Hurtownie danych

na podstawie referatów IV Szkoły PLOUG

http://www.ploug.org.pl/

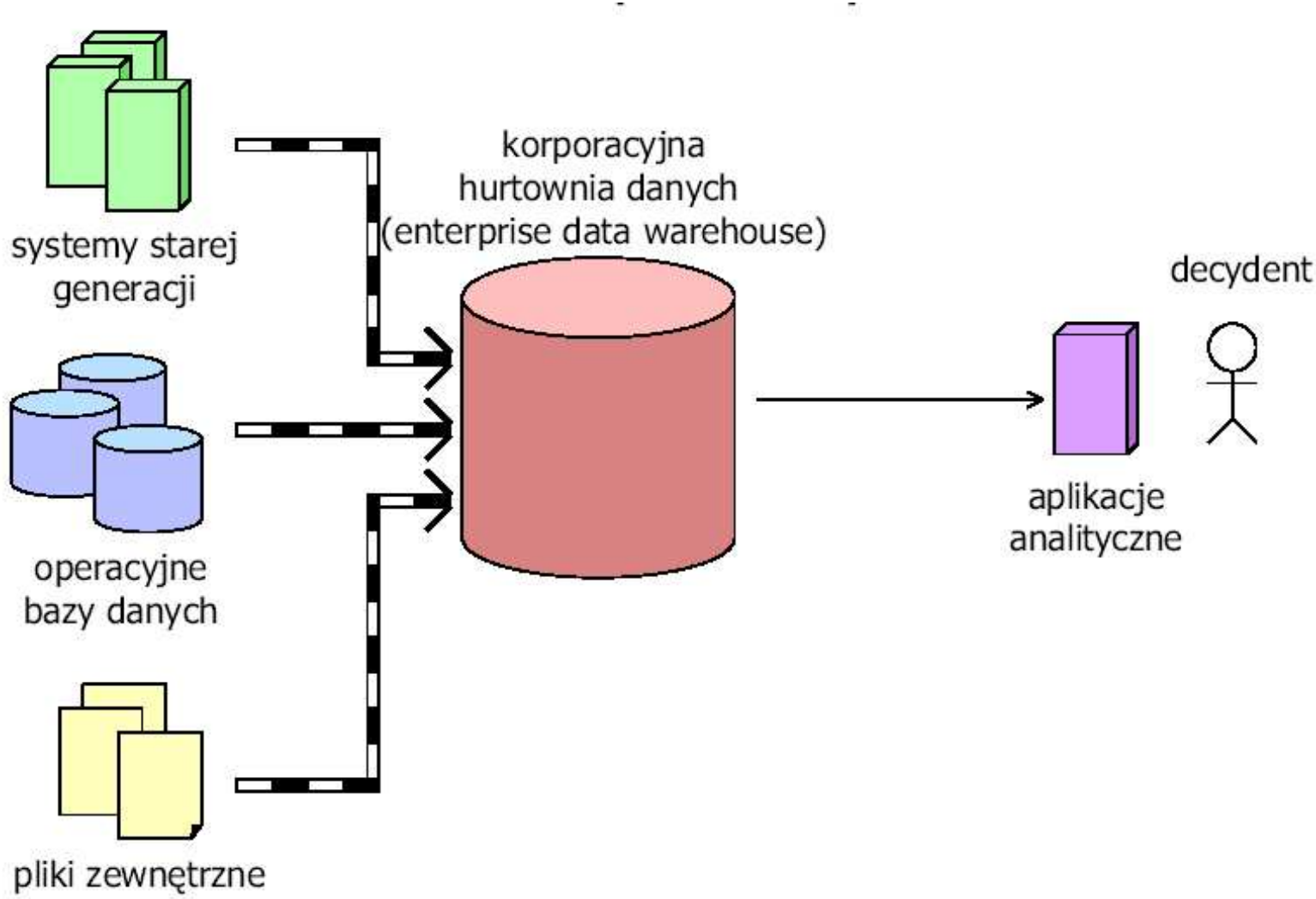
Plan seminarium

- Wprowadzenie
- Schematy logiczne dla hurtowni danych
- Struktury fizyczne
- Wydajność
- Ekstrakcja, transformacja i ładownaie danych (ETL)
- Funkcje analityczne i rozszerzenia grupowania SQL

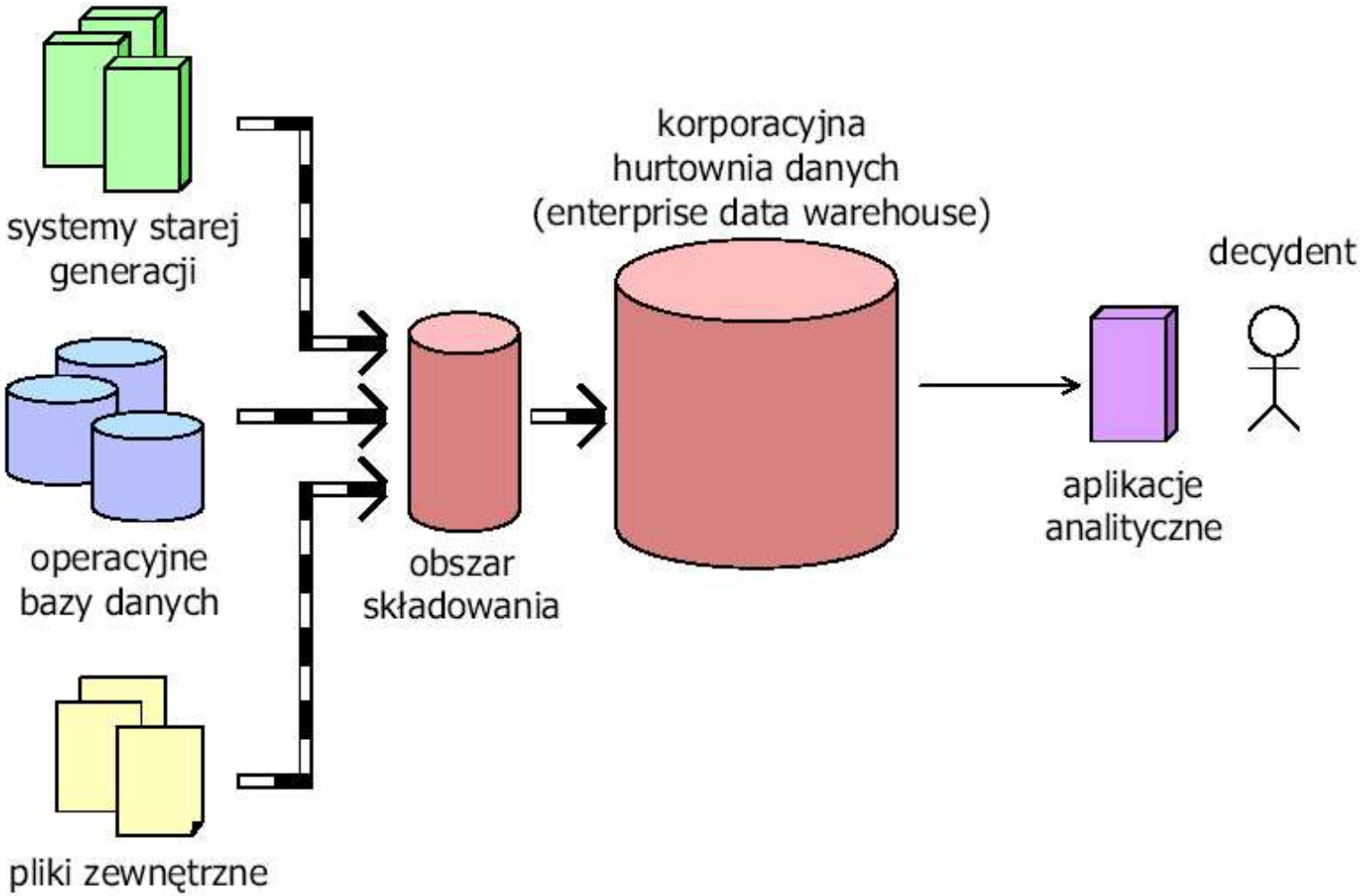
Business Intelligence

- Technologia informatyczna służąca do przekształcania dużych wolumenów danych w informacje, a następnie do przekształcania tych informacji w wiedzę
- Adresowana do decydentów
- Stawiająca drastyczne wymagania wydajnościowe
- Skupiona wokół technologii hurtowni danych

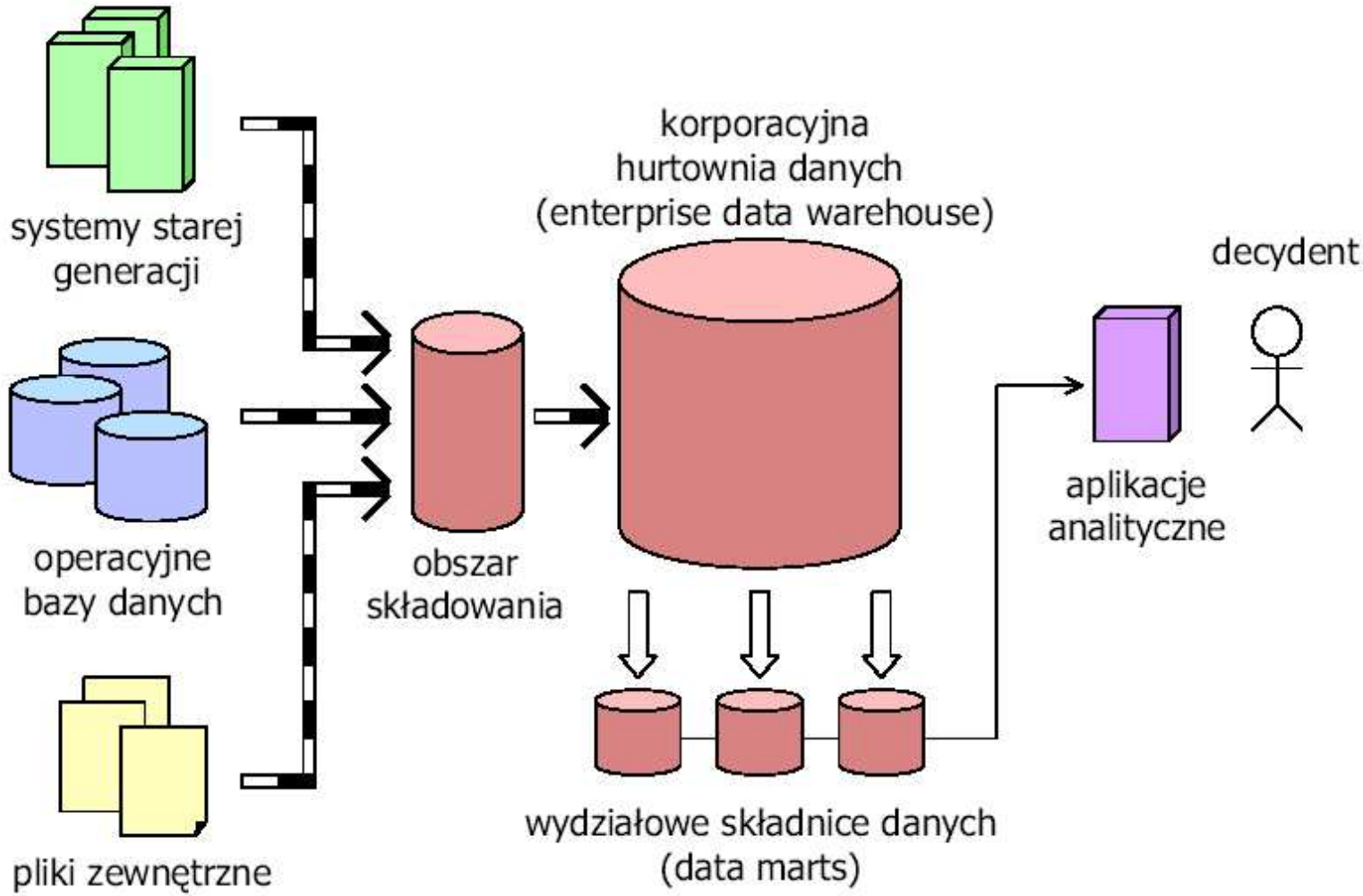
Środowisko hurtowni danych - model podstawowy



