UNIVERZITET U BEOGRADU · MATEMATIČKI FAKULTET

University of Belgrade
Faculty of Mathematics

Uvod u relacione baze podataka

Arhitektura, modeli podataka i relacioni model

Nina Radojičić Matić

2026/2027

NAPOMENA O MATERIJALIMA

Slajdovi su nastali korišćenjem materijala **prof. dr Saše Malkova** za kurs „Uvod u relacione baze podataka” na Matematičkom fakultetu Univerziteta u Beogradu, skripte „Relacione baze podataka” (M. Maljković, I. Tanasijević), kao i ostale literature navedene na kraju.

Od arhitekture do relacionog modela

TEMA 2

Arhitektura baza podataka

Ciljevi SUBP, ANSI-SPARC, tri nivoa šema, nezavisnost podataka

TEMA 3

Modeli podataka

Mrežni, hijerarhijski, relacioni, ER, objektni i nerelacioni modeli

TEMA 4.1

Struktura relacionog modela

Relacija, domen, atribut, torka, ključ, modeliranje odnosa

TEMA 4.2

Manipulativni deo

Relaciona algebra, NULL vrednosti, relacioni račun

TEMA 4.3

Integritetni deo

Domen, ključevi, strani ključ, CHECK, okidači

Ciljevi učenja

Posle ovih časova trebalo bi da umete da:

1	objasnite tri nivoa ANSI-SPARC arhitekture i nezavisnost podataka;
2	uporedite osnovne modele podataka i obrazložite prednosti relacionog modela;
3	precizno definišete relaciju, domen, atribut, torku, natključ i ključ;
4	zapišete jednostavne upite u relacionoj algebri i relacionom računu i prevedete ih u SQL;
5	razlikujete vrste uslova integriteta i definišete ih u sistemu Db2.

TEMA 2

Arhitektura baza podataka

Ciljevi SUBP · komponente sistema · ANSI-SPARC arhitektura · nezavisnost podataka

Šta očekujemo od SUBP

Višekorisnički rad

Deljivost podataka, rad u složenim mrežnim okruženjima, različite potrebe aplikacija i korisnika.

Integritet podataka

Sadržaj baze mora da bude logički ispravan.

Pouzdan rad

ACID svojstva transakcija i otpornost na otkaze.

Bezbedan rad

Ograničen pristup podacima — svako vidi i menja samo ono za šta ima prava.

Efikasnost i pristupačnost

Brz rad sistema i lakoća upotrebe.

Kontrola i transparentnost

Puno administriranje; standardizovana upotreba bez tehničkih detalja.

Komponente sistema baza podataka

Hardver

- računarski sistem
- infrastruktura: mreža, skladišta podataka

Softver

SUBP — najvažnija komponenta

- korisničke aplikacije
- alati za razvoj softvera

Korisnici

- krajnji korisnici
- razvijaoci
- administratori

Tri vrste administratora

Uloga	Odgovoran je za	Stara se o
Administrator SUBP	rad i performanse celog SUBP	raspodeli resursa i prioriteta između baza podataka
Administrator baze podataka	pravljenje i održavanje objekata konkretne baze	ispravnosti i efikasnosti baze; fizičkom projektu
Administrator podataka	sadržaj baze podataka	samo konceptualnom projektu; nije nužno tehničko lice

Ilustracija: jedan administrator održava Db2 server na kome radi više baza; drugi pravi tabele i indekse baze STUD2020; treći (npr. iz studentske službe) odlučuje koji se podaci o studentima uopšte čuvaju.

Zadaci SUBP

- Upravljanje organizovanjem i čuvanjem podataka
- Upravljanje prenosom podataka između komponenti
- Upravljanje algoritmima i metodima upotrebe podataka na niskom nivou
- Pružanje korisničkih i aplikativnih interfejsa
- Staranje o integritetu podataka
- Staranje o bezbednosti (prava pristupa)
- Zaštita podataka u slučaju neispravnosti ili otkaza

Put jednog SQL zahteva

1. Korisnik postavlja zahtev jezikom podataka (npr. SQL upit)
2. SUBP prihvata zahtev i analizira ga
3. Proverava spoljašnju šemu, preslikavanja, konceptualnu i internu šemu
4. Izvršava potrebne operacije nad bazom

Šta je arhitektura sistema?

