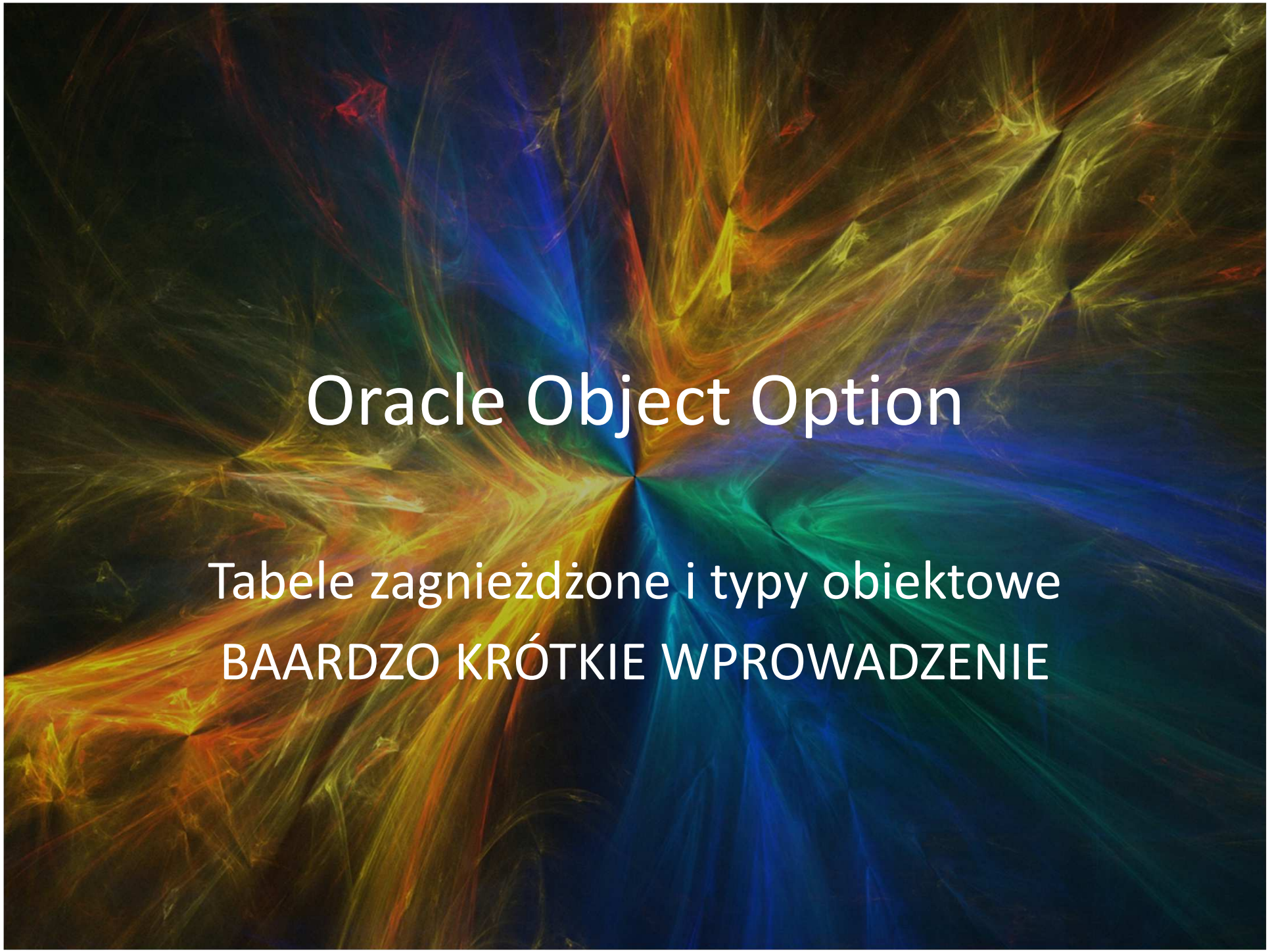

Oracle Object Option
Tabele zagnieżdżone i typy obiektowe
BAARDZO KRÓTKIE WPROWADZENIE

Typ kolekcji

- Utworzenie typu tabeli zagnieżdżonej

```
CREATE TYPE NAZWA AS TABLE OF TYP;
/
```

```
CREATE TYPE LICZBY AS TABLE OF NUMERIC;
/
```

Utworzenie kolekcji w zapytaniu

- Kolekcja wszystkich pensji

```
SELECT CAST(COLLECT(PLACA_POD) AS LICZBY)
FROM PRACOWNICY;
```

