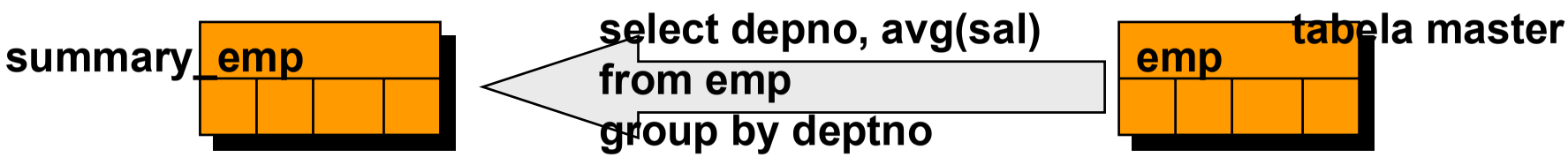
Materializowanie perspektyw

Robert Wrembel, Juliusz Jezierski
Politechnika Poznańska, Instytut Informatyki

Materializowanie wyników zapytania

- **Cel → optymalizacja zapytań analitycznych**
- **Mechanizmy**
  - **Perspektywy zmaterializowane (materialized views)**
  - **Przepisywanie zapytań (query rewrite)**
  - **Oprogramowanie Summary Advisor**
  - **Wymiary (dimensions)**



- **Odświeżanie perspektyw zmaterializowanych**
  - **synchroniczne / asynchroniczne**
  - **pełne / przyrostowe**

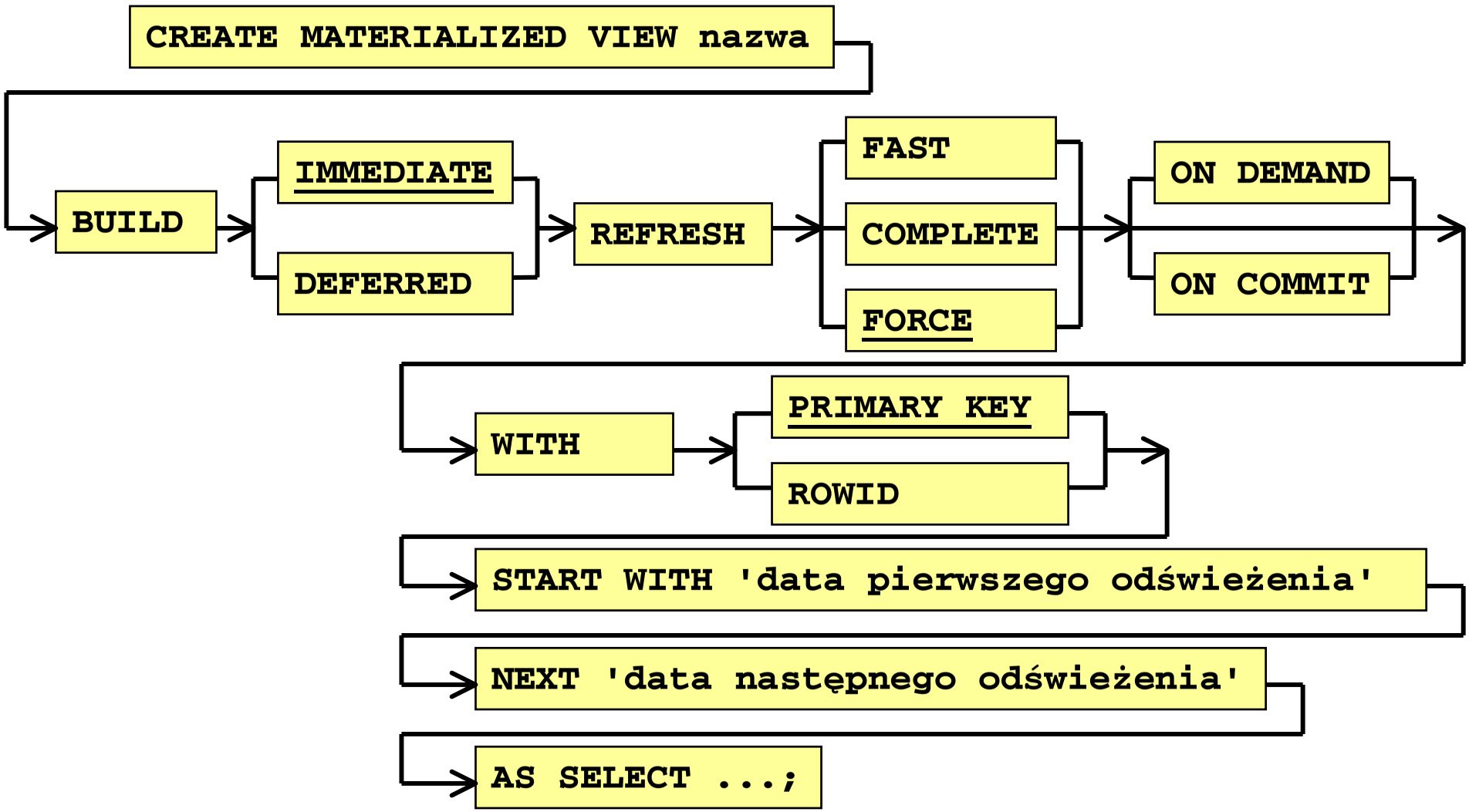
Perspektywa zmaterializowana (1)

- **Definicja zawiera:**
  - **zapytanie odwołujące się do tabel**
  - **sposób odświeżania danych (pełne, przyrostowe)**
  - **częstotliwość odświeżania danych**
- **Standardowo tylko do odczytu**
- **Implementacja**
  - **tabela + indeks**

Perspektywa zmaterializowana (2)

- **Rodzaje (sposób identyfikowania rekordów)**
  - **PRIMARY KEY**
    - **tabela master musi posiadać włączone ograniczenie PRIMARY KEY**
    - **klauzula SELECT musi zawierać wszystkie atrybuty wchodzące w skład klucza podstawowego tabeli master**
  - **ROWID**
- **Rodzaje (struktura zapytania)**
  - **perspektywa zawierająca agregaty**
  - **perspektywa zwierająca jedynie połączenia**
  - **perspektywa zagnieżdżona**

Definiowanie perspektywy zmaterializowanej



Przykład

```
CREATE MATERIALIZED VIEW mv_suma_sprzedazy
BUILD IMMEDIATE
REFRESH FAST
NEXT sysdate+(1/(24*60*30))
as
select sklep_id, produkt_id,
       sum(l_sztuk), sum(l_sztuk*cena_jedn),
       count(l_sztuk), count(l_sztuk*cena_jedn), count(*)
from sprzedaz
group by sklep_id, produkt_id;
```