Arhitektura sistema je konceptualna definicija njegove strukture: komponente, njihove funkcije, međusobni odnosi i interakcije.

U idealnom slučaju

Arhitektura SUBP ispunjava sve postavljene ciljeve i sakriva sve elemente implementacije od korisnika.

U praksi

To nije moguće: model upotrebe podataka je tesno povezan sa modelom njihove organizacije.

Odgovor: **pristup više šema** — različiti pogledi na strukturu sistema i razdvajanje korisničke reprezentacije od implementacije.

Šta ako postoji samo jedna šema?

SCENARIO 1

Aplikacije čitaju direktno fajlove na disku

Administrator promeni format zapisa ili doda indeks — i sve aplikacije prestanu da rade.

Potreba: sakriti fizičku organizaciju

→ interna šema

SCENARIO 2

Svi korisnici vide celu bazu

Student vidi tuđe ocene; studentska služba mora da razume desetine tabela da bi dobila jedan spisak.

Potreba: prilagođeni pogledi

→ spoljašnje šeme

SCENARIO 3

Svaka aplikacija ima svoju kopiju podataka

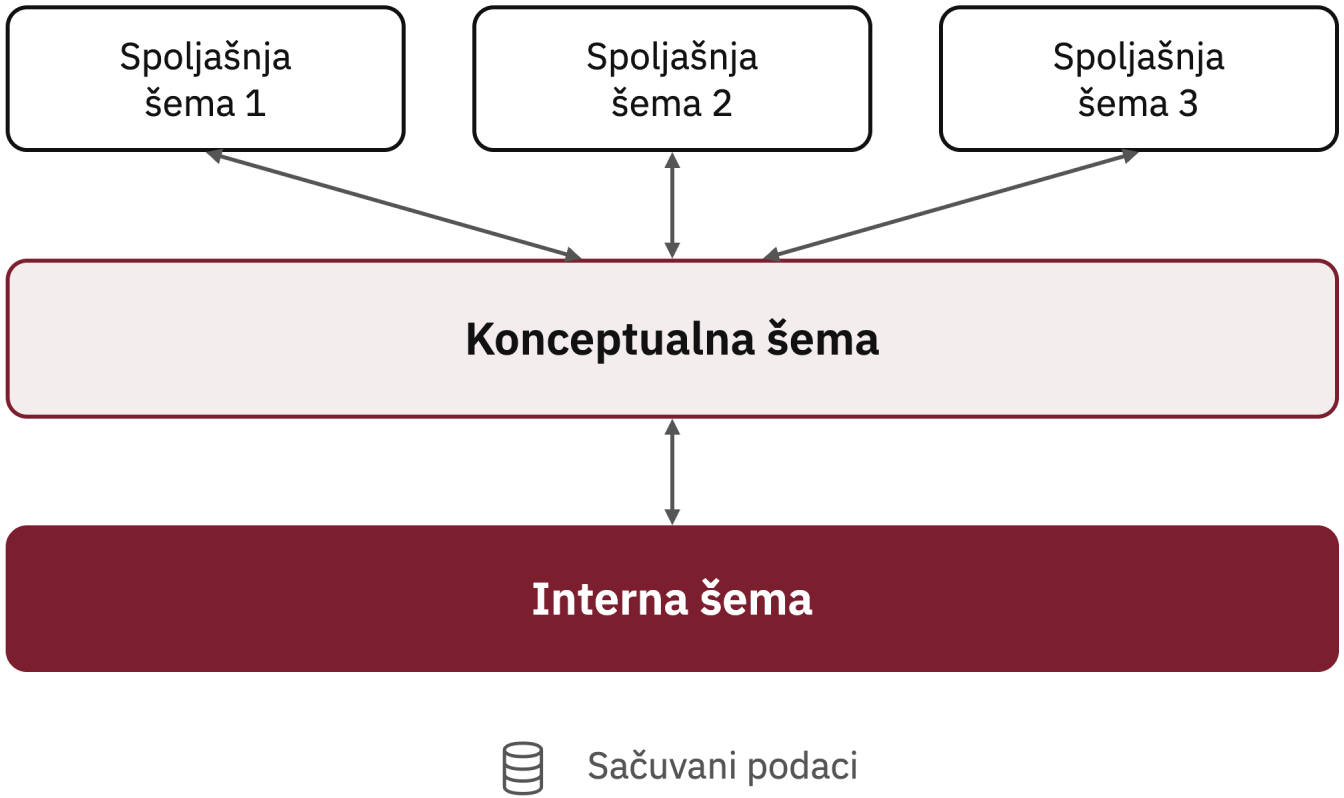
Ista ocena je zapisana na pet mesta — pre ili kasnije kopije se razilaze.

