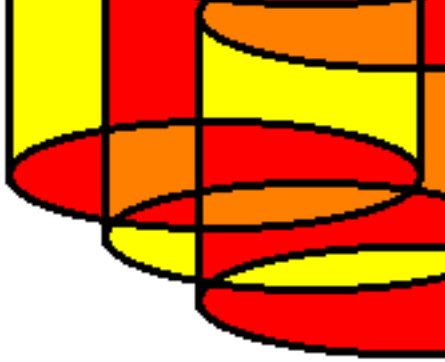
Aktivne baze podataka

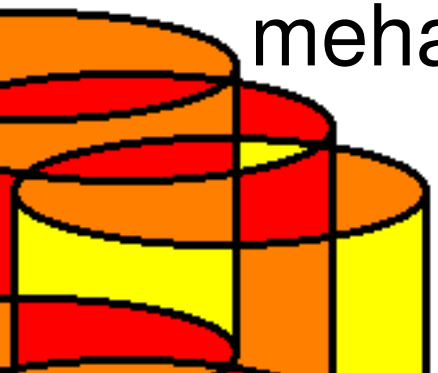


- Aktivne baze podataka su baze podataka koje uključuju aktivna pravila najčešće u obliku ECA (DUA)

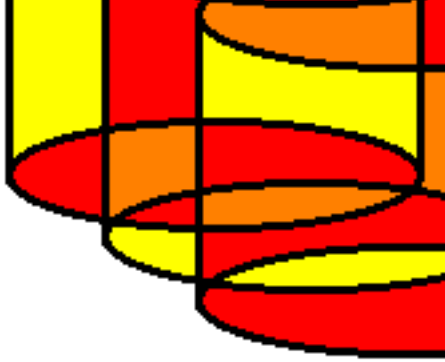
EVENT – **C**ONDITION – **A**CTION

DOGAĐAJ – **U**VJET – **A**KCIJA

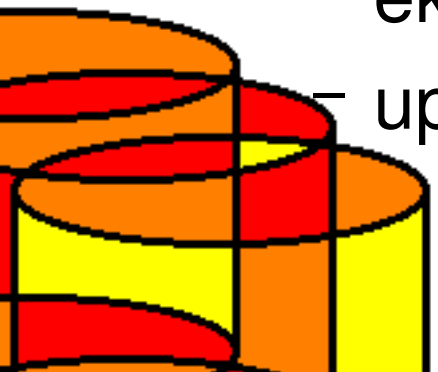
- Obogaćuju tradicionalne baze podataka s mehanizmima za procesiranje pravila



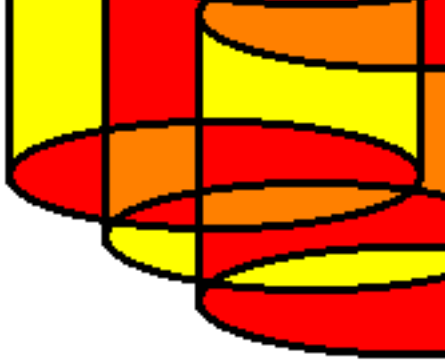
Aplikacijske domene



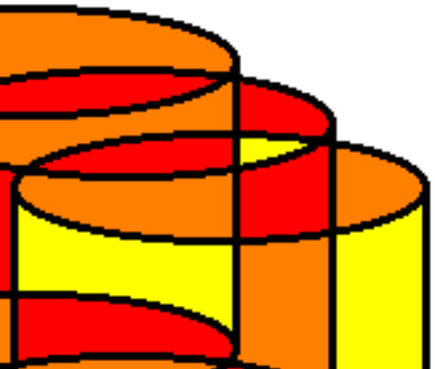
- Moguće aplikacije aktivnih baza podataka uključuju
  - napredne uvjete integriteta
  - napredne poglede
  - statističko praćenje
  - sigurnost baze podataka
  - sustavi temeljeni na znanju
  - ekspertni sustavi
  - upravljanje tokovima rada ...