Środowisko hurtowni danych - architektura z obszarem składowania



Środowisko hurtowni danych - architektura z obszarem składowania i składnicami danych



Hurtownia danych - definicja

- **"Hurtownia danych to tematyczna baza danych, która trwale przechowuje zintegrowane dane opisane wymiarem czasu "** [Inmon96]
- "tematyczna baza danych" - dane dotyczą głównych obszarów działalności przedsiębiorstwa
- "trwale przechowuje" - dane nie są zmieniane ani usuwane; hurtownia danych ma charakter przyrostowy
- "zintegrowane dane" - dane dotyczące tego samego podmiotu stanowią całość
- "opisane wymiarem czasu" - dane opisują zdarzenia historyczne, a nie tylko stan aktualny

Porównanie hurtowni danych z bazami operacyjnymi

Cecha	**OLTP**	**Hurtownia danych**
czas odpowiedzi aplikacji	ułamki sekundy - sekundy	sekundy - godziny
wykonywane operacje	DML	select
czasowy zakres danych	30-60 dni	2-10 lat
organizacja danych	według aplikacji	tematyczna
rozmiar	małe - duże	duże - wielkie
intensywność operacji dyskowych	mała - średnia	wielka

Metody projektowania i wdrażania hurtowni danych

- **Wielki wybuch**
  - budowa korporacyjnej hurtowni danych w ramach jednego projektu
  - globalna analiza wymagań, implementacja całościowej hurtowni danych, implementacja aplikacji analitycznych
  - długi czas realizacji, początkowo wybrane technologie przestają by wspierane przez dostawców
- **Od ogółu do szczegółu**
  - iteracyjna realizacja hurtowni danych, po kolei dla każdego obszaru tematycznego, konstrukcja zależnych składnic danych
- **Od szczegółu do ogółu**
  - realizacja niezależnych składnic danych, z wizją ich przyszłej integracji w jedną hurtownię danych

ETL: Extraction, Transformation, Loading

- **Ekstrakcja:** odczyt źródłowych danych z operacyjnych baz danych, systemów starej generacji, plików zewnętrznych
- **Transformacja:** łączenie danych, ich weryfikacja, walidacja, czyszczenie i znakowanie czasowe
- **Wczytywanie:** wprowadzanie danych do docelowej hurtowni danych

Kroki implementacji systemu Business Intelligence

- Analiza wymagań - zgromadzenie wiedzy o wymaganiach biznesowych w zakresie przetwarzania analitycznego
- Projekt logiczny hurtowni danych - pojęciowa definicja wymaganych struktur danych
- Implementacja struktur fizycznych hurtowni danych - tworzenie bazy danych, tabel, indeksów, materializowanych perspektyw
- Implementacja oprogramowania ETL - konstrukcja modułów programowych służących do zasilania hurtowni danych nowymi danymi
- Realizacja aplikacji analitycznych - implementacja programów dla użytkowników końcowych
- Strojenie hurtowni danych - rekonfiguracja serwera bazy danych, tworzenie dodatkowych indeksów i materializowanych perspektyw

Technologie Oracle dla hurtowni danych [1]

- Implementacja hurtowni danych
  - Oracle Database
  - Oracle OLAP
  - Oracle Warehouse Builder
- Implementacja ładowania danych
  - SQL*Loader
  - Replikacja
  - Tabele zewnętrzne
  - ODBC/JDBC
  - Oracle Gateways

Technologie Oracle dla hurtowni danych [2]

- Implementacja aplikacji analitycznych
  - Oracle Discoverer
  - Oracle Business Intelligence Beans
  - Oracle Reports
  - Oracle Data Mining

Etapy projektowania hurtowni danych

- **Model biznesowy**
  - Efekt analizy strategicznej
  - Identyfikacja miar i wymiarów dla poszczególnych procesów biznesowych
- **Model logiczny** (wymiarowy)
  - Model abstrakcyjny, konceptualny
  - Encje i atrybuty (reprezentowane w modelu relacyjnym jako tabele i powiązania między nimi)
- **Model fizyczny**
  - Wybór sposobu składowania danych
  - Formaty danych
  - Strategie partycjonowania
  - Wybór indeksów

Miary

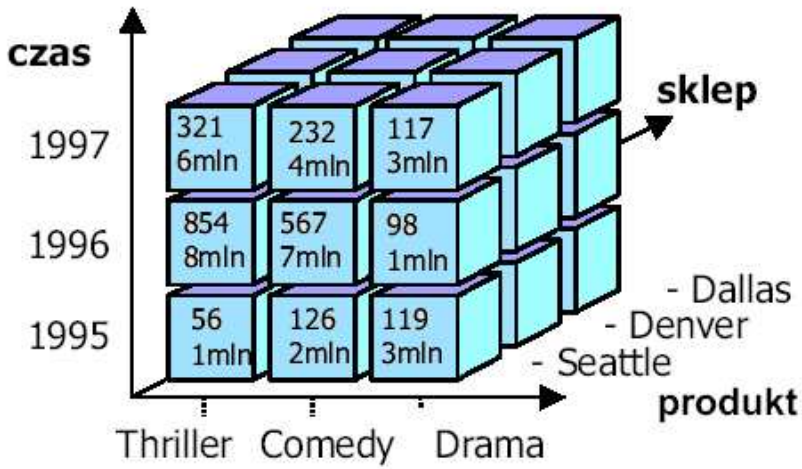
- **Miary**, inaczej: fakty
  - Wartości ciągłe, numeryczne
  - Typowe miary: wartość sprzedaży, koszt, zysk, sprzedana ilość
  - Rodzaje miar
    - addytywne (we wszystkich wymiarach) - np. liczba sprzedanych sztuk
    - częściowo addytywne (addytywne w niektórych wymiarach) - np. stan w magazynie
    - nieaddytywne

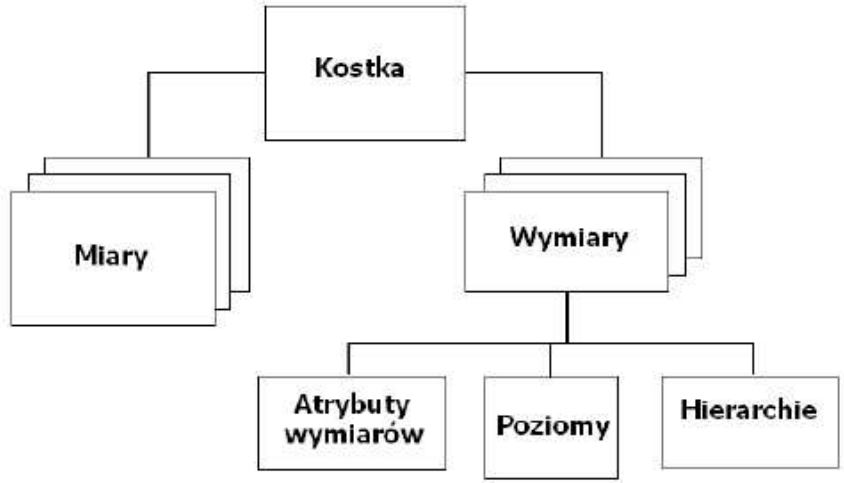
Wymiary

- **Wymiary**
  - Wartości dyskretne, niezmienne lub rzadko zmienne
  - Nadają znaczenie danym (miarom, faktom)
  - Typowe wymiary: klient, czas, produkt, sklep
  - Hierarchie - umożliwiają organizację danych na różnych poziomach agregacji
  - Poziom - reprezentuje pozycję w hierarchii
  - Atrybuty - dostarczają dodatkowych informacji o danych, np. kolor, smak, dzień tygodnia

Wielowymiarowy model danych

- Dane na potrzeby przetwarzania OLAP są w naturalny sposób przedstawiane w postaci wielowymiarowej (3 lub więcej wymiarów) - logiczny model wielowymiarowy
- Logiczne kostki stanowią sposób organizacji miar mających te same wymiary





