

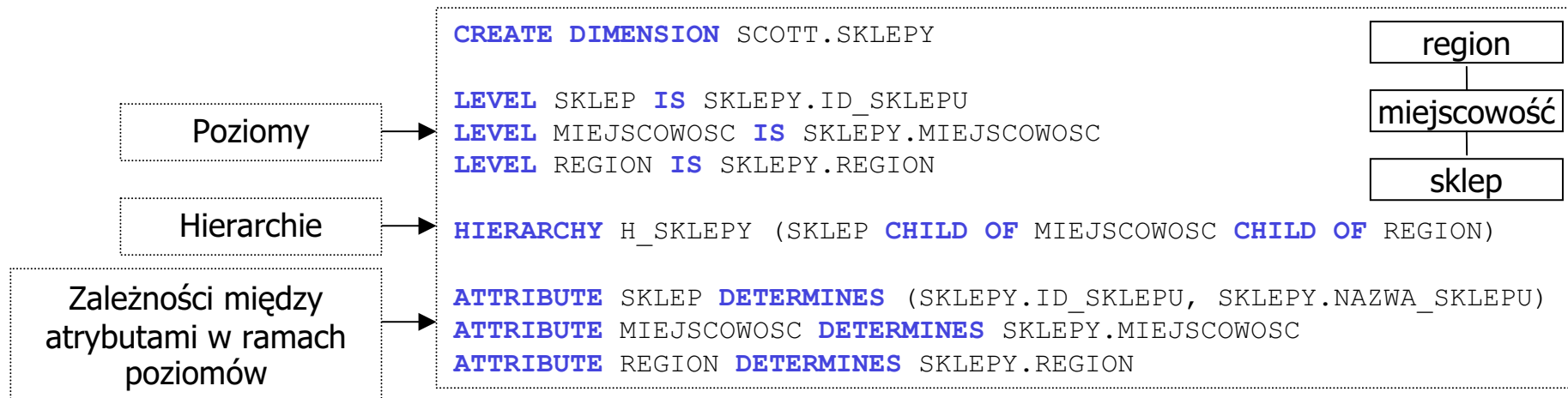
# **Oracle10g Database: struktury fizyczne dla hurtowni danych**

# Plan rozdziału

- Obiekty DIMENSION
- Materializowane perspektywy
- Przepisywanie zapytań
- Partycjonowanie tabel i indeksów
- Indeksy bitmapowe

# Obiekty DIMENSION

- Obiekty bazy danych wspierające organizację danych wymiarowych zawartych w tabelach wymiarów (zdenormalizowanych lub znormalizowanych) i grupujące informacje wymiarowe w hierarchie
- Zastosowania obiektów DIMENSION
  - Wspierają mechanizm przepisывania zapytań w oparciu o MV
  - Wspierają niektóre typy odświeżania MV
  - Wykorzystywane przez SQLAccess Advisor
  - Umożliwiają walidację struktury wymiaru (programowo)



- Operacje: ALTER DIMENSION, DROP DIMENSION, procedura DBMS\_DIMENSION.DESCRIBE\_DIMENSION, ...

# Walidacja wymiaru

- Organizacja danych zdefiniowana w obiekcie DIMENSION nie jest automatycznie wymuszana przez b.d.
- Walidację wymiaru można przeprowadzić programowo:
  - Wywołanie DBMS\_DIMENSION.VALIDATE\_DIMENSION
  - Błędy w tabeli DIMENSION\_EXCEPTIONS (utldim.sql)

<u>SKLEPY</u>
id_sklepu
nazwa_sklepu
miejscowosc
region



```

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

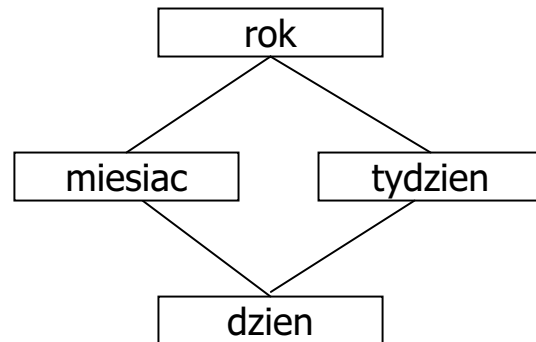
Test NOT NULL

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

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

# Wymiary z wieloma hierarchiami

- Dla danego wymiaru można zdefiniować kilka hierarchii



```
CREATE DIMENSION CZAS_FISK  
  
LEVEL DZIEN IS CZAS_FISKALNY.DATA  
LEVEL TYDZIEN IS CZAS_FISKALNY.TYDZIEN  
LEVEL MIESIAC IS CZAS_FISKALNY.MIESIAC  
LEVEL ROK IS CZAS_FISKALNY.ROK  
  
