BAZE PODATAKA

SQL
Nedostajući podaci

Neđeljko Lekić
Irena Orović
www.etf.ac.me

U OVOJ LEKCIJI

- Nedostajući podaci
  - NULLe i relacioni model
  - OUTER JOIN
  - Podrazumijevane vrijednosti

NEDOSTAJUĆI PODACI

- Ponekad vrijednost koju treba unijeti u relaciju nije poznata.
  - Poznato je da postoji vrijednost ali se ne zna koja je.
  - Ne postoji vrijednost koja ima smisla.
- Postoje dva postupka koji se predlažu.
  - NULL-e (praznine) se mogu koristiti da pokažu da podatak nedostaje.
  - Može se koristiti predefinisana vrijednost, koja će takođe, ali na drugačiji način, ukazivati na nedostatak podatka.

NULL-e

- NULL čuva mjesto nedostajućim podacima atributa. To nije vrijednost, sama po sebi.
- Codd predlaže da se razlikuju dvije vrste NULL-a:
  - Podatak postoji ali je nepoznat (ne primjer, nečiji datum rođenja, broj telefona)
  - Podatak ne postoji (telefonski broj nekoga ko nema telefon, ili ime roditelja za nekoga za koga to nije poznato, itd..)

PROBLEMI SA NULL-ama

- Problemi sa proširenjem operacija relacione algebre na NULL-e:
  - Definisanje operacije selekcije: ako ispitujemo neku osobinu zapisa, kao Bodovi > 40, i za neki zapis Bodovi su NULL, što da radimo?
  - Definisanje presjeke i razlike dviju relacija: jesu li dva zapisa <Jovan,NULL> i <Jovan,NULL> isti ili ne?
- Dodatni problemi za SQL: da li tretirati NULL-e kao duplikate? Dali da ih uključimo u račun (suma, srednja vrijednost, ...) i ako da, kako? Kako da se aritmetičke operacije ponašaju kada je argument NULL?

RJEŠENJE 1

- Za procjenu uslova upotrijebiti tro-vrijednosnu logiku umjesto uobičajene dvo-vrijednosne logike.
- Kada nema NULL-a, uslovi se procjenjuju kao istina (true) ili neistina (false), ali ako NULL-e postoje, uslov će se procjenjivati i kao treća vrijednost - ‘nedefinisano’, ili ‘nepoznato’).
- Ovo je ideja za provjeru uslova u WHERE klauzuli SQL SELECT-a: Biće vraćeni jedino zapisi gdje je rezultat provjere uslova istina.

3 – VRIJEDNOSNA LOGIKA

- Ako uslov uključuje Boolean vrijednosti, ocjenjuje se na sljedeći način:

x	y	x AND y	x OR y	NOT x
true	true	true	true	false
true	unknown	unknown	true	false
true	false	false	true	false
un	true	un	true	un
un	un	un	un	un
un	false	false	un	un
false	true	false	true	true
false	un	false	un	true
false	false	false	false	true

3 – VRIJEDNOSNA LOGIKA

false=0, true=1, unknown=1/2,

NOT(x)=1-x,

AND(x,y) = min(x,y),

OR(x,y) =max(x,y):

x	y	x AND y	x OR y	NOT x
true	true	true	true	false
true	unknown	unknown	true	false
true	false	false	true	false
un	true	un	true	un
un	un	un	un	un
un	false	false	un	un
false	true	false	true	true
false	un	false	un	true
false	false	false	false	true

RJEŠENJE 2

- Za predstavljanje nepoznate vrijednosti koristiti promjenjive umjesto NULLa.
- Različita nepoznata vrijednost odgovara različitoj promjenjivoj.
- Kada se primjenjuje operacija, kao što je selekcija, iz tabela sa promjenjivim, promjenjive mogu zadovoljiti zadate uslove (ograničenja). Na primjer x > 40, ako je x nepoznata vrijednost bodova, može biti veća od 40 i uključuje se u rezultat selekcije, Bodovi > 40.

SQL RJEŠENJE

NULL-e u uslovima

```
SELECT *
FROM Radnik
Where Plata > 1500
```

- Plata>1500 ocjenjuje se kao ‘nepoznato’
- zadnji red se ne uključuje se u rezultat.

