Rozdział 8
Perspektywy

Stosowanie perspektyw, tworzenie perspektyw prostych i złożonych, perspektywy modyfikowalne i niemodyfikowalne, pseudokolumny ROWID i ROWNUM.

Perspektywa

Perspektywa (ang. *view*) jest strukturą logiczną udostępniającą wybrane informacje przechowywane w relacjach bazy danych

Własności

- **definiowana w oparciu o relacje (relacje bazowe) lub inne perspektywy (perspektywy bazowe),**
- **nie posiada własnych danych, nie jest materializowana na dysku**
- **przechowywana w postaci zapytania**

Cel stosowania

- **ograniczenie dostępu do danych - zabezpieczenie przed nieautoryzowanym dostępem**
- **uproszczenie schematu bazy danych, uproszczenie zapytań**
- **uniezależnienie aplikacji od zmian w strukturze bazy danych**
- **prezentowanie danych w inny sposób niż dane w relacjach i perspektywach bazowych (m.in. zmiana nazw atrybutów, formatów danych, itp.)**
- **dodatkowa kontrola poprawności wprowadzanych danych (perspektywy z kontrolą ograniczeń integralnościowych)**

Tworzenie perspektywy

- **Polecenie CREATE VIEW**

```
CREATE [OR REPLACE] VIEW nazwa_perspektywy
        [ ( kolumna1, kolumna2, ...) ]
AS
SELECT zapytanie definiujące perspektywę
[ WITH READ ONLY  | WITH CHECK OPTION [ CONSTRAINT
nazwa_ograniczenia ] ];
```

```
CREATE OR REPLACE VIEW prac_zesp_30
(id, nazwisko, posada, pensja) AS
     SELECT id_prac, nazwisko, etat, placa_pod
     FROM pracownicy WHERE id_zesp = 30;
```

```
SELECT * FROM prac_zesp_30;
```


Rodzaje perspektyw

proste

- **Oparte na jednej relacji bazowej**
- **Nie zawierają: operatorów zbiorowych, operatora DISTINCT, funkcji grupowych i analitycznych, grupowania, sortowania, klauzul CONNECT BY i START WITH, kolekcji i podzapytań w klauzuli SELECT**

złożone

- **Oparte na wielu relacjach i perspektywach bazowych**
- **Wykorzystują operatory zbiorowe, funkcje, grupowanie, sortowanie, połączenia, funkcje analityczne, itd.**

- **Perspektywy proste mogą służyć do wstawiania, modyfikowania i usuwania krotek z relacji bazowej. Perspektywy złożone służą tylko i wyłącznie do odczytu (istnieją nieliczne wyjątki od tej reguły).**

Perspektywy modyfikowalne (1)

- Perspektywa jest modyfikowalna (ang. *updatable*) jeśli nie zawiera: operatorów zbiorowych, operatora DISTINCT, funkcji grupowej lub analitycznej, klauzul GROUP BY, ORDER BY, CONNECT BY, START WITH, połączeń (z pewnymi wyjątkami), podzapytań w klauzuli SELECT.
- Jeśli perspektywa zawiera formuły lub pseudokolumny to polecenia INSERT i UPDATE nie mogą dotyczyć tych pseudokolumn.
- Jeśli perspektywa zawiera połączenie, to operacja DML musi dotyczyć tylko jednej relacji bazowej, a ponadto:
  - Dla operacji INSERT wszystkie wypełniane wartościami atrybuty muszą pochodzić z relacji zachowującej klucz, brak klauzuli WITH CHECK OPTION.
  - Dla operacji UPDATE wszystkie modyfikowane atrybuty muszą pochodzić z relacji zachowującej klucz; nie można stosować klauzuli WITH CHECK OPTION jeśli konieczne jest modyfikowanie wartości atrybutów z warunku połączeniowego; perspektywa nie może zawierać połączenia zwrotnego (przy obecności kl. WITH CHECK OPTION).

Perspektywy modyfikowalne (2)

- Dla operacji DELETE operacja połączenia może dotyczyć tylko jednej relacji zachowującej klucz, w przypadku kilku tabel zachowujących klucz rekordy są usuwane z tabeli pierwszej w kolejności w klauzuli FROM.
- Relacja zachowuje klucz, jeśli każda unikalna wartość w relacji jest też unikalna w perspektywie.