HIERARCHY ROLLUP_MIESIAC (DZIEN CHILD OF  
MIESIAC CHILD OF  
ROK)  
  
HIERARCHY ROLLUP _TYDZIEN (DZIEN CHILD OF  
TYDZIEN CHILD OF  
ROK)
```

# Perspektywy materializowane

- **Perspektywa materializowana** (ang. materialized view) to perspektywa, której zawartość jest fizycznie składowana w bazie danych - implementowana jako: tabela + indeks
- Zastosowania materializowanych perspektyw
  - Hurtownie danych (jako summaries)
    - Perspektywy materializowane zawierające wstępnie wyliczone agregaty
    - Perspektywy mat. zawierające tylko połączenia (bez agregacji)
    - Perspektywy materializowane zagnieżdżone
  - Rozproszone bazy danych
  - Systemy mobilne

```
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

- 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 źródłowych (dla tabel lokalnych, dla metody FAST, wymagany przywilej ON COMMIT)
    - **ON DEMAND** – ręcznie (DBMS\_MVIEW.REFRESH)
  - 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

- Polega na inkrementalnej aplikacji do perspektywy zmian, które 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 tabeli (tabelach) źródłowych musi (muszą) być założony(e) dzienniki (ang. materialized view log)
- Dodatkowo, szczególne warunki muszą być spełnione dla:
  - Perspektyw mat. wykorzystujących połączenie tabel
    - m.in. ROWID wszystkich tabel źródłowych w klauzuli SELECT zapytania perspektywy i ROWID w dziennikach
  - Perspektyw mat. zawierających agregaty
    - m.in. ROWID, INCLUDING NEW VALUES i wszystkie kolumny z MV w dzienniku, w klauzuli SELECT: COUNT(\*), wszystkie kolumny GROUP BY, COUNT(expr) dla każdego agregatu np. dla SUM(expr)

# Dziennik materializowanej perspektywy

- Zawiera historię zmian w tabeli na potrzeby przyrostowego odświeżania materializowanych perspektyw
  - Wiersze identyfikowane przez ROWID lub PRIMARY KEY
  - Opcjonalnie może zawierać wartości innych kolumn
  - Opcjonalnie oprócz starych wartości również nowe
  - Opcjonalnie sekwencyjny numer porządkujący operacje

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

```
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;
```

# Konwersja tabeli na materializowaną perspektywę

- Operacja określana jako rejestracja materializowanej perspektywy
  - Dane z istniejącej tabeli staną się dostępne dla mechanizmu przepisania zapytań
- Mechanizm użyteczny gdy:
  - Złożone obliczeniowo podsumowania zostały już wyznaczone i umieszczone w tabeli
  - Do wyznaczania zawartości mają być wykorzystywane mechanizmy niedostępne przy tworzeniu perspektyw materializowanych np. wielotablicowe wstawianie (multi-INSERT)

```
CREATE TABLE SPRZEDAZ_SUM  
AS  
SELECT id_sklepu, id_produktu,  
SUM(suma_sprzedazy) AS suma_sprzedana  
FROM sprzedaz  
GROUP BY id_sklepu, id_produktu;
```

MV musi mieć taką  
samą nazwę jak tabela

```
CREATE MATERIALIZED VIEW SPRZEDAZ_SUM  
ON PREBUILT TABLE  
WITHOUT REDUCED PRECISION  
ENABLE QUERY REWRITE AS  
SELECT id_sklepu, id_produktu,  
SUM(suma_sprzedazy) AS suma_sprzedana  
FROM sprzedaz  
GROUP BY id_sklepu, id_produktu;
```

# Przepisywanie zapytań

- **Przepisywanie zapytań** (ang. **query rewrite**)
  - 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
  - Czy mechanizm włączony: param. inicjalizacyjne, CREATE MV, wskazówki
  - Poziomy integralności przepisywania zapytań (ang. query rewrite integrity)
  - Dostępność więzów integralności i obiektów DIMENSION
- Procedura DBMS\_MVIEW.EXPLAIN\_REWRITE
  - Sprawdza czy dane zapytanie może być przepisane
  - Podaje jakie perspektywy zostaną użyte
  - Wyjaśnia dlaczego dana perspektywa nie może być użyta do przepisania danego zapytania

```
DBMS_MVIEW.EXPLAIN_REWRITE('SELECT ...', 'SCOTT.SPRZEDAZ_MV', 'ID')
```

