Kreiranje pogleda

```
CREATE VIEW <table name> [(<view column list>)]
AS <query expression>
[WITH [<levels clause>] CHECK OPTION]

<levels clause> ::= CASCADED | LOCAL
```

Ukoliko se koristi CHECK OPTION, ali LEVEL izostavi, onda se podrazumeva CASCADED.

Ažurabilni pogled

1. Kreiran nad samo jednom tabelom

2. Ne koristi GROUP BY klauzulu

3. Ne koristi HAVING klauzulu

4. Ne sadrži agregatne funkcije

5. Ne sadrži izračunate kolone

6. Ne koristi UNION, INTERSECT, ili EXCEPT

7. Ne koristi SELECT DISTINCT klauzulu

8. Svaka kolona osnovne tabele koja nije obuhvaćena pogledom mora biti NULL-abilna ili mora imati definisanu DEFAULT vrednost u osnovnoj tabeli, tako da se čitav red može konstruisati prilikom INSERT instrukcije

```
CREATE VIEW Test1 -- ažurabilan pogled, obuhvata primarni ključ!
AS SELECT *
   FROM Tabela
   WHERE x IN (1,2);



CREATE VIEW Test2 -- nije anije ažurabilan!
AS SELECT *
   FROM Tabela
   WHERE x = 1

   UNION ALL

   SELECT *
   FROM Tabela
   WHERE x = 2;
```

Ekspanzija definicije pogleda

Na primer pogled:

```
CREATE VIEW OdeljenjaProdaje (odeljenje, grad, ...)
AS SELECT 'Prodaja', grad, ...
   FROM Odeljenja
   WHERE odeljenje = 'Prodaja';
```

Ukoliko se iskoristi u upitu:

```
SELECT *
FROM OdeljenjaProdaje
WHERE grad = 'Beograd';
```

Dolazi do ekspanzije definicije upita, tako da nastaje:

```
SELECT *
FROM (SELECT 'Prodaja', grad, ...
      FROM Odeljenja
      WHERE odeljenje = 'Prodaja') AS OdeljenjaProdaje (odeljenje, grad, ...)
WHERE grad = 'Beograd';
```

Što dalje može biti optimizovano kao:

```
SELECT *
FROM Odeljenja
WHERE (odeljenje = 'Prodaja')
AND (grad = 'Beograd');
```

Materijalizacija pogleda

Materijalizacija pogleda znači da neće doći do INLINE ekspanzije definicije pogleda, već da će doći do fizičkog kreiranja tabele i njenog popunjavanja podacima koji se dobijaju kao rezultat izvršavanja definicionog upita. Materijalizaciju pogleda obično kontroliše DBMS, ali postoje i oni kod kojih se to može kontrolisati (npr. SNAPSHOT, RESULT SET). Ukoliko definicioni upite sadrži SELECT DISTINCT, UNION, GROUP BY, HAVING, ili bilo šta drugo što zahteva eliminaciju duplikata ili sortiranje rezultata definicionog upita, automatski se obavlja materijalizacija pogleda. Za poglede sa velikom količinom podataka može biti veoma zahtevna operacija. Takođe tabela koja je rezultat materijalizacije pogleda nema podršku za indekse i optimizacije.

Klauzula WITH CHECK OPTION

Ukoliko se posmatra pogled definisan sa:

```
CREATE VIEW V1
AS SELECT *
   FROM TestTabela
   WHERE kolona1 = 'A';
```

Potom izvrši iskaz za promenu podataka:

```
UPDATE V1 SET kolona1 = 'B';
```

To će biti izvršeno, ali će sledeća upotreba pogleda V1 vratiti prazan rezultat.

Ukoliko se koristi WITH CHECK OPTION onda sistem proverava uslov naveden u WHERE klauzuli prilikom INSERT i UPDATE naredbi. Ukoliko novi red, ili promena nad redom ne prođe proveru, onda će provera biti odbijena i pogled ostaje nepromenjen.