Implementacje logicznego wielowymiarowego modelu danych

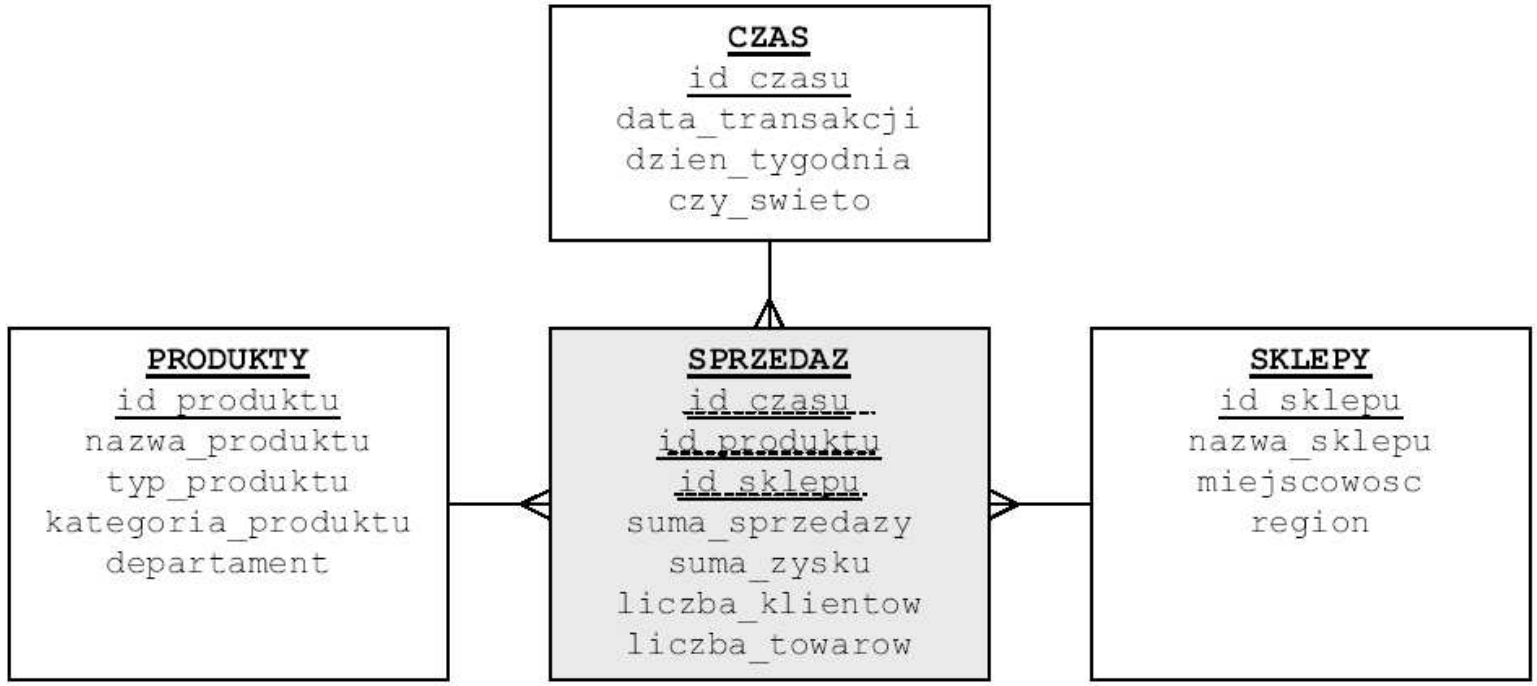
- **ROLAP** Relacyjna implementacja modelu
  - Powiązane ze sobą tabele relacyjne: tabele faktów i wymiarów
  - Schematy logiczne:
    - Schemat gwiazdy
    - Schemat płatka śniegu
    - Konstelacja faktów
  - Materializowane perspektywy dla agregatów
  - Logiczny model wielowymiarowy definiowany poprzez OLAP Catalog lub na poziomie aplikacji analitycznej
- **MOLAP** Wielowymiarowa reprezentacja modelu
  - Dane fizycznie składowane w postaci wielowymiarowej
  - W Oracle jako analityczne przestrzenie robocze (ang. Analytic Workspaces - AW)

Charakterystyka tabeli faktów i wymiarów

- **Tabela faktów:**
  - Zawiera numeryczne miary
  - Posiada wieloatrybutowy klucz główny złożony z kluczy obcych odwołujących się do wymiarów
  - Największy rozmiar spośród tabel tworzących gwiazdę (typowo zawiera ponad 90% danych)
  - Jej rozmiar szybko się powiększa
- **Tabele wymiarów:**
  - Zawierają atrybuty opisowe
  - Nadają znaczenie faktom (definiują przestrzeń faktów)
  - Zawierają dane rzadko podlegające zmianom (zmiany typu: pojawianie się nowych klientów, produktów)

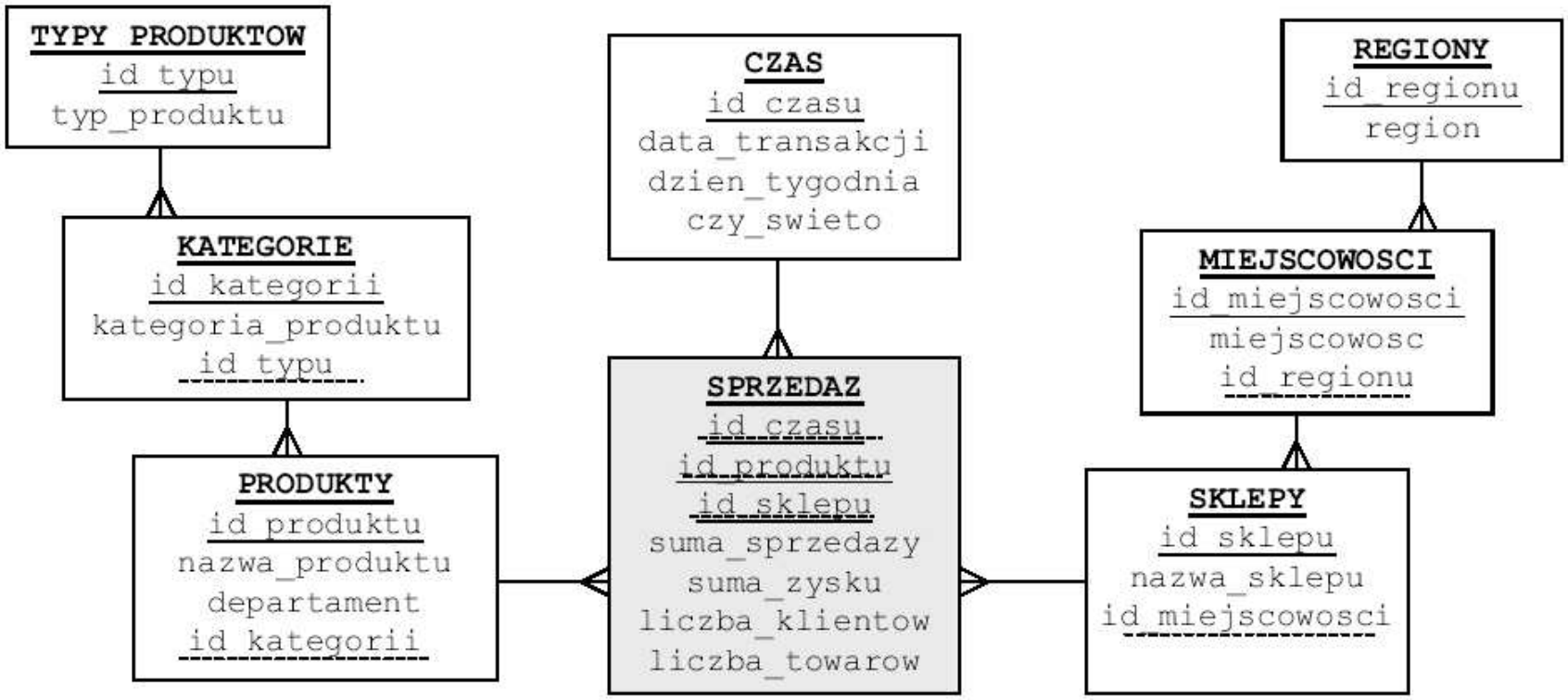
Schemat gwiazdy

- Centralna tabela faktów
- Wymiary zdenormalizowane
- Tabela faktów połączona z tabelami wymiarów poprzez klucze główne i klucze obce



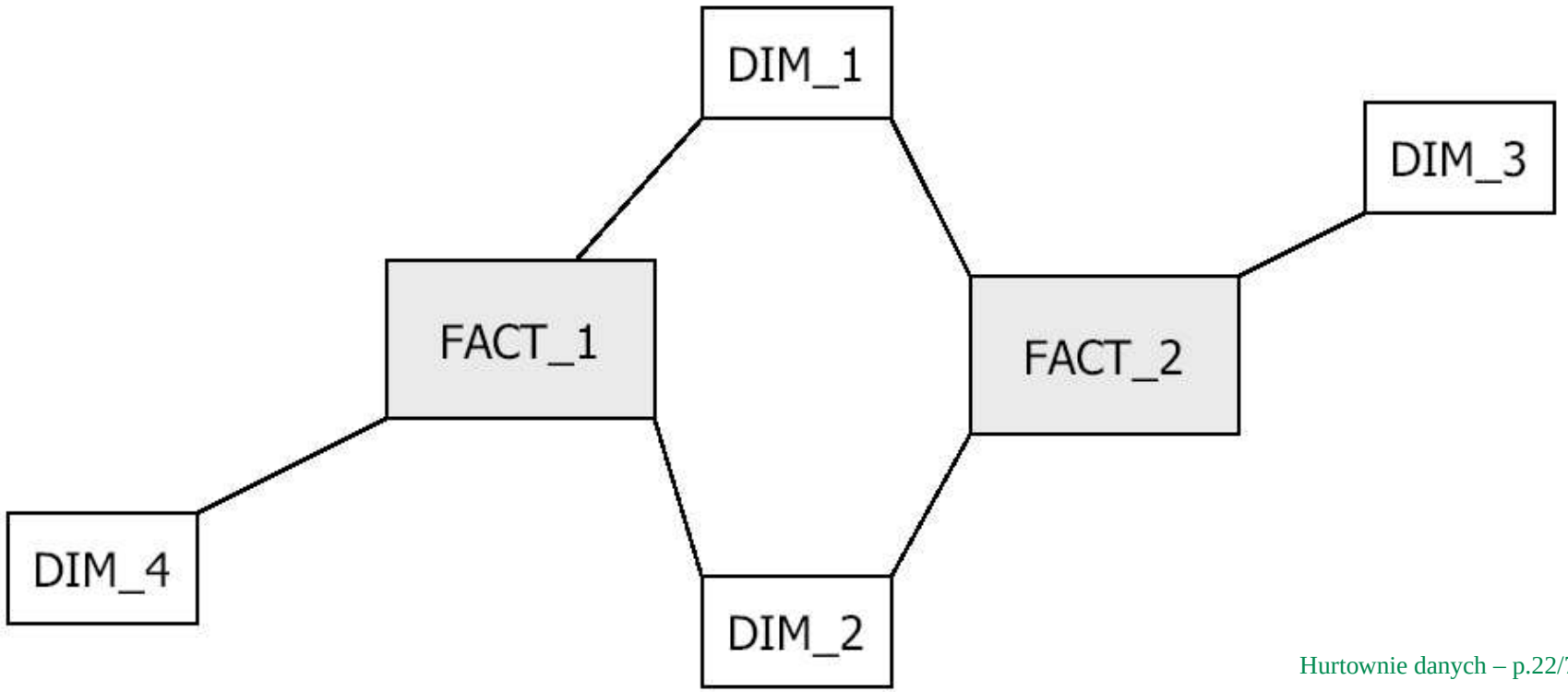
Schemat płatka śniegu

- Centralna tabela faktów
- Wymiary znormalizowane



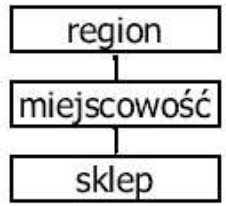
Schemat konstelacji faktów

- Schemat stanowiący kombinację schematów gwiazd współdzielących niektóre wymiary
  - Różne tabele faktów mogą odwoływać się do różnych poziomów danego wymiaru