  - Wyniki w tabeli REWRITE\_TABLE (utlxrw.sql)

# Przepisywanie zapytań - Przykład

- Wykonanie zapytania bez perspektywy materializowanej

```
SELECT id_sklepu, id_produktu, SUM(suma_sprzedazy) AS suma_sprzedana
FROM sprzedaz
GROUP BY id_sklepu, id_produktu;
```

Plan wykonywania

```
-----
0          SELECT STATEMENT Optimizer=ALL_ROWS (Cost=91 Card=1995 Bytes=23940)

1   0      SORT (GROUP BY) (Cost=91 Card=1995 Bytes=23940)
2   1      TABLE ACCESS (FULL) OF 'SPRZEDAZ' (TABLE) (Cost=84 Card=86205 Bytes=1034460)
```

- To samo zapytanie z perspektywą materializowaną

```
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;
```

Plan wykonywania

```
-----
0          SELECT STATEMENT Optimizer=ALL_ROWS (Cost=4 Card=2818 Bytes=109902)

1   0      MAT_VIEW REWRITE ACCESS (FULL) OF 'SPRZEDAZ_MV' (MAT_VIEW REWRITE)
          (Cost=4 Card=2818 Bytes=109902)
```

# 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
- Optymalizator kosztowy (ważne w Oracle9i)

(!) Na możliwość przepisania zapytania wpływa również ustawiony poziom integralności przepisywania zapytań (**QUERY\_REWRITE\_INTEGRITY**, na poziomie systemu/sesji):  
ENFORCED | STALE\_TOLERATED | TRUSTED

# QUERY\_REWRITE\_INTEGRITY

- **ENFORCED** (domyślnie)
  - Wykorzystane tylko aktualne dane i związki oparte o więzy (klucze główne, unikalne, obce) z klauzulą `ENABLE VALIDATE`
- **TRUSTED**
  - Wykorzystane obiekty `DIMENSION` i więzy `NOVALIDATE RELY`  
`ALTER TABLE sprzedaz MODIFY CONSTRAINT czas_fk RELY;`
  - Optymalizator ufa, że dane w perspektywach są aktualne a związki zdefiniowane w obiektach `DIMENSION` i więzach `RELY` poprawne
  - Zalecane w hurtowniach danych gdzie integralność zapewniona w inny sposób (np. w operacyjnych b.d., na poziomie ETL)
- **STALE\_TOLERATED**
  - Wykorzystuje również dane z perspektyw, które mogą nie być aktualne
  - Daje największą szansę na przepisanie zapytań
  - Wyniki przepisanego zapytania mogą nie być dokładne!

# Przepisywanie zapytań – wykorzystanie obiektów DIMENSION <sup>100</sup>

```
CREATE DIMENSION SCOTT.PRODUKTY
LEVEL KATEGORIA_PRODUKTU IS SCOTT.PRODUKTY.KATEGORIA_PRODUKTU
LEVEL PRODUKT IS SCOTT.PRODUKTY.ID_PRODUKTU
LEVEL TYP_PRODUKTU IS SCOTT.PRODUKTY.TYP_PRODUKTU
HIERARCHY H_PRODUKTY (PRODUKT CHILD OF
                      KATEGORIA_PRODUKTU CHILD OF
                      TYP_PRODUKTU)
ATTRIBUTE ...
```

```
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;
```

```
SELECT id_sklepu, typ_produktu, SUM(suma_sprzedazy) AS
suma_sprzedana
FROM sprzedaz s, produkty p
WHERE s.id_produktu = p.id_produktu
GROUP BY id_sklepu, typ_produktu
```

Plan wykonywania

```
-----
0      SELECT STATEMENT Optimizer=ALL_ROWS (Cost=9 Card=3 Bytes=147)
1      0      SORT (GROUP BY) (Cost=9 Card=3 Bytes=147)
2      1      HASH JOIN (Cost=8 Card=2818 Bytes=138082)
3      2      TABLE ACCESS (FULL) OF 'PRODUKTY' (TABLE) (Cost=3 Card=141 Bytes=1410)
4      2      MAT_VIEW REWRITE ACCESS (FULL) OF 'SPRZEDAZ_MV' (MAT_VIEW REWRITE)
          (Cost=4 Card=2818 Bytes=109902)
```