Radnik

Ime	Plata
Jovan	2500
Marko	1500
Ana	2000
Krsto	NULL

Ime	Plata
Jovan	2500
Ana	2000

Vraćeni jedino zapisi gdje je rezultat provjere uslova istina.

SQL RJEŠENJE

NULL-e u uslovima

```
SELECT *
FROM Radnik
WHERE Plata > 1500
OR Ime = 'Krsto'
```

- Plata > 1500 OR Ime = 'Krsto' uključuje zadnji red

Radnik

Ime	Plata
Jovan	2500
Marko	1500
Ana	2000
Krsto	NULL

Ime	Plata
Jovan	2500
Ana	2000
Krsto	NULL

SQL RJEŠENJE

ARITMETIKA

```
SELECT
Plata*1.1 AS NovaPlata
FROM Radnik
```

- Aritmetička operacija sa NULL-ama rezultuje NULL-ama

Radnik

Ime	Plata
Jovan	2500
Marko	1500
Ana	2000
Krsto	NULL

NovaPlata
2750
1650
2200
NULL

SQL RJEŠENJE

AGREGATNE FUNKCIJE

```
SELECT
  AVG(Plata) AS Avg,
  COUNT(Plata) AS Num,
  SUM(Plata) AS Sum
FROM Radnik
```

Radnik

Ime	Plata
Jovan	2500
Marko	1500
Ana	2000
Krsto	NULL

- Avg = 2000
- Num = 3
- Sum = 6000
- **`SELECT COUNT(*)`** . . . daje rezultat 4

OUTER JOIN

- Kada se povezuju dvije relacije, povezuju se redovi koji zadovoljavaju uslov povezivanja.
  - Neki redovi ne zadovoljavaju uslov i nema ih u rezultatu.
  - Takvi redovi se nazivaju ‘viseći’
- Outer joins uključuje ‘viseće’ redove u rezultat i koristi NULL-e za popunjavanje praznina.
  - Left outer join
  - Right outer join
  - Full outer join

PRIMJER: INNER JOIN

Student

ID	Ime
123	Jovan
124	Marija
125	Marko
126	Jana

Rezultat

ID	Kod	Bodovi
123	BPD	60
124	PRG	70
125	BPD	50
128	BPD	80

viseći

Student inner join Rezultat

ID	Ime	ID	Kod	Bodovi
123	Jovan	123	BPD	60
124	Marija	124	PRG	70
125	Marko	125	BPD	50

PRIMJER: LEFT OUTER JOIN

Student

ID	Ime
123	Jovan
124	Marija
125	Marko
126	Jana

Rezultat

ID	Kod	Bodovi
123	BPD	60
124	PRG	70
125	BPD	50
128	BPD	80

viseći

Student left outer join Rezultat

ID	Ime	ID	Kod	Bodovi
123	Jovan	123	BPD	60
124	Marija	124	PRG	70
125	Marko	125	BPD	50
126	Jana	null	null	null

PRIMJER: RIGHT OUTER JOIN

Student

ID	Ime
123	Jovan
124	Marija
125	Marko
126	Jana

Rezultat

ID	Kod	Bodovi
123	BPD	60
124	PRG	70
125	BPD	50
128	BPD	80

viseći

Student right outer join Rezultat

ID	Ime	ID	Kod	Bodovi
123	Jovan	123	BPD	60
124	Marija	124	PRG	70
125	Marko	125	BPD	50
null	null	128	BPD	80

PRIMJER: FULL OUTER JOIN

Student

ID	Ime
123	Jovan
124	Marija
125	Marko
126	Jana

Rezultat

ID	Kod	Bodovi
123	BPD	60
124	PRG	70
125	BPD	50
128	BPD	80

viseći

Student full outer join Rezultat

ID	Ime	ID	Kod	Bodovi
123	Jovan	123	BPD	60
124	Marija	124	PRG	70
125	Marko	125	BPD	50
126	Jana	null	null	null
null	null	128	BPD	80

SINTAKSA ZA OUTER JOIN U MySQL-u

```
SELECT <kolone>
   FROM <tabela1> <tip> OUTER JOIN
  <tabela2>
     ON <uslov>
```

<tip> je **LEFT**, **RIGHT**

Primjer:

```
SELECT *
   FROM Student LEFT OUTER JOIN  Rezultat
     ON Student.ID = Rezultat.ID
```

