Perspektywy zmaterializowane

1. Uruchom skrypty perspektywy_schemat.sql i perspektywy_dane.sql tworzące schemat hurtowni danych i wypełniające tabele danymi.

2. Utwórz perspektywę materializowaną o nazwie PURCHASES_SUM_MV, która będzie przechowywała zagregowane dane o sumarycznej kwocie (PURCHASE_PRICE) i liczbie sprzedanych produktów (PRODUCT_ID). Perspektywa powinna wykorzystywać tylko tabelę PURCHASES.

Parametry odświeżania perspektywy to:

- pierwsze odświeżenie natychmiast po utworzeniu,
- odświeżanie automatyczne co 1 minutę,
- sposób odświeżania: domyślny (FORCE),
- wybierz samodzielnie sposób identyfikowania rekordów (na podstawie definicji tabeli PURCHASES i zapytania definiującego perspektywę).

Sprawdź w słowniku danych informacje o zdefiniowanej perspektywie (sposób odświeżania, zapytanie definiujące, świeżość danych, itd.) oraz o momencie i typie ostatniego odświeżenia perspektywy. Odczytaj zawartość perspektywy.

3. Wstaw wiersz do tabeli PURCHASES poniższym poleceniem, a następnie zatwierdź transakcję.

INSERT INTO purchases VALUES ('SP1067', DATE '1999-01-01', 'AB123460', DATE '1999-01-01', 1024, 3.14, 2.71, 'TX', 'N');

4. Poczekaj kilka minut na automatyczne odświeżenie perspektywy raz po raz odczytując jej zawartość.

5. Ponownie wstaw wiersz do tabeli PURCHASES (tym samym poleceniem INSERT co poprzednio), a następnie zatwierdź transakcję.

6. Odśwież ręcznie perspektywę (korzystając z pakietu DBMS_MVIEW). Po odświeżeniu odczytaj jej zawartość. W jakim trybie (FAST/COMPLETE) wg Ciebie nastąpiło odświeżenie perspektywy?

7. Uruchom skrypt utlxmv.sql tworzący tabelę MV_CAPABILITIES_TABLE, wykorzystywaną przez procedurę z której skorzystamy w kolejnym punkcie ćwiczenia.

8. Sprawdź możliwości perspektywy PURCHASES_SUM_MV za pomocą procedury DBMS_MVIEW.EXPLAIN_MVIEW. Czy może ona być odświeżania w trybie FAST? Jak myślisz, dlaczego?

9. Usuń perspektywę PURCHASES_SUM_MV.

10. Utwórz ponownie perspektywę materializowaną o nazwie PURCHASES_SUM_MV, opartą na tym samym zapytaniu co poprzednio, ale z innymi parametrami odświeżania:

- odświeżanie po zatwierdzeniu transakcji,
- sposób odświeżania: FAST,
- ten sam sposób identyfikowania rekordów co poprzednio.

Czy udało się utworzyć perspektywę? Jeśli nie, zinterpretuj błąd zwrócony przez system.

11. Utwórz dziennik (materialized view log) na tabeli PURCHASES, spełniający wymogi odświeżania przyrostowego perspektywy PURCHASES_SUM_MV, którą chcemy utworzyć:

- sposób identyfikacji rekordów
- kolumny filtrujące potrzebne do odświeżania agregatu
- parametr zapewniający przechowywanie w dzienniku porządku operacji DML na tabeli bazowej
- klauzula zapewniająca przechowywanie zarówno starych jak i nowych wartości w dzienniku (wymagane do przyrostowego odświeżania agregatów)

12. Po utworzeniu dziennika ponów próbę utworzenia perspektywy z pkt 11. Jeśli ponownie utworzenie perspektywy się nie uda z powodu niewystarczających informacji w dzienniku, popraw definicję dziennika (DROP/CREATE lub ALTER), aż uda się utworzyć perspektywę. Po utworzeniu perspektywy odczytaj jej zawartość.

13. Wstaw kolejny wiersz do tabeli PURCHASES (tym samym poleceniem INSERT co wcześniej), a następnie zatwierdź transakcję.

14. Odczytaj zawartość perspektywy. Czy nastąpiło odświeżenie?

15. Korzystając z funkcji EXPLAIN PLAN sprawdź czy system skorzystałby z mechanizmu przepisywania zapytań (query rewrite) dla zapytania identycznego z tym na którym oparta jest utworzona perspektywa zmaterializowana.

16. Dlaczego system nie przepisał zapytania na perspektywę? Zanotuj szacowany koszt wykonania zapytania bez wykorzystania perspektywy zmaterializowanej.

17. Zmodyfikuj perspektywę (poleceniem ALTER MATERIALIZED VIEW) tak aby była dostępna dla mechanizmu przepisywania zapytań.

18. Ponownie sprawdź plan wykonania zapytania identycznego z tym na którym oparta jest utworzona perspektywa zmaterializowana. Czy plan się zmienił? Czy szacowany koszt wykonania jest zauważalnie mniejszy niż poprzednio?