# SQLAccess Advisor

- Dobór odpowiednich perspektyw materializowanych i indeksów w celu poprawy efektywności wykonywania zapytań nie jest zadaniem trywialnym
- **SQLAccess Advisor** - narzędzie do strojenia systemu generujące rekomendacje dotyczące:
  - Materializowanych perspektyw
  - Indeksów
  - Dzienników materializowanych perspektyw
- Narzędzie dostępne poprzez:
  - Interfejs **Oracle Enterprise Manager** (strona **Advisor Central**)
  - Pakiet **DBMS\_ADVISOR**
- Do korzystania wymagany przywilej **ADVISOR**
- W Oracle9i: **Summary Advisor**

# OEM – SQLAccess Advisor – Przykład (1/7)

- Uruchomienie narzędzia z poziomu Oracle Enterprise Manager

[Home](#) [Performance](#) [Administration](#) [Maintenance](#)

---

**Related Links**

<a href="#">Advisor Central</a>	<a href="#">Alert History</a>	<a href="#">Alert Log Content</a>
<a href="#">All Metrics</a>	<a href="#">Blackouts</a>	<a href="#">iSQL*Plus</a>
<a href="#">Jobs</a>	<a href="#">Manage Metrics</a>	<a href="#">Metric Collection Errors</a>
<a href="#">Monitoring Configuration</a>	<a href="#">User-Defined Metrics</a>	

---

Database: [marek10g.world](#) > [Advisor Central](#) Logged in As SCOTT

[Advisor Central](#)

---

Page Refreshed 2005-02-13 11:42:48

**Advisors**

<a href="#">ADDM</a>	<a href="#">Memory Advisor</a>	<a href="#">Segment Advisor</a>
<a href="#">SQL Tuning Advisor</a>	<a href="#">MTTR Advisor</a>	<a href="#">Undo Management</a>
<a href="#">SQL Access Advisor</a>		

# OEM – SQLAccess Advisor – Przykład (2/7)

- Wybór obciążenia

ORACLE Enterprise Manager 10g Database Control Help Logout Database

Workload Source Recommendation Options Schedule Review

## SQL Access Advisor: Workload Source

Database **marek10g.world** Cancel Step 1 of 4 Next  
 Logged in As **SCOTT**

Select the source of the workload that you want to use for the analysis. The best workload is one that fully represents all the SQL statements that access the underlying tables.

**Current and Recent SQL Activity**  
SQL will be selected from the cache.

**Import Workload from SQL Repository**  
Choose any SQL Tuning Set from the SQL Repository that you own.  
 SQL Tuning Set  [SQL Tuning Sets](#)

**User-Defined Workload; Import SQL from a Table**  
The table must contain at least SQL\_TEXT and USERNAME columns.  
 Table

**Create a Hypothetical Workload from the Following Schemas and Tables**  
The advisor can create a hypothetical workload if the tables contain dimension or primary/foreign key constraints.  
 Tables   
Comma-separated list

**TIP** Enter "Schema.%" to specify all the tables belonging to a particular schema.

**Overview**

The performance of SQL queries can often be improved by creating additional structures, like Indexes and Materialized Views, that help in data retrieval. The SQL Access Advisor evaluates SQL statements in a workload, and can suggest indexes and materialized views that will improve the performance of the workload as a whole.

Ostatnie polecenia z cache'a

→

SQL Tuning Set

→

Polecenia SQL zapisane w tabeli

→

Automatycznie wygenerowane obciążenie w oparciu o wymiary i klucze

→

# OEM – SQLAccess Advisor – Przykład (3/7)

- Wybór zakresu rekomendacji
  - Materializowane perspektywy
  - Indeksy

ORACLE Enterprise Manager 10g Database Control

Help Logout Database

Workload Source Recommendation Options Schedule Review

## SQL Access Advisor: Recommendation Options

Database marek10g.world  
Logged in As SCOTT

Cancel Back Step 2 of 4 Next

### Recommendation Types

The advisor may recommend indexes or materialized views to reduce the time it takes to read data. However you must balance this benefit against the cost to maintain the additional structures. Select the type of structures to be recommended by the advisor.

- Indexes
- Materialized Views
- Both Indexes and Materialized Views

**TIP** Access Advisor jobs generating Materialized View recommendations should be scheduled in maintenance windows.

### Advisor Mode

The advisor can run in one of two modes, Limited or Comprehensive. Limited Mode is meant to return quickly after processing the statements with the highest cost, potentially ignoring statements with a cost below a certain threshold. Comprehensive Mode will perform an exhaustive analysis.

- Limited Mode  
Analysis will focus on highest cost statements
- Comprehensive Mode  
Analysis will be exhaustive

# OEM – SQLAccess Advisor – Przykład (4/7)

- Zaszeregowanie zadania do wykonania (JOB)

ORACLE Enterprise Manager 10g Database Control Help Logout Database

Workload Source Recommendation Options **Schedule** Review

---

## SQL Access Advisor: Schedule

Database **marek10g.world**  
 Logged in As **SCOTT** Cancel Back Step 3 of 4 Next

The Advisor Task will be scheduled to run immediately by default. You may choose to schedule the task to run at a later time. A database job will be submitted to complete this task. It will be named the same as the Task Name, with "ADV\_" prepended.