- Kolekcja wszystkich pensji dla każdego etatu

```
SELECT ETAT,CAST(COLLECT(PLACA_POD) AS LICZBY)
FROM PRACOWNICY
GROUP BY ETAT;
```

Tabela z tabelą zagnieżdżoną

- Utworzenie tabeli z tabelą zagnieżdżoną

```
CREATE TABLE Pensje (
 etat varchar2(100) primary key,
 wartosci liczby
)
nested table wartosci store as nested_pensje_wartosci;
```

- Wstawienie nowego wiersza z kolekcją

```
Insert into pensje(etat,wartosci) values(
 'test',liczby(1,2,3,4)
);
```

- Wstawienie nowej wartości do kolekcji

```
insert into table(
 select wartosci from pensje where etat='test',
) values(5);
```

Odczyt danych z tabeli zagnieżdżonej

- Odczyt danych z jednej konkretnej kolekcji

```
SELECT value(x) FROM TABLE(
 select wartosci from pensje where etat='test'
) x
```

- Odczyt danych ze wszystkich kolekcji w tabeli

```
SELECT etat, value(x)
FROM pensje CROSS JOIN TABLE(wartosci) x;
```

Ćwiczenie 1

1. Utwórz typ kolekcji **ListaDat** reprezentujący zbiór dat.
2. Utwórz perspektywę **DatyZatrudnień** w której dla każdego identyfikatora zespołu (id_zesp) znajduje się kolekcja dat zatrudnienia.
3. Napisz zapytanie do tej perspektywy, w którym wyliczasz ile dni zatrudniony jest każdy pracownik zespołu 10.
4. Napisz zapytanie do tej perspektywy, w którym dla każdego id_zesp znajdowana jest najwcześniejsza data zatrudnienia.

Ćwiczenie 2

1. Utwórz typ „ListaTabel”, który jest kolekcją łańcuchów typu varchar2(50).
2. Utwórz tabelę „Mirror” o dwóch kolumnach: nazwa użytkownika i kolekcja nazw tabel użytkownika.
3. Wypełnij utworzoną tabelę danymi (wykorzystaj polecenie insert..select i tabelę all_tables).
4. Do kolekcji odpowiadającej Twojemu użytkownikowi wstaw nazwę tabeli „NewOne”.

Typ obiektowy

- Utworzenie typu

```
create type Czlowiek as object (
 imie varchar2(50),
 nazwisko varchar2(50)
);
```

- Utworzenie obiektu w zapytaniu

```
Select id_prac,czlowiek(imie,nazwisko) as obiekt
From pracownicy;
```

Kolekcja obiektów

- Utworzenie typu kolekcji

```
CREATE TYPE LUDZIE AS TABLE OF CZLOWIEK;
/
```

- Kolekcja wszystkich pracowników

```
SELECT CAST(COLLECT(CZLOWIEK(IMIE,NAZWISKO)) AS LUDZIE)
FROM PRACOWNICY;
```

- Kolekcja wszystkich pracowników w każdym z etatów

```
SELECT ETAT,
 CAST(COLLECT(CZLOWIEK(IMIE,NAZWISKO)) AS LUDZIE)
FROM PRACOWNICY
GROUP BY ETAT;
```

Tabela z tabelą zagn. obiektów

- Utworzenie tabeli z tabelą zagnieżdżoną

```
CREATE TABLE Prac2 (
 etat varchar2(100) primary key,
 zatrudnieni ludzie
)
nested table zatrudnieni
store as nested_zatrudnieni_ludzie;
```

- Wstawienie nowego wiersza z kolekcją

```
Insert into prac2(etat,zatrudnieni) values(
 'test',ludzie(
   czlowiek('Jan','Kowalski'),
   czlowiek('Jakub','Palka')
 )
);
```

Tabela z tabelą zagn. obiektów

- Wstawienie nowej wartości do kolekcji

```
insert into table(
 select zatrudnieni from prac2 where etat='test'
) values(czlowiek('Derp','Herp'));
```

Odczyt danych z tabeli zagnieżdżonej

- Odczyt obiektów z jednej kolekcji

```
SELECT value(x) FROM TABLE(
 select zatrudnieni from prac2 where etat='test'
) x;
```

- Odczyt wartości pól obiektów

```
SELECT value(x).imie, value(x).nazwisko FROM TABLE(
 select zatrudnieni from prac2 where etat='test'
) x;
```

- Zapis skrótowy

```
SELECT x.imie, x.nazwisko FROM TABLE(
 select zatrudnieni from prac2 where etat='test'
) x;
```