Odświeżanie perspektywy zmaterializowanej (2)

- **Sposób odświeżania**
  - **REFRESH FAST -> odświeżanie przyrostowe**
  - **REFRESH COMPLETE -> odświeżanie pełne**
  - **REFRESH FORCE -> automatyczny wybór metody odświeżania; jeżeli możliwe to Oracle wybiera FAST**
- **Częstotliwość odświeżania**
  - **START WITH -> data pierwszego odświeżenia**
  - **NEXT -> wyrażenie określające częstotliwość odświeżania**
- **Odświeżanie automatyczne gdy:**
  - **wyspecyfikowana klauzula NEXT**
  - **określona częstotliwość odświeżania**
    - **REFRESH FAST START WITH sysdate NEXT sysdate+1**
    - **REFRESH FAST NEXT sysdate+1**

Odświeżanie przyrostowe

- **Dla perspektyw z agregatami:**
  - **Tylko dla agregatów SUM, COUNT, AVG, STDDEV, VARIANCE, (MIN i MAX tylko dla operacji INSERT)**
  - **klauzula SELECT musi zawierać wszystkie wyrażenia z klauzuli GROUP BY oraz COUNT(*) i COUNT(*wyrażenie*) dla wszystkich agregowanych wyrażeń**
  - **dla agregatów VARIANCE(*wyrażenie*) i STDEV(*wyrażenie*) klauzula SELECT musi zawierać również SUM(*wyrażenie*), zaleca się zawarcie SUM(*wyrażenie* wyrażenie*)**
  - **dziennik utworzony z klauzulą INCLUDING NEW VALUES, ROWID i SEQUENCE**
  - **dziennik zawiera wszystkie atrybuty wymienione po SELECT oraz będące argumentami wywołania funkcji grupowych**
  - **dla połączeń zewnętrznych tylko modyfikacje tabeli zewnętrznej mogą być odświeżane przyrostowo**
  - **wykorzystywane perspektywy muszą dać się scalić z zapytaniem def. MV**

Odświeżanie przyrostowe

- **Dla tylko z połączeniami:**
  - **klauzula SELECT musi zawierać ROWID wszystkich tabel *detail***
  - **każda tabela musi posiadać dziennik z klauzulą ROWID**
  - **jeżeli połączenie jest połączeniem zewnętrznym (może być tylko równościowym) to zapytanie def. MV nie może wykonywać operacji selekcji**
  - **dla połączenia zewnętrznego atrybuty połączeniowe tabeli wewnętrznej muszą stanowić klucz unikalny**

Odświeżanie perspektywy zmaterializowanej (1)

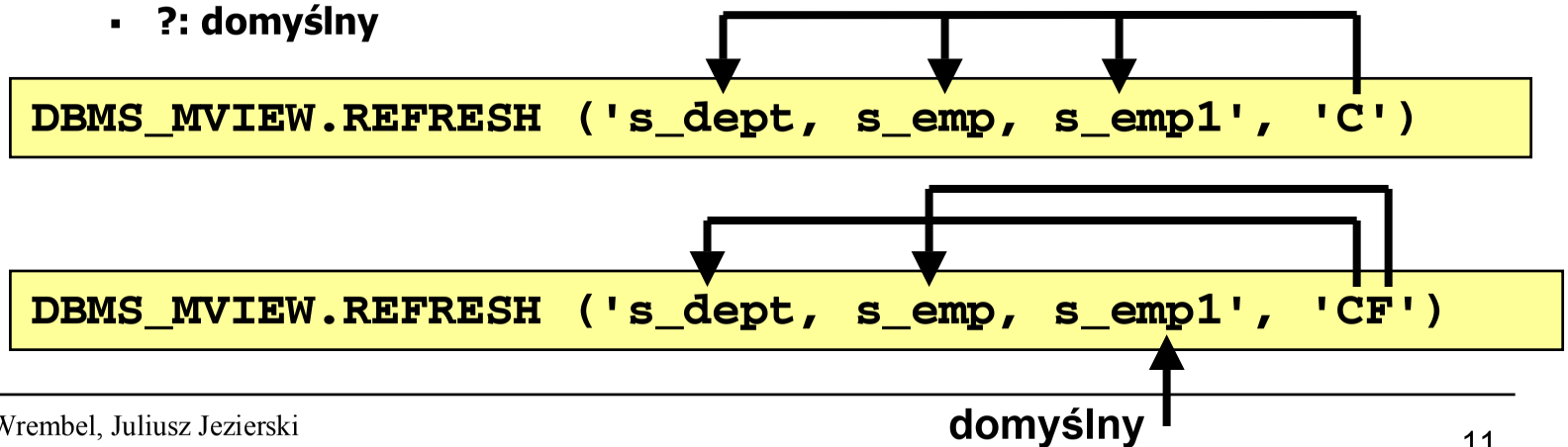
- **ON COMMIT** można stosować jedynie, gdy:
  - zapytanie korzysta z tabel lokalnych
- Uprawnienia wymagane do doświeżania ON COMMIT
  - uprawnienie obiektowe **ON COMMIT** dla wszystkich tabel, do których odwołuje się perspektywa zmaterializowana
  - uprawnienie systemowe **ON COMMIT**

Odświeżanie perspektywy zmaterializowanej (2)

- **Odświeżanie ręczne, gdy:**
  - **wyspecyfikowano ON DEMAND**
  - **brak klauzuli NEXT**
    - **perspektywa odświeżona raz w momencie jej tworzenia jeśli wyspecyfikowano BUILD IMMEDIATE**

```
DBMS_MVIEW.REFRESH ('sn1, sn2, ..., snn', 'metoda')
```

  - **sn_1, sn_2, ..., sn_n: perspektywy zmaterializowane**
  - **metoda: metoda odświeżania**
    - **f lub F: FAST**
    - **c lub C: COMPLETE**
    - **?: domyślny**



Prebuilt table (1)

- W systemie znajdują się "zwykłe" tabele zawierające dane
- Zamiana zwykłych tabel na perspektywy zmaterializowane → wykorzystanie do przepisywania zapytań
  - klauzula ON PREBUILT TABLE