19. Usuń perspektywę PURCHASES_SUM_MV i dziennik na tabeli PURCHASES.

20. Utwórz perspektywę zmaterializowaną SALES_MV, której definicję przedstawiono poniżej.

```
CREATE MATERIALIZED VIEW SALES_MV
BUILD IMMEDIATE
REFRESH FORCE
ON DEMAND
ENABLE QUERY REWRITE
AS
SELECT SUM (purchase_price), state_id
FROM purchases
GROUP BY state_id;
```

21. Sprawdź plan wykonania poniższego zapytania.

```
SELECT SUM (purchase_price), state_id
FROM purchases
GROUP BY state_id;
```

22. Sprawdź aktualność danych perspektywy SALES_MV odpytując słownik bazy danych (kolumna STALENESS perspektywy USER_MVIEWS).

23. Zmodyfikuj zawartość tabeli PURCHASES (możesz wykorzystać polecenie INSERT z wcześniejszych punktów ćwiczenia) i zatwierdź transakcję.

24. Ponownie sprawdź aktualność danych perspektywy SALES_MV odpytując słownik bazy danych.

25. Ponownie sprawdź plan wykonania zapytania z pkt 22.

26. Wykonaj polecenie
```
ALTER SESSION SET QUERY_REWRITE_INTEGRITY=STALE_TOLERATED;
```

27. Ponownie sprawdź plan wykonania zapytania z pkt 22.

28. Przywróć domyślną wartość parametru QUERY_REWRITE_INTEGRITY poleceniem
```
ALTER SESSION SET QUERY_REWRITE_INTEGRITY=ENFORCED;
```

29. Ponownie sprawdź plan wykonania zapytania z pkt 22.

30. Co należy zrobić, aby perspektywa SALES_MV była znowu używana przy przepisywaniu zapytań w trybie poziomu integralności ENFORCED? Wykonaj niezbędne kroki i jeszcze raz sprawdź plan wykonania zapytania z pkt 22 (należy osiągnąć stan w którym system podejmie decyzję o przepisaniu zapytania na perspektywę w trybie poziomu integralności ENFORCED).

31. Usuń perspektywę SALES_MV.

32. Utwórz perspektywę zmaterializowaną CUST_SALES_MV, której definicję przedstawiono poniżej.

```
CREATE MATERIALIZED VIEW cust_sales_mv
ENABLE QUERY REWRITE
AS
SELECT c.customer_id, SUM(ps.purchase_price) as sum_of_sales,
     COUNT (ps.purchase_price) as total_sales
FROM customer c, purchases ps
WHERE c.customer_id = ps.customer_id
GROUP BY c.customer_id;
```

33. Sprawdź plan wykonania poniższego zapytania.

```
SELECT c.occupation, SUM(ps.purchase_price) as sum_of_sales
FROM purchases ps, customer c
WHERE c.customer_id = ps.customer_id
GROUP BY c.occupation;
```

Dlaczego system mógł przepisać zapytanie na perspektywę mimo, że w perspektywie nie ma podsumowań na wymaganym poziomie agregacji?

34. Usuń klucz główny tabeli CUSTOMER (kaskadowo usuwając odwołujące się do niego klucze obce).

35. Ponownie sprawdź plan wykonania zapytania z pkt 34. Czy system dalej planuje przepisać zapytanie na perspektywę? Dlaczego?

36. Zdefiniuj wymiar (obiekt DIMENSION), który umożliwi przepisanie zapytania z pkt 34.
Wskazówka: Wymiar powinien zawierać hierarchię wiążącą zawody z klientami albo zależność funkcyjną między klientem a zawodem.

37. Aby system korzystał z informacji zawartej w obiektach wymiarów przy przepisywaniu zapytań, parametr QUERY_REWRITE_INTEGRITY musi mieć odpowiednią wartość. Ustaw ją, jeśli aktualnie obowiązuje domyślna (ENFORCED).

38. Ponownie sprawdź plan wykonania zapytania z pkt 34. Czy teraz system planuje przepisać zapytanie na perspektywę? Jeśli nie, popraw definicję wymiaru lub wartość parametru QUERY_REWRITE_INTEGRITY aż system skorzysta z przepisania zapytania mimo braku klucza głównego w tabeli CUSTOMER.

39. Usuń perspektywę CUST_SALES_MV.

40. (zadanie dla chętnych, ale obowiązkowe dla osób nieobecnych na zajęciach)
Utwórz tabelę PURCHASES_AVG o następującej strukturze:

```
PRODUCT_ID         varchar2(8) not null
AVG_PURCHASE_PRICE  number
```

Wypełnij tabelę PURCHASES_AVG danymi nt. średniej ceny sprzedaży produktów (wykorzystaj polecenie CREATE TABLE ... AS SELECT). Następnie dokonaj transformacji tej tabeli do perspektywy

zmaterializowanej (użyj opcji ON PREBUILT TABLE). Spróbuj dokonać modyfikacji danych w PURCHASES_AVG. Następnie usuń perspektywę PURCHASES_AVG. Co się dzieje z wcześniej utworzoną tabelą?

41. Usuń wszystkie tabele utworzone w ramach ćwiczeń.