Odczyt danych z tabeli zagnieżdżonej

- Odczyt danych ze wszystkich kolekcji w tabeli

```
SELECT etat, value(x)
FROM prac2 CROSS JOIN TABLE(zatrudnieni) x;
```

- To samo, ale z dostępem do pól

```
SELECT etat, value(x).imie, value(x).nazwisko
FROM prac2 CROSS JOIN TABLE(zatrudnieni) x;
```

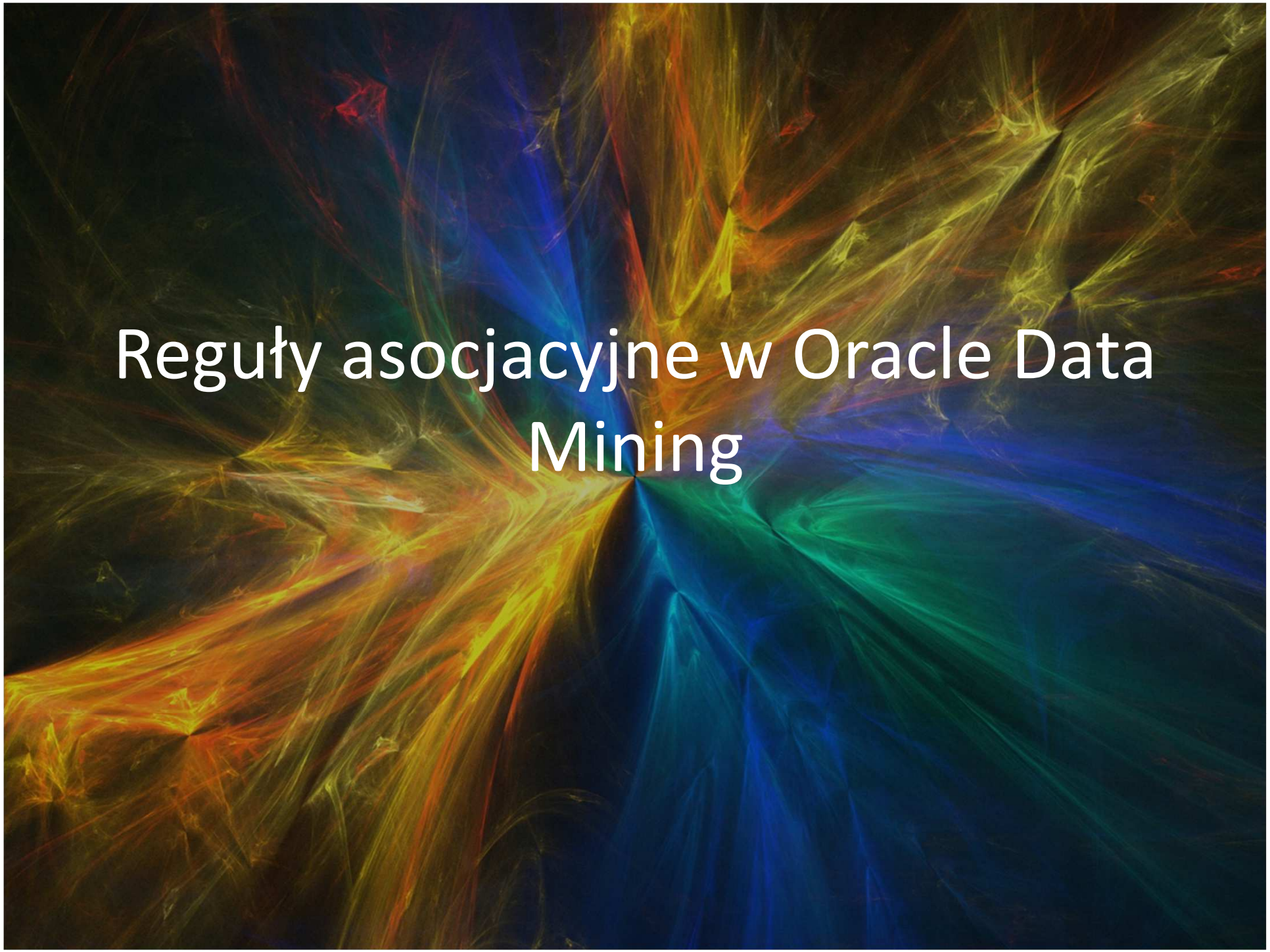
```
SELECT etat, x.imie, x.nazwisko
FROM prac2 CROSS JOIN TABLE(zatrudnieni) x;
```

Ćwiczenie 3a

1. Utwórz typ obiektowy **Zwierz** reprezentujący zwierzaki. W typie powinny znajdować się następujące pola:
   - Nazwa (varchar2(50))
   - Liczba nóg (number(1))
   - Czy ma ogon (char(1))
2. Utwórz typ kolekcji Zwierzaki, który jest kolekcją obiektów typu **Zwierz**

Ćwiczenie 3b

3. Wykorzystując zestaw danych ZOO (patrz opis zestawów danych) utwórz perspektywę, która dla każdego typu zwierzaka (ssak, gad itp.) zawiera listę obiektów reprezentujących zwierzęta tego typu.
4. Napisz zapytanie do tej perspektywy, które dla każdego typu zwierzaka wylicza średnią liczbę nóg.