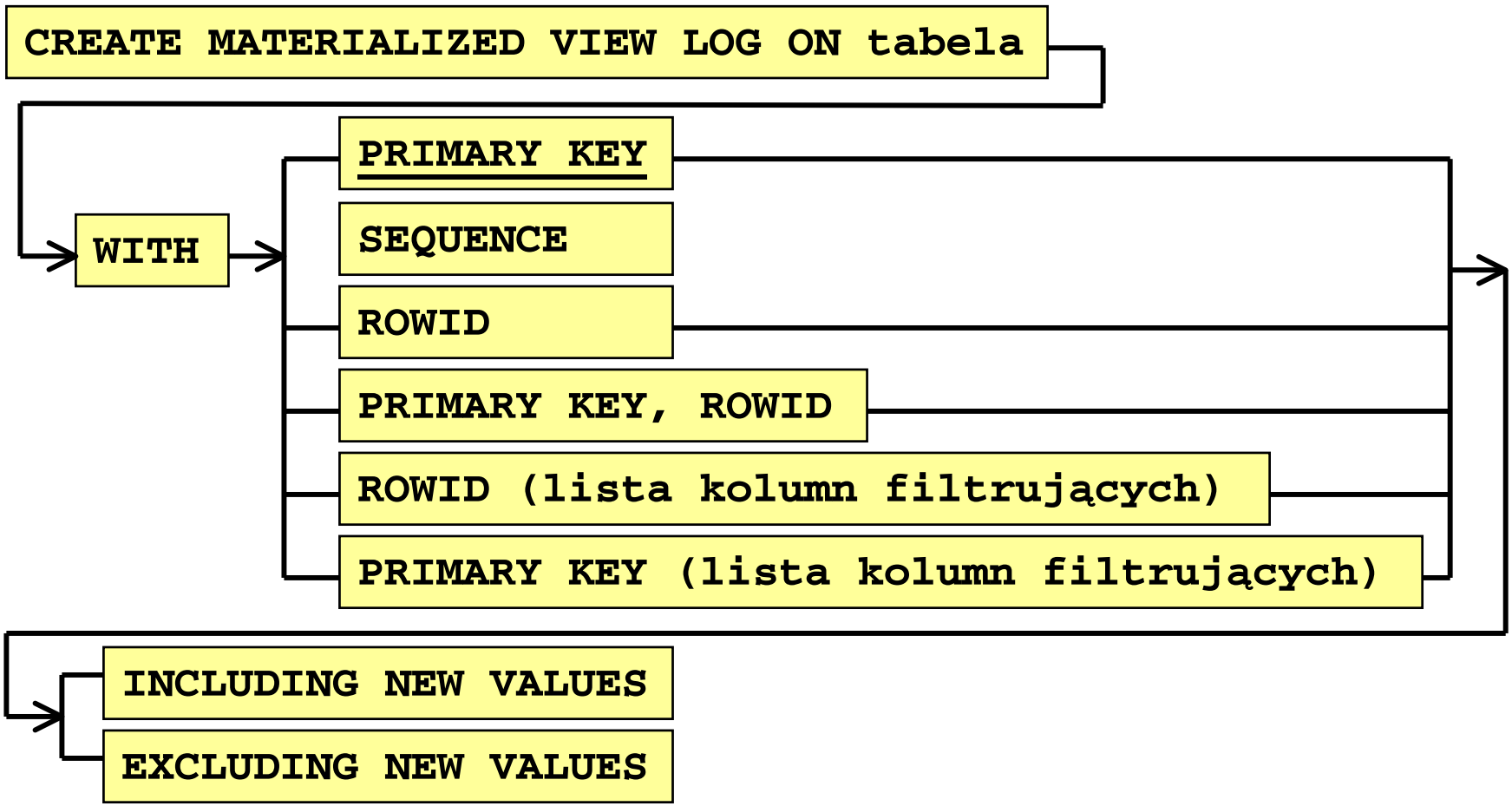
```
CREATE TABLE monthly_customer_sales
( YEAR NUMBER(4), MONTH NUMBER(2),
  CUSTOMER_ID VARCHAR2(10), DOLLAR_SALES NUMBER);
```

```
CREATE MATERIALIZED VIEW monthly_customer_sales
ON PREBUILT TABLE
ENABLE QUERY REWRITE
AS
SELECT t.year, t.month, c.customer_id,
       SUM(f.purchase_price) dollar_sales
FROM time t, purchases f, customer c
WHERE f.time_key = t.time_key AND f.customer_id = c.customer_id
GROUP BY t.year, t.month, c.customer_id;
```

Prebuilt table (2)

- **Nazwa perspektywy zmaterializowanej budowanej na istniejącej tabeli musi być identyczna z nazwą tej tabeli**
- **System nie sprawdza poprawności zawartości tabeli w kontekście zapytania definiującego perspektywę**
  - **tabela może zawierać dane nieodpowiadające zapytaniu**
- **Perspektywa będzie odświeżana z wykorzystaniem standardowych mechanizmów, jeśli je zdefiniowano**

Dziennik perspektywy zmaterializowanej (1)



Dziennik perspektywy zmaterializowanej (2)

- **WITH PRIMARY KEY**
  - **w dzienniku rejestrowane wartości atrybutów wchodzących w skład klucza**
- **WITH ROWID**
  - **w dzienniku rejestrowane ROWID rekordów**
- **WITH PRIMARY KEY, ROWID**
  - **w dzienniku rejestrowane wartości atr. kluczowych i ROWID**
- **WITH SEQUENCE**
  - **konieczne do odświeżania przyrostowego, gdy do tabeli bazowej są wstawiane rekordy, modyfikowane i usuwane**

Dziennik perspektywy zmaterializowanej (3)

- **kolumna filtrująca**
  - **atrybut występujacy w klauzuli WHERE zapytania definiującego perspektywę**

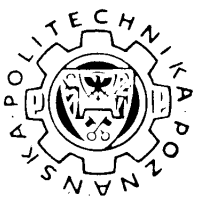
```
select sk.nazwa, sk.sklep_id
from scott.sklepy
where exists (select sp.sklep_id
              from scott.sprzedaz
              where sp.sklep_id=sk.sklep_id
              and sp.produkt_id=100
              and sp.data='23.01.2002'
              and sp.l_sztuk>1)
```

- **including new values**
  - **konieczne dla perspektyw odświeżanych przyrostowo zawierających agregaty**

Przykład

```
CREATE MATERIALIZED VIEW mv_suma_sprzedazy
BUILD IMMEDIATE
REFRESH FAST
NEXT sysdate+(1/(24*60*30))
as
select sklep_id, produkt_id, sum(l_sztuk), sum(l_sztuk*cena_jedn),
       count(l_sztuk), count(l_sztuk*cena_jedn), count(*)
from sprzedaz
group by sklep_id, produkt_id;
```

```
CREATE MATERIALIZED VIEW LOG on sprzedaz
WITH PRIMARY KEY, ROWID (l_sztuk, cena_jedn)
INCLUDING NEW VALUES;
```