LEFT OUTER JOIN: PRIMJER

```
SELECT *
  FROM Student LEFT OUTER JOIN  Rezultat
    ON Student.ID = Rezultat.ID
```

Rezultat:

ID	Ime	Prezime	ID	Kod	Bodovi
123	Jovan	Simić	123	BPD	60
124	Marija	Jokić	124	PRG	70
125	Marko	Jokić	125	BPD	50
126	Jana	Brkić			

RIGHT OUTER JOIN: PRIMJER

```
SELECT *
  FROM Student RIGHT OUTER JOIN  Rezultat
    ON Student.ID = Rezultat.ID
```

Rezultat:

ID	Ime	Prezime	ID	Kod	Bodovi
123	Jovan	Simić	123	BPD	60
124	Marija	Jokić	124	PRG	70
125	Marko	Jokić	125	BPD	50
			128	RHD	80

FULL OUTER JOIN: PRIMJER

U MySQL-u FULL OUTHER JOIN se ne može primijeniti direktno. Ipak moguće ga je izvesti posredno, pomoću LEFT i RIGHT OUTER JOIN.

```
SELECT * FROM Student LEFT OUTER JOIN Rezultati
    ON Student.ID = Rezultati.ID

 UNION

SELECT * FROM Student RIGHT OUTER JOIN Rezultati
    ON Student.ID = Rezultati.ID
```

Rezultat:

ID	Ime	Prezime	ID	Kod	Bodovi
123	Jovan	Simić	123	BPD	60
124	Marija	Jokić	124	PRG	70
125	Marko	Jokić	125	BPD	50
126	Jana	Brkić			
			128	RHD	80

PODRAZUMIJEVANE (DEFAULT) VRIJEDNOSTI

- Podrazumijevane vrijednosti su alternativa NULL vrijednostima.
  - Ako je vrijednost nepoznata, kao ″čuvar mjesta″ može se koristiti podrazumijevana vrijednost.
  - Ovo je stvarna vrijednost, pa nije potrebna 3VL i slično.
- Podrazumijevane vrijednosti mogu imati više smisla od NULL-a.
  - ‘nijedan’ (‘none’)
  - ‘nepoznat’ (‘unknown’)
  - ‘neispručeno’ (‘not supplied’)
  - ‘neprimjenjivo’ (‘not applicable’)

DEFAULT VRIJEDNOSTI: PRIMJER

Artikli

ID	Ime	Tez	Kol
1	Orah	10	20
2	Urma	15	-1
3	Ekser	3	100
4	Čaša	-1	30
5	???	20	20
6	Šraf	-1	-1
7	Šina	150	0

- Podrazumijevane vrijednosti su
  - ??? za Ime
  - -1 za Tez i Kol

- -1 je upotrijebljeno za Tez i Kol jer nema smisla, odnosno, ne može se slučajno pojaviti, ali što sa:

```
UPDATE Artikli
  SET Kol = Kol + 5
```

PROBLEMI SA PODRAZ. VRIJEDNOSTIMA

- Kako su podraz. vrijednosti stvarne vrijednosti
  - mogu se ažurirati kao druge vrijednosti.
  - potrebno je koristiti vrijednosti koje se ne mogu pojaviti u bilo kom drugom slučaju.
  - mogu se pogrešno protumačiti.
- Također, u SQL-u predefinisana vrijednost mora biti istog tipa kao kolona.
  - Ne može se uzeti string kao ‘unknown’ ako je kolona INTEGER tipa.

PODJELA TABELA

- NULL-e i podrazumijevane vrijednosti zamjenjuju nedostajuće podatke.
  - NULL-e označavaju podatke kao nedostajuće.
  - Podrazumijevane vrijednosti daju neke indikacije, kao koji je tip nedostajućih informacija.
- Često je moguće ukloniti zapise koji imaju nedostajuće podatke.
  - Moguće je podijeliti tabelu tako da kolone koje mogu imati NULL-e se odvoje u posebne tabele.
  - Zapis koji bi sadržavao NULL ne postoji u novonastalim tabelama.

