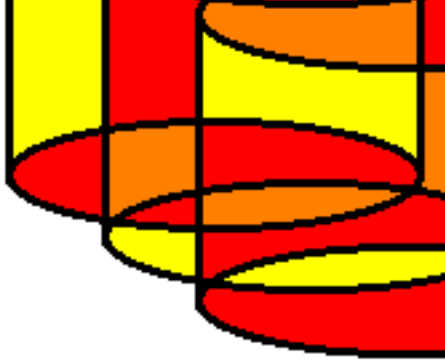
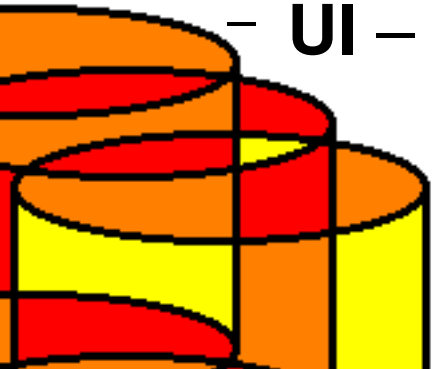
Deduktivne baze podataka

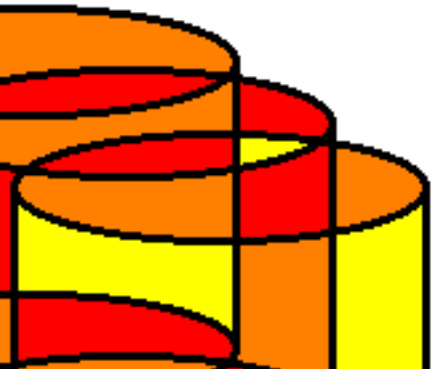


- proširenje relacijskih baza podataka deduktivnom komponentom
- Sastoje se od:
  - **EBP** – Eksplicitne (ekstenzijske) baze podataka – baza činjenica
  - **IBP** – Implicitne (intenzijske) baze podataka – baza znanja, baza pravila
  - **UI** – Uvjeta integriteta – integritetna ograničenja

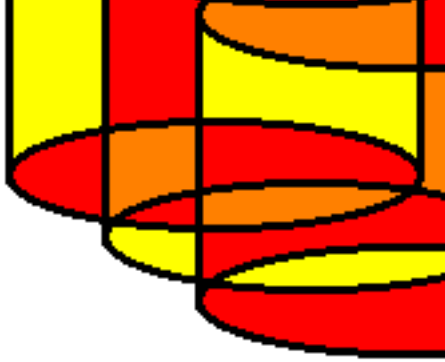


Deduktivna baza podataka je dakle uređena trojka:

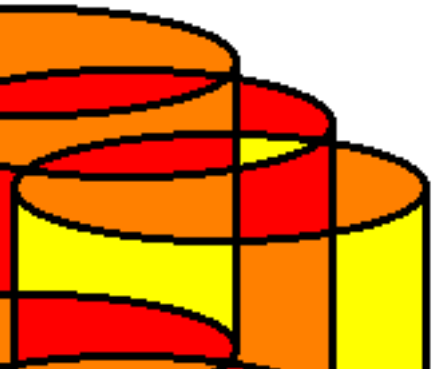
$$DBP = (EBP, IBP, UI)$$



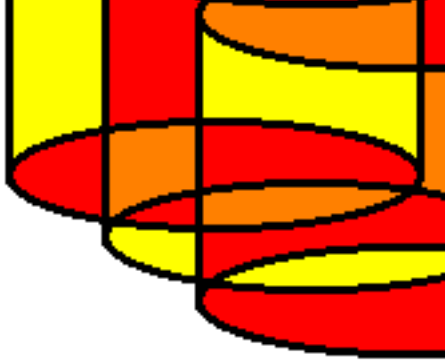
Datalog



- Datalog – jezik za rad s deduktivnim bazama podataka
- Nastao povezivanjem relacijskih baza podataka i logičkog programiranja (Prolog).



Sintaksa Dataloga



- Abeceda se sastoji od:

  1) Konstanti

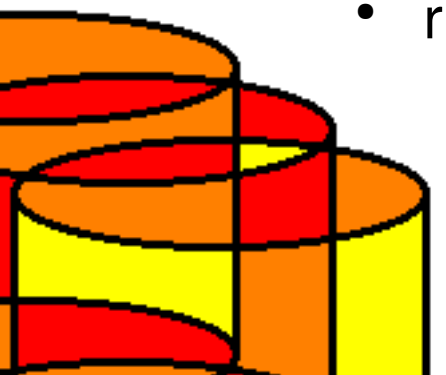
  - a, b, c, mrkva, ivo, 123, dugacka_konstanta, 'Jozo' ...

  2) Varijabli

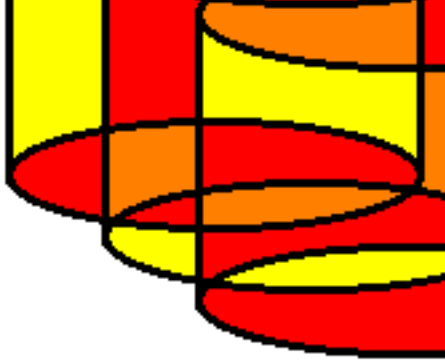
  - X, Y, Z, Avion, _, _nevazna, VrLo_dUgAcKa_VaRiJaBlA ...

  3) Relacija ili predikata

  - r1, r2, naziv_relacije, osoba, knjiga, posudba ...



Sintaksa Dataloga



4) (Ugrađenih) **relacija uspoređivanja**

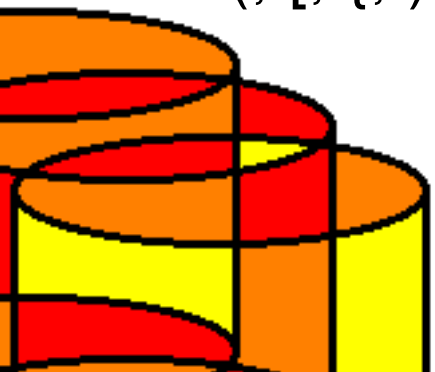
=, <, >, =<, >= **...**

5) Logički veznici

; , :- **...**

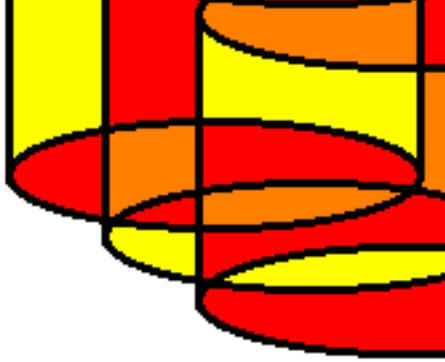
6) Zagrade

(, [, {,),], }

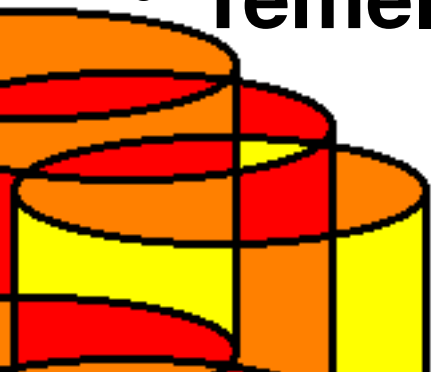


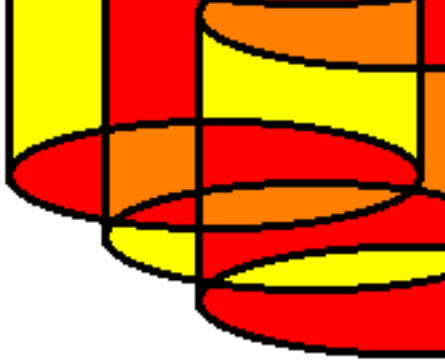
Pri čemu su skupovi 1) – 6) konačni

Osnovni pojmovi



- **Term** je konstanta ili varijabla.
- **Atom** je izraz oblika $r(t_1, t_2, .., t_n)$, gdje je r n-mjesna (n-arna) relacija i $t_1, t_2,.., t_n$ su termi.
- Izraz $(t_1, t_2,.., t_n)$ je **slog**.
- **Temeljni atom** je atom bez varijabli.





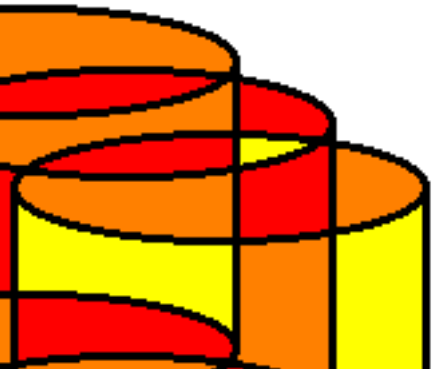
Primjeri

a, a_1, Y, Y_1, Z su termi;

(a, b, X) je slog;

r(a, b, X) je atom ; r(a, b, c) je temeljni atom.