**Advisor Task Name**

\* Task Name

Task Description

**Scheduling Options**


Schedule Type

**Repeating**

Repeat

**Start**

Immediately  
 Later

Date    
(example: 2005-02-13)

Time     AM  PM

Jednorazowo →

Natychmiast →

# OEM – SQLAccess Advisor – Przykład (5/7)

- Ekran podsumowujący; ostateczne potwierdzenie zadania

ORACLE Enterprise Manager 10g Database Control Help Logout Database

Workload Source Recommendation Options Schedule Review

---

## SQL Access Advisor: Review

Database **marek10g.world**  
 Logged in As **SCOTT**

Step 4 of 4

This page allows for review of your chosen input parameters to the SQL Access Advisor.

Task Name           **SQLACCESS1521101**  
 Task Description   **SQL Access Advisor**  
 Scheduled Start Time **Run Immediately**

**Options**

OPTION <small>△</small>	VALUE
Advisor Mode	Limited Mode
Hypothetical	SCOTT.CZAS, SCOTT.PRODUKTY, SCOTT.SKLEPY, SCOTT.SPRZEDAZ
Recommendation Type	All Methods
Workload Scope	Partial Workload
Workload Source	Hypothetical
Workload Type	Allow Advisor to Determine Workload Type Based on Workload Content

# OEM – SQLAccess Advisor – Przykład (6/7)

- Monitorowanie statusu zadania

Results

View Result Delete Actions Re-schedule Go

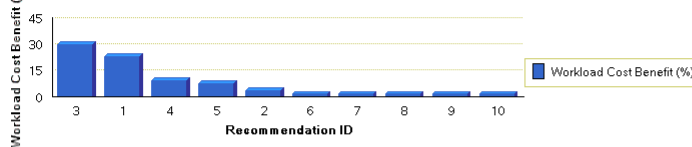
Select	Advisory Type	Name	Description	User	Status	Start Time	End Time	Expires In (days)
<input type="radio"/>	SQL Access Advisor	SQLACCESS4724595	SQL Access Advisor	SCOTT	COMPLETED	2005-02-13 11:26:28	2005-02-13 11:26:49	30
<input type="radio"/>	SQL Access Advisor	SQLACCESS1521101	SQL Access Advisor	SCOTT	CREATED			30

- Podgląd wyników

### Recommendations

The following chart and table initially show the top recommendations ordered by their percentage improvement to the total cost of the whole workload. The top recommendation will have the biggest total performance improvement.

Recommendations Ordered by Workload Cost Benefit (%)



### Select Recommendations for Implementation

Select All | Select None

Schedule Implementation Show SQL

Select	Recommendation ID	Actions	Workload Cost Benefit (%)	Estimated Space Used (MB)	Affected SQL Statements
<input checked="" type="checkbox"/>	3	5	29.52	8.813	1
<input checked="" type="checkbox"/>	1	5	22.52	1.141	3
<input checked="" type="checkbox"/>	4	6	9.00	0.234	1
<input checked="" type="checkbox"/>	5	4	7.63	0.164	1
<input checked="" type="checkbox"/>	2	4	3.39	0.039	2
<input checked="" type="checkbox"/>	6	5	1.76	0.086	1
<input checked="" type="checkbox"/>	7	5	1.76	0.086	1
<input checked="" type="checkbox"/>	8	4	1.72	0.008	1
<input checked="" type="checkbox"/>	9	4	1.72	0.008	1
<input checked="" type="checkbox"/>	10	4	1.72	0.023	1

Schedule Implementation Show SQL

# OEM – SQLAccess Advisor – Przykład (7/7)

- Przeglądanie rekomendacji

## Recommendation: 3

The Actions table lists the actions of the selected recommendation. You may change the values of any of the editable fields. If you edit any name, dependent names, which are shown as readonly, will be updated accordingly. If the Tablespace field is left blank the default tablespace of the schema will be used. When you click OK, the SQL script is modified, but it is not actually executed until you select 'Schedule Implementation' on the Recommendations page.