Upotreba WITH CHECK OPTION se razlikuje od CHECK klauzule u CREATE TABLE iskazu. Razlog je to što CHECK pored TRUE rezultata prihvata i UNKNOWN, odnosno NULL, dok WITH CHECK OPTION prihvata isključivo TRUE.

Na primer:

```
CREATE TABLE TestTabela (col_a INTEGER);

CREATE VIEW TestPogled (col_a)
AS SELECT col_a FROM TestTabela WHERE col_a > 0 WITH CHECK OPTION;

INSERT INTO TestPogled VALUES (NULL); -- Ovo ne uspeva!

CREATE TABLE TestTabela_2 (col_a INTEGER CHECK (col_a > 0));

INSERT INTO TestTabela_2(col_a) VALUES (NULL); -- Ovo uspeva!
```

Kada se koristi CASCADED, koji se inače implicitno podrazumeva, onda se pored WHERE klauzule definicionog upita proveravaju WHERE klauzule i svih pogleda od kojih definicioni upit zavisi. Ukoliko se koristi LOCAL onda se proverava samo WHERE klazula definicionog upita, a ne i pogleda od kojih on zavisi.

Na primer:

```
CREATE VIEW NiskaPlata
AS SELECT *
   FROM Osoblje
   WHERE plata <= 250;

CREATE VIEW SrednjaPlata
AS SELECT *
   FROM NiskaPlata
   WHERE plata >= 100;
```

Ukoliko se uradi:

```
UPDATE SrednjaPlata SET plata = 1000;
```

To prolazi, ali onda će sledeći poziv pogledu SrednjaPlata, vratiti prazan rezultat. Da je pogled SrednjaPlata sadržao WITH CASCADED CHECK OPTION onda bi UPDATE bio neuspešan. Međutim, da je pogled SrednjaPlata sadržao WITH LOCAK CHECK OPTION onda bi UPDATE bio uspešan, baš kao i kada nije bilo WITH CHECK OPTION.

Primer:

```
CREATE TABLE TABLE T1 (c1 INT, c2, INT, c3 INT, c4 INT, c5 INT);

INSERT INTO T1 VALUES (10, 10, 10, 10, 10);


CREATE VIEW V1 AS SELECT * FROM T1 WHERE (c1 > 5);

CREATE VIEW V2 AS SELECT * FROM V1 WHERE (c2 > 5) WITH xx CHECK OPTION;

CREATE VIEW V3 AS SELECT * FROM V2 WHERE (c3 > 5);

CREATE VIEW V4 AS SELECT * FROM V3 WHERE (c4 > 5) WITH yy CHECK OPTION;

CREATE VIEW V5 AS SELECT * FROM V4 WHERE (c5 > 5);
```

xx/yy	c1	c2	c3	c4	c5
cascade/cascade	F	F	F	F	S
local/cascade	F	F	F	F	S
local/local	S	F	S	F	S
cascade/local	F	F	S	F	S

```
UPDATE V5 SET c1 = 0;
```

Ovo za kombinaciju LOCAL/LOCAL prolazi, jer V5 ne proverava uslov, V4 ima LOCAL odnosno proverava samo njegov WHERE koji inače ne proverava c1, potom V3 ne proverava uslov, V2 proverava ali zbog LOCAL samo njegov WHERE koji takođe ne proverava c1, a poslednji V1 ne proverava uslov.

```
UPDATE V5 SET c1 = 0, c2 = 0, c3 = 0, c4 = 0, c5 = 0;
```