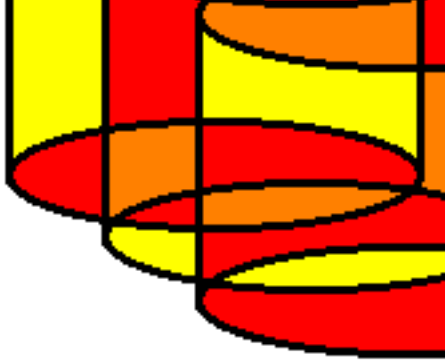
Napomena: U praksi koristimo simbole (imena) koji prirodno odgovaraju opisu aplikacijske domene.



Pravila

- Koristeći atome gradimo pravila:
  - Pravilo je izraz oblika $r(u) :- r_1(u_1), r_2(u_2),.., r_n(u_n)$, gdje je $n \neq 0$, r je obična relacija, r_i su relacije, a u, $u_1, u_2,.., u_n$ su slogovi odgovarajuće arnosti.
  - Relacija r je glava pravila (zaključak), a $r_1(u_1), r_2(u_2)$, .. , $r_n(u_n)$ je tijelo pravila (pretpostavke).

Spajanje na bazu podataka



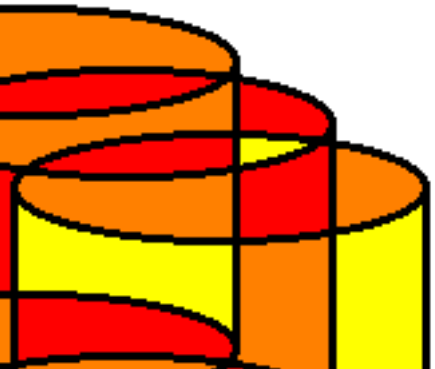
- Rekreirajte Vašu bazu podataka **s prvih laboratorijskih vježbi** (bez dodataka funkcija, okidača, dodatnih tablica itd.), npr.

```
dropdb vjezbefoi && createdb vjezbefoi && psql < kreiranje.sql
```

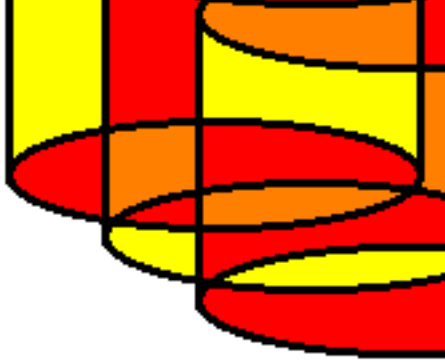
- Pokrećemo DES (engl. Datalog Educational System) sa **`des_start`** (ako javlja pogrešku, instalirati DES prema uputama sa sljedeće stranice **https://tinyurl.com/des-instalacija**)
- Na bazu podataka spajamo se preko ODBC (engl. Open DataBase Connectivity) sučelja:

```
/open_db baza
```

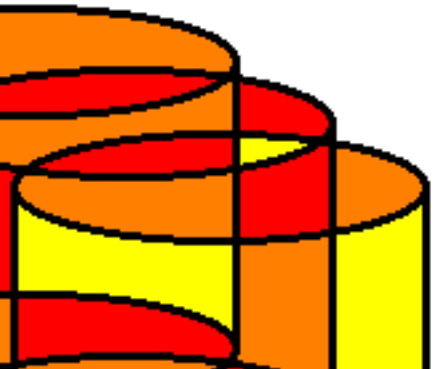
- Pri čemu je baza DSN (engl. Data Source Name) koji je definiran u datoteci **`.odbc.ini`**)



Upute



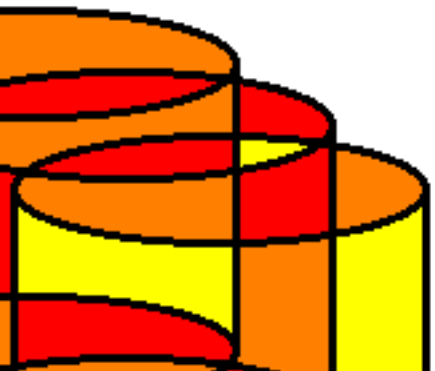
- U nastavku slijede primjeri rada s deduktivnom bazom podataka.
- Sve upite koje ste postavili (kopiju konzole) pohranite u datoteku **ime_prezime.txt**



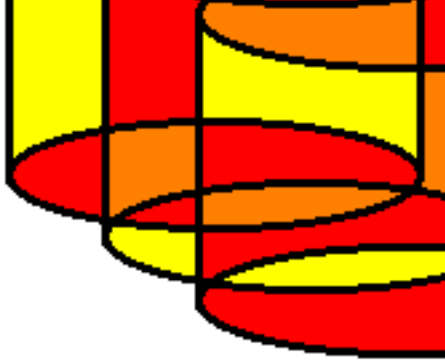
Provjera

- Možemo provjeriti je li se DES spojio na ispravnu bazu podataka s:

```
DES> /list_tables
clanstvo
dual
grupa
osoba
poruka
veza
vrsta_veze
```

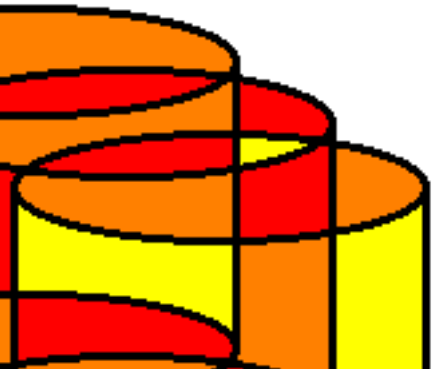


Primjeri upita

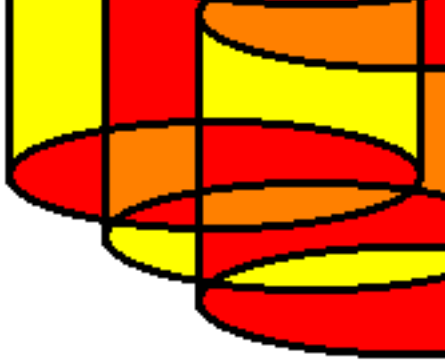


- Imena i prezimena osoba u bazi

```
osoba( _, Ime, Prezime, _, _, _ ).
```

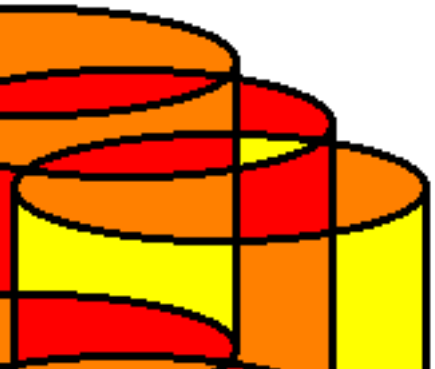


Primjeri upita



- Imena i prezimena osoba koje žive u Hrvatskoj

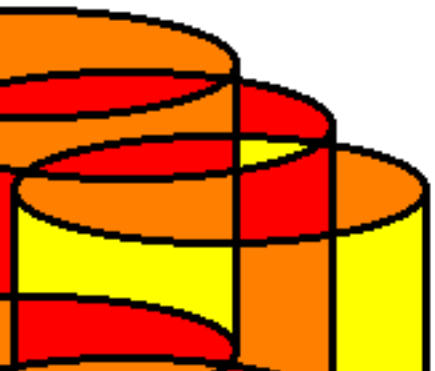
```
osoba( _, Ime, Prezime, 'Hrvatska', _, _ ).
```



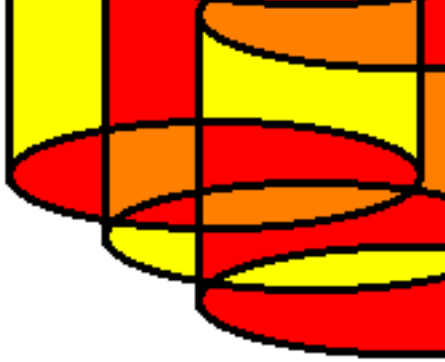
Primjeri upita

- E-mail adrese muških osoba iz Varaždina

```
osoba( Email, _, _, _, 'Varaždin',
                    'musko' ).
```

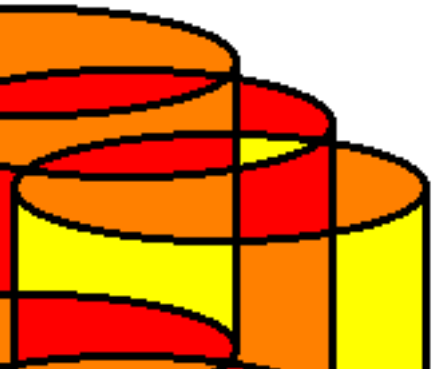


Primjeri upita



- Imena, prezimena i e-mail adrese osoba koje su primile poruke.