Implementacija



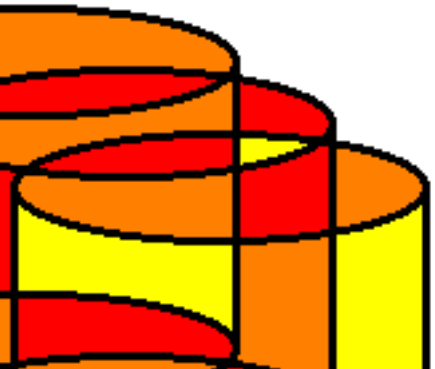
- Aktivne BP mogu se implementirati na dva načina:
  - Pomoću aktivnih pravila (CREATE RULE)
  - Pomoću okidača (CREATE TRIGGER)



Pomoću pravila

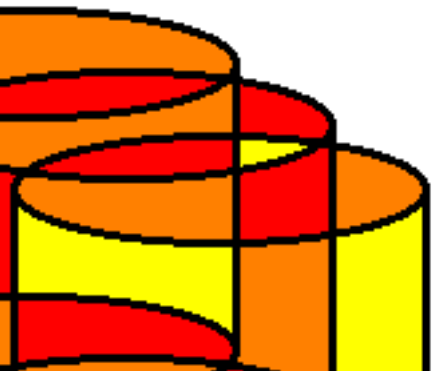
- Primjer – pravilo za kreiranje log zapisa za svaku promjenu e-mail adrese

```
CREATE TABLE osoba_log ( email TEXT, vrijeme TIMESTAMP
DEFAULT now() );

CREATE RULE osoba_log AS ON UPDATE TO osoba WHERE
NEW.email <> OLD.email DO INSERT INTO osoba_log VALUES
( OLD.email );

UPDATE osoba SET email = 'ivek@foi.hr' WHERE email =
'ivo@foi.hr';

SELECT * FROM osoba_log;
```

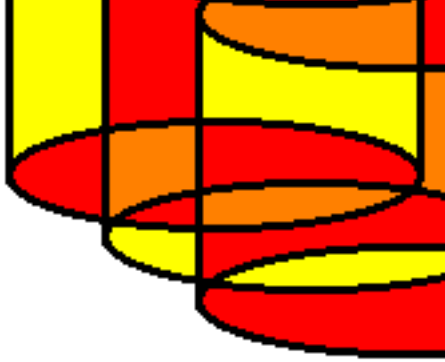


Okidači - sintaksa

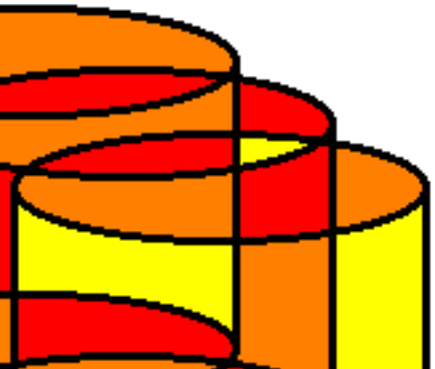
```
CREATE TRIGGER name
 { BEFORE | AFTER } { event [ OR ... ] }
 ON table
 [ FOR [ EACH ] { ROW | STATEMENT } ]
  EXECUTE PROCEDURE funcname ( arguments )
```



Primjer



- Okidač koji će prilikom upisivanja novog korisnika provjeriti je li njegova e-mail adresa sadrži znak '@'

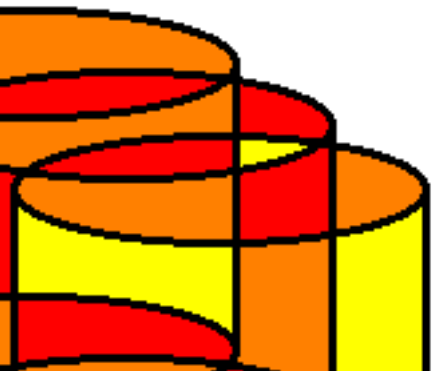


Rješenje - funkcija

```
CREATE OR REPLACE FUNCTION provjeri_email()
    RETURNS TRIGGER
    AS $$
       BEGIN
          IF NEW.email LIKE '%@%' THEN
             RETURN NEW;
          ELSE
             RAISE EXCEPTION '%',
             'E-mail adresa ne sadrzi znak ''@''';
          END IF;
       END;
    $$
    LANGUAGE plpgsql;
```

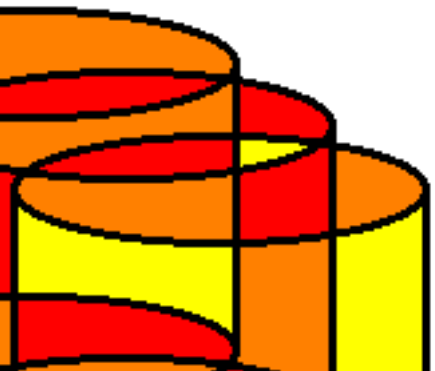
Rješenje - okidač

```
CREATE TRIGGER email_provjera
    BEFORE INSERT OR UPDATE
    ON osoba
    FOR EACH ROW
    EXECUTE PROCEDURE provjeri_email();
```