Potreba: jedan zajednički opis

→ konceptualna šema

Tri problema vode do tri nivoa šema — ANSI-SPARC arhitekture.

ANSI-SPARC arhitektura: tri nivoa šema



Predlog standarda iz **1975.** godine — nije formalno usvojen, ali je zaživeo i široko se koristi.

Više spoljašnjih šema — po jedna za korisnika ili grupu korisnika.

Jedna konceptualna i **jedna** interna šema.

Strelice su **preslikavanja**: više spoljašnje/konceptualnih i jedno konceptualno/interno.

Tri nivoa — uporedni pregled

Šema	Šta opisuje	Drugi nazivi	Apstrakcija
Spoljašnja	pogled na bazu iz ugla korisnika; različiti korisnici vide različite, različito organizovane skupove podataka	nivo pogleda, nivo korisnika	relativno visoka
Konceptualna	zaista implementiran skup podataka i odnosa među njima; prilagođena modelu podataka (kod nas relacionom)	logički nivo, nivo administratora	umerena
Interna	implementaciju na fizičkom nivou, onako kako su podaci zapisani u SUBP	nivo sistema	najniža

Konceptualna šema je odvojena od fizičke implementacije, ali ne i od modela organizacije podataka; spoljašnja je uglavnom odvojena i od jednog i od drugog.

Primer: tri nivoa u bazi STUD2020

Spoljašnji
pogled korisnika

```
create view polozeni as
  select indeks, idpredmeta, ocena from da.ispit
  where status = 'o' and ocena > 5
```

Korisnik vidi samo položene ispite, ne celu tabelu.

Konceptualni
logički nivo

Tabele **DA.DOSIJE**, **DA.PREDMET**, **DA.ISPIT**, … sa kolonama, tipovima, ključevima i stranim ključevima.

Interni
nivo sistema

Prostori tabela, stranice na disku, indeksi — u upitima ih uopšte ne pominjemo.

Nezavisnost podataka

Nezavisnost podataka je otpornost aplikacija na promene reprezentacije podataka i tehnika pristupa. Ostvaruje se preslikavanjima između nivoa.

FIZIČKA NEZAVISNOST

Promena interne šeme ne menja konceptualnu šemu niti aplikacije.

Čuva je **jedno** konceptualno/interno preslikavanje.

Npr. dodat je indeks nad DA.ISPIT — upiti ostaju isti, samo su brži.