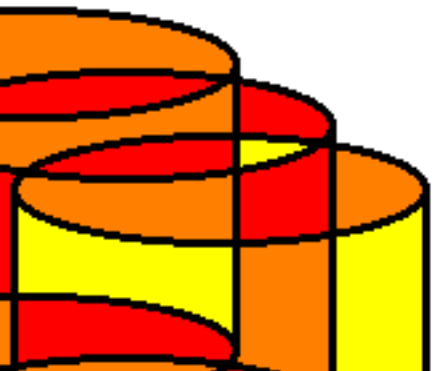
```
osoba( Email, Ime, Prezime, _, _, _ ),
  poruka( _, _, Email, _, _, _ ).
```



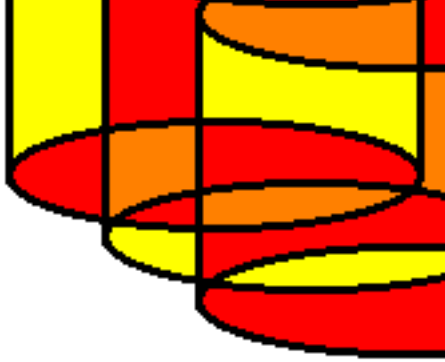
Primjeri upita

- Imena i prezimena osoba koje su primile poruke.

```
osoba( _email, Ime, Prezime, _, _, _ ),
  poruka( _, _, _email, _, _, _ ).
```

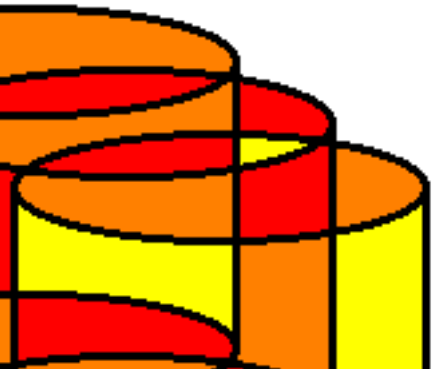


Primjeri upita



- Prezimena pošiljatelja i primatelja poruka.

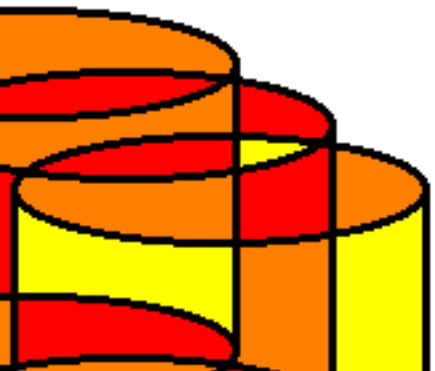
```
poruka( _, _posiljatelj, _primatelj, _, _, _ ),
osoba( _primatelj, _, Prez_prim, _, _, _ ),
osoba( _posiljatelj, _, Prez_posilj, _, _, _ ).
```



Pravila

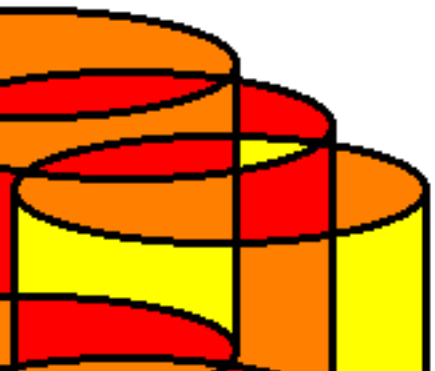
- Izrazi oblika

$$r(u) \leftarrow r_1(u_1), r_2(u_2),.., r_n(u_n)$$

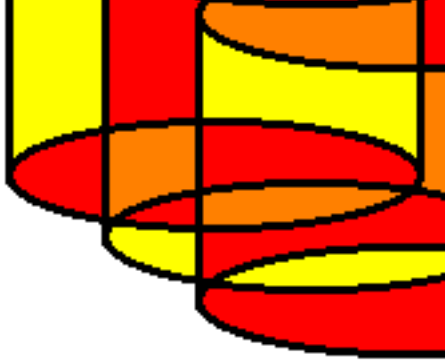


Upute

- Pravila dodajemo u DES s naredbom **`/assert`**, npr. **`/assert x( Y ) :- z( Y ), r( Y ).`**
- Pravila brišemo iz DES-a naredbom **`/retractall`**, npr. **`/retractall x( Y )`**

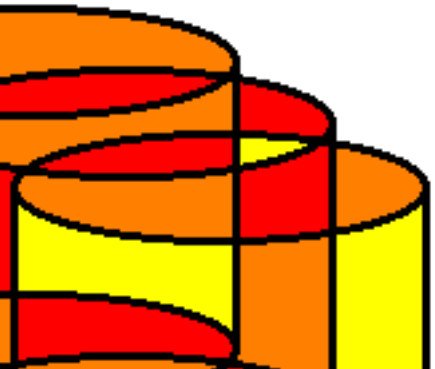


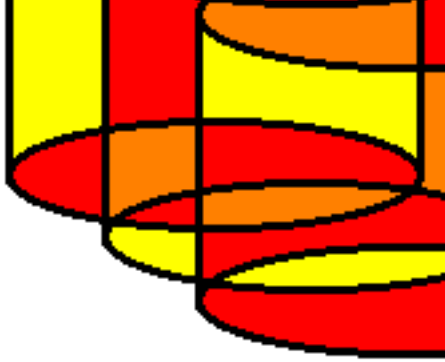
Primjer



- Definirajmo pogled (pravilo) koje će nam izlistati samo nazive grupa

```
grupa( Naziv ) :-
    grupa( _, Naziv ).
```

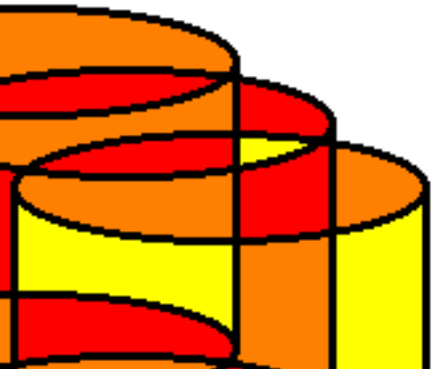




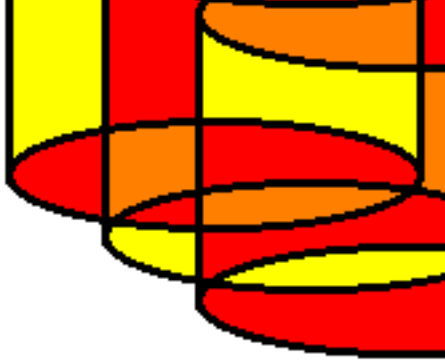
Primjer

- Definirajmo pogled (pravilo) koje će nam izlistati samo nazive grupa

```
grupa( Naziv ) :-
    grupa( _, Naziv ).

?- grupa( X ).
```

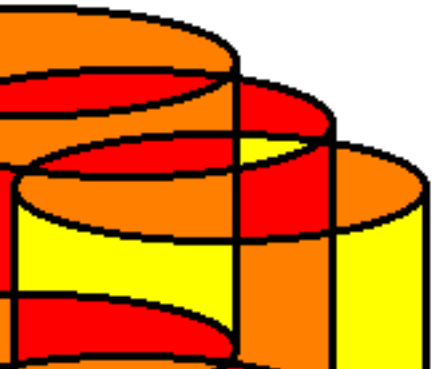


Primjer

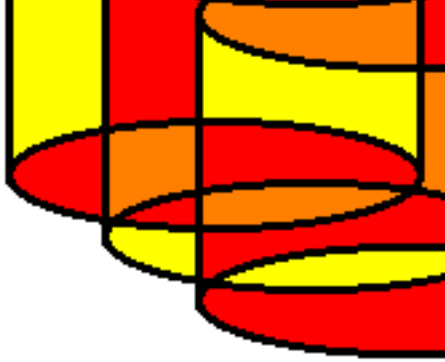


- Definirajmo pogled (pravilo, predikat) koje će nam izlistati samo imena i prezimena osoba

```
osoba( Ime, Prezime ) :-
    osoba( _, Ime, Prezime, _, _, _ ).
```

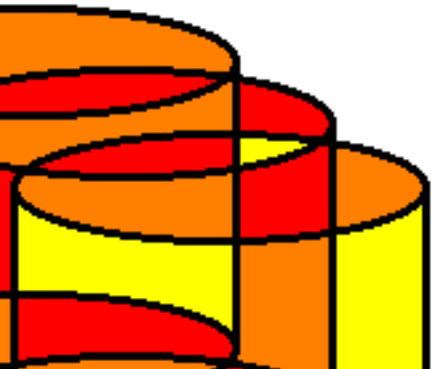


Primjer

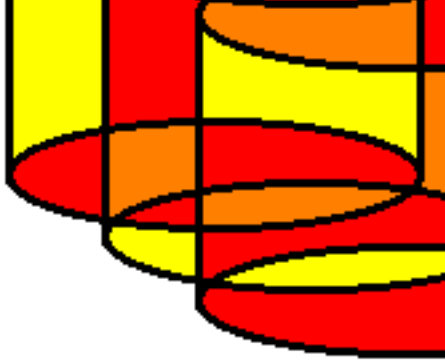


- Definirajmo predikat koje će nam izlistati sva imena i prezimena osoba koje su slale poruke

```
posiljatelj( Ime, Prezime ) :-
    osoba( Email, Ime, Prezime, _, _, _ ),
    poruka( _, Email, _, _, _, _ ).
```