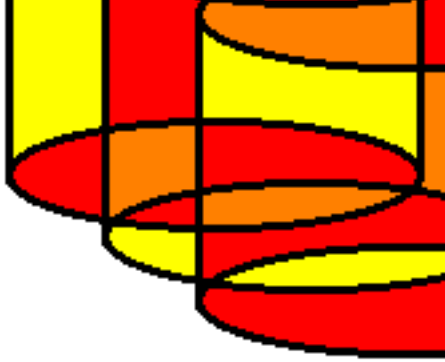


Isprobavanje

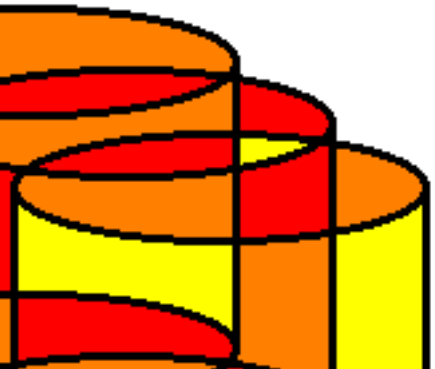
```
INSERT INTO osoba VALUES( 'spambot',
  'spambot', 'spambot', 'spambot',
  'spambot', 'musko' );
```



Primjer

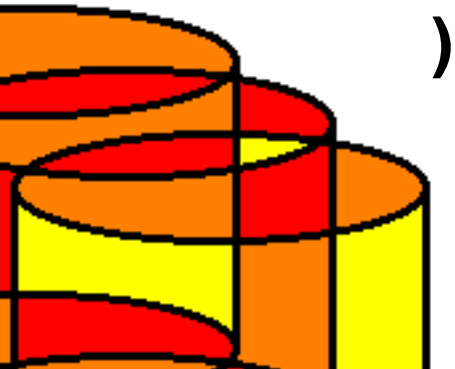


- Okidač koji će prilikom slanja zamolbe za vezu primatelju automatski poslati poruku da je dobio novu zamolbu



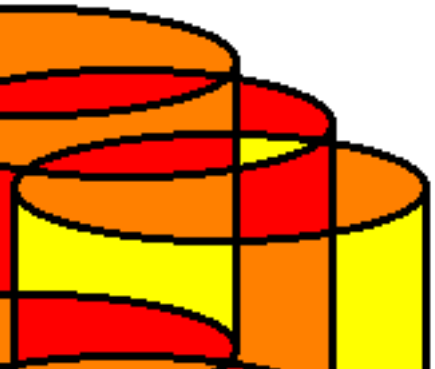
Rješenje – funkcija 1/2

```
CREATE OR REPLACE FUNCTION nova_zamolba()
 RETURNS TRIGGER
 AS $$
   DECLARE molitelj osoba;
   DECLARE sadrzaj_poruke TEXT;
   BEGIN
     IF NEW.status = 'poslano' THEN
       molitelj := (
         SELECT osoba
         FROM osoba
         WHERE email = NEW.poslao
       );
```



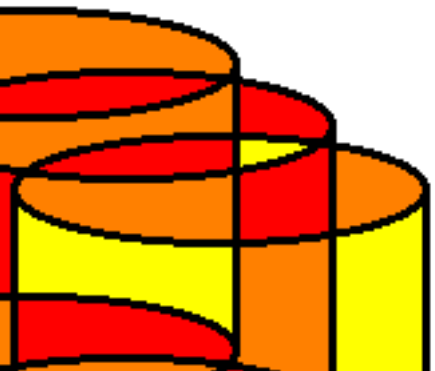
Rješenje funkcija 2/2

```
    sadrzaj_poruke := molitelj.ime || ' ' ||
      molitelj.prezime || ' ti je poslao
      zamolbu za prijateljstvom!';
    INSERT INTO poruka(
       posiljatelj, primatelj, naslov, sadrzaj )
    VALUES( NEW.poslao, NEW.prihvatio, 'Nova
      zamolba', sadrzaj_poruke );
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```



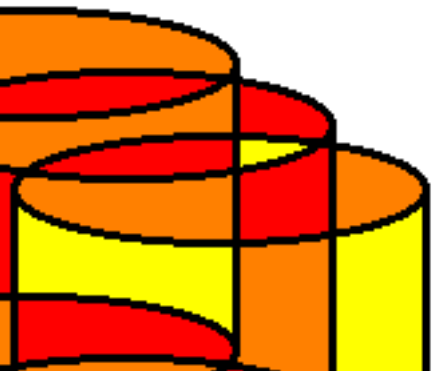
Rješenje - okidač