PODJELA TABELA: PRIMJER

Artikli

ID	Ime	Tez	Kol
1	Orah	10	20
2	Urma	15	-1
3	Ekser	3	100
4	Čaša	-1	30
5	???	20	20
6	Šraf	-1	-1
7	Šina	150	0

ID	Ime
1	Orah
2	Urma
3	Ekser
4	Čaša
6	Šraf
7	Šina

ID	Tez
1	10
2	15
3	3
5	20
7	150

ID	Kol
1	20
3	100
4	30
5	20
7	0

PROBLEMI SA PODJELOM TABELA

- Dijeljenje tabela ima svoje probleme.
  - Može se uvesti mnogo dodatnih tabela
  - Potrebna informacija je razasuta po cijeloj bazi.
  - Upiti postaju znatno složeniji i zahtijevaju puno spajanja tabela.
- Moguće je rekonstrisati originalnu tabelu, ali
  - Mora se koristiti OUTER JOIN da bi se to uradilo.
  - Time se ponovo ukuljučuju NULL-e, sa svim problemima koje nose.

SQL PODRŠKA

- SQL dozvoljava i NULL-e i default vrijednosti. Primjer:
  - Naka tabela sadrži podatke o radnicima.
  - Svaki radnik ima ime.
  - Svaki radnik ima platu (default 600)
  - Neki radnici imaju telefonski broj. Ako nemaju koriste se NULL-e.

```
CREATE TABLE Radnik
(
   Ime CHAR(50)
         NOT NULL,
   Plata INT
         DEFAULT 600,
   Telefon CHAR(15)
         NULL
)
```

SQL PODRŠKA

SQL dozvoljava unos NULL-a

```
INSERT INTO Radnik
VALUES ('Jovan',
12000,NULL)

UPDATE Radnik
SET Telefon = NULL
WHERE Ime = 'Marko'
```

Moguće je i pretraživati po NULL-ama.

```
SELECT Ime FROM
 Radnik WHERE
 Telefon IS NULL

SELECT Ime FROM Radnik
  WHERE Telefon IS NOT
  NULL
```

KOJI METOD KORISTITI?

- Često pitanje ličnog izbora, ali
  - Default vrijednosti se ne trebaju koristiti ako mogu da se pomiješaju sa ‘realnim’ vrijednostima
  - Dijeljenje tabela ne treba previše forsirati, inače će se dobiti previše tabela.
  - NULL-e se mogu koristiti (i često se koriste) gdje drugačiji pristup izgleda neadekvatan.
  - Ne mora se uvijek koristiti isti metod – bolje je miješati i prilagođavati potrebi.

PRIMJER

- U on-line prodavnicama postoje različiti proizvodi - knjige, CD-ovi, i DVD-evi.
  - Svi artikli imaju ime, cijenu i ID (kataloški broj).
  - Neki artilkl može imati ekstra troškove isporuke, ali neki ne.
- Postoje također i specifični podaci za svaki tip proizvoda.
  - Knjige moraju imati autora i mogu imati izdavača.
  - CD-ovi moraju imati umjetnika
  - DVD-evi moraju imati producenta ili direktora.

PRIMJER

- Mogu se svi podaci staviti u jednu tabelu

Artikli

ID	Naziv	Cijena	Transport	Autor	Izdavač	Umjetnik	Producent	Direktor

  - Postojaće puno unosa sa nedostajućim podacima.
  - Svaki red će imati nedostajuće podatke.
  - Smješteni su podaci o tri tipa proizvoda u jednu tabelu.

PRIMJER

- Moguće da je najbolje odvojiti rezličite tipove proizvoda u različite tabele.
  - Postojaće glavna tabela – tabela Artikli.
  - Takođe će postojati tabele Knjiga, CD, i DVD sa referenciranjem na tabelu Artikli.

Artikli

ID	Naziv	Cijena	Transport

Knjiga

ID	Autor	Izdavac

CD

ID	Umjetnik

DVD

ID	Producent	Direktor

PRIMJER

- I dalje, svaka od tabela može imati nedostajuće podatke.
  - Cijena transporta u Artiklima može imati default vrijednost 0
  - To neće remetiti izračunavanja.
  - Ako vrijednost nije zadata, transport je besplatan.
- Za druge kolone dozvoliti NULL-e
  - Izdavač, direktor i producent su opcioni.
  - Nema izgleda da će biti upotrijebljeni u izračunavanjima.