Obiekt Dimension

Obiekty bazy danych wspierające organizację danych wymiarowych zawartych w tabelach wymiarów (zdenormalizowanych lub znormalizowanych) i grupujące informacje wymiarowe w hierarchie.




```
CREATE DIMENSION SCOTT.SKLEPY
LEVEL SKLEP IS SKLEPY.ID_SKLEPU
LEVEL MIEJSCOWOSC IS SKLEPY.MIEJSCOWOSC
LEVEL REGION IS SKLEPY.REGION
HIERARCHY H_SKLEPY (SKLEP CHILD OF
                    MIEJSCOWOSC CHILD OF
                    REGION)
ATTRIBUTE ...
```

```
ID NAZWA_SKLEPU MIEJSCOWOSC     REGION
-- ------------ --------------- ------
 1 Store No. 1  New York        East
 3 Store No. 3  Atlanta         East
 4 Store No. 4  Los Angeles     West
 5 Store No. 5  San Francisco   West
```

Tylko nowe wiersze

Test NOT NULL

```
EXECUTE DBMS_DIMENSION.VALIDATE_DIMENSION ('SCOTT.SKLEPY', FALSE, TRUE, 'test1');

SELECT * FROM dimension_exceptions WHERE statement_id = 'test1';
```

Perspektywy materializowane [1]

- Perspektywa materializowana to perspektywa, której zawartość jest fizycznie składowana w bazie danych - implementowana jako: tabela + indeks
- Zastosowania materializowanych perspektyw:
  - Hurtownie danych
  - Rozproszone bazy danych
  - Systemy mobilne

Perspektywy materializowane [2]

```
CREATE MATERIALIZED VIEW SCOTT.SPRZEDAZ_MV
  BUILD IMMEDIATE
  REFRESH COMPLETE
  ENABLE QUERY REWRITE
  AS
  SELECT id_sklepu, id_produktu,
    SUM(suma_sprzedazy) AS suma_sprzedana
    FROM sprzedaz
    GROUP BY id_sklepu, id_produktu;
```

Parametry perspektyw materializowanych [1]

Definicja perspektywy materializowanej obejmuje m.in.:

- Specyfikację zawartości (zapytanie SQL)
- Sposób odświeżania zawartości:
  - REFRESH FAST - szybkie, przyrostowe
  - REFRESH COMPLETE - pełne
  - REFRESH FORCE - FAST gdy możliwe, jeśli nie - COMPLETE
- Tryb odświeżania:
  - ON COMMIT - przy zatwierdzeniu transakcji modyfikującej którąś z tabel zródłowych (dla tabel lokalnych, dla metody FAST, wymagany przywilej ON COMMIT)
  - ON DEMAND - ręcznie (DBMS_MVIEW.REFRESH)

Parametry perspektyw materializowanych [2]

- Częstotliwość odświeżania (nie można gdy ON COMMIT/DEMAND)
  - START WITH - pierwsze odświeżenie
  - NEXT - interwał automatycznego odświeżania
- Możliwość wykorzystania do przepisywania zapytań
  - ENABLE/DISABLE QUERY REWRITE

Odświeżanie przyrostowe [1]

- Polega na inkrementalnym dodaniu do perspektywy zmian, jakie miały miejsce w danych źródłowych
- Aby odświeżanie przyrostowe było możliwe muszą być spełnione następujące ogólne warunki:
  - Zapytanie definiujące zawartość nie może zawierać:
    - Odwołań do niepowtarzalnych wyrażeń, np. SYSDATE, ROWNUM
    - Odwołań do kolumn typu RAW i LONG RAW
  - Na tabelach źródłowych muszą być założone dzienniki

Odświeżanie przyrostowe [2]

- Dodatkowo, szczególne warunki muszą być spełnione dla:
  - Perspektyw materializowanych wykorzystujących połączenie tabel
    - m.in. ROWID wszystkich tabel źródłowych w klauzuli SELECT zapytania perspektywy i ROWID w dziennikach
  - Perspektyw materializowanych zawierających agregaty
    - m.in. ROWID, INCLUDING NEW VALUES i wszystkie kolumny z perspektywy materializowanej w dzienniku, w klauzuli SELECT: COUNT(*), wszystkie kolumny GROUP BY, COUNT(expr) dla każdego agregatu np. dla SUM(expr)

Dziennik materializowanej perspektywy

```
CREATE MATERIALIZED VIEW sprzedaz_prod_mv
  BUILD IMMEDIATE
  REFRESH FAST ON COMMIT
  ENABLE QUERY REWRITE
  AS
  SELECT id_produktu, SUM(suma_sprzedazy) AS suma_sprzedana,
    COUNT(suma_sprzedazy) AS liczba_s, COUNT(*) AS liczba
    FROM sprzedaz
    GROUP BY id_produktu;

CREATE MATERIALIZED VIEW LOG ON sprzedaz
WITH SEQUENCE, ROWID(id_produktu, suma_sprzedazy)
INCLUDING NEW VALUES;
```

Przepisywanie zapytań [1]

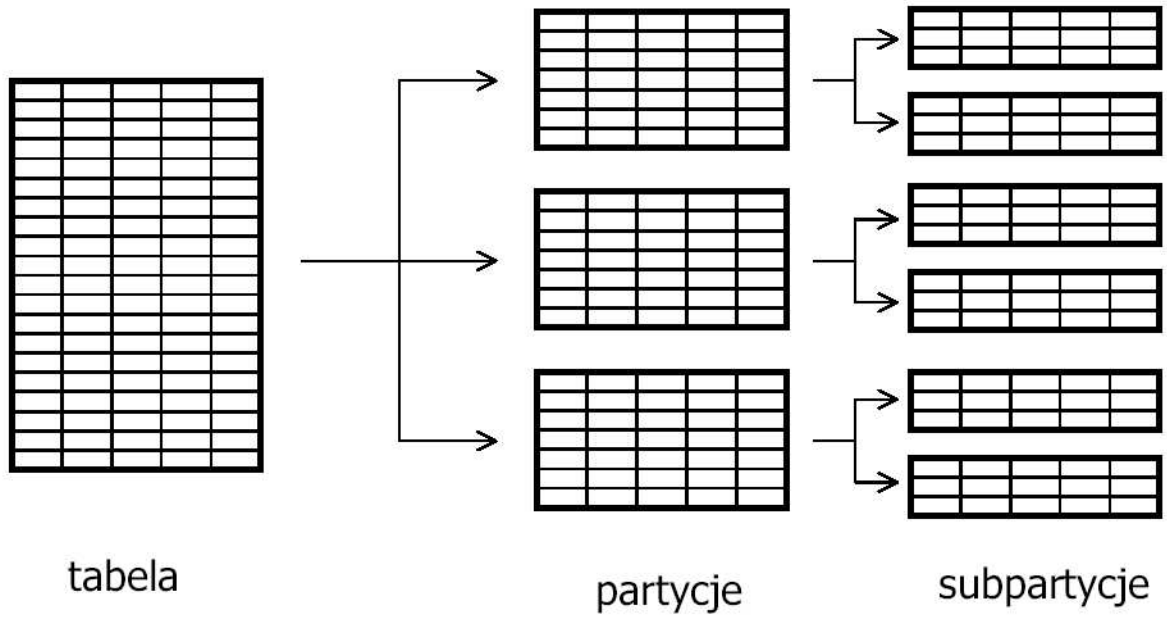
- Przepisywanie zapytań
  - Transformacja zapytania SQL odwołującego się do tabeli/perspektywy do zapytania korzystającego z perspektywy materializowanej
  - Mechanizm transparentny dla użytkownika - perspektywy materializowane mogą być tworzone/usuwane bez zmian w aplikacji
- Możliwość przepisania zapytania na perspektywę materializowaną zależy od kilku czynników
  - Włączenia mechanizmu przy tworzeniu perspektywy materializowanej
  - Poziomu integralności przepisywania zapytań
  - Dostępności więzów integralności i obiektów DIMENSION

Przepisywanie zapytań [2]

- Procedura DBMS_MVIEW.EXPLAIN_REWRITE:
  `DBMS_MVIEW.EXPLAIN_REWRITE('SELECT ... ','SCOTT.SPRZEDAZ_MV`
  - Sprawdza czy dana zapytanie może być przepisane
  - Podaje jakie perspektywy zostaną użyte
  - Wyjaśnia dlaczego dana perspektywa nie może być użyta do przepisania danego zapytania
  - Wyniki umieszcza w tabeli REWRITE_TABLE
- Przepisywanie zapytań - warunki konieczne
  - Uprawnienia użytkownika: przywilej obiektowy QUERY REWRITE lub systemowy GLOBAL QUERY REWRITE
  - Perspektywa materializowana z klauzulą ENABLE QUERY REWRITE
  - Mechanizm przepisywania włączony na poziomie systemu (QUERY_REWRITE_ENABLED=TRUE) lub sesji

Partycjonowanie

- Partycjonowane tabele i indeksy umożliwiają fizyczny podział danych na niewielkie, łatwe w zarządzaniu podzbiory, nazywane partycjami
- Każda partycja stanowi odrębny segment w bazie danych
- Partycje mogą być opcjonalnie dzielone na subpartycje
- Partycjonowanie umożliwia równoległą realizację poleceń DML



Metody partycjonowania

- **Partycjonowanie zakresowe** - rozdział rekordów pomiędzy partycje odbywa się według przynależności wartości kolumny-klucza do predefiniowanych przedziałów
- **Partycjonowanie haszowe** - rozdział rekordów odbywa się według wartości funkcji haszowej (modulo) wyliczanej dla kolumny-klucza
- **Partycjonowanie wg listy** - rozdział rekordów odbywa się według przynależności wartości kolumny-klucza do predefiniowanych list wartości
- **Partycjonowanie dwupoziomowe zakresowo-haszowe** - rozdział rekordów na partycje wg zakresów, a następnie na subpartycje wg wartości funkcji haszowej
- **Partycjonowanie dwupoziomowe zakresowo-listowe** - rozdział rekordów na partycje wg zakresów, a następnie na subpartycje wg przynależności do list wartości