EXPLAIN_MVIEW (1)

- **Procedura pakietu DBMS_MVIEW**
- **Analizuje zapytanie definiujące perspektywę i sprawdza, czy perspektywa może być odświeżana przyrostowo**
- **W schemacie użytkownika należy uruchomić skrypt %ORACLE_HOME%\rdbms\admin\utlxmv.sql tworzący tabelę MV_CAPABILITIES_TABLE przechowującą wyniki analiz**

EXPLAIN_MVIEW (2)

```
BEGIN
dbms_mview.explain_mview (
'select sp.rowid sp_rowid, sp.produkt_id, sp.l_sztuk,
 sk.rowid sk_rowid, sk.nazwa
 from sprzedaz sp, sklepy sk
 where sp.sklep_id=sk.sklep_id', 1);
END;
/
```

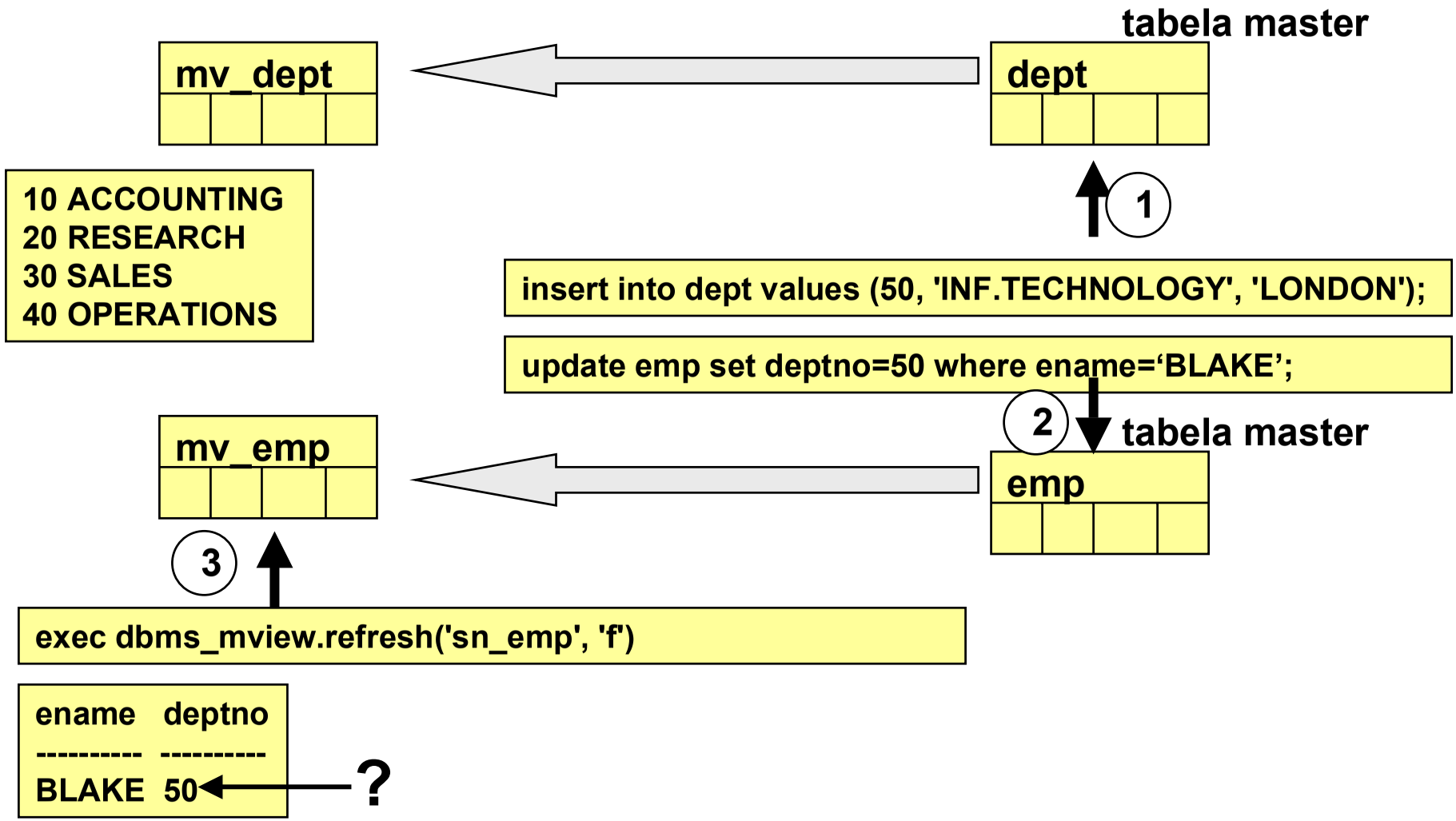
statement_id

```
SELECT capability_name, possible,
       related_text "table", msgtxt explanation
FROM MV_CAPABILITIES_TABLE
where statement_id='1';
```

EXPLAIN_MVIEW (3)

CAPABILITY_NAME	P	table	EXPLANATION
PCT	N		
REFRESH_COMPLETE	Y		
REFRESH_FAST	N		
REWRITE	Y		
PCT_TABLE	N	SPRZEDAZ	relation is not a partitioned table
PCT_TABLE	N	SKLEPY	relation is not a partitioned table
REFRESH_FAST_AFTER_INSERT	N	DEMO.SPRZEDAZ	the detail table does not have a materialized vlog
REFRESH_FAST_AFTER_ONETAB_DML	N		see the reason why REFRESH_FAST_AFTER_INSERT is disabled
REFRESH_FAST_AFTER_ANY_DML	N		see the reason why REFRESH_FAST_AFTER_ONETAB_DML is disabled
REFRESH_FAST_PCT	N		PCT is not possible on any of the detail tables in the materialized view
REWRITE_FULL_TEXT_MATCH	Y		
REWRITE_PARTIAL_TEXT_MATCH	Y		
REWRITE_GENERAL	Y		
REWRITE_PCT	N		general rewrite is not possible and PCT is not possible on any of the detail tables

Zależności między perspektywami



Grupy odświeżania

```
DBMS_REFRESH.MAKE
   ( name,
     list,
     next_date,
     interval,
     implicit_destroy)
```