```
CREATE TRIGGER poruka_uz_zamolbu
    BEFORE INSERT
    ON veza
    FOR EACH ROW
    EXECUTE PROCEDURE nova_zamolba();
```

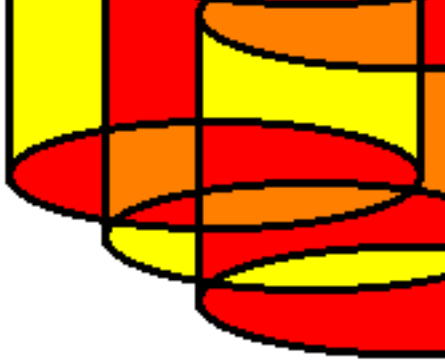


Isprobavanje

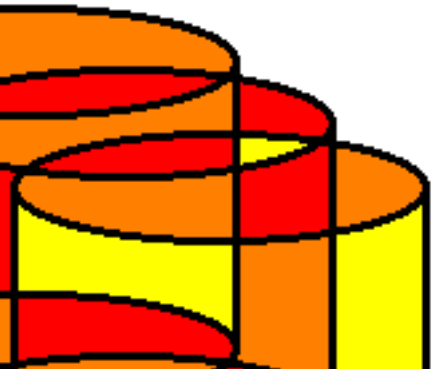
```
SELECT zamolba(
  'markus.schatten@foi.hr',
  'mirko.malekovic@foi.hr',
  'suradnik' );
```



Primjer



- Implementirajmo okidač i odgovarajuću funkciju koji će prilikom prihvaćanja nove veze (**`status = 'prihvaceno'`**) automatski upisati i suprotnu (komutativnu) vezu.



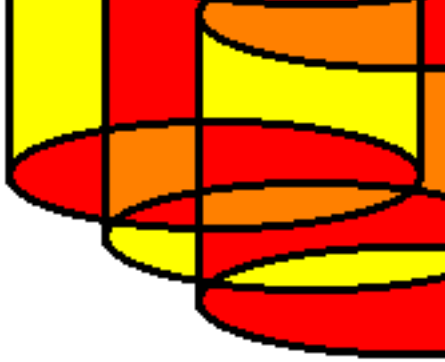
Rješenje – funkcija 1/2

```
CREATE OR REPLACE FUNCTION veza_prihvacena()
    RETURNS TRIGGER
    AS $$
       DECLARE postoji BOOLEAN;
       BEGIN
          IF NEW.status = 'prihvaceno' THEN
             postoji := EXISTS(
                SELECT veza
                FROM veza
                WHERE poslao = NEW.prihvatio
                AND prihvatio = NEW.poslao
                AND vrsta = NEW.vrsta
             );
```

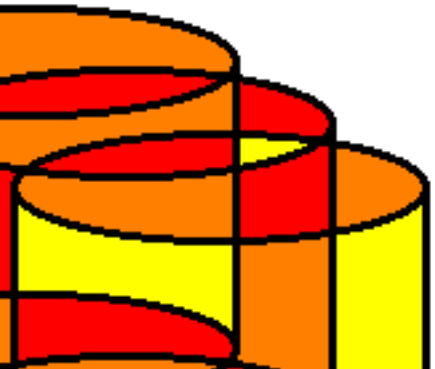
Rješenje – funkcija 2/2