Partycjonowanie zakresowe

```
CREATE TABLE pracownicy (
  id NUMBER(10),
  imie VARCHAR2(20), n
  azwisko VARCHAR2(20),
  pensja NUMBER(10,2)
)
PARTITION BY RANGE (pensja) (
  PARTITION male_pensje VALUES LESS THAN (1000),
  PARTITION srednie_pensje VALUES LESS THAN (10000),
  PARTITION duze_pensje VALUES LESS THAN (MAXVALUE)
)
```

Partycjonowanie haszowe

```
CREATE TABLE pracownicy (
  id NUMBER(10),
  imie VARCHAR2(20),
  nazwisko VARCHAR2(20),
  pensja NUMBER(10,2)
)
PARTITION BY HASH (id)
  PARTITIONS 3;
```

Partycjonowanie wg listy

```
CREATE TABLE pracownicy (
  id NUMBER(10),
  imie VARCHAR2(20),
  nazwisko VARCHAR2(20),
  pensja NUMBER(10,2)
)
PARTITION BY LIST (imie) (
  PARTITION p1 VALUES ('ADAM', 'JOANNA'),
  PARTITION p2 VALUES ('JERZY', 'ANNA'),
  PARTITION p3 VALUES ('MACIEJ', 'MARIA')
)
```

Partycjonowanie dwupoziomowe zakresowo-haszowe

```
CREATE TABLE pracownicy (
  id NUMBER(10),
  imie VARCHAR2(20),
  nazwisko VARCHAR2(20),
  pensja NUMBER(10,2)
)
PARTITION BY RANGE (pensja) SUBPARTITION BY HASH (id)
  SUBPARTITIONS 4 (
    PARTITION male_pensje VALUES LESS THAN (1000),
    PARTITION srednie_pensje VALUES LESS THAN (10000),
    PARTITION duze_pensje VALUES LESS THAN (MAXVALUE)
  )
```

Partycjonowanie dwupoziomowe zakresowo-listowe

```
CREATE TABLE pracownicy (
  id NUMBER(10),
  imie VARCHAR2(20),
  nazwisko VARCHAR2(20),
  pensja NUMBER(10,2)
)
PARTITION BY RANGE (pensja) SUBPARTITION BY LIST (imie) (
  PARTITION male_pensje VALUES LESS THAN (1000) (
    SUBPARTITION male_pensje_s1 VALUES ('ADAM'),
    SUBPARTITION male_pensje_s2 VALUES ('JOANNA')
  ),
  PARTITION srednie_pensje VALUES LESS THAN (10000) (
    SUBPARTITION srednie_pensje_s1 VALUES ('ADAM'),
    SUBPARTITION srednie_pensje_s2 VALUES ('JOANNA')
  ),
  PARTITION duze_pensje VALUES LESS THAN (MAXVALUE) (
    SUBPARTITION duze_pensje_s1 VALUES ('ADAM'),
    SUBPARTITION duze_pensje_s2 VALUES ('JOANNA')
  )
```

Partycje a przestrzenie tabel

```
CREATE TABLE pracownicy (
  id NUMBER(10),
  imie VARCHAR2(20),
  nazwisko VARCHAR2(20),
  pensja NUMBER(10,2)
)
PARTITION BY RANGE (pensja) (
  PARTITION male_pensje VALUES LESS THAN (1000) TABLESPACE ts1,
  PARTITION srednie_pensje VALUES LESS THAN (10000) TABLESPACE ts
  PARTITION duze_pensje VALUES LESS THAN (MAXVALUE) TABLESPACE ts
)
```

Operacje na tabelach partycjonowanych

- Odczyt partycji

```
select ... from <nazwa_tabeli> partition (<nazwa_partycji>)
```

- Definiowanie nowej partycji

```
alter table ... add partition ...
```

- Usuwanie partycji

```
alter table ... drop partition ...
```

- Wymiana partycji z tabelą

```
alter table ... exchange partition ... with ...
```

- Przenoszenie partycji do innej przestrzeni tabel

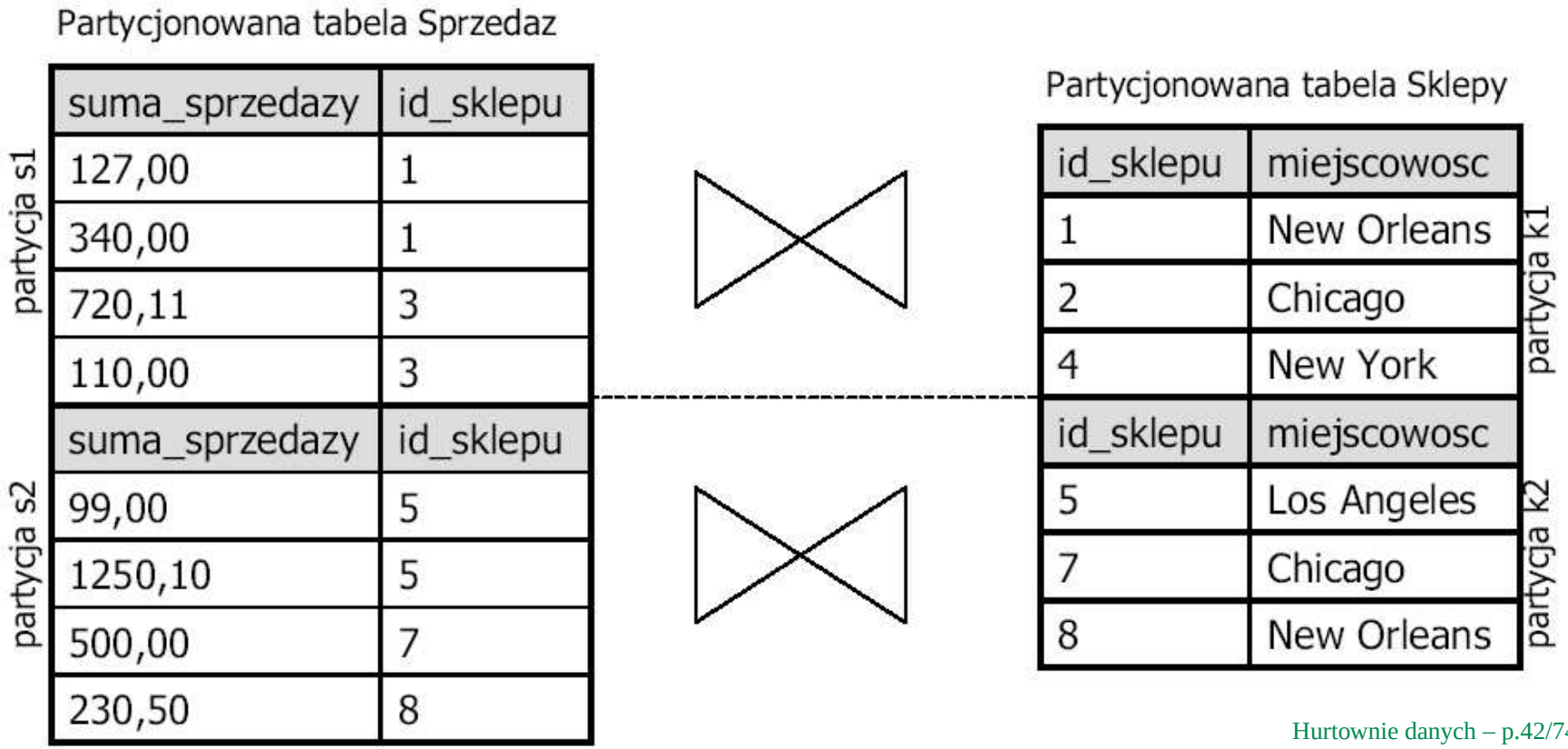
```
alter table ... move partition ... tablespace ...
```

- Podział partycji

```
alter table ... split partition ... at ... into
(partition ..., partition ...)
```

Połączenie partycji (Partition-wise Joins)

```
SELECT SUM(suma_sprzedazy)
  FROM sprzedaz NATURAL JOIN sklepy
  WHERE miejscowosc= 'New Orleans';
```



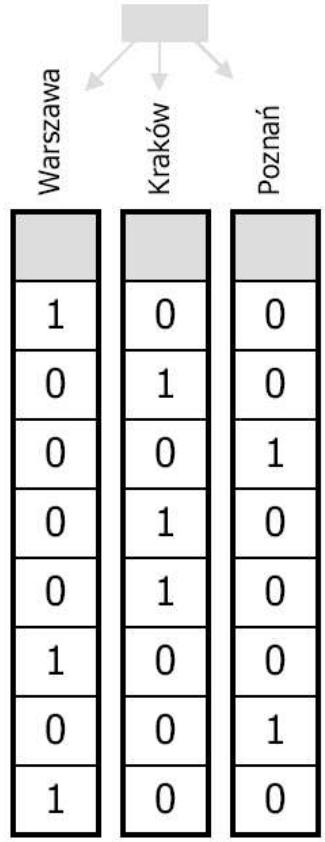
Indeks bitmapowy

```
CREATE BITMAP INDEX m_ind ON sprzedaz(miasto)
```

- Zalecany dla kolumn o relatywnie niskiej kardynalności
- Niewielki rozmiar nawet dla bardzo dużych tabel
- Niska efektywność operacji DML
- Możliwość odwzorowania operatorów AND, OR, NOT z zapytania