Cancel OK

### Actions

Action	Object Name	Index Object Table	Schema	Tablespace
CREATE MATERIALIZED VIEW LOG		"SCOTT"."SPRZEDAZ"	"SCOTT"	
CREATE MATERIALIZED VIEW LOG		"SCOTT"."SKLEPY"	"SCOTT"	
CREATE MATERIALIZED VIEW LOG		"SCOTT"."PRODUKTY"	"SCOTT"	
CREATE MATERIALIZED VIEW	"MV\$\$_002B0001"		"SCOTT"	
GATHER TABLE STATISTICS	"MV\$\$_002B0001"		"SCOTT"	

### SQL Affected by Recommendation

Statement ID	Statement	Workload Cost Benefit (%)	Original Cost	New Cost	SQL Cost Benefit (%)	Execution Count
2	SELECT SKLEPY.ID_SKLEPU, SKLEPY.MIEJSCOWOSC, SKLEPY.REGION, PRODUKTY.ID_PRODUKTU, PRODUKTY.KATEGORIA_PRODUKTU, PRODUKTY.TYP_PRODUKTU, COUNT(*), SUM (SPRZEDAZ.LICZBA_KLIENTOW), COUNT (SPRZEDAZ.LICZBA_TOWAROW), COUNT (SPRZEDAZ.LICZBA_TOWAROW), SUM (SPRZEDAZ.SUM	29.52	1845	250	86.45	1

### Show SQL

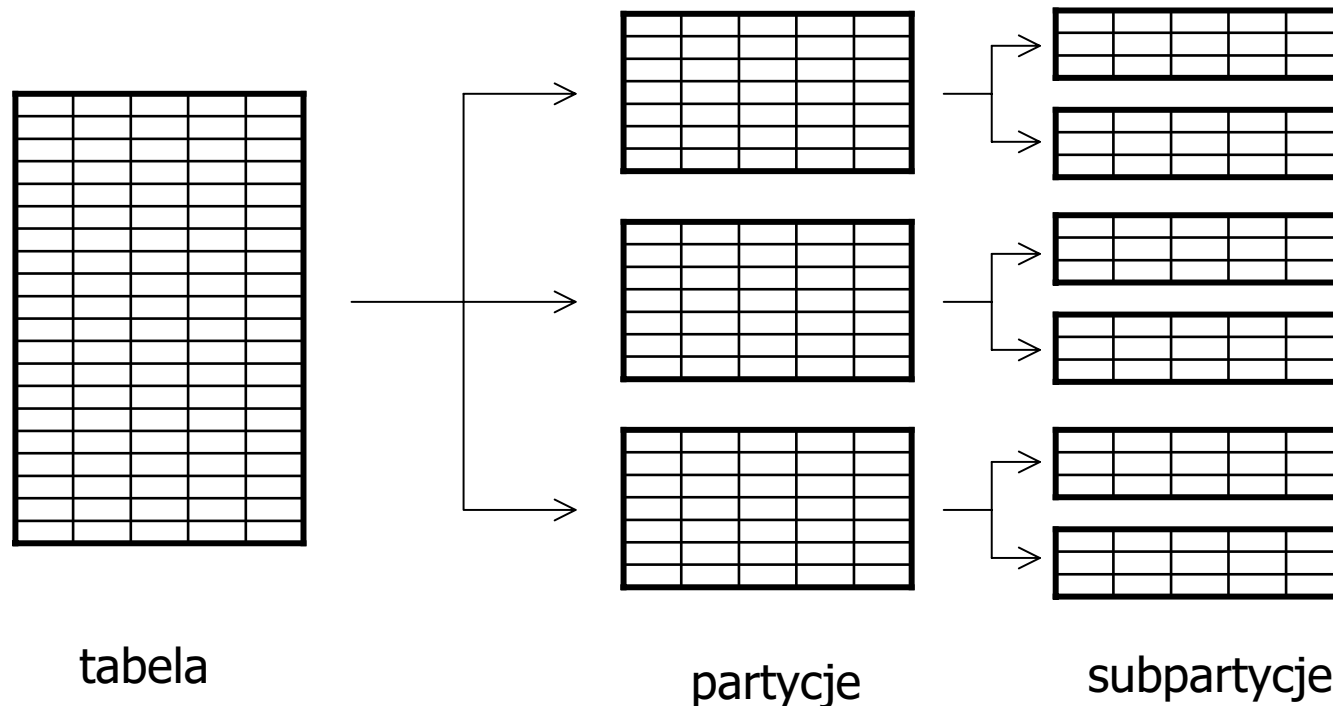
```
CREATE MATERIALIZED VIEW "SCOTT"."MV$$_002B0001"
REFRESH FAST WITH ROWID
ENABLE QUERY REWRITE
AS SELECT SCOTT.SKLEPY.REGION C1, SCOTT.SKLEPY.MIEJSCOWOSC C2, SCOTT.SKLEPY.ID_SKLEPU
C3, SCOTT.PRODUKTY.KATEGORIA_PRODUKTU C4, SCOTT.PRODUKTY.TYP_PRODUKTU
C5, SCOTT.PRODUKTY.ID_PRODUKTU C6, SUM("SCOTT"."SPRZEDAZ"."LICZBA_KLIENTOW")
M1, COUNT("SCOTT"."SPRZEDAZ"."LICZBA_KLIENTOW") M2, SUM("SCOTT"."SPRZEDAZ"."LICZBA_TOWAROW")
M3, COUNT("SCOTT"."SPRZEDAZ"."LICZBA_TOWAROW") M4, SUM("SCOTT"."SPRZEDAZ"."SUMA_SPRZEDAZY")
M5, COUNT("SCOTT"."SPRZEDAZ"."SUMA_SPRZEDAZY") M6, SUM("SCOTT"."SPRZEDAZ"."SUMA_ZYSKU")
M7, COUNT("SCOTT"."SPRZEDAZ"."SUMA_ZYSKU") M8, COUNT(*) M9 FROM SCOTT.SPRZEDAZ,
SCOTT.SKLEPY, SCOTT.PRODUKTY WHERE SCOTT.SKLEPY.ID_SKLEPU = SCOTT.SPRZEDAZ.ID_SKLEPU
AND SCOTT.PRODUKTY.ID_PRODUKTU = SCOTT.SPRZEDAZ.ID_PRODUKTU GROUP BY SCOTT.SKLEPY.REGION,
SCOTT.SKLEPY.MIEJSCOWOSC, SCOTT.SKLEPY.ID_SKLEPU, SCOTT.PRODUKTY.KATEGORIA_PRODUKTU,
SCOTT.PRODUKTY.TYP_PRODUKTU, SCOTT.PRODUKTY.ID_PRODUKTU;
```

