESC%[0m
sqlldr userid=%login%/%password% control=produkt.ctl log=produkt.log

echo.
echo.
echo %ESC%[30;48;5;118m Import danych za pomocą SQL*Loader'a do tabeli SPRZEDAZ %ESC%[0m
sqlldr userid=%login%/%password% control=sprzedaz.ctl log=sprzedaz.log

goto:eof
:error
echo %ESC%[30;48;5;196m Musisz podać dane dostępowe %ESC%[0m
```

Dane zostały przygotowane w formie plików **CSV** (pełne pliki zawarte w katalogu /dane). Źródłem danych były skrypty **PHP** bezpośrednio generujące dane oraz parsujące katalogi oraz sklepy interetowe. Wszystkim plikom **CSV** utworzone zostały odpowiadające pliki **CTL**. (pełne pliki zawarte w katalogu /dane). Dane zostały zaimportowane za pomocą **SQL*Loader'a** (pliki logów zawarte w katalogu /dane). Automatyczny import został zrealizowany za pomocą pliku wsadowego **BAT**. Plik wsadowy pobiera od użytkownika login oraz hasło potrzebne do utworzenia połączenia z bazą danych. Następnie wszystkie tabele są wypełniane danymi. Dodatkowo plik zmienia stronę kodową na **UTF-8** (aby widoczne były polskie znaki) oraz używa kolorowania **ANSICON**.

4. Zapytania

Wszystkie wyniki zapytań są umieszczone w folderze **/wyniki_zapytan.**

Operatory grupujące

```
SELECT produkt_id, klient_id, count(produkt_id) FROM sprzedaz
GROUP BY rollup(produkt_id, klient_id);
```

Opis

Funkcja rollup - Powyżej przedstawiono zapytanie, które w jednym przebiegu zlicza ilość produktów sprzedaży, które kupił konkretny kontrahent, ilosc sztuk sprzedanego produktu oraz ile sztuk wszytkich produktów zostało sprzedane.

```
SELECT produkt_id, data_sprzedazy_id, sum(wartosc) FROM sprzedaz
GROUP BY cube (produkt_id, data_sprzedazy_id);
```

Opis

Funkcja cube - Powyżej przedstawiono zapytanie, które w jednym przebiegu wyznacza sumy produktów w ramach sprzedaży całej firmy, w obrębie daty, w ramach konkretnego produktu oraz sprzedaży produktu kokretnego dnia.

```
SELECT k.nazwa, b.nazwa AS "BRANZA", concat(m.nazwa, r.numer) AS "Data Sprzedazy", srednia FROM
(
SELECT s.klient_id AS klient, b.id AS branzaid, s.data_sprzedazy_id AS data, Round(Avg(s.ilosc)) AS srednia
FROM sprzedaz s
JOIN klient k ON s.klient_id = k.id
JOIN branza b ON k.branza_id = b.id
JOIN data_sprzedazy d ON s.data_sprzedazy_id = d.id
group BY GROUPing sets (s.klient_id,b.id,s.data_sprzedazy_id)
) query1

LEFT JOIN branza b ON query1.branzaid = b.id
LEFT JOIN klient k ON query1.klient = k.id
left JOIN data_sprzedazy d ON query1.data = d.id
left JOIN miesiac m ON d.miesiac_id = m.id
left JOIN rok r ON m.rok_id = r.id;
```

Opis

Funkcja grouping sets - Powyżej przedstawiono zapytanie, które zwraca średnią w odniesieniu do daty sprzedaży, średnią w odniesieniu do klienta, średnią w odniesieniu do branży. W zapytaniu zastosowano podzapytanie *query1* oraz funkcje *concat* (wyświetlenie wyniku w jednej kolumnie) oraz *join* w celu wyswietlenia roku i miesiąca w dacie sprzedaży (zamiast ich id).

Funkcje analityczne

```
select sprzedawca_id, produkt_id, ilosc,
sum(ilosc) over (partition by sprzedawca_id)
sum_ilosc, round(100*ilosc/sum(ilosc) OVER (PARTITION BY sprzedawca_id))  "udzial%"
from sprzedaz
order by sprzedawca_id, produkt_id;
```

Opis

Zapytanie zwraca wartosc funkcji sum oddzielnie dla kazdego rekordu w ktorym jest sprzedawca i produkt posortowaną po sprzedawcy, oraz procent sprzedawcy w całości sprzedaży.

Funkcje analityczne - ruchome okno obliczeniowe

```
SELECT r.numer AS rok,m.nazwa AS miesiac, wartosc_all FROM
(
SELECT data_sprzedazy_id,sum(wartosc) over (partition by r.numer order by d.id range between unbounded
preceding and current row)
AS wartosc_all
from sprzedaz s
JOIN data_sprzedazy d ON s.data_sprzedazy_id = d.id
JOIN miesiac m ON d.miesiac_id = m.id
JOIN rok r ON m.rok_id = r.id

)query2
left JOIN data_sprzedazy d ON query2.data_sprzedazy_id = d.id
left JOIN miesiac m ON d.miesiac_id = m.id
left JOIN rok r ON m.rok_id = r.id
GROUP BY r.numer, m.nazwa,wartosc_all;
```

Opis

Powyższe zapytanie wylicza sumę sprzedaży narastająco w kolejnych miesiącach w ramach każdego roku. W zapytaniu zastosowano okno logiczne *range* zaczynające sie od pierwszego rekordu grypy (*unbounded preceeding*) oraz od wartości pobranej z aktualnego rekordu (*current row*). W zapytaniu zastosowano podzapytanie *query2* oraz funkcje *join* w celu wyswietlenia roku i miesiąca w dacie sprzedaży (po ich nazwie a nie numerze id). Funkcja *group by* wyeliminowała miesiące z powtarzającymi się wartościami.

Funkcje rankingowe

```
select sprzedawca_id, produkt_id, wartosc,
rank() over (partition by sprzedawca_id order by wartosc desc) as rank,
dense_rank() over (partition by sprzedawca_id order by wartosc desc) as
dense_rank
from sprzedaz
order by sprzedawca_id, produkt_id;
```

Opis

Powyższe zapytanie wyświetla numerów pozycji rankingowych sprzedawców według sumy ich sprzedaży. Z powyższego zapytania można się dowiedzieć, że najlepszym sprzedawcą był sprzedawca o numerze id 35, który sprzedał produkt o numerze id 56 o wartości 1978. W zapytaniu zastosowano funkcje rank oraz dense_rank – pierwsza spowodowała dziurę w numeracji, ponieważ wystąpiły jednakowe pozycje rankingowe.