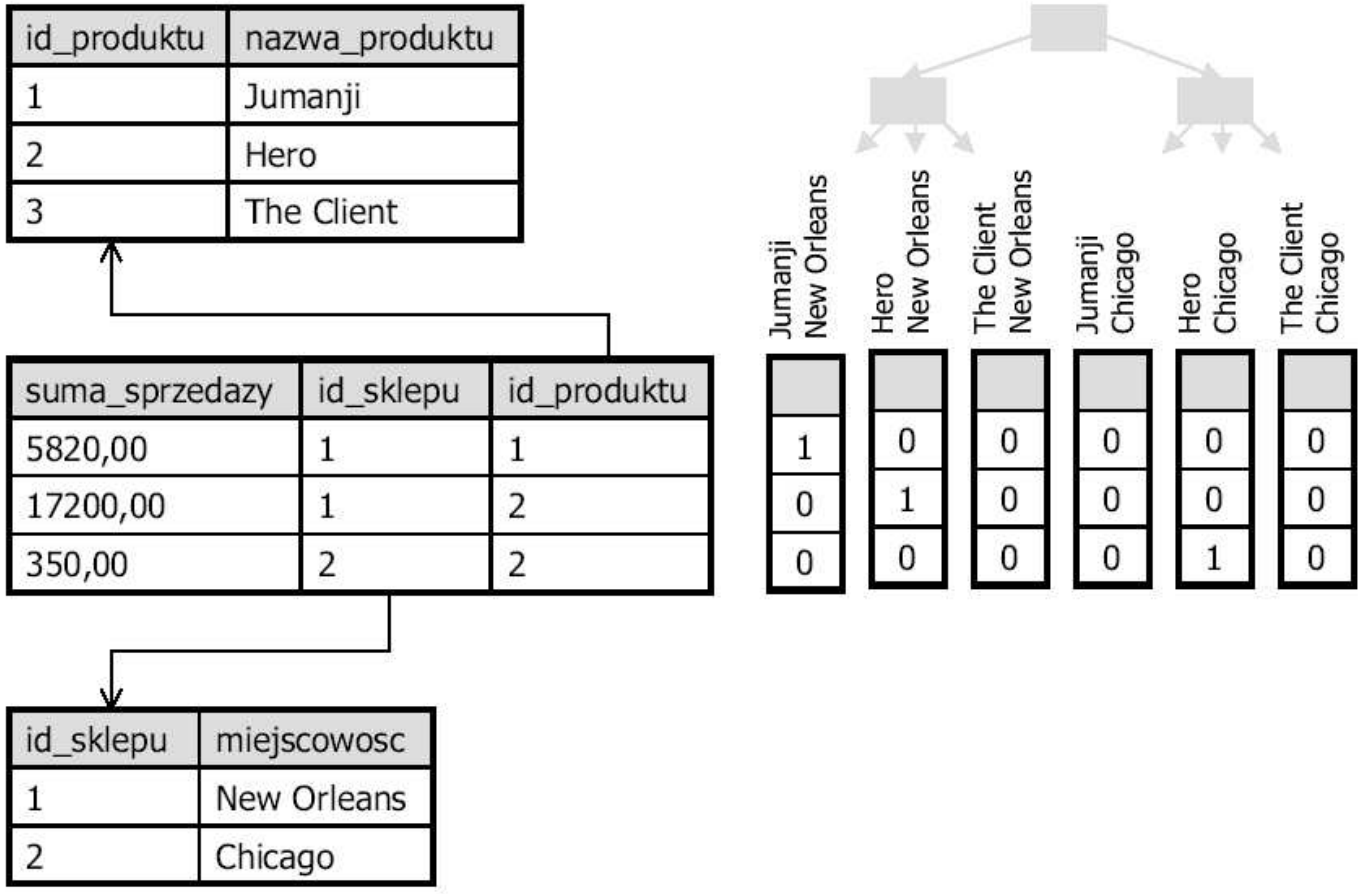
Sprzedaz

kwota	miasto
950,00	Warszawa
800,00	Kraków
755,00	Poznań
320,00	Kraków
110,00	Kraków
230,00	Warszawa
170,50	Poznań
190,00	Warszawa



Bitmapowy indeks połączeniowy [1]

```
CREATE BITMAP INDEX sprz_prod_sklep_ind ON sprzedaz(p.nazwa_produktu,k.miejscowos
  FROM sprzedaz s, sklepy k, produkty p
  WHERE s.id_sklepu=k.id_sklepu AND s.id_produktu=p.id_produktu
```



Bitmapowy indeks połączeniowy [2]

- Wg benchmarków Oracle, bitmapowy indeks połączeniowy może być nawet 8-krotnie szybszy od tradycyjnych metod indeksowania
- Z bitmapowego indeksu połączeniowego mogą korzystać tylko te zapytania, które w klauzuli WHERE nie odwołują się do kolumn nie objętych indeksem
- Ze względu na bardzo niską efektywność operacji DML, w praktyce indeksy takie usuwa się przed ładowaniem hurtowni danych, a następnie tworzy się je ponownie

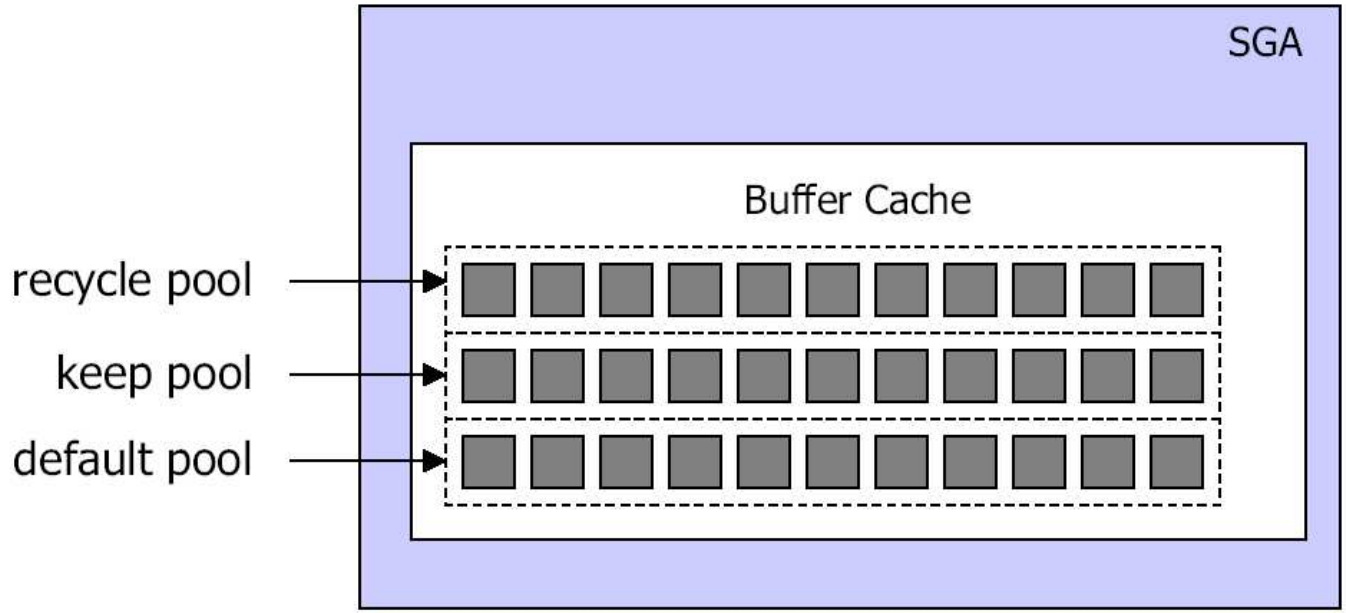
Rozpraszanie przestrzeni tabel

Przestrzenie tabel przechowujące obiekty hurtowni danych powinny być rozłożone pomiędzy wiele fizycznych urządzeń dyskowych.

- poprawa wydajności przetwarzania równoległego
- zalecenie Oracle: liczba fizycznych urządzeń dyskowych co najmniej równa liczbie procesorów
- manualna dystrybucja przestrzeni tabel:

```
CREATE TABLESPACE dw_1 DATAFILE
  'c:\oradata\dw_1_1.dbf'  SIZE 100M,
  'd:\oradata\dw_1_2.dbf'  SIZE 100M,
  'e:\oradata\dw_1_3.dbf'  SIZE 100M
```

Wielokrotny bufor danych



- Przydział obiektu bazy danych do obszaru bufora danych

```
ALTER TABLE sklepy STORAGE (BUFER_POOL KEEP)
```

Buforowanie tabel podczas operacji pełnego odczytu

- Gdy serwer bazy danych wykonuje operację pełnego odczytu tabeli, nie umieszcza jej bloków w buforze danych
- Wymuszenie buforowania tabel poddawanych operacjom pełnego odczytu (np. tabele wymiarów)
  `ALTER TABLE sklepy CACHE`
- Tabela faktów nie powinna być buforowana, ponieważ jej rozmiar zwykle dalece przekracza rozmiar dostępnej pamięci operacyjnej

Zapytania równoległe

Wydanie jednego z poniższych poleceń spowoduje, że zapytania wykorzystujące operację pełnego odczytu tabeli będą wykonywane równolegle z użyciem maks. 2 procesów.

```
ALTER TABLE sprzedaz PARALLEL 2
```

```
SELECT /*+ PARALLEL(sprzedaz,2) */ FROM sprzedaz
```

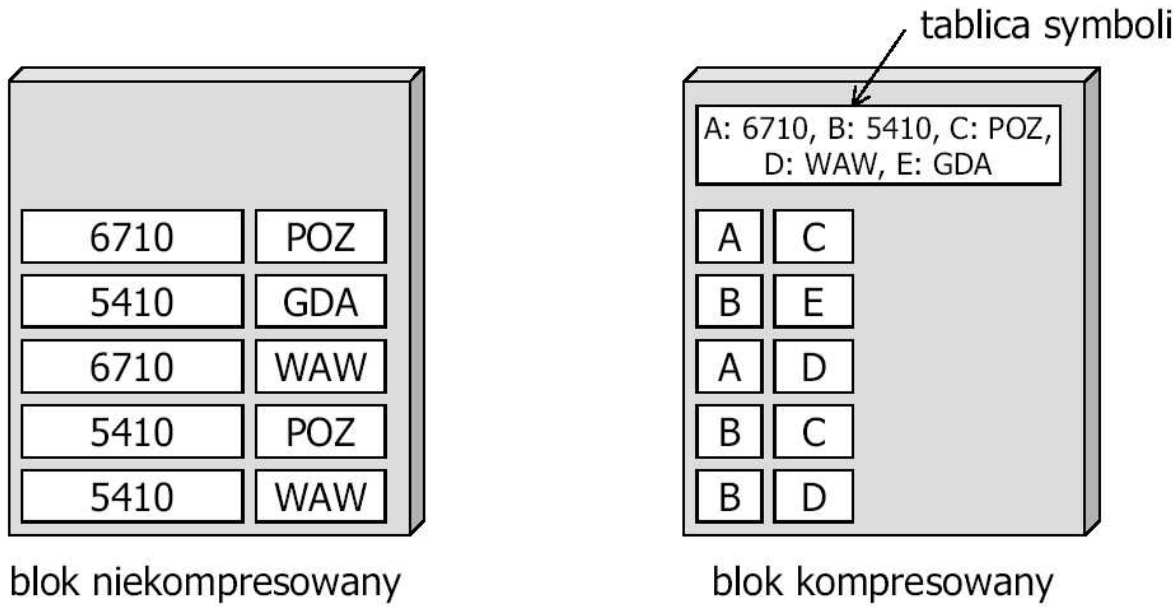
Różnicowanie wielkości bloków bazy danych