Ovo ne prolazi ni za jednu kombinaciju nivoa.

```
local/local
V1 = none
V2 = (c2 > 5)
V3 = (c2 > 5)
V4 = (c2 > 5) AND (c4 > 5)
V5 = (c2 > 5) AND (c4 > 5)

cascade/cascade
V1 = none
V2 = (c1 > 5) AND (c2 > 5)
V3 = (c1 > 5) AND (c2 > 5)
V4 = (c1 > 5) AND (c2 > 5) AND (c3 > 5) AND (c4 > 5)
V5 = (c1 > 5) AND (c2 > 5) AND (c3 > 5) AND (c4 > 5)
```

```
local/cascade
V1 = none
V2 = (c2 > 5)
V3 = (c2 > 5)
V4 = (c1 > 5) AND (c2 > 5) AND (c3 > 5) AND (c4 > 5)
V5 = (c1 > 5) AND (c2 > 5) AND (c3 > 5) AND (c4 > 5)

cascade/local
V1 = none
V2 = (c1 > 5) AND (c2 > 5)
V3 = (c1 > 5) AND (c2 > 5)
V4 = (c1 > 5) AND (c2 > 5) AND (c4 > 5)
V5 = (c1 > 5) AND (c2 > 5) AND (c4 > 5)
```

Upotreba WITH CHECK OPTION umesto CHECK()

S obzirom na to da većina DBMS ne podržava CHECK() klauzulu sa podupitima, onda se može koristiti pogled sa WITH CHECK OPTION.

Na primer umesto da se koristi provera na nivou tabele:

```
CREATE TABLE Hotel
(broj_sobe INTEGER NOT NULL,
datum_dolaska DATE NOT NULL,
datum_odlaska DATE NOT NULL,
ime_gosta CHAR(30) NOT NULL,
CONSTRAINT dobar_raspored CHECK (H1.datum_dolaska <= H1.datum_odlaska),
CONSTRAINT nema_duplog_izdavanja CHECK (NOT EXISTS (SELECT *
                                                    FROM Hotel AS H1, Hotel AS H2
                                                    WHERE H1.broj_sobe = H2.broj_sobe
                                                    AND H2.datum_dolaska < H1.datum_dolaska
                                                    AND H1.datum_dolaska < H2.datum_odlaska)));
```

Može se koristi:

```
CREATE VIEW Hotel_V (broj_sobe, datum_dolaska, datum_odlaska, ime_gosta)
AS SELECT H1.broj_sobe, H1.datum_dolaska, H1.datum_odlaska, H1.ime_gosta
   FROM Hotel AS H1
   WHERE NOT EXISTS (SELECT *
                     FROM Hotel AS H2
                     WHERE H1.broj_sobe = H2.broj_sobe
                     AND H2.datum_dolaska < H1.datum_dolaska
                     AND H1.datum_dolaska < H2.datum_odlaska)
   AND H1.datum_dolaska <= H1.datum_odlaska
WITH CHECK OPTION;


INSERT INTO Hotel_V VALUES (1, '2006-01-01', '2006-01-03', 'Pera Peric');
COMMIT;

INSERT INTO Hotel_V VALUES (1, '2006-01-03', '2006-01-05', 'Mika Mikic'); -- ovo ne prolazi!
```

Uklanjanje pogleda

```
DROP VIEW <table name> <drop behavior>

<drop behavior> ::= [CASCADE | RESTRICT]
```

Pri tome opcija CASCADE dovodi do toga da se prilikom uklanjanja pogleda, uklone i svi pogledi koji zavise od onog koji se uklanja.

Treba obratiti pažnju na to da prilikom ukanjanja tabele, mogu ostati pogledi kreirani nad tom tabelom.

```
CREATE TABLE TestTabela (col_a INTEGER);

CREATE VIEW TestView
AS SELECT col_a
   FROM TestTabela;

DROP TABLE TestTabela; -- uklanjanje osnovne tabele


CREATE TABLE TestTabela
(test_id CHAR(5) NOT NULL PRIMARY KEY,
col_a REAL NOT NULL); -- kreirana je nova tabela koja se isto zove i ima kolonu sa istim nazivom


-- stari pogled i dalje postoji, iako se tip kolone promenio

INSERT INTO TestTabela VALUES ('BrojPI', 3.14159); -- ovo prolazi!
```