FUNKCIJE KONTROLE TOKA

Ime funkcije	Opis funkcije
CASE	Case operator
IF(izraz1, izraz2, izraz3)	if/else konstukcija
IFNULL(izraz1, izraz2)	NULL if/else konstukcija
NULLIF(izraz1, izraz2)	Vraća NULL ako je izraz1 = izraz2

CASE – vidi predhodnu lekciju.

IF

IF(*izraz1*, *izraz2*, *izraz3*)

Ako je *izaz1* **TRUE** (*izraz1* **<> 0** i *izraz1* **<> NULL**) **IF()** vraca *izraz2*;
inače vraća *izraz3*.

IF() može vratiti numeričku ili znakovnu vrijednost.

Primjeri:

SELECT IF(1>2,2,3); Rezultat: 3.

SELECT IF(1<2,'yes','no'); Rezultat: 'yes'

SELECT IF(STRCMP('test','test1'), 'yes', 'no'); Rezultat: 'no‘.

IFNULL

IFNULL(*izraz1*, *izraz2*)

Ako *izraz1* nije **NULL**, **IFNULL()** vraća *izraz1*; u suprotnom vraća *izraz2*.

IFNULL() može vratiti numeričku ili znakovnu vrijednost.

Primjeri:

SELECT IFNULL(1,0); Rezultat: 1

SELECT IFNULL(NULL,10); Rezultat: 10

SELECT IFNULL(1/0,10); Rezultat: 10

SELECT IFNULL(1/0,'yes'); Rezultat: 'yes'

IFNULL: PRIMJER

Primjer: Prikazati ime, platu, stimulaciju i ukupno primanje radnika relacije radnik. Ukoliko radnik ne prima stimulaciju, smatrati da mu je stimulacija 0.

RJEŠENJE:

```
SELECT ime, plata, stimul,
plata+IFNULL(stimul, 0) AS Ukupno
FROM radnik;
```

NULLIF

NULLIF(*izraz1*, *izraz2*)

Vraća **NULL** ako je *izraz1* **=** *izraz2*, u suprotnom vraća *izraz1*.

Ovo je isto što i:
CASE WHEN *izraz1* **=** *izraz2* **THEN NULL ELSE** *izraz1* **END**.

Primjeri:

SELECT NULLIF(1,1); Rezultat: NULL

SELECT NULLIF(1,2); Rezultat: 1

ZA VJEŽBU

P1. Napisati upit pretraživanja koji će kao rezultat vratiti kompletne zapise radnika relacije radnik koji ne primaju stimulaciju.

P2. Izračunati odnos stimulacije i plate za svakog radnika relacije radnik. Odnos prikazati sa dvije decimalne cifre, sa zaokruživanjem. Za radnike koji ne primaju stimulaciju smatrati da im je stimulacija 0. U izvještaju, osim pomenutog odnosa prikazati i ime, radno mjesto i ime i lokaciju odjeljenja kojemu radnik pripada.

P3. Napraviti upit pretrazivanja koji će kao rezultat vratiti ime radnika, ime odjela radnika, mjesečni dohotak radnika i godišnji dohotak radnika. Zapise sortirati po platama u opadajucem nizu.

ZA VJEŽBU

P4. Izračunati prosječnu stimulaciju, prosječnu platu i ukupnu zaradu za pojedina radna mjesta radnika relacije radnik. Za radnike koji ne primaju stimulaciju smatrati da im je stimulacija 0.

P5. Napraviti upit pretraživanja koji će kao rezultat vratiti sifru odjela, ime odjela, radno mjesto i ime radnika. U izvjestaju treba prikazati i odjele koji nemaju nijednog radnika. Zapise izvještaja posložiti u po imenu odjela u opadajućem redosljedu.

P6. Napraviti upit pretrazivanja nad relacijama RADNIK i ODJEL koji će kao rezultat vratiti ime radnika, ime njegovog rukovodioca, ime i lokaciju odjela kojemu radnik pripada. Prikazati i radnike koji nemaju rukovodioca.

SLJEDEĆA LEKCIJA

- Normalizacija
  - Redudantnost podataka
  - Funkcionalne zavisnosti
  - Normalne forme
  - Prva, Druga i Treća normalna forma.
  - Dekompozicija bez gubitaka. Zašto je redukcija na 2NF i 3NF bez gubitaka.
  - Boyce-Codd-ova normalna forma (BCNF)
  - Više normalne forme.
  - Denormalizacija.