- Pliki przestrzeni tabel są logicznie podzielone na bloki, stanowiące jednostki wymiany w operacjach I/O
- Rozmiar bloku jest stały dla całej przestrzeni tabel, lecz może być różny dla różnych przestrzeni tabel
- Dla obiektów hurtowni danych, które podlegają operacjom pełnego odczytu (full scan) zaleca się stosowanie dużych rozmiarów bloków bazy danych, np. 32kB

```
CREATE TABLESPACE dw_1 DATAFILE
'/oradata/dw_1_1.dbf'  SIZE 100M BLOCKSIZE 16K
```

- Duże bloki danych umożliwiają uzyskanie wyższego stopnia kompresji
- Ustawienie PCTFREE>0 pozwala uniknąć tzw. migracji rekordu w przypadku, gdy modyfikacja rekordu powoduje zwiększenie jego rozmiaru (tabela faktów nie jest modyfikowana - zaleca się stosowanie PCTFREE=0)

Kompresja bloków tabeli




```
CREATE TABLE ... COMPRESS
```

- Każdy blok jest kompresowany niezależnie
- Kompresja następuje podczas operacji: ładowania danych ścieżką bezpośrednią SQL*Loadera, CREATE TABLE AS SELECT, zrównoleglonego INSERT, ALTER TABLE MOVE

Procesy ETL

- Ekstrakcja danych źródłowych
- Transformacja i czyszczenie danych źródłowych
- Indeksowanie i podsumowywanie danych
- Ładowanie danych do hurtowni danych
- Wykrywanie zmian w danych źródłowych
- Odświeżanie danych

Źródła danych

- **Produkcyjne** - systemy operacyjne, operacyjne bazy danych (IMS, DB2, Oracle, Sybase, Informix), systemy plików, dedykowane aplikacje (SAP, PeopleSoft, Oracle Financials)
- **Zarchiwizowane** - dane historyczne, potrzebne do inicjalizacji hurtowni, mogą wymagać unikalnej transformacji
- **Zewnętrzne** - komercyjne bazy danych, Internet, problemy związane z formatem, częstotliwością odświeżania, przewidywalnością
- **Wewnętrzne** - wewnętrzne bazy danych, dokumenty, arkusze kalkulacyjne

Techniki ekstrakcji

- Programy w C, C++, COBOL, Java, PL/SQL
- Bramy - Oracle Transparent Gateways (MS SQL Server, DB2, Informix, Sybase, Ingres, Teradata, AS/400, MQSeries)
- Narzędzia do ekstrakcji - `http://www.dbmsmag.com/9706d16.html`
- Własne rozwiązania

Transformacja

- Najtrudniejszy element ETL, zazwyczaj wymaga istnienia obszaru tymczasowego (ang. staging area), w którym następuje konsolidacja, czyszczenie, restrukturyzacja
- Techniki
  - czyszczenie
  - eliminacja niespójności
  - dodawanie i łączenie danych
  - integracja danych

Anomalie i problemy

- Nazewnictwo i kodowanie
- Semantyka danych
- Błędy literowe
- Brak unikalnych kluczy globalnych
- Klucze złożone
- Różne formaty danych wejściowych
- Wiele sprzecznych źródeł danych
- Brakujące bądź zduplikowane wartości
- Brakujące więzy referencyjne

Ładowanie danych

- Proces przesyłania danych z obszaru tymczasowego do docelowej hurtowni danych w czasie:
  - inicjalizacji hurtowni: bardzo duża ilość danych, unikalne procesy transformacji, znaczące przetwarzanie po załadowaniu
  - odświeżanie hurtowni: wykonywane wg. cyklów biznesowych, prostsze procesy transformacji i przetwarzania, mniejsza objętość danych
- Metody ładowania danych
  - Specjalizowane narzędzia
  - Własne programy
  - Bramy
  - Replikacja synchroniczna i asynchroniczna
  - Ręczne ładowanie

Przetwarzanie po załadowaniu

- Indeksowanie danych
- Odtwarzanie kluczy
- Tworzenie perspektyw materializowanych
- Sprawdzanie spójności danych

ETL w środowisku Oracle

- Podstawowe narzędzie: Oracle Warehouse Builder
- Programy narzędziowe: Export/Import, Data Pump, SQL*Loader
- Zaawansowane elementy SZBD
  - przenaszalne przestrzenie tabel (transportable tablespaces)
  - mechanizm strumieni (Oracle Streams)
  - tabele zewnętrzne
  - zaawansowane elementy SQL (wstawianie wielotablicowe, instrukcja MERGE, funkcje tablicowe, równoległy DML, perspektywy materializowane)

Wprowadzenie do funkcji analitycznych [1]

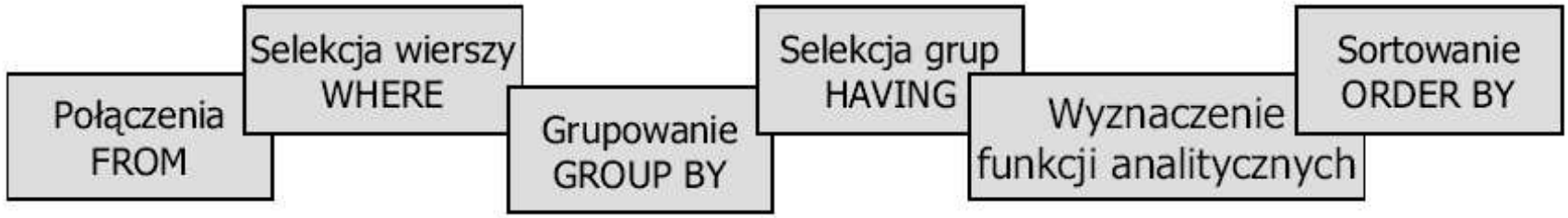
- Funkcje analityczne stanowią analityczne rozszerzenie funkcji SQL
- Podstawowy podział funkcji analitycznych:
  - **funkcje rankingu** - wykorzystywane do wyznaczania rankingów, podziałów zbiorów wierszy na grupy (n-tki) (RANK, DENSE_RANK, CUME_DIST, PERCENT_RANK, NTILE, ROWNUMBER)
  - **funkcje okna** - wyznaczają wartości agregatów dla zbiorów wierszy wyznaczanych przy użyciu definicji okna (MAX, MIN, AVG, SUM, COUNT, STDDEV, VARIANCE, FIRST_VALUE, LAST_VALUE)
  - **funkcje raportujące** - wyznaczają wartości agregatów dla zbiorów wierszy w ramach tzw. partycji (MAX, MIN, AVG, SUM, COUNT, STDDEV, VARIANCE, RATIO_TO_REPORT)
  - **funkcje LAG/LEAD** - znajdują wartości określonych atrybutów w wierszach sąsiednich

Wprowadzenie do funkcji analitycznych [2]

- Podstawowy podział funkcji analitycznych:
  - **funkcje FIRST/LAST** - znajdują początkową lub końcową wartość w uporządkowanym zbiorze
  - **odwrotne funkcje procentyli** - wyznaczają wartość występującą w określonym miejscu w uporządkowanym zbiorze (PERCENTILE_DISC, PERCENTILE_CONT)
  - **funkcje rankingu hipotetycznego** - wyznaczają hipotetyczny ranking zadanych wartości w uporządkowanym zbiorze (RANK, DENSE_RANK, CUME_DIST, PERCENT_RANK)
  - **funkcja WIDTH_BUCKET** - dzieli uporządkowany zbiór na określoną liczbę przedziałów o zadanej szerokości
  - **funkcje statystyczne** - wyliczają zmiany poziomów i inne statystyki (EXPONENTIAL_DIST_FIT, NORMAL_DIST_FIT, POISSON_DIST_FIT, WEIBULL_DIST_FIT, SUMMARY)

Miejsce funkcji analitycznych [1]

- Funkcje analityczne wyznaczając wynik dla bieżącego wiersza z reguły korzystają z informacji znajdujących się w wierszach sąsiednich
- Wartości funkcji analitycznych są wyliczane po wszystkich składowych operacjach (połączeniach, selekcji wierszy, grupowaniu, selekcji grup itd.)
- Po wyznaczeniu wyników funkcji analitycznych wykonywana jest jedynie operacja sortowania



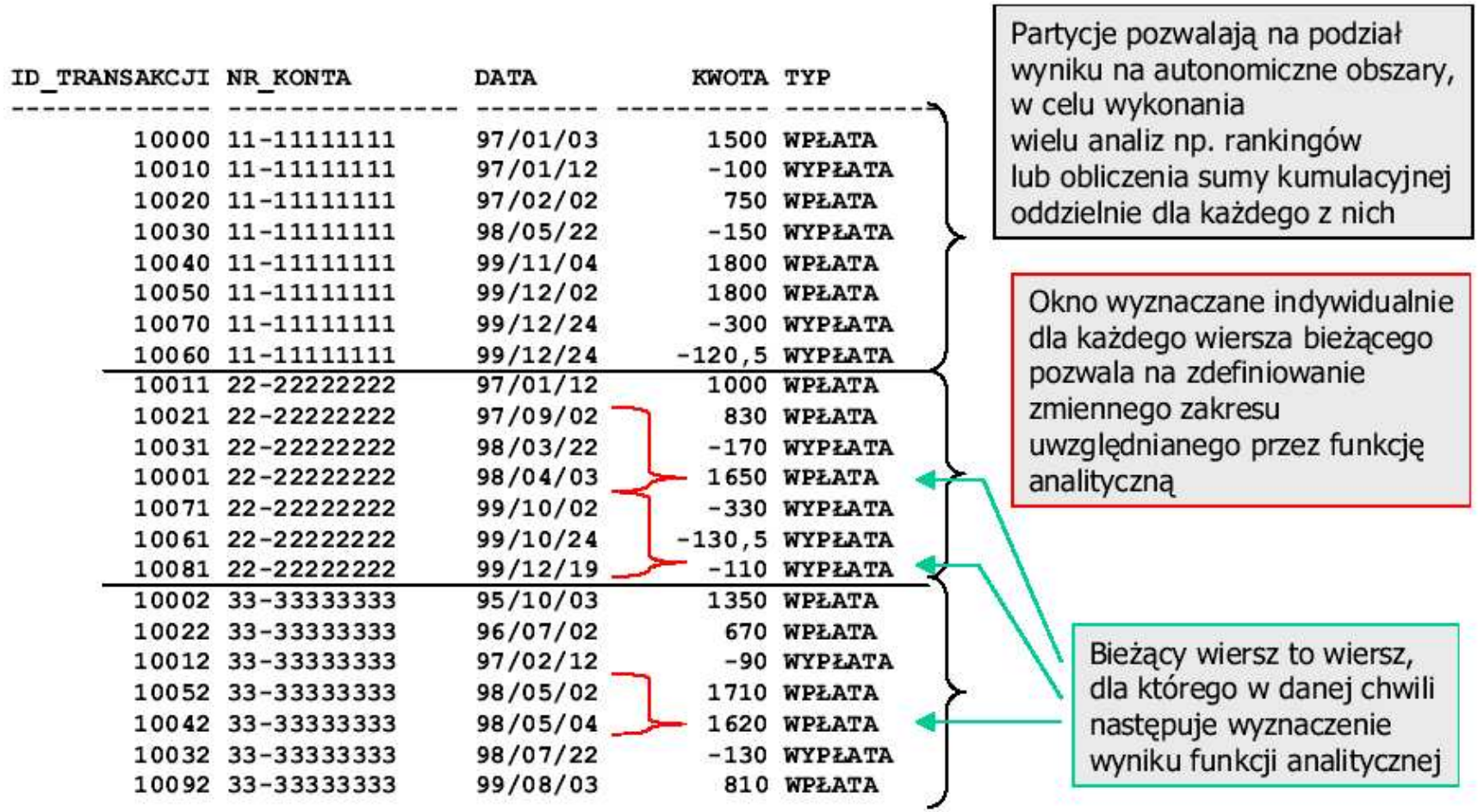
Miejsce funkcji analitycznych [2]