# Pakiet DBMS\_ADVISOR

- **CREATE\_TASK** – utworzenie zadania
- **CREATE\_SQLWKLD** – utworzenie obiektu obciążenia
- **ADD\_SQLWKLD\_REF** – powiązanie zadania z obciążeniem
- **IMPORT\_SQLWKLD\_SQLCACHE,**  
**IMPORT\_SQLWKLD\_STS ,**  
**IMPORT\_SQLWKLD\_USER,**  
**IMPORT\_SQLWKLD\_SCHEMA**  
- definicja obciążenia (4 warianty jak w kreatorze OEM)
- **EXECUTE\_TASK** – uruchomienie zadania
- **GET\_TASK\_SCRIPT** – generacja skryptu, alternatywa dla przeglądania rekomendacji w perspektywach słownika

# 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),
 nazwisko 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))
```

- Tabela podzielona na trzy partycje, „male\_pensje”, „srednie\_pensje” i „duze\_pensje”
- Rekordy, w których pensja < 1000 trafiają do partycji „male\_pensje”
- Rekordy, w których 1000 ≤ pensja < 10000 trafiają do partycji „srednie\_pensje”
- Rekordy, w których 10000 ≤ pensja trafiają do partycji „duze\_pensje”
- Wszystkie partycje są przechowywane w tej samej przestrzeni tabel

# Partycjonowanie haszowe

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

- Tabela podzielona na trzy partycje o nazwach wygenerowanych automatycznie (np. SYS\_P1397)
- Rekordy są rozpraszane pomiędzy partycje na podstawie wartości funkcji haszowej wyliczanej dla kolumny „id”
- Wszystkie partycje są przechowywane w tej samej przestrzeni tabel

# 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'))
```

- Tabela podzielona na trzy partycje: „p1”, „p2”, „p3”
- Pracownicy, których imię to „Adam” lub „Joanna” trafiają do partycji „p1”
- Pracownicy, których imię to „Jerzy” lub „Anna” trafiają do partycji „p2”
- Pracownicy, których imię to „Maciej” lub „Maria” trafiają do partycji „p3”
- Wszystkie partycje są przechowywane w tej samej przestrzeni tabel

# 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))
```

- Tabela podzielona jest na 12 subpartycji, należących do 3 partycji
- Rekordy są rozpraszane pomiędzy partycjami wg przynależności wartości kolumny „pensja” do zdefiniowanych przedziałów
- W obrębie każdej partycji, rekordy są rozpraszane pomiędzy 4 subpartycje wg wartości funkcji haszowej wyliczonej dla kolumny „id”

# 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'))
)
```