- nazwa grupy
- lista perspektyw przypisywanych do grupy
- data następnego odświeżenia
- okres odświeżania
- TRUE: usunięcie grupy jeżeli nie zawiera MV, domyślnie FALSE

- **lista perspektyw zmaterializowanych**
  - **muszą być w tej samej bd**
  - **mogą być w różnych schematach**
  - **max. 100 MV w grupie**

Przykłady

```
exec DBMS_REFRESH.MAKE
  (name => 'orc1.rg_dept_emp', -
   list => 'orc1.sn_dept, orc1.sn_emp', -
   next_date => sysdate+1/48, -
   interval => 'next_day(trunc(sysdate), ''FRIDAY'')+10/24', -
   implicit_destroy => TRUE, -
   rollback_seg => 'rb1')
```

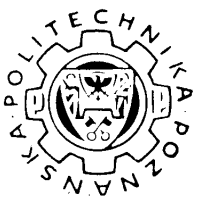
```
exec DBMS_REFRESH.ADD ('orc1.rg_dept_emp', 'orc1.sn_emp1')
```

```
exec DBMS_REFRESH.SUBTRACT('orc1.rg_dept_emp', 'orc1.sn_emp1')
```

```
exec DBMS_REFRESH.REFRESH('orc1.rg_dept_emp')
```

```
exec DBMS_REFRESH.DESTROY('orc1.rg_dept_emp')
```

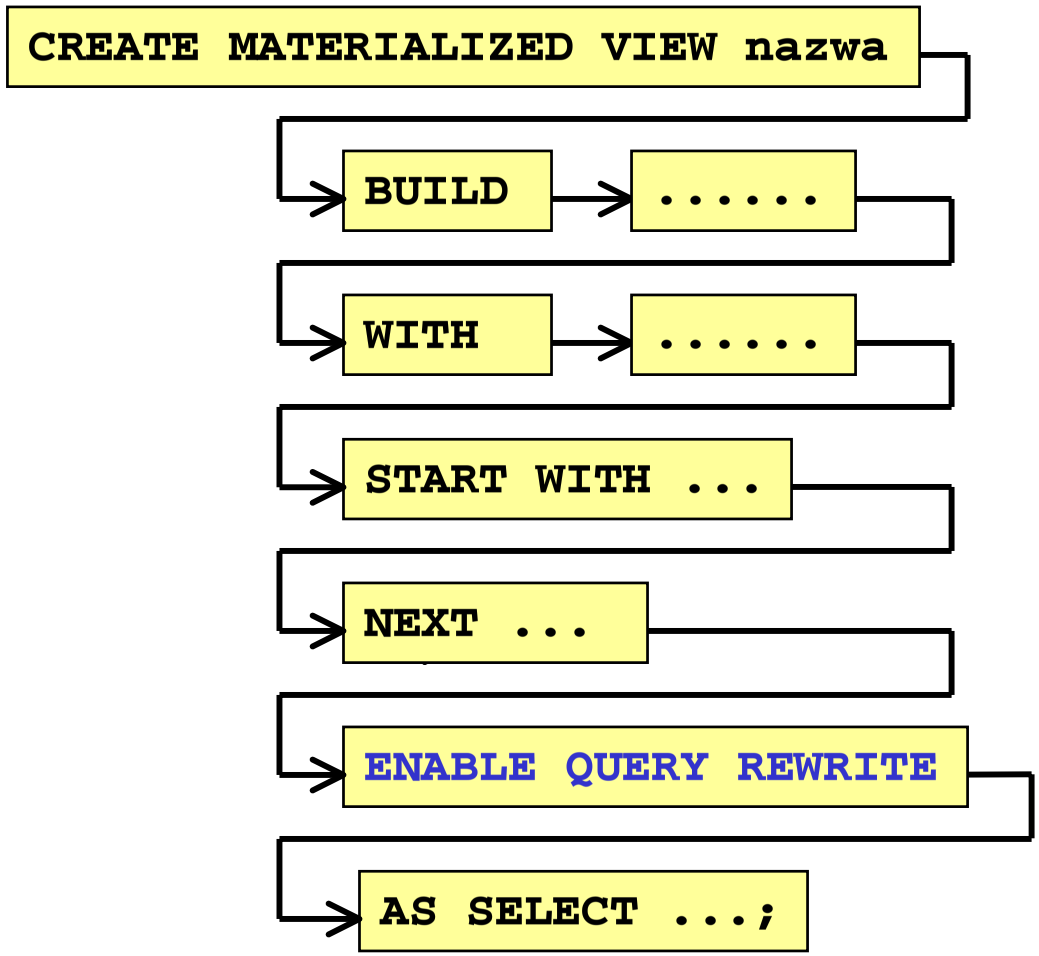
usuwa grupę z perspektywami lub pustą



Informacje słownikowe

- Zmaterializowane perspektywy
  - USER_MVIEWS
- Dzienniki zmaterializowanych perspektyw
  - USER_MVIEW_LOGS
- Informacje nt. odświeżania
  - USER_MVIEW_REFRESH_TIMES
- Grupy odświeżania
  - USER_REFRESH_CHILDREN
- Zmaterializowane perspektywy w grupie
  - USER_REFRESH

Przepisywanie zapytań (1)



Przepisywanie zapytań (2)

```
create materialized view suma_sprzedazy
build immediate
refresh complete
ENABLE QUERY REWRITE
as select nazwa, sum(l_sztuk*cena_jedn) suma
      from sprzedaz sp, sklepy sk
      where sp.sklep_id=sk.sklep_id
      group by nazwa;
```

```
select nazwa, sum(l_sztuk*cena_jedn) suma
from sprzedaz sp, sklepy sk
where sp.sklep_id=sk.sklep_id
having sum(l_sztuk*cena_jedn) > 30000
group by nazwa;
```

```
Execution Plan
----------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'SUMA_SPRZEDAZY'
```

Przepisywanie zapytań (3)

```
select nazwa, sum(l_sztuk*cena_jedn) suma
from sprzedaz sp, sklepy sk
where sp.sklep_id=sk.sklep_id
and sk.miasto='Poznań'
group by nazwa;
```

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   SORT (GROUP BY)
   2    1     NESTED LOOPS
   3    2       TABLE ACCESS (FULL) OF 'SKLEPY'
   4    2       TABLE ACCESS (FULL) OF 'SPRZEDAZ'
```