```
            IF NOT postoji THEN
             INSERT INTO veza
             VALUES ( NEW.prihvatio, NEW.poslao,
                        NEW.vrsta, 'prihvaceno' );
            END IF;
         END IF;
         RETURN NEW;
      END;
$$
LANGUAGE plpgsql;
```

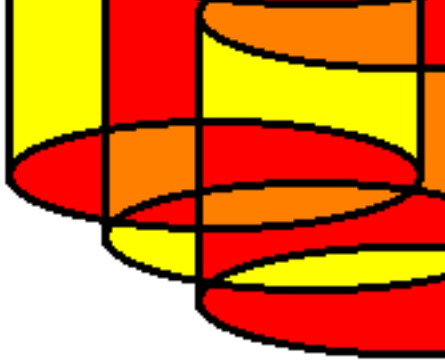
Rješenje - okidač



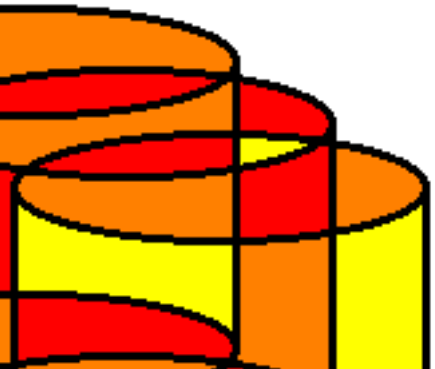
```
CREATE TRIGGER prihvat_veze
    AFTER INSERT OR UPDATE
    ON veza
    FOR EACH ROW
    EXECUTE PROCEDURE veza_prihvacena();
```



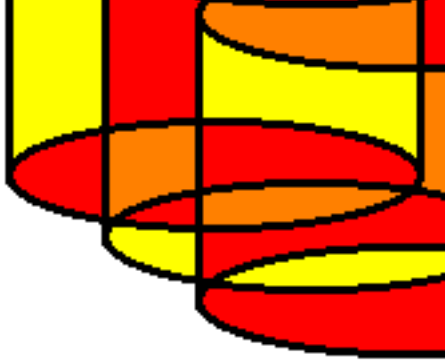
Isprobavanje



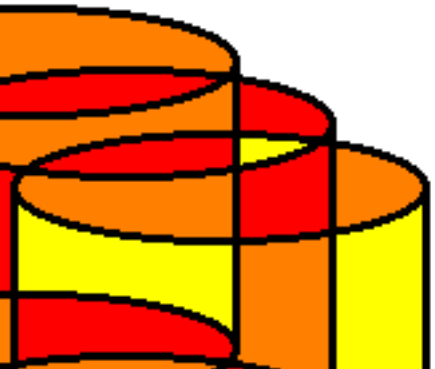
```
UPDATE veza SET status = 'prihvaceno';

SELECT * FROM veza;
```



Primjer

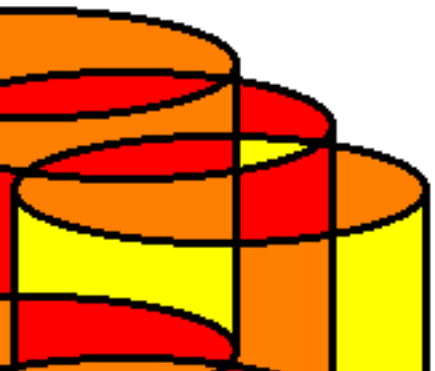


- Okidač koji će svim prijateljima osobe koja je postala član neke grupe poslati poruku o tome



Rješenje – funkcija 1/2

```
CREATE OR REPLACE FUNCTION novo_clanstvo()
    RETURNS TRIGGER
    AS $$
       DECLARE prijatelj TEXT;
       DECLARE naziv_grupe TEXT;
       DECLARE sadrzaj_poruke TEXT;
       DECLARE novi_clan osoba;
       BEGIN
          naziv_grupe := (
             SELECT naziv
             FROM grupa
             WHERE sifra = NEW.grupa
          );
```

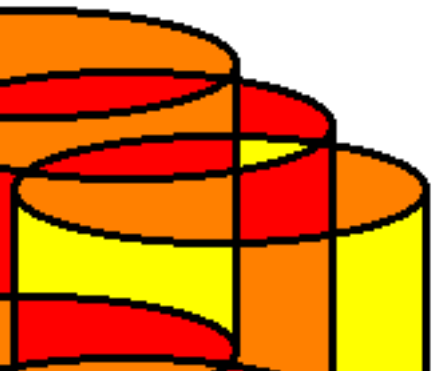


Rješenje – funkcija 2/2

```
        novi_clan := (
        SELECT osoba
           FROM osoba
           WHERE email = NEW.clan
        );
        sadrzaj_poruke := novi_clan.ime || ' '
         || novi_clan.prezime || ' je postao clan grupe '
         || naziv_grupe || '!';
        FOR prijatelj IN SELECT DISTINCT prihvatio FROM
                         veza WHERE poslao = NEW.clan LOOP
              PERFORM posalji_poruku( NEW.clan,
prijatelj,
                          'Novo clanstvo', sadrzaj_poruke );
        END LOOP;
        RETURN NEW;
    END;
    $$ LANGUAGE plpgsql;
```

Rješenje - okidač

```
CREATE TRIGGER novi_clan_grupe
    AFTER INSERT
    ON clanstvo
    FOR EACH ROW
    EXECUTE PROCEDURE novo_clanstvo();
```



Isprobavanje

```
SELECT upisi_u_grupu(
  'mirko.malekovic@foi.hr',
  'Baze podataka' );
```