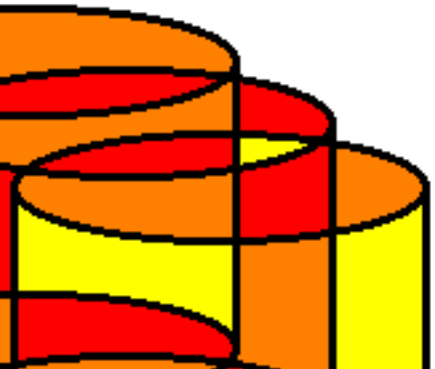


Primjer

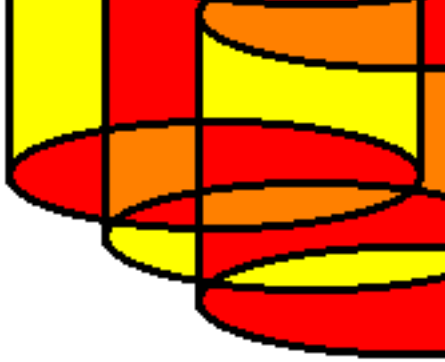


- Definirajmo predikat koje će nam izlistati imena i prezimena osoba koje su primale poruke

```
primatelj( Ime, Prezime ) :-
    osoba( Email, Ime, Prezime, _, _, _ ),
    poruka( _, _, Email, _, _, _ ).
```

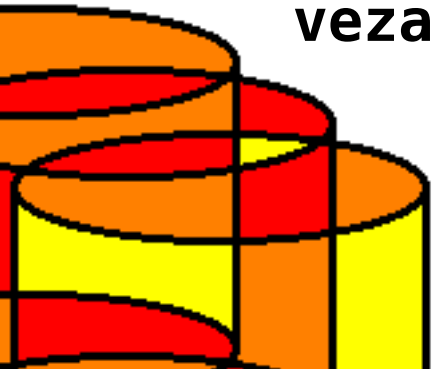


Primjer

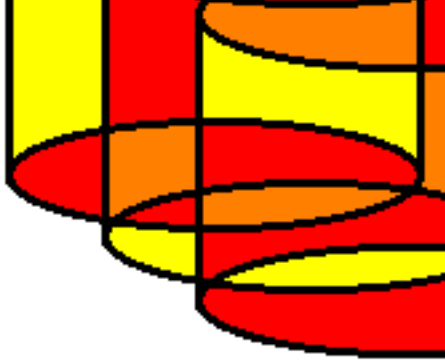


- Definirajmo predikat veza_k koji će definirati da je veza komutativna relacija (svako pravilo potrebno je dodati zasebno!)

```
veza_k( Osoba1, Osoba2, Vrsta, Status ) :-
     veza( Osoba1, Osoba2, Vrsta, Status ).
veza_k( Osoba1, Osoba2, Vrsta, Status ) :-
     veza( Osoba2, Osoba1, Vrsta, Status ).
```

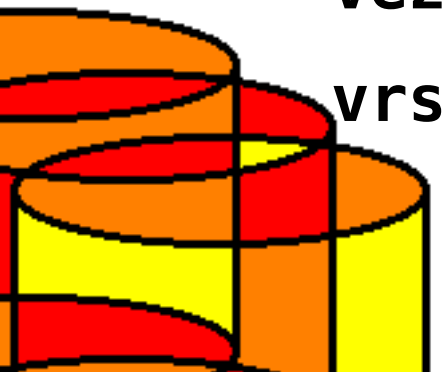


Primjer

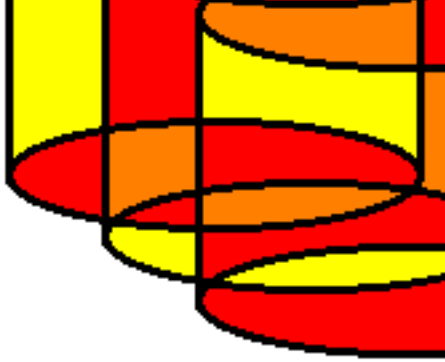


- Definirajmo predikat koje će nam izlistati imena i prezimena osoba koje su u vezi vrste 'prijatelj'

```
prijatelj( Ime1, Prezime1, Ime2, Prezime2 ) :-
      osoba( Email1, Ime1, Prezime1, _, _, _ ),
      osoba( Email2, Ime2, Prezime2, _, _, _ ),
      veza( Email1, Email2, Vrsta, _ ),
      vrsta_veze( Vrsta, 'prijatelj' ).
```

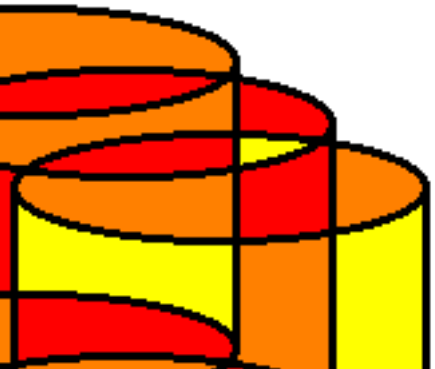


Primjer

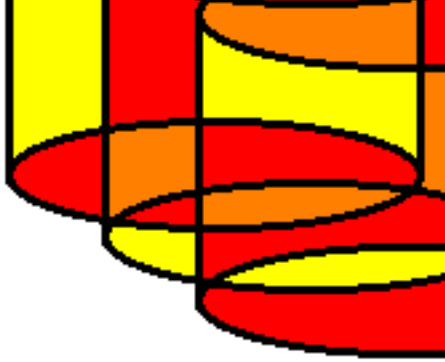


- Definirajmo predikat koje će nam izlistati imena i prezimena osoba koje su prijatelji prijatelja

```
prijatelj_prijatelja( Ime1, Prezime1, Ime2,
  Prezime2 ) :-
     prijatelj( Ime1, Prezime1, ImeX, PrezimeX ),
     prijatelj( ImeX, PrezimeX, Ime2, Prezime2 ).
```

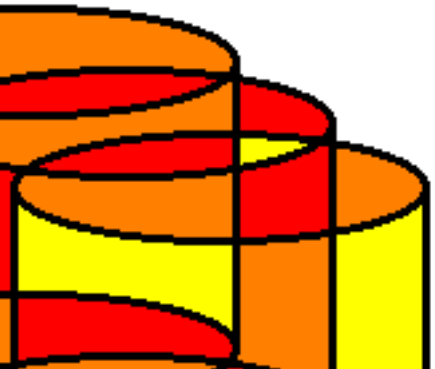


Primjer

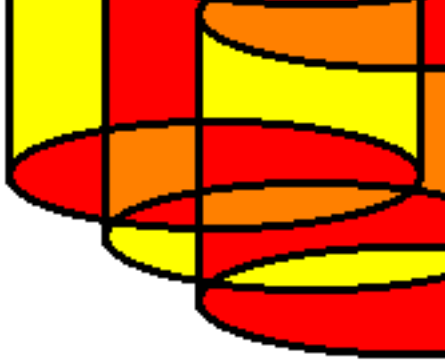


- Defnirajmo predikat koje će nam izlistati imena i prezimena ženskih prijatelja

```
zenski_prijatelj( Ime1, Prezime1, Ime2, Prezime2 ) :-
    prijatelj( Ime1, Prezime1, Ime2, Prezime2 ),
    osoba( _, Ime2, Prezime2, _, _, 'zensko' ).
```

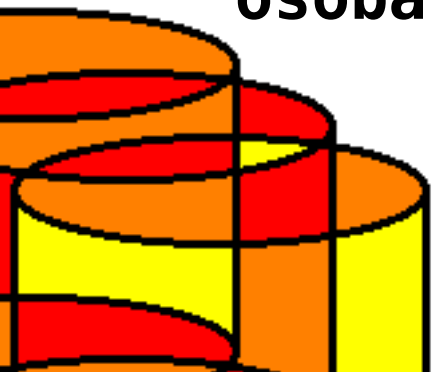


Primjer

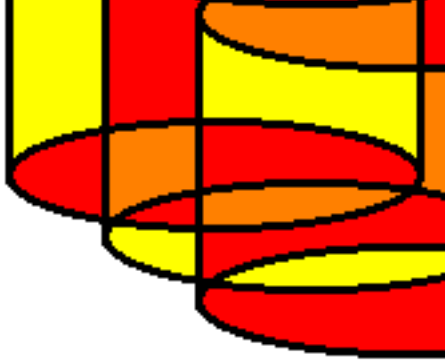


- Definirajmo predikat koje će nam izlistati imena i prezimena muških prijatelja

```
muski_prijatelj( Ime1, Prezime1, Ime2,
  Prezime2 ) :-
     prijatelj( Ime1, Prezime1, Ime2, Prezime2 ),
     osoba( _, Ime2, Prezime2, _, _, 'musko' ).
```