```
alter materialized view nazwa_perspektywy
{enable | disable} query rewrite;
```

Przepisywanie zapytań (4)

- **Przepisanie zapytania jest możliwe tylko wówczas, gdy:**
  - **system wykorzystuje optymalizator kosztowy**
    - **parametr konfiguracyjny instancji OPTIMIZER_MODE=CHOOSE**
    - **dynamiczny wybór optymalizatora dla sesji**

```
alter session set optimizer_mode='choose';
alter session set optimizer_mode='all_rows';
```

    - **dla tabel bazowych i perspektyw zmaterializowanych zebrano statystyki**
  - **parametr konfiguracyjny QUERY_REWRITE_ENABLED=TRUE**
    - **dynamiczne włączenie przepisywania dla sesji**

```
alter session set query_rewrite_enabled=true;
```

  - **parametr konfiguracyjny QUERY_REWRITE_ENABLED=FORCE**
    - **jeżeli istnieje odpowiednia zmaterializowana perspektywa, zostanie wykorzystana do przepisania zapytania niezależnie od tego, czy z jej wykorzystaniem zapytanie zostanie wykonane szybciej, czy wolniej → nie jest porównywany koszt wykonania zapytania**

Przepisywanie zapytań (5)

- **Uprawnienia użytkownika niezbędne do wykorzystania przepisywania zapytań:**
  - **uprawnienie obiektowe QUERY REWRITE na wszystkich tabelach, do których odwołuje się perspektywa zmaterializowana**
  - **lub uprawnienie systemowe GLOBAL QUERY REWRITE**
- **Techniki przepisywania zapytań**
  - **text match**
  - **aggregate rollup**
  - **join-back**
  - **filtered data**
  - **rollup using a dimension**
- **Uwaga: perspektywa z operatorem BETWEEN AND wskazującym zakres dat nie będzie wykorzystana do przepisywania zapytań**

```
ORA-30353: expression not supported for query rewrite
```

Text match

- **Tekst zapytania perspektywy zmaterializowanej i tekst zapytania użytkownika:**
  - **są identyczne**
  - **spacje nie są uwzględniane**
  - **wielkość liter nie jest uwzględniana**
  - **kolejność atrybutów/wyrażeń w obu zapytaniach nie ma znaczenia**
  - **kolejność składników klauzuli WHERE nie ma znaczenia**
  - **kolejność tabel w klauzuli FROM nie ma znaczenia**

Aggregate rollup (1)

- **Wynik zapytania użytkownika jest wyznaczany za pomocą agregowania zawartości perspektywy zmaterializowanej**

```
create materialized view mv_suma_sprzedazy
build immediate
refresh force on commit
enable query rewrite
as
select sk.nazwa, pr.prod_nazwa, sum(l_sztuk*cena_jedn), count(l_sztuk)
from sprzedaz sp, sklepy sk, produkty pr
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id
group by sk.nazwa, pr.prod_nazwa;
```

```
select sum(l_sztuk*cena_jedn)
from sprzedaz sp, sklepy sk, produkty pr
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id;
```

```
Plan wykonywania
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=2 Card=1 Bytes=13)
   1    0   SORT (AGGREGATE)
   2    1     TABLE ACCESS (FULL) OF 'MV_SUMA_SPRZEDAZY' (Cost=2
                                                          Card=82 Bytes=1066)
```

Aggregate rollup (2)

```
select sk.nazwa, sum(l_sztuk*cena_jedn)
from sprzedaz sp, sklepy sk, produkty pr
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id
group by sk.nazwa;
```

```
Plan wykonywania
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=4 Card=82 Bytes=2050)

   1    0   SORT (GROUP BY) (Cost=4 Card=82 Bytes=2050)
   2    1     TABLE ACCESS (FULL) OF 'MV_SUMA_SPRZEDAZY'
                                          (Cost=2 Card=Bytes=2050)
```

Join-back (1)

- **Perspektywa zmaterializowana jest łączona z jedną z jej tabel bazowych, w celu wyznaczenia wyników zapytania użytkownika**
- **Perspektywa musi zawierać albo klucz podstawowy tej tabeli bazowej lub ROWID rekordów tabeli bazowej**

```
create materialized view mv_suma_sprzedazy1
build immediate
refresh force
next sysdate+(1/(24*60*30))
enable query rewrite
as
select sp.sklep_id, pr.prod_nazwa, sum(l_sztuk*cena_jedn),
       count(l_sztuk)
from sprzedaz sp, produkty pr
where sp.produkt_id=pr.produkt_id
group by sp.sklep_id, pr.prod_nazwa;
```

Join-back (2)

```
select sk.nazwa, pr.prod_nazwa, sum(l_sztuk*cena_jedn),
       count(l_sztuk)
from sprzedaz sp, sklepy sk, produkty pr
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id
group by sk.nazwa, pr.prod_nazwa;
```

- **zależność funkcyjna SKLEP_ID → NAZWA**

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=7 Card=2 Bytes=136)
   1    0   SORT (GROUP BY) (Cost=7 Card=2 Bytes=136)
   2    1     HASH JOIN (Cost=5 Card=82 Bytes=5576)
   3    2     TABLE ACCESS (FULL) OF 'SKLEPY' (Cost=2 Card=2 Bytes=14)

   4    2       TABLE ACCESS (FULL) OF 'MV_SUMA_SPRZEDAZY1' (Cost=2
                                                 Card=82 Bytes=5002)
```

Join-back (3)

- **Również w poniższym przypadku mechanizm przepisywania zapytań zostanie wykorzystany**

```
create materialized view mv_suma_sprzedazy1
build immediate
refresh force on commit
enable query rewrite
as
select sp.sklep_id, pr.prod_nazwa, sum(l_sztuk*cena_jedn),
       count(l_sztuk)
from sprzedaz sp, produkty pr
where sp.produkt_id=pr.produkt_id
group by sp.sklep_id, pr.prod_nazwa;
```

```
select sk.nazwa, pr.prod_nazwa, sum(l_sztuk*cena_jedn),
       count(l_sztuk)
from sprzedaz sp, produkty pr, sklepy sk
where sp.produkt_id=pr.produkt_id
and sp.sklep_id=sk.sklep_id
group by sk.nazwa, pr.prod_nazwa;
```

Filtered data (1)

```
create materialized view mv_suma_sprzedazy3
build immediate
refresh force on commit
enable query rewrite
as
select sp.sklep_id, pr.prod_nazwa, sum(l_sztuk*cena_jedn), count(l_sztuk)
from sprzedaz sp, sklepy sk, produkty pr
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id and sk.sklep_id in (1, 2)
group by sp.sklep_id, pr.prod_nazwa;
```

```
select sp.sklep_id, pr.prod_nazwa, sum(l_sztuk*cena_jedn),
       count(l_sztuk)
from sprzedaz sp, sklepy sk, produkty pr
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id
and sk.sklep_id=2
group by sp.sklep_id, pr.prod_nazwa;
```