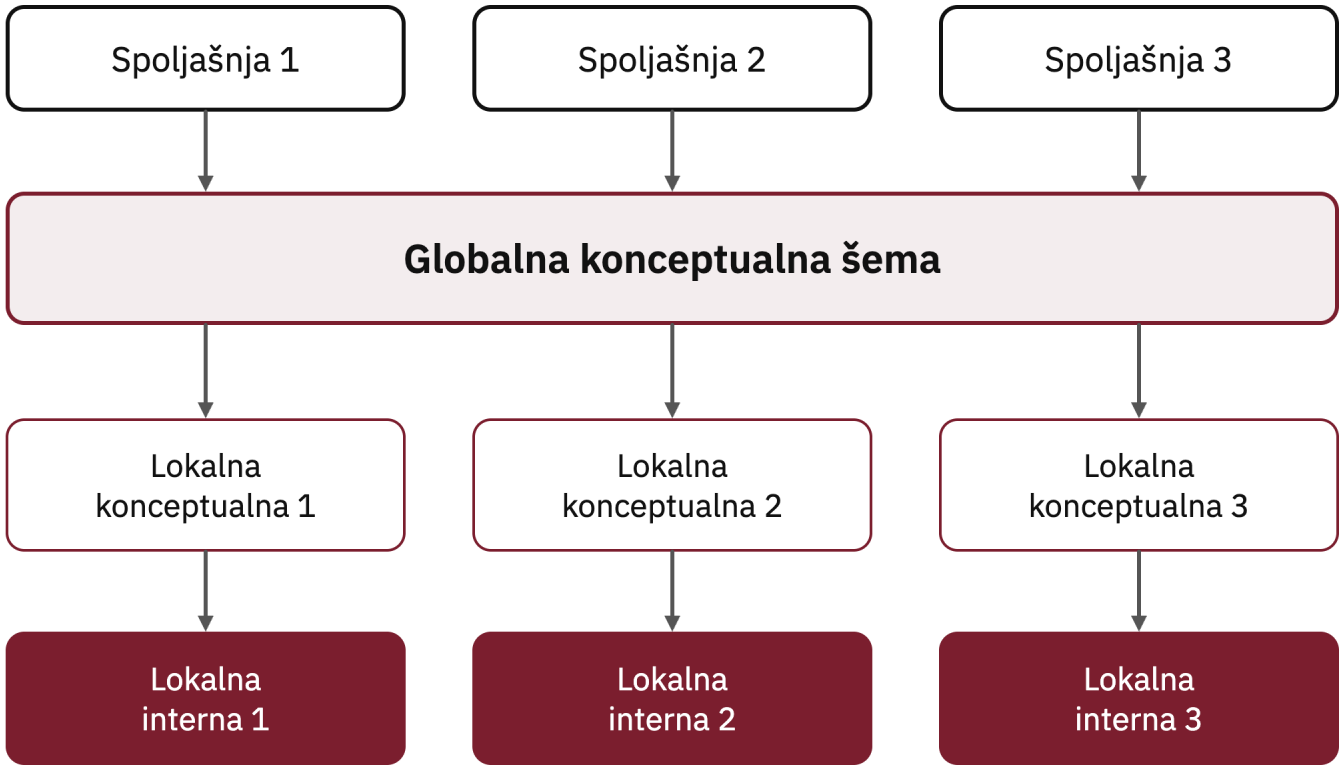
LOGIČKA NEZAVISNOST

Promena konceptualne šeme ne menja spoljašnje šeme niti aplikacije.

Čuvaju je **spoljašnje/konceptualna** preslikavanja (više njih).

Npr. tabeli je dodata kolona — pogled „polozeni” se ne menja.

Arhitektura distribuiranih SUBP



I dalje postoji više spoljašnjih šema.

Konceptualna šema se deli na dva nivoa: **globalnu**, jedinstvenu i centralizovanu, i **lokalne** — po jednu u svakom čvoru.

Svaki čvor ima svoju lokalnu internu šemu.

Kod federativnih baza globalna konceptualna šema ne mora ni da postoji.

Proverite razumevanje

1. Koja su tri nivoa ANSI-SPARC arhitekture i koliko šema svakog nivoa postoji?

2. Šta je fizička, a šta logička nezavisnost podataka?

3. Kom nivou arhitekture pripada pogled (VIEW), a kom indeks?

Proverite razumevanje — odgovori

1. Koja su tri nivoa ANSI-SPARC arhitekture i koliko šema svakog nivoa postoji?

Spoljašnji (više šema — po jedna za korisnika ili grupu), konceptualni (jedna šema) i interni (jedna šema).

2. Šta je fizička, a šta logička nezavisnost podataka?

Fizička: promena interne šeme ne utiče na konceptualnu šemu i aplikacije. Logička: promena konceptualne šeme ne utiče na spoljašnje šeme i aplikacije.

3. Kom nivou arhitekture pripada pogled (VIEW), a kom indeks?

Pogled je deo spoljašnje šeme; indeks pripada internoj šemi. Tabele sa ključevima čine konceptualnu šemu.

TEMA 3

Modeli podataka

Uloga modela · mrežni i hijerarhijski model · relacioni model · ER, objektni i nerelacioni modeli

Šta je model podataka?

Model podataka je relativno apstraktan, standardizovan pogled na problem organizovanja, projektovanja, implementiranja i upotrebe podataka.

Osnovna ideja: podizanjem nivoa apstrakcije ostvariti što „univerzalnije” rešenje. Način modeliranja se vremenom menjao.

Manje opšti modeli mogu da pomognu...

...kada nas opšta rešenja ograničavaju — obično u pogledu performansi (sužen domen primene, konkretizacija).

...ali imaju cenu

Manja fleksibilnost. Može biti korak nazad ako ne znamo gde smo već bili i zašto tamo nismo ostali.

Model podataka i arhitektura

Nezavisno od arhitekture, **model podataka određuje kako korisnik vidi podatke i kako ih upotrebljava**.

Najveće performanse se ostvaruju kada se na svim nivoima arhitekture koristi isti model.

Ipak, na spoljašnjem nivou se često koristi drugačiji model, radi prilagođavanja korisnicima.



Najvažniji modeli podataka kroz vreme

1960-te

Mrežni

IDS, C. Bachman;
CODASYL DBTG 1971.

1968.

Hijerarhijski

IMS, IBM
(program Apolo)

1970.

Relacioni

E. F. Codd

1976.

Entiteti i odnosi

P. Chen

1980-te i 1990-te

Objektni i OR

objektne baze,
objektno-
relacioni model

2000-te

Nerelacioni

„NoSQL” baze

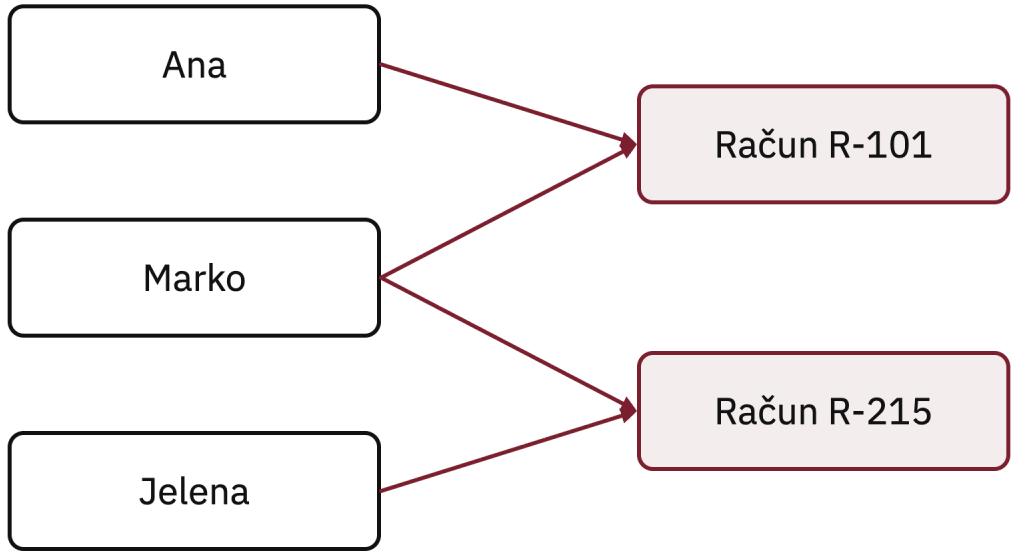
Mrežni model podataka

Slogovi — kao slogovi u programskim jezicima; sadrže podatke jedne instance entiteta i sastoje se od polja.

Veze (link) — kao pokazivači; slično povezivanju elemenata dijagrama strelicama.

Raznolikost veza otežava upotrebu, pa CODASYL (DBTG 1971) propisuje samo **usmerene veze više-jedan**; veza više-više se gradi kao par veza više-jedan.

Primer: klijenti i bankovni računi



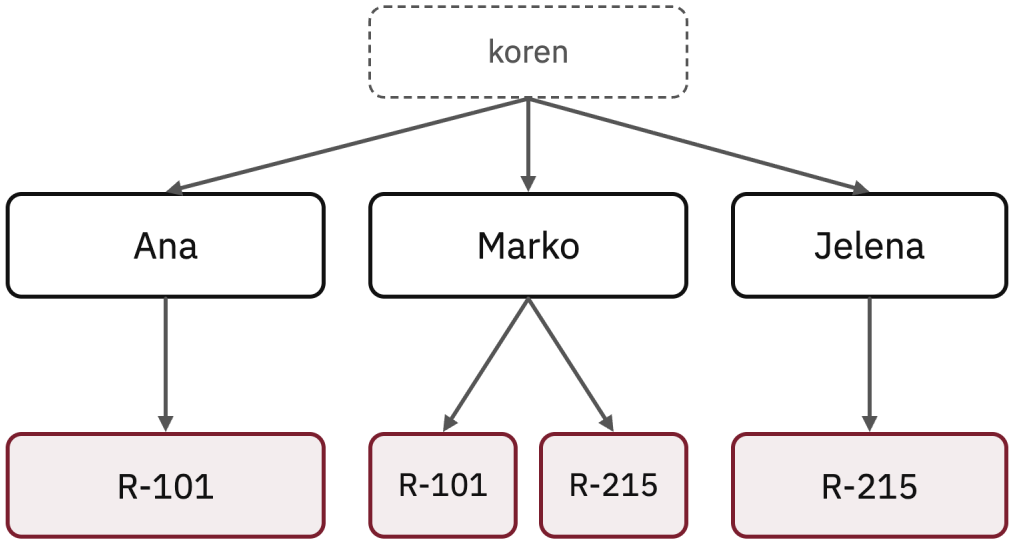
Strelice su veze-pokazivači; račun može imati više vlasnika (više-više).

Hijerarhijski model podataka

Skup slogova povezanih vezama tako da grade **hijerarhiju** (stablo); smer veza je eksplicitno jedan-više.

Do svakog čvora vodi **tačno jedan put** od korena; višestruko vezivanje čvorova nije dopušteno.

Umesto redundantnosti veza imamo **redundantnost podataka**; složeni odnosi i različiti načini pristupa zahtevaju više hijerarhija.



Isti podaci kao na prethodnom slajdu: računi R-101 i R-215 su zapisani po dva puta.

SUŠTINSKI POMAK

Mrežni i hijerarhijski model povezuju mesta gde su podaci zapisani. Relacioni model povezuje same podatke.

Čvor ≈ promenljiva, veza ≈ pokazivač. Relacioni model potpuno zanemaruje semantiku implementacije i bavi se neposredno sadržajem — integritet se obezbeđuje ključevima.

ER, objektno-relacioni i objektni model

Model entiteta i odnosa

P. Chen, 1976.

Zamišljen kao alternativa postojećim modelima. Nije zaživeo kao model implementacije, ali jeste kao **alat za modeliranje** baza podataka.

Objektno-relacioni model

proširenje relacionog

Složeni i korisnički definisani tipovi, ponašanje (klase i metodi), hijerarhije tipova. Potpunije proširenje razmatraju Date i Darwen.

Objektni „model”

objektne baze podataka

Nije ponuđen konzistentan model — gotovo svaki OO SUBP ima sopstveni. Većina uvodi ID objekta, koji ima osobine „pokazivača”.

Nerelacione baze podataka

Šta dobijamo

- visoke performanse — ali ne uvek na punom opsegu primene
- jednostavnije i efikasnije distribuiranje
- veću fleksibilnost

Šta plaćamo

- nisku strukturiranost i često višu redundantnost
- sužene domene primene
- van osnovnih mogućnosti rad se komplikuje ili gubi na performansama

Najčešće vrste

- kataloške baze
- baze sadržaja (multimedijalne kolekcije)
- baze za veb (JSON)
- naučne baze (višedimenzione matrice)
- grafovske baze

Proverite razumevanje

1. Zašto kažemo da su mrežni i hijerarhijski model bliski implementaciji?

2. Koju cenu plaća hijerarhijski model u odnosu na mrežni?

3. Kakvu ulogu danas ima ER model, a kakvu ORM biblioteke?

Proverite razumevanje — odgovori

1. Zašto kažemo da su mrežni i hijerarhijski model bliski implementaciji?

Rade sa čvorovima (≈ promenljive) i vezama (≈ pokazivači), tj. povezuju mesta gde su podaci zapisani, a ne same podatke.

2. Koju cenu plaća hijerarