**View**

**რა არის View?**

View არის ბაზის ობიექტი, რომელიც არის queryს result set ი. ანუ ის ინახავს მასთან ასოცირებულ query\_იებს და მისი გამოყენება შეიძლება ვირტუალურ ცხრილად. ანუ მასზეც შეგვიძლია დავწეროთ ნებისმიერი ტიპის query და გამოვიყენოთ როგორც ცხრილი. ეს ცხრილი შეიძლება იყოს, ერთი ცხრილიდან ამორჩევა, რამდენიმე ცხრილის გაერთიანება ან აგრეგატული მონაცემი.

მაგალითად:

CREATE VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

შემდეგ შეგვიძლია Select ის დაწერა ამ view ზე:

SELECT column1, column2, ...  
FROM view\_name   
WHERE condition2;

**Materialized View**

**რა არის Materialized View?**

Materialized view \_ არის მონაცემთა ბაზის ობიექტი, რომელიც ინახავს შესასრულებელ query სა და მის მიერ დაბრუნებულ მნიშვნელობას.

ანუ, მისი პირველი გამოძახებისას ის გამოიძახებს ნამდვილ query\_ს, შემდეგი გამოძახებებისას კი დააბრუნებს პირველი გამოძახების მიერ დაბრუნებულ მნიშვნელობას. ანუ მომდევნო შემდგომი გამოძახებები ამ materialized\_vew ზე იქნება ბევრად სწრაფი რადგან უბრალოდ მოუწევს ამ მნიშვნელობის მეხსიერებიდან ამოღება.

მაგრამ არის ერთი პრობლემა, როცა ცხრილი იცვლება, საჭიროა ამ materialized\_view განახლებაც. ეს განახლება არ ხდება ავტომატურად, მისთვის უნდა გამოვიძახოთ refresh.

CREATE MATERIALIZED VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

REFRESH MATERIALIZED VIEW view\_name;  
ამიტომ ის არ უნდა გამოვიყენოთ ზედმეტად. საუკეთესო გამოყენება არის მისი მაშინ, როცა ხდება ბევრად მეტი წაკითხვა ვიდრე ჩაწერა, ან მცირე აცდენას არ აქვს დიდი მნიშვნელობა რეალური მნიშვნელობიდან. მაგალითად: როცა არის რეპორტები მოსამზადებელი დიდ მონაცემებზე, მაშინ სწორია მისი გამოყენება.