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=2 Card=2 Bytes=28)
   1    0   TABLE ACCESS (FULL) OF 'MV_SUMA_SPRZEDAZY3' (Cost=2 Card=2 Bytes=28)
```

Filtered data (2)

- **Wyliczanie dodatkowych agregatów, np. AVG, na podstawie perspektywy zmaterializowanej**

```
create materialized view mv_suma_sprzedazy4
build immediate
refresh force on commit
enable query rewrite
as
select sp.sklep_id, pr.prod_nazwa, sum(l_sztuk*cena_jedn),
       count(l_sztuk*cena_jedn)
from sprzedaz sp, sklepy sk, produkty pr
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id
and sk.sklep_id in (1, 2)
group by sp.sklep_id, pr.prod_nazwa;
```

muszą wystąpić identyczne atrybuty/wyrażenia

```
select sk.nazwa, pr.prod_nazwa, avg(l_sztuk*cena_jedn)
from sprzedaz sp, sklepy sk, produkty pr
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id
and sk.sklep_id=2
group by sk.nazwa, pr.prod_nazwa;
```

Filtered data (3)

- **Wyliczenie AVG(x) w zapytaniu użytkownika będzie możliwe jeśli perspektywa zmaterializowana będzie zawierała SUM(x) i COUNT(x)**

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=5 Card=1 Bytes=55)
   1    0   SORT (GROUP BY) (Cost=5 Card=1 Bytes=55)
   2    1     NESTED LOOPS (Cost=3 Card=2 Bytes=110)
   3    2       TABLE ACCESS (BY INDEX ROWID) OF 'SKLEPY' (Cost=1
                                                  Card=1 Bytes=7)

   4    3         INDEX (UNIQUE SCAN) OF 'SKLEPY_PK' (UNIQUE)
   5    2       TABLE ACCESS (FULL) OF 'MV_SUMA_SPRZEDAZY4' (Cost=2
                                                  Card=2 Bytes=96)
```

Rollup using dimension (1)

```
create table czas
(data date CONSTRAINT czas_pk PRIMARY KEY,
 nr_dnia_mies number (2,0),
 nr_mies number (2,0),
 nr_kwart number (2,0),
 rok number (4,0),
 nr_dnia_roku number (3,0),
 dzien_tyg varchar2(12),
 nr_tyg number (2,0));
```

```
create dimension CZAS
  level l_data is czas.data
  level l_mies is czas.nr_mies
  level l_kwart is czas.nr_kwart
  level l_rok is czas.rok
  HIERARCHY czas_hier (
    l_data child of
    l_mies child of
    l_kwart child of
    l_rok);
```

```
create materialized view mv_suma_sprzedazy5
build immediate
refresh force on commit
enable query rewrite
as
select pr.prod_nazwa, cz.nr_mies, sum(l_sztuk*cena_jedn),
       count(l_sztuk*cena_jedn)
from sprzedaz sp, produkty pr, czas cz
where sp.produkt_id=pr.produkt_id
and sp.data=cz.data
group by pr.prod_nazwa, cz.nr_mies;
```

Rollup using dimension (2)

- Należy ustalić wartość parametru QUERY_REWRITE_INTEGRITY na TRUSTED lub STALE_TOLERATED

```
alter session set QUERY_REWRITE_INTEGRITY='TRUSTED';
```

```
select pr.prod_nazwa, cz.rok, sum(l_sztuk*cena_jedn)
from sprzedaz sp, produkty pr, czas cz
where sp.produkt_id=pr.produkt_id
and sp.data=cz.data
group by pr.prod_nazwa, cz.rok;
```

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=13 Card=2 Bytes=92)
   1    0   SORT (GROUP BY) (Cost=13 Card=2 Bytes=92)
   2    1     HASH JOIN (Cost=8 Card=682 Bytes=31372)
   3    2       TABLE ACCESS (FULL) OF 'MV_SUMA_SPRZEDAZY5' (Cost=2 Card=5
                                                             Bytes=100)

   4    2       VIEW (Cost=5 Card=409 Bytes=10634)
   5    4         SORT (UNIQUE) (Cost=5 Card=409 Bytes=10634)
   6    5           TABLE ACCESS (FULL) OF 'CZAS' (Cost=2 Card=409
                                                   Bytes=10634)
```

Rollup using dimension (3)

- **Wykorzystanie zależności funkcyjnych między atrybutami tabeli, zdefiniowanych w wymiarze**

```
create table sklepy
(sklep_id number(3),
 nazwa varchar2(20) not null,
 rodzaj_sklepu varchar2(10) not null,
 miasto varchar2(15) not null,
 wojewodztwo varchar2(30) not null);
```

```
create dimension SKLEPY
 level l_sklep is sklepy.sklep_id
 level l_miasto is sklepy.miasto
 level l_wojew is sklepy.wojewodztwo
 HIERARCHY sklepy_hier (
   l_sklep child of
   l_miasto child of
   l_wojew)
 ATTRIBUTE l_sklep
    DETERMINES (rodzaj_sklepu);
```

```
create materialized view
         mv_suma_sprzedazy6
build immediate
refresh force on commit
enable query rewrite
as
select sk.sklep_id, sum(l_sztuk*cena_jedn), count(l_sztuk*cena_jedn)
from sprzedaz sp, sklepy sk
where sp.sklep_id=sk.sklep_id
group by sk.sklep_id;
```

Rollup using dimension (4)

```
select sk.rodzaj_sklepu, sum(l_sztuk*cena_jedn)
from sprzedaz sp, sklepy sk
where sp.sklep_id=sk.sklep_id
group by sk.rodzaj_sklepu;
```

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=6 Card=2 Bytes=30)
   1    0   SORT (GROUP BY) (Cost=6 Card=2 Bytes=30)
   2    1     NESTED LOOPS (Cost=4 Card=2 Bytes=30)
   3    2       TABLE ACCESS (FULL) OF 'MV_SUMA_SPRZEDAZY6' (Cost=2
                                                 Card=2 Bytes=8)

   4    2       TABLE ACCESS (BY INDEX ROWID) OF 'SKLEPY' (Cost=1
                                                 Card=1 Bytes=11)

   5    4         INDEX (UNIQUE SCAN) OF 'SKLEPY_PK' (UNIQUE)
```