- Ściśle określone miejsce wyznaczania wartości funkcji analitycznych ma swoje znaczące konsekwencje:
  - nie mogą być używane w klauzulach WHERE, GROUP BY, HAVING
  - wykorzystywane są tylko i wyłącznie w klauzuli SELECT lub ORDER BY
  - działają tylko i wyłącznie na wierszach, grupach będących efektem finalnym zapytania
  - wiersze lub grupy odrzucone za pomocą klauzul WHERE lub HAVING nie są uwzględniane przez funkcje analityczne

Partycje, okna, wiersz bieżący [1]

- **Partycje** - umożliwiają podział rezultatu zapytania na autonomiczne, niezależne zbiory, w ramach których funkcje analityczne będą mogły wyznaczać oddzielne rankingi, średnie itp.
- **Okna** - występują tylko w przypadku funkcji okna. Pozwalają na zdefiniowane ruchomego zakresu, określanego indywidualnie dla każdego wiersza, w ramach którego funkcja będzie wyznaczała swoją wartość
- **Bieżący wiersz** - wiersz, dla którego w danym momencie wyznaczany jest wynik funkcji analitycznej. W szczególności stanowi on punkt odniesienia przy wyznaczaniu zakresu okna

Partycje, okna, wiersz bieżący [2]



ID_TRANSAKCJI	NR_KONTA	DATA	KWOTA	TYP
10000	11-11111111	97/01/03	1500	WPŁATA
10010	11-11111111	97/01/12	-100	WYPŁATA
10020	11-11111111	97/02/02	750	WPŁATA
10030	11-11111111	98/05/22	-150	WYPŁATA
10040	11-11111111	99/11/04	1800	WPŁATA
10050	11-11111111	99/12/02	1800	WPŁATA
10070	11-11111111	99/12/24	-300	WYPŁATA
10060	11-11111111	99/12/24	-120,5	WYPŁATA
10011	22-22222222	97/01/12	1000	WPŁATA
10021	22-22222222	97/09/02	830	WPŁATA
10031	22-22222222	98/03/22	-170	WYPŁATA
10001	22-22222222	98/04/03	1650	WPŁATA
10071	22-22222222	99/10/02	-330	WYPŁATA
10061	22-22222222	99/10/24	-130,5	WYPŁATA
10081	22-22222222	99/12/19	-110	WYPŁATA
10002	33-33333333	95/10/03	1350	WPŁATA
10022	33-33333333	96/07/02	670	WPŁATA
10012	33-33333333	97/02/12	-90	WYPŁATA
10052	33-33333333	98/05/02	1710	WPŁATA
10042	33-33333333	98/05/04	1620	WPŁATA
10032	33-33333333	98/07/22	-130	WYPŁATA
10092	33-33333333	99/08/03	810	WPŁATA

Składnia funkcji analitycznych

```
NAZWA FUNKCJI ANALITYCZNEJ ( parametry - różne dla różnego rodza
OVER ( definicja partycji - opcjonalna
       definicja porządku wierszy w partycji - zależna od typu f
       definicja okna - tylko w przypadku funkcji okna)

RANK() OVER ( partition by numer_konta
              order by kwota)
AVG() OVER ( partition by numer_konta
             order by data
             ROWS UNBOUNDED PRECEDING)
LEAD(sum(kwota)) OVER ( partition by numer_konta
                        order by data)
```

Rozszerzenia grupowania danych

Polecenia poszerzające grupowanie

- ROLLUP
- CUBE
- GROUPING SETS
- GROUPING

ROLLUP

- Polecenie ROLLUP jest rozszerzeniem klauzuli GROUP BY, które pozwala wyliczać dodatkowe podsumowania częściowe i ogólne. Polecenie ROLLUP jest wyrażeniem wyjątkowo wydajnym. Dla n-kolumn grupujących, ROLLUP tworzy n + 1 podsumowań.
- Dodatkowe podsumowania wyznaczone przez ROLLUP wyliczane są przez eliminowanie kolejno kolumn grupujących począwszy od ostatniej a skończywszy na pierwszej
- Przykładowo klauzula GROUP BY ROLLUP(kontynent, kraj, miejscowosc , dzielnica) wyznaczy następujące podsumowania:
  - GROUP BY kontynent, kraj, miejscowosc, dzielnica
  - GROUP BY kontynent, kraj, miejscowosc
  - GROUP BY kontynent, kraj
  - GROUP BY kontynent
  - bez GROUP BY - podsumowanie całkowite

CUBE

- Operator CUBE tworzy podsumowania dla wszystkich możliwych kombinacji grupowanych kolumn. W terminologii analiz wielowymiarowych, CUBE generuje podsumowania częściowe i ogólne tabeli faktów dla wszystkich możliwych wymiarów. Dla n-kolumn grupujących CUBE tworzy 2^n podsumowań.
- Przykładowo klauzula GROUP BY CUBE(kontynent, kraj, miejscowosc) wyznaczy następujące podsumowania:
  - GROUP BY kontynent, kraj, miejscowosc
  - GROUP BY kontynent, kraj,
  - GROUP BY kontynent, miejscowosc
  - GROUP BY kraj, miejscowosc
  - GROUP BY kontynent
  - GROUP BY kraj
  - GROUP BY miejscowosc
  - bez GROUP BY - podsumowanie całkowite

GROUPING SETS

- Zarówno operacja CUBE jak i ROLLUP może wyznaczać oprócz pożądanych podsumowań, także te, które są zbędne
- Operacja GROUPING SETS pozwala na jednoznaczne wskazanie tych podsumowań, które chcemy uzyskać
- Dla przykładu

```
GROUPING SETS( (nr_konta, typ, kategoria),
               (nr_konta, typ),
               (nr_konta, kategoria) )
```

  wyznaczy następujące podsumowania:

  - GROUP BY nr_konta, typ, kategoria
  - GROUP BY nr_konta, typ
  - GROUP BY nr_konta, kategoria

Funkcja GROUPING

- Oparcie się na wartościach pustych występujących w kolumnach grupowania może prowadzić do błędów
- Funkcja GROUPING posiadająca jako argument wyrażenie występujące w klauzuli GROUP BY jednoznacznie wskazuje na "zwinięcie" danego wymiaru. Wartość funkcji 1 wskazuje na "zwinięcie", wartość 0 na "normalny" wiersz jaki pojawiłby się bez wykorzystania rozszerzeń grupowania
- Funkcję GROUPING bardzo często wykorzystuje się także do zastąpienia pustej wartości bardziej znaczącą informacją

Użycie funkcji GROUPING [1]

```
SELECT nr_konta, typ, grouping(nr_konta) gnr, grouping(typ) gt, sum(kwota)
  FROM transakcje
  GROUP BY grouping sets((nr_konta, typ),(typ),(nr_konta), ())
  ORDER BY nr_konta, typ

NR_KONTA      TYP         GNR         GT          SUM(KWOTA)
------------ ---------- ---------- ---------- ----------
11-11111111   WPŁATA     0          0          9800
11-11111111   WYPŁATA    0          0          -2320,5
11-11111111              0          1          7479,5
22-22222222   WPŁATA     0          0          11900
22-22222222   WYPŁATA    0          0          -2400,5
22-22222222              0          1          9499,5
33-33333333   WPŁATA     0          0          8900
33-33333333   WYPŁATA    0          0          -2030,5
33-33333333              0          1          6869,5
              WPŁATA     1          0          30600
              WYPŁATA    1          0          -6751,5
                         1          1          23848,5
```

Użycie funkcji GROUPING [2]

```
SELECT decode(grouping(nr_konta),0,nr_konta,'Wszystkie konta') nr_konta,
       decode(grouping(typ),0,typ,'Wszystkie typy') typ, sum(kwota)
       FROM TRANSAKCJE
       GROUP BY grouping sets((nr_konta, typ),(typ),(nr_konta), ())
```

NR_KONTA	TYP	SUM(KWOTA)
11-11111111	WPŁATA	9800
22-22222222	WPŁATA	11900
33-33333333	WPŁATA	8900
11-11111111	WYPŁATA	-2320,5
22-22222222	WYPŁATA	-2400,5
33-33333333	WYPŁATA	-2030,5
Wszystkie konta	WPŁATA	30600
Wszystkie konta	WYPŁATA	-6751,5
11-11111111	Wszystkie typy	7479,5
22-22222222	Wszystkie typy	9499,5
33-33333333	Wszystkie typy	6869,5
Wszystkie konta	Wszystkie typy	23848,5

Materiały do poszerzenia wiedzy:

- *Building the Data Warehouse*, W. H. Inmon
- *The Data Warehouse Toolkit*, R. Kimball
- *Data Warehouse from Architecture to Implementation*, B. Devlin
- *Data Warehousing in the Real World*, S. Anahory, D. Murray
- http://www.ploug.org.pl/ - materiały szkoleniowe