Tworzenie perspektywy złożonej (1)

- Perspektywa złożona niemodyfikowalna oparta na jednej relacji
  - perspektywa jest niemodyfikowalna, ponieważ zawiera funkcje agregujące oraz klauzulę GROUP BY

```
CREATE OR REPLACE VIEW place_etaty
(etat, srednia, maksymalna, liczba)
AS
SELECT etat, AVG(placa_pod), MAX(placa_pod), COUNT(*)
FROM pracownicy
GROUP BY etat
ORDER BY MAX(placa_pod) DESC;
```


Tworzenie perspektywy złożonej (2)

- Perspektywa złożona niemodyfikowalna oparta na wielu relacjach bazowych
  - perspektywa jest niemodyfikowalna ponieważ żadna z trzech tabel nie zachowuje kluczy, krotka z każdej tabeli występuje w perspektywie wielokrotnie

```
CREATE OR REPLACE VIEW prac_zesp_etat
(id, id_zesp, nazwisko, nazwa, etat,kategoria)
AS
SELECT p.id_prac, id_zesp, p.nazwisko, z.nazwa, p.etat, e.nazwa
FROM pracownicy p JOIN zespoly z USING (id_zesp)
        JOIN etaty e
        ON (p.placa_pod BETWEEN e.placa_min AND e.placa_max)
WHERE p.etat IN ('DYREKTOR','ASYSTENT','SEKRETARKA');
```


Tworzenie perspektywy złożonej (3)

- Perspektywa złożona modyfikowalna oparta na wielu relacjach bazowych
  - perspektywa jest modyfikowalna ponieważ tabela *Pracownicy* zachowuje klucz, każdy pracownik występuje w perspektywie co najwyżej raz

```
CREATE OR REPLACE VIEW prac_zesp (id, nazwisko, nazwa)
AS SELECT id_zesp, nazwisko, nazwa
FROM pracownicy JOIN zespoly USING (id_zesp);
```


Perspektywa weryfikująca ograniczenia

- Perspektywa weryfikująca ograniczenia

```
CREATE OR REPLACE VIEW
prac_minimum (id, nazwisko, placa, etat)
AS SELECT id_prac, nazwisko, placa_pod, etat
FROM pracownicy WHERE placa_pod < 1000
WITH CHECK OPTION CONSTRAINT za_wysoka_placa;
```

Polecenie modyfikujące dane udostępniane przez perspektywę zostanie odrzucone, jeśli w wyniku jego działania zbiór rekordów perspektywy zostałby pomniejszony.

```
UPDATE prac_minimum
SET placa = 2000 WHERE nazwisko = 'MAREK';

ORA-01402: naruszenie klauzuli WHERE dla perspektywy
z WITH CHECK OPTION
```

Mechanizm działa również dla polecenia INSERT.

Kompilowanie perspektywy

```
ALTER VIEW nazwa_perspektywy [COMPILE]
[ADD | MODIFY | DROP CONSTRAINT ograniczenie];
```

Usuwanie perspektywy

```
DROP VIEW nazwa_perspektywy [CASCADE CONSTRAINTS];
```

Słownik bazy danych

- opis zdefiniowanych perspektyw
  - USER_VIEWS
- informacje o możliwości uaktualniania danych dostępnych przez perspektywę
  - USER_UPDATABLE_COLUMNS

Pseudokolumny ROWID i ROWNUM

- ROWID – fizyczny adres krotki, składa się z numeru pliku, numeru bloku dyskowego wewnątrz pliku i numeru krotki wewnątrz bloku
- ROWNUM – numer krotki w zbiorze wynikowym, przyznawany w momencie odczytywania krotki z dysku (ale przed sortowaniem!)

```
SELECT ROWID, ROWNUM, nazwisko, etat, placa_pod
FROM pracownicy;
```

```
SELECT ROWID, ROWNUM, nazwisko, etat, placa_pod
FROM pracownicy ORDER BY placa_pod DESC;
```

```
SELECT ROWNUM, T.rnum, T.nazwisko, T.etat, T.pensja
FROM (
        SELECT ROWNUM AS rnum,
        nazwisko, etat, placa_pod AS pensja
        FROM pracownicy ORDER BY pensja DESC ) T
WHERE ROWNUM <= 3;
```