- **Uwaga: jeżeli tabela SPRZEDAZ będzie posiadała klucz podstawowy, to powyższe zapytanie zostanie również przepisane nawet jeśli nie zdefiniowano zależności funkcyjnych w wymiarze**

QUERY_REWRITE_INTEGIRTY (1)

- **Parametr określający jakie perspektywy zmaterializowane, ograniczenia integralnościowe i obiekty typu DIMENSION zostaną wykorzystane do przepisywania zapytań**
- **Modyfikowany dla całej instancji lub sesji**
- **Dopuszczalne wartości**
  - **ENFORCED (wartość domyślna)**
    - **Oracle zapewnia, że wynik zapytania przepisanego będzie identyczny z wynikiem oryginalnego zapytania na tabelach bazowych**
    - **Sprawdza poprawność danych weryfikując za każdym razem ograniczenia integralnościowe**
    - **Nie wykorzystuje obiektów typu DIMENSION**
  - **STALE_TOLERATED**
    - **Oracle wykorzysta perspektywy zmaterializowane, których zawartość albo nie jest aktualna albo nie odzwierciedla (zagregowanych) danych z tabel bazowych (dla perspektyw modyfikowalnych)**

QUERY_REWRITE_INTEGIRTY (2)

- **TRUSTED** (zalecany w magazynach danych) pod warunkiem, że integralność danych zostanie zapewniona przez oprogramowanie zarządzające magazynem (również aplikacje)
  - Oracle wykorzysta:
    - obiekty typu DIMENSION
    - ograniczenia integralnościowe oznaczone jako poprawne → RELY

```
ALTER TABLE nazwa MODIFY CONSTRAINT nazwa_ogr RELY;
```

    - perspektywy zbudowane w oparciu o istniejące tabele (ON PREBUILT TABLE)
    - tylko te perspektywy zmaterializowane, które zawierają dane aktualne
- Uwaga: jeżeli zapytanie nie zostanie przepisane w trybie STALE_TOLERATED, to w innych trybach na pewno nie zostanie przepisane
- Określenie aktualności danych zmaterializowanej perspektywy
  - **USER_MVIEWS.STALENESS** → wartości FRESH lub STALE (NEEDS_COMPILE)

Warunki niezbędne do przepisania zapytania - podsumowanie

- **Uprawnienia użytkownika:**
  - **uprawnienie obiektowe QUERY REWRITE na wszystkich tabelach, do których odwołuje się perspektywa zmaterializowana**
  - **lub uprawnienie systemowe GLOBAL QUERY REWRITE**
- **Włączony mechanizm przepisywania zapytań dla perspektywy zmaterializowanej**
  - **klauzula ENABLE QUERY REWRITE**
  - **lub włączenie dynamicznie (`alter materialized view`)**
- **Włączony mechanizm przepisywania w systemie**
  - **parametr konfiguracyjny QUERY_REWRITE_ENABLED=TRUE**
  - **lub włączenie dynamicznie dla sesji (`alter session`)**
- **Wykorzystywany optymalizator kosztowy**
  - **OPTIMIZER_MODE=CHOOSE (parametr konfiguracyjny, ustawienie dla sesji)**
  - **dla tabel bazowych i perspektyw zmaterializowanych zebrano statystyki**
- **QUERY_REWRITE_INTEGRITY= ENFORCED | STALE_TOLERATED | TRUSTED (parametr konfiguracyjny, ustawienie dla sesji)**

Wskazówki optymalizatora zapytań

- **REWRITE (nazwa_perspektywy)**
- **NOREWRITE**

```
select /*+ REWRITE(MV_SUMA_SPRZEDAZY6) */ sk.sklep_id,
                                           sum(l_sztuk*cena_jedn)
from sprzedaz sp, sklepy sk
where sp.sklep_id=sk.sklep_id
group by sk.sklep_id;
```

```
select /*+ NOREWRITE */ sk.sklep_id,
                         sum(l_sztuk*cena_jedn)
from sprzedaz sp, sklepy sk
where sp.sklep_id=sk.sklep_id
group by sk.sklep_id;
```

- **Uwaga: jeżeli we wskazówce REWRITE podamy perspektywę, której nie da się wykorzystać do przepisania zapytania, to optymalizator wybierze perspektywę właściwą**

GROUPING SETS (1)

```
create materialized view mv_suma_sprzedazy7
build immediate
refresh force
next sysdate+(1/(24*60*30))
enable query rewrite
as
select sk.nazwa, pr.prod_nazwa, cz.rok, sum(l_sztuk*cena_jedn),
       count(l_sztuk), GROUPING_ID(sk.nazwa, pr.prod_nazwa, cz.rok)
from sprzedaz sp, sklepy sk, produkty pr, czas cz
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id
and sp.data=cz.data
group by GROUPING SETS ((sk.nazwa, pr.prod_nazwa),
                        (pr.prod_nazwa, cz.rok));
```

Perspektywa zmaterializowana musi zawierać funkcję GROUPING lub GROUPING_ID, w przeciwnym przypadku zapytanie nie zostanie przepisane

```
select sk.nazwa, pr.prod_nazwa, sum(l_sztuk*cena_jedn), count(l_sztuk)
from sprzedaz sp, sklepy sk, produkty pr
where sp.sklep_id=sk.sklep_id
and sp.produkt_id=pr.produkt_id
group by GROUPING SETS ((sk.nazwa, pr.prod_nazwa));
```

Procedura EXPLAIN_REWRITE

- Analizuje możliwość przepisania zapytania w oparciu o wskazaną perspektywę zmaterializowaną
- Wynik analizy w tabeli REWRITE_TABLE
  - należy ją utworzyć w schemacie użytkownika → skrypt ORACLE_HOME\rdbms\admin\utlxrw.sql

```
BEGIN
DBMS_MVIEW.EXPLAIN_REWRITE
('select sk.nazwa, pr.prod_nazwa, sum(l_sztuk*cena_jedn),
  count(l_sztuk)
  from sprzedaz sp, sklepy sk, produkty pr
  where sp.sklep_id=sk.sklep_id
  and sp.produkt_id=pr.produkt_id
  group by GROUPING SETS ((sk.nazwa, pr.prod_nazwa))',
                                        'MV_SUMA_SPRZEDAZY7');
END;
```

```
select mv_name, message from rewrite_table;
```

```
MV_NAME                MESSAGE
---------------------- ----------------------------------------------
MV_SUMA_SPRZEDAZY7     QSM-01033: query rewritten with materialized
                       view, MV_SUMA_SPRZEDAZY7
```