- Tabela podzielona jest na 6 subpartycji, należących do 3 partycji
- Rekordy są rozpraszane pomiędzy partycjami wg przynależności wartości kolumny „pensja” do zdefiniowanych przedziałów
- W obrębie każdej partycji, rekordy są rozpraszane pomiędzy 2 subpartycje – jedna dla pracowników o imieniu „Adam”, druga dla pracowników o imieniu „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 tbsp1,
 PARTITION srednie_pensje VALUES LESS THAN (10000) TABLESPACE tbsp2,
 PARTITION duze_pensje VALUES LESS THAN (MAXVALUE) TABLESPACE tbsp3)
```

- W każdej metodzie partycjonowania partycje mogą być rozpraszane pomiędzy różne przestrzenie tabel

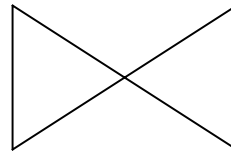
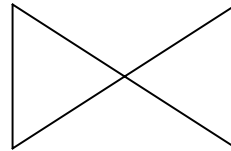
# Operacje na tabelach partycjonowanych

- 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 ...)`
- Obcinanie partycji
  - `alter table ... truncate partition ...`

# Połączenie partycji (Partition-wise Joins)

Partycjonowana tabela Sprzedaz

	suma_sprzedazy	id_sklepu
partycja s1	127,00	1
	340,00	1
	720,11	3
	110,00	3
partycja s2	suma_sprzedazy	id_sklepu
	99,00	5
	1250,10	5
	500,00	7
	230,50	8



Partycjonowana tabela Sklepy

id_sklepu	miejscowosc
1	New Orleans
2	Chicago
4	New York
partycja k1	
id_sklepu	miejscowosc
5	Los Angeles
7	Chicago
8	New Orleans
partycja k2	

Niezależne partycje są łączone przez równoległe pracujące procesy → redukcja czasu wykonania zapytania

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

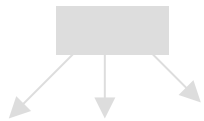
# Indeks bitmapowy

```
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



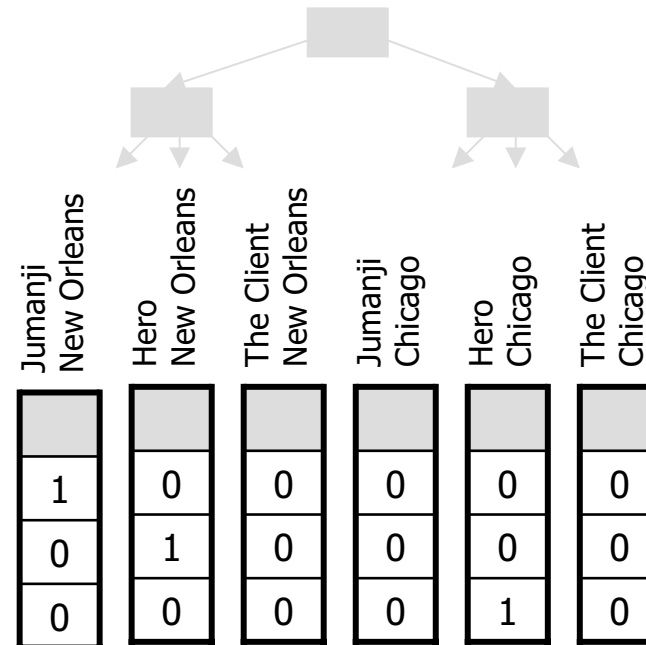
Warszawa	Kraków	Poznań
1	0	0
0	1	0
0	0	1
0	1	0
0	1	0
1	0	0
0	0	1
1	0	0

# Bitmapowy indeks połączeniowy

id_produktu	nazwa_produktu
1	Jumanji
2	Hero
3	The Client

suma_sprzedazy	id_sklepu	id_produktu
5820,00	1	1
17200,00	1	2
350,00	2	2

id_sklepu	miescowosc
1	New Orleans
2	Chicago



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

# Bitmapowy indeks połączeniowy

- 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

Plan wykonywania

```
-----  
0  SELECT STATEMENT Optimizer=ALL_ROWS (Cost=2 Card=1 Bytes=39)  
1 0   SORT (AGGREGATE)  
2 1   TABLE ACCESS (BY INDEX ROWID) OF 'SPRZEDAZ' (TABLE) (Cost=2 Card=1 Bytes=39)  
3 2   BITMAP CONVERSION (TO ROWIDS)  
4 3   BITMAP INDEX (SINGLE VALUE) OF 'SPRZ_PROD_SKLEP_IND' (INDEX (BITMAP))
```