Reguły asocjacyjne w Oracle Data Mining

Typy DM_Nested_Numerical(+s)

```
CREATE TYPE DM_Nested_Numerical AS OBJECT (
 ATTRIBUTE_NAME VARCHAR2(4000),
 VALUE NUMBER
);
/
```

```
CREATE TYPE DM_Nested_Numericals AS
 TABLE OF DM_Nested_Numericals;
/
```

Oracle Data Mining (1)

- Przygotowanie danych (format poziomy):

```
create or replace view market_basket_data as
SELECT distinct a.cust_id, b.prod_name
FROM SH.SALES a join SH.PRODUCTS b on
 (a.prod_id = b.prod_id)
where cust_id between 100000 AND 100500;
```

Ci klienci dokonali zakupu tylko raz, dzięki czemu to zapytanie jest poprawne

- Transformacja do zbioru kolekcji (format pionowy):

```
CREATE OR REPLACE VIEW MARKET_BASKET_DATA_AR AS
SELECT CUST_ID, CAST(COLLECT(
 DM_Nested_Numerical(SUBSTR(PROD_NAME,1,30), 1)
 ) AS DM_Nested_Numericals) BASKET
FROM MARKET_BASKET_DATA
GROUP BY CUST_ID;
```

Oracle Data Mining (2)

- Porównaj format pionowy i poziomy bazy danych

```
SELECT * FROM MARKET_BASKET_DATA
WHERE cust_id = 100001;

SELECT * FROM MARKET_BASKET_DATA_AR
WHERE cust_id = 100001;
```

- Usuń model jeżeli wykonywałeś ćwiczenie wcześniej

```
BEGIN
  DBMS_DATA_MINING.DROP_MODEL('Associations');
END;
```

Oracle Data Mining (3)

- Utwórz tabelę przechowującą parametry konfiguracyjne algorytmu

```
DROP TABLE settings;
CREATE TABLE settings (
  setting_name VARCHAR2(30),
  setting_value VARCHAR2(128)
);
```

- Wypełnij tabelę danymi:

```
BEGIN
 INSERT INTO settings VALUES
  (dbms_data_mining.asso_max_rule_length,3);
 INSERT INTO settings VALUES
  (dbms_data_mining.asso_min_support,0.1);
 INSERT INTO settings VALUES
  (dbms_data_mining.asso_min_confidence,0.5);
END;
COMMIT;
```

Oracle Data Mining (4)

- Zbuduj model który będzie zawierał reuły asocjacyjne.

```
BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
    model_name           => 'Associations',
    mining_function      => DBMS_DATA_MINING.ASSOCIATION,
    data_table_name      => 'market_basket_data_ar',
    case_id_column_name => 'cust_id',
    settings_table_name => 'settings' );
END;
```

- Odczytaj parametry zbudowanego modelu

```
SELECT * FROM TABLE (
  DBMS_DATA_MINING.GET_MODEL_SETTINGS('Associations'))
ORDER BY setting_name;
```

Oracle Data Mining (5)

- Odczytaj znalezione zbiory częste

```
SELECT
 t.itemset_id,
 t.support,
 t.items
FROM TABLE (
 DBMS_DATA_MINING.GET_FREQUENT_ITEMSETS('Associations')
) t;
```

Oracle Data Mining (6)

- Odczytaj znalezione reguły asocjacyjne

```
SELECT t.rule_id, t.rule_support,
t.rule_confidence,t.antecedent,t.consequent
FROM
 TABLE (DBMS_DATA_MINING.
  GET_ASSOCIATION_RULES('Associations')) t
order by t.rule_support DESC, t.rule_confidence;
```

Zadanie

- Napisz wariant zapytania odczytującego zbiory częste, który przedstawia wynik bez kolekcji (jako listę rekordów).

Zadanie (Oracle Data Mining)

- Wykorzystując tabele DMDATA.Movies i DMDATA.Ratings znajdź reguły asocjacyjne mówiące:
  - Jeżeli klient lubi jakieś filmy to polubi też inne