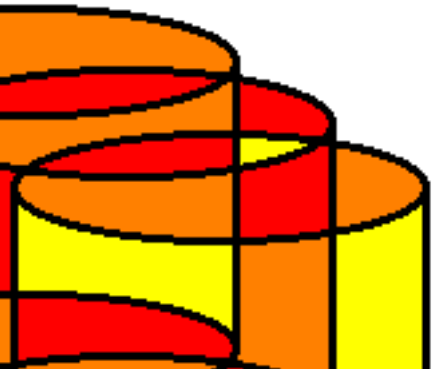


Primjer



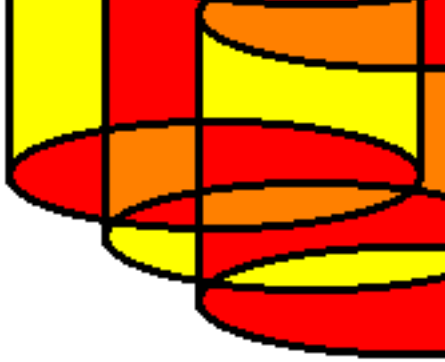
- Definirajmo predikat koji će nam izlistati imena i prezimena svih osoba koje nisu iz Varaždina

```
nije_varazdinec( Ime, Prezime ) :-
  osoba( Email, Ime, Prezime, _, _, _ ),
  not( osoba( Email, _, _, _, 'Varazdin', _ ) ).
```



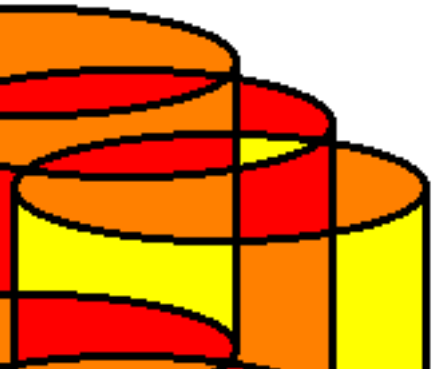
Primjer negacije!

Primjer

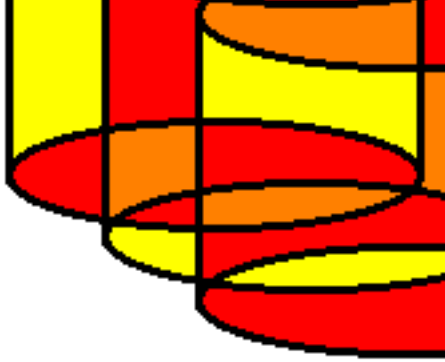


- Definirajmo pravilo koje kaže da ako barem jedna osoba živi u državi X, tada postoji grupa pod nazivom X

```
grupa( Drzava ) :-
    osoba( _, _, _, Drzava, _, _ ).
```

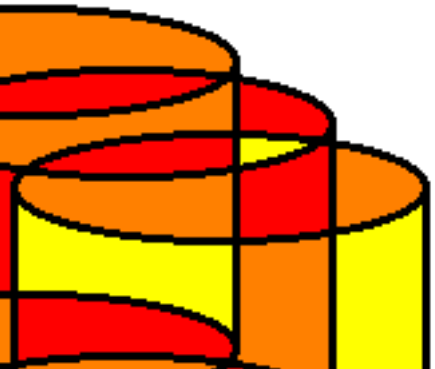


Primjer

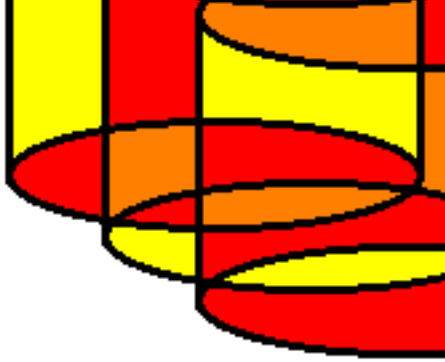


- Definirajmo pravilo koje kaže da ako barem jedna osoba živi u gradu X, tada postoji grupa pod nazivom X

```
grupa( Grad ) :-
    osoba( _, _, _, _, Grad, _ ).
```

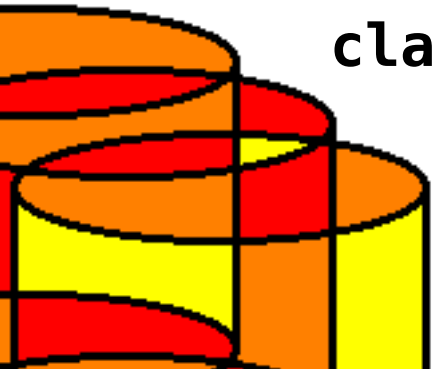


Primjer

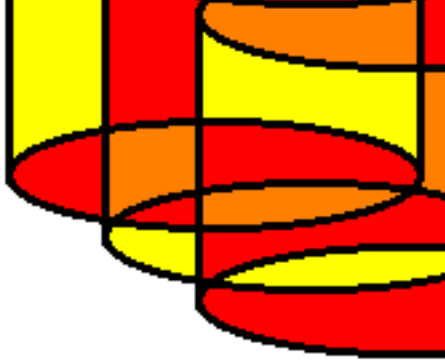


- Definirajmo predikat clan_grupe koje će nam izlistati e-mail adrese clanova i nazive grupa.

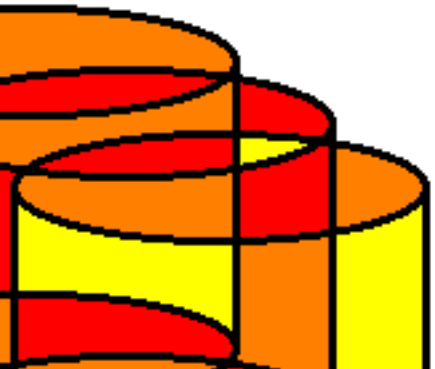
```
clan_grupe( Email, Grupa ) :-
      osoba( Email, _, _, _, _, _ ),
      clanstvo( Email, SifGrupe ),
      grupa( SifGrupe, Grupa ).

clan_grupe( Email, Drzava ) :-
      osoba( Email, _, _, Drzava, _, _ ).

clan_grupe( Email, Grad ) :-
      osoba( Email, _, _, _, Grad, _ ).
```



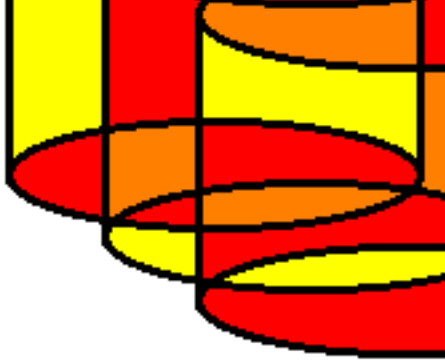
Aritmetičke operacije



- Za pridruživanje vrijednosti izračuna aritmetičkih operacija koristimo operator **`is\2`**

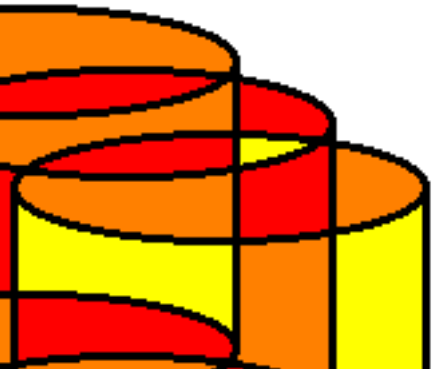


Aritmetičke operacije

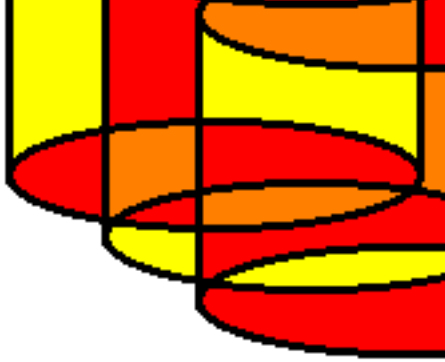


- Za pridruživanje vrijednosti izračuna aritmetičkih operacija koristimo operator **`is\2`**

```
X is 2 + 2.
```

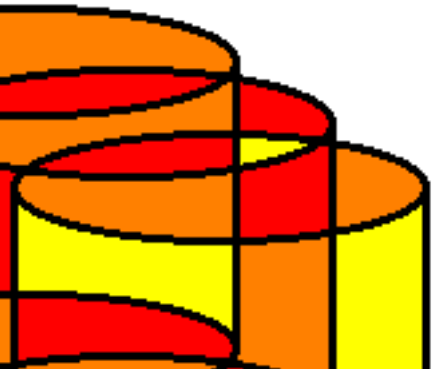


Aritmetičke operacije

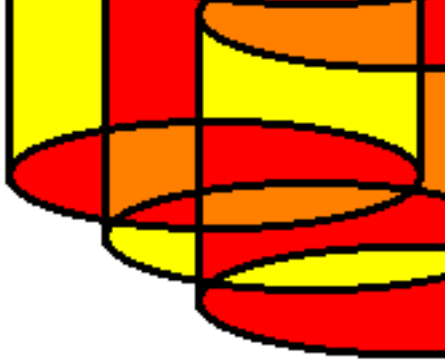


- Za pridruživanje vrijednosti izračuna aritmetičkih operacija koristimo operator **`is\2`**

```
X is 2 + 2.
X is 9 / 4.
```

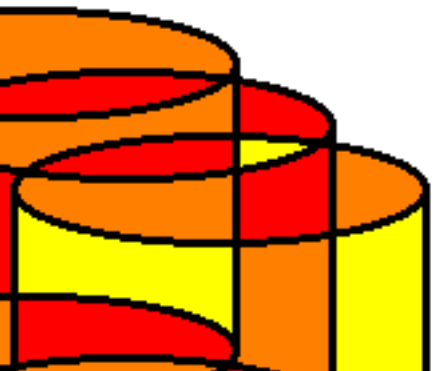


Aritmetičke operacije

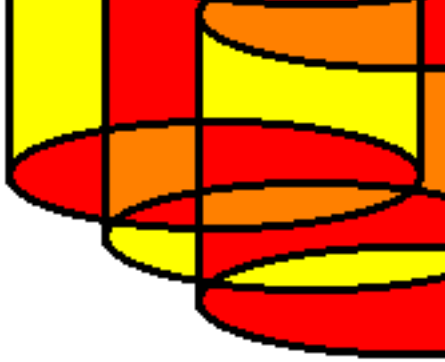


- Za pridruživanje vrijednosti izračuna aritmetičkih operacija koristimo operator **`is\2`**

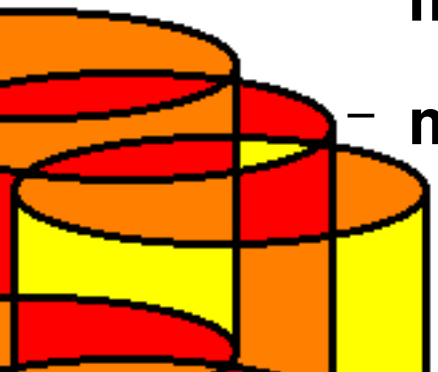
```
X is 2 + 2.
X is 9 / 4.
Y = 7, Z is Y + 3.
```



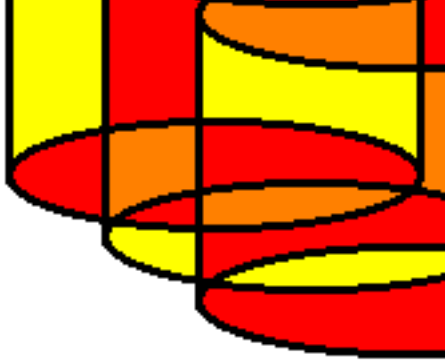
Grupiranje i agregatne funkcije



- Datalog podržava niz agregatnih funkcija koje nam omogućavaju grupiranje podataka:
  - **count** - prebrojava vrijednosti u grupi
  - **sum** - izračunava sumu vrijednosti u grupi
  - **times** - izračunava umnožak vrijednosti u grupi
  - **avg** - izračunava prosjek vrjednosti u grupi
  - **min** - izračunava minimalnu vrijednost u grupi
  - **max** - izračunava maksimalnu vrijednost u grupi

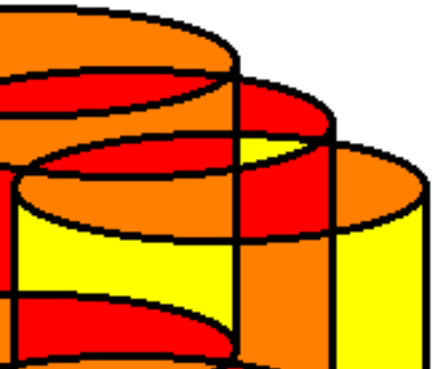


Struktura upita

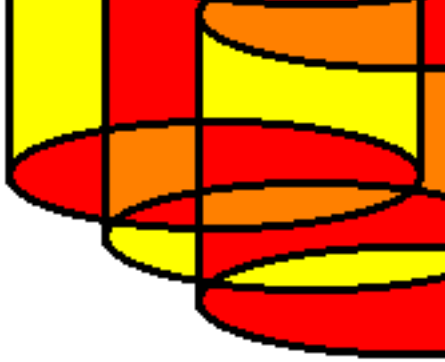


- Kada se koriste kao samostalni upit, agregirajuće funkcije imaju sljedeći oblik:

```
funkcija( Upit , VarijablaAgregacije, Rezultat )
```

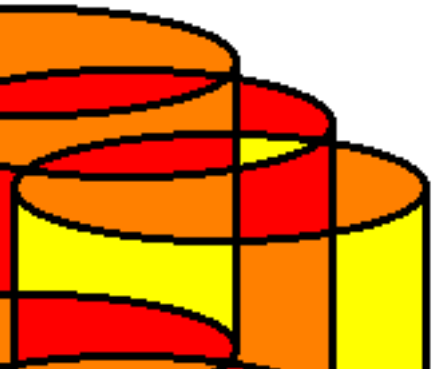


Primjeri

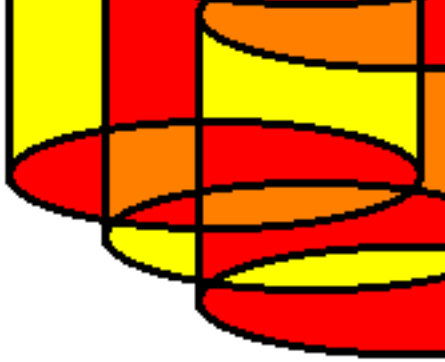


- Upit koji prebrojava sve države u tablici osoba:

```
count( osoba( _, _, _, Drzava , _, _ ) , Drzava , Rezultat )
```



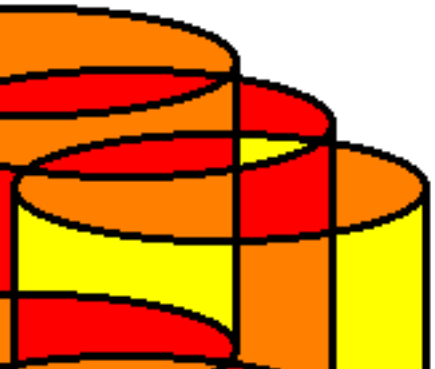
Primjeri



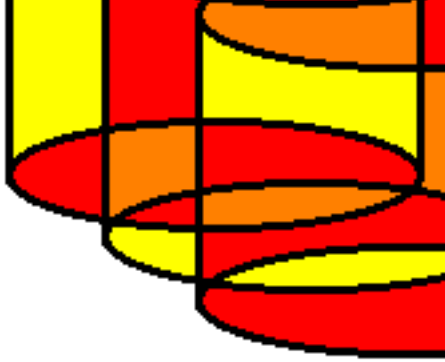
- Upit koji prebrojava sve države u tablici osoba:

```
count( osoba( _, _, _, Drzava , _, _ ) , Drzava , Rezultat )
```

Zašto je odgovor neočekivan?

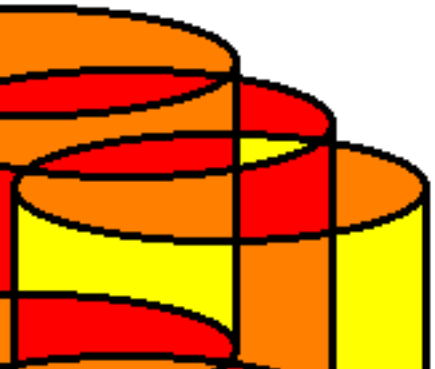


Primjeri

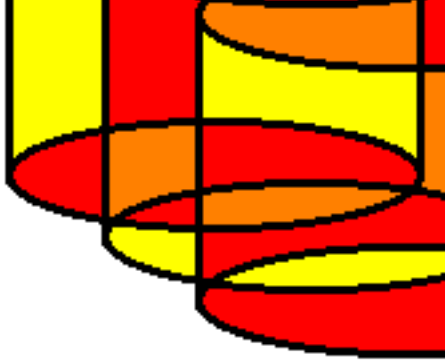


- Da bismo eliminirali duplikate koristimo distinct:

```
count (
   distinct (
      [ Drzava ] ,
      osoba( _, _, _, Drzava , _, _ )
   ),
  Drzava ,
  Rezultat
)
```

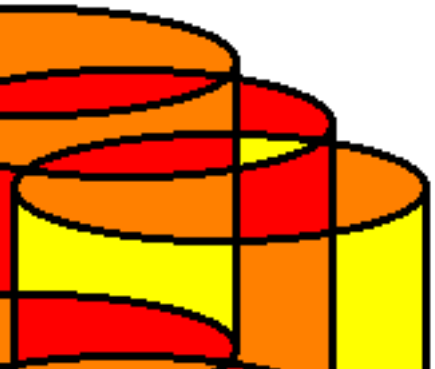


Primjeri

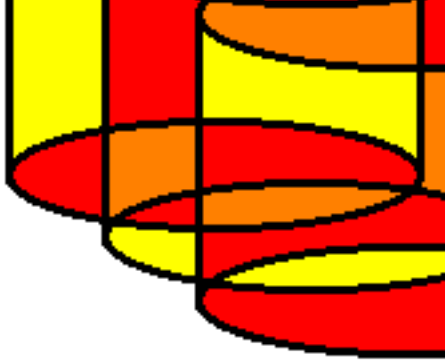


- Ili jednostavnije count_distinct:

```
count_distinct( osoba( _, _, _, Drzava , _, _ ), Drzava, Rezultat)
```

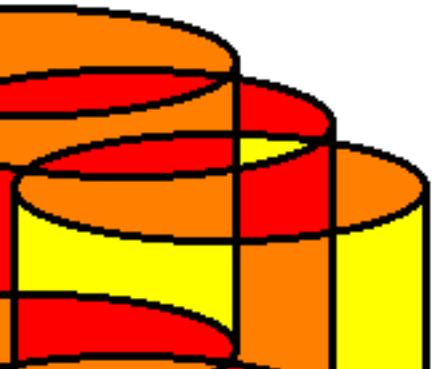


Group by

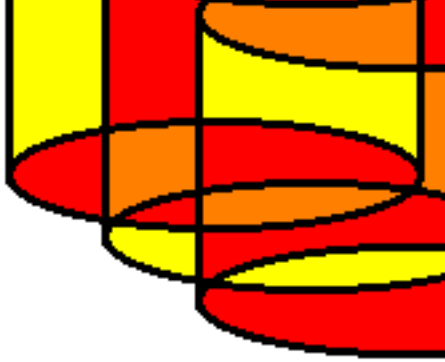


- Drugi način korištenja agregirajućih funkcija

```
group_by( Upit, Varijable, UvjetiGrupiranja )
```

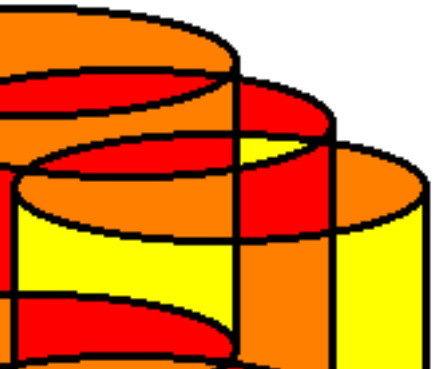


Primjeri

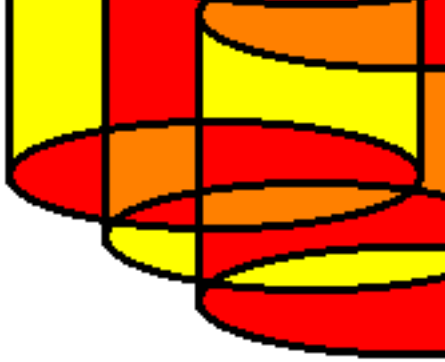


- Upit koji vraća nazive gradova i ukupan broj korisnika koji su naveli da žive u tom gradu:

```
group_by( osoba(_,_,_,_,Grad,_), [ Grad ], Ukupno=count( Grad ) )
```

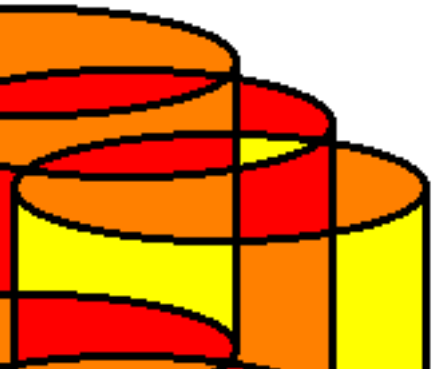


Primjeri

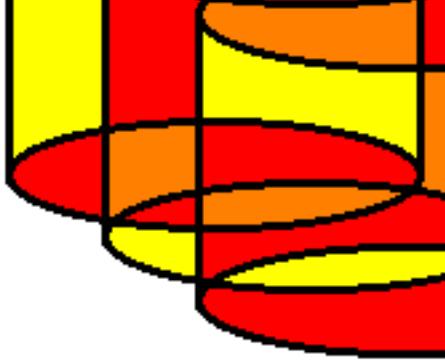


- Moguće je postavljati i složene uvjete (slično HAVING klauzuli u SQL-u):

```
group_by (
   osoba( _, _, _, _, Grad , _ ),
   [ Grad ],
   ( Ukupno=count( Grad ), Ukupno > 1 )
)
```

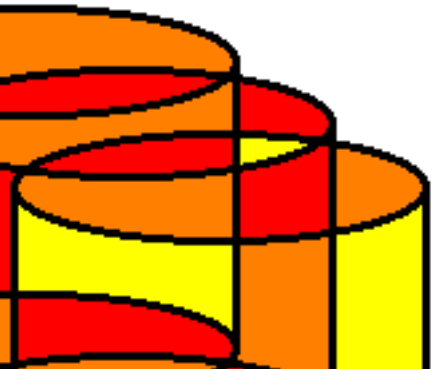


Primjer



- Definirajmo predikat koji će nam pronaći udaljenost između dvaju osoba brojeći veze:

```
udaljen( Email1, Email2, 1 ) :-
     veza( Email1, Email2, _, _ ).
udaljen( Email1, Email2, X ) :-
    udaljen( EmailX, Email2, X1 ),
     veza( Email1, EmailX, _, _ ),
    count( veza( E1, E2, _, _ ), Max ),
    X1 < Max,
     X is X1 + 1.
```

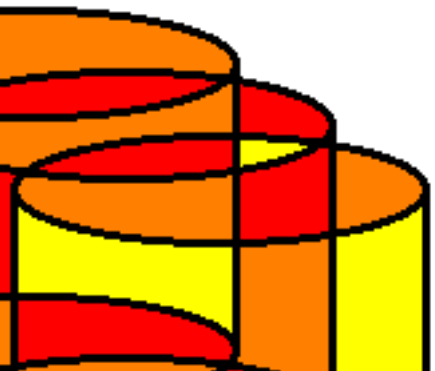


Rekurzivna definicija predikata!

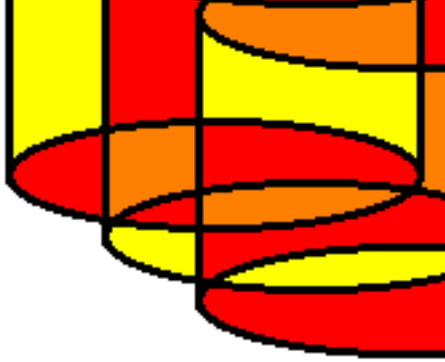
Primjer

- Želimo samo najkraću udaljenost

```
najkraci_put( E1, E2, X ) :-
        min( udaljen( E1, E2, U ), U, X ).
```



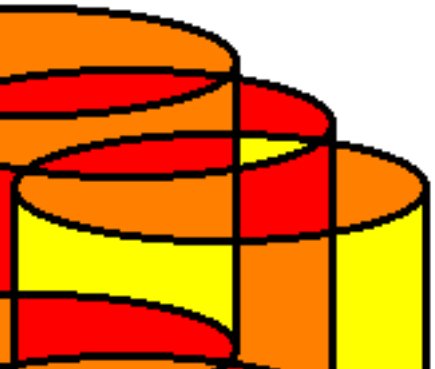
Ugrađeni predikati



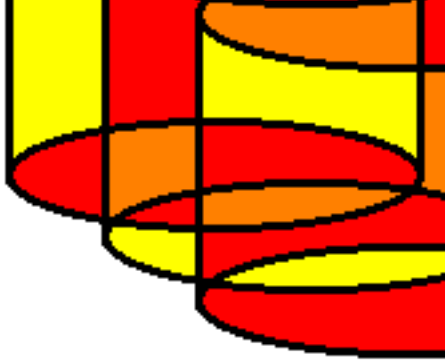
- Osim prethodno navedenih predikata, postoji još nekoliko koji su relevantni za rad s deduktivnim bazama podataka, npr. predikati za testiranje NULL vrijednosti

`is_null/1` - provjerava je li argument vezan uz null vrijednost.

`is_not_null/1` - uspjeva ako argument nije vezan uz null vrijednost.

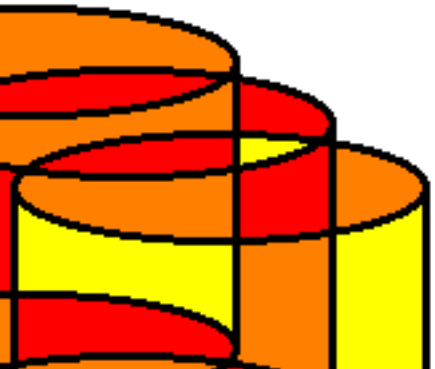


Distinct

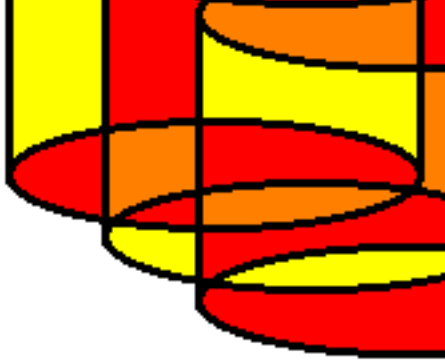


- Postavite sljedeća sva upita:

```
distinct( osoba( _, _, Prezime, Drzava, _, _ ) )
distinct([Prezime , Drzava], osoba( _, _, Prezime, Drzava, _, _))
```



Distinct



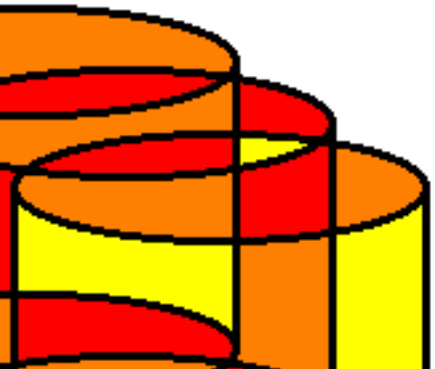
- Postavite sljedeća sva upita:

```
distinct( osoba( _, _, Prezime, Drzava, _, _ ) )
```

```
distinct([Prezime , Drzava], osoba( _, _, Prezime, Drzava, _, _))
```

Rezultat prvog upita se čini neočekivanim, ali ako razmotrimo detaljnije nije neočekivan. Nebitne varijable također se vežu uz vrijednosti! Je li bi rezultat bio neočekivan da je upit izgledao ovako:

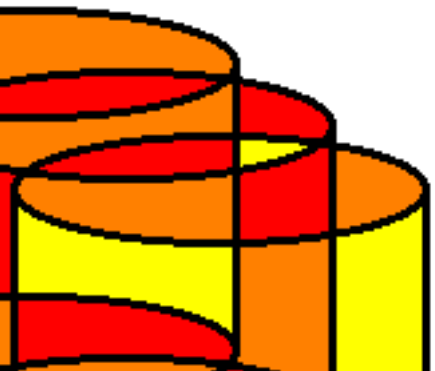
```
distinct( osoba( Email, Ime, Prezime, Drzava, Grad, Spol ) )
```



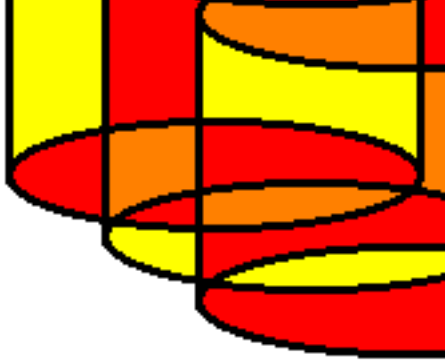
Top

- S predikatom top možemo ograničiti broj slogova u rješenju:

```
top( 3, grupa( Sifra, Naziv ) )
```

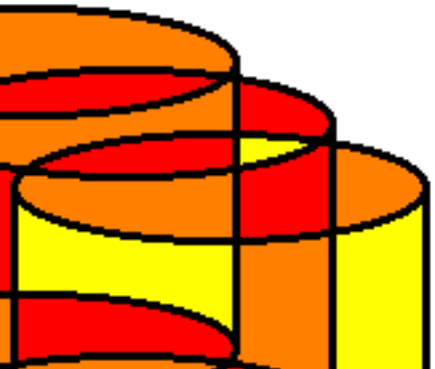


Order by

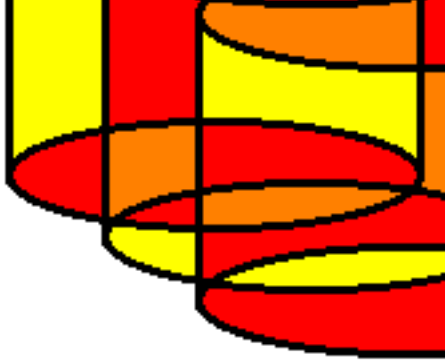


- S predikatom order by možemo sortirati izlaz:

```
order_by(
   osoba(Email,Ime,Prezime,Drzava,Grad,Spol),
   [ Drzava, Grad, Spol ]
)
```

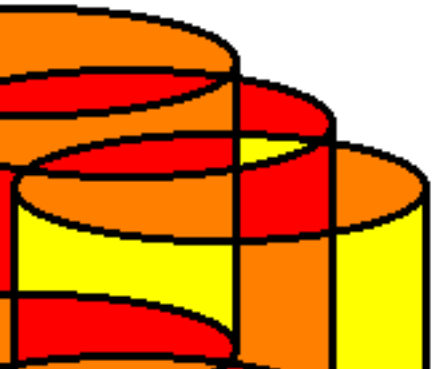


Order by

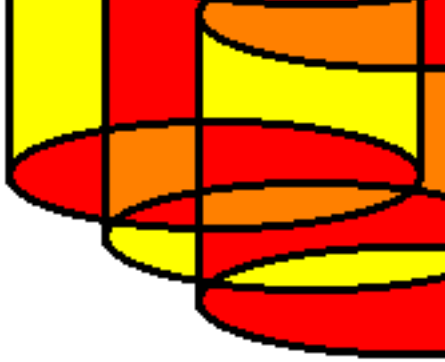


- Možemo odrediti i poredak sortiranja (ascending, descending):

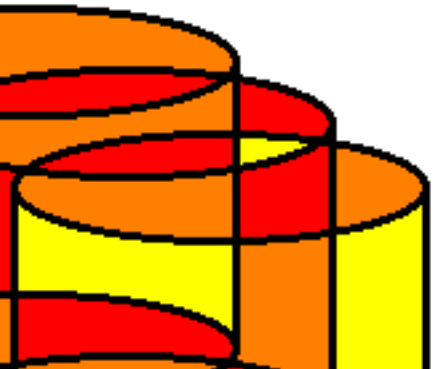
```
order_by(
   osoba(Email,Ime,Prezime,Drzava,Grad,Spol),
   [ Drzava, Grad, Spol ],
   [ d, d, a ]
)
```



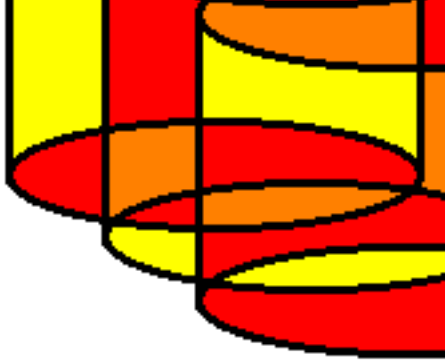
Meta naredbe



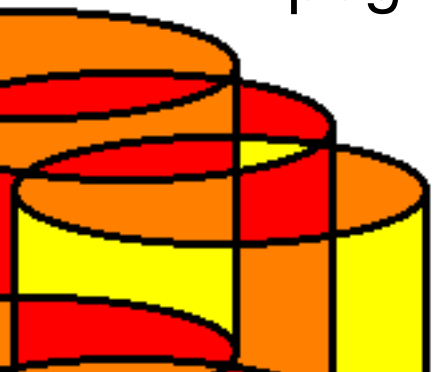
- Da bismo izlistali aktivnu deduktivnu bazu podataka koristimo metanaredbu **/listing**
- Da bismo pospremili trenutnu deduktivnu bazu podataka koristimo metanaredbu **/save_ddb <naziv_datoteke.ddb>**
- Da bismo ponovno učitali pospremljenu deduktivnu bazu podataka koristimo metanaredbu **/restore_ddb <naziv_datoteke.ddb>**



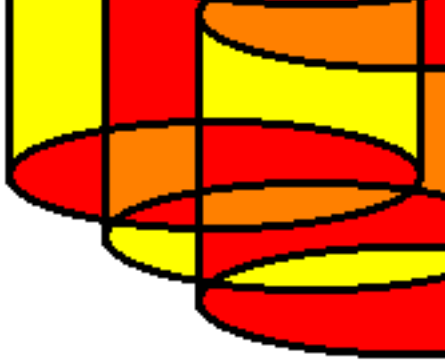
Diskusija



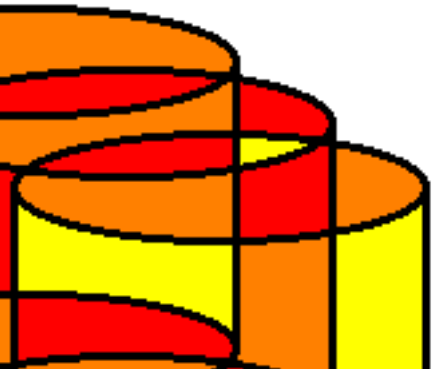
- Kada koristiti deduktivne baze podataka?
  - Za analizu ponašanja/modela baze podataka – je li sve u skladu s očekivanjima.
  - Za integraciju heterogenih baza podataka, relacija, pogleda ...
  - Za implementaciju kompleksnijih skupova pravila i pogleda (rekurzija u SQL-u standardu nije podržana!).



Diskusija



- Kada ne koristiti deduktivne baze podataka?
  - Za implementaciju jednostavnih transakcijskih baza podataka u kojima su bitne performanse
  - Za implementaciju baza podataka u kojima je količina podataka potencijalno vrlo velika (kombinatorička eksplozija!)



Zadatak

- Implementirajte sljedeća deduktivna pravila
  - **klika(Email1, Email2, Email3)** koje će vraćati klike osoba razine 3 (klika znači da su svi čvorovi međusobno povezani)
  - **kratki_put(Email1, Email2, Vrsta, Duljina)** koje će vraćati najkraće puteve između osoba ali određene vrste veze
- Deduktivnu bazu podataka pohranite u datoteku **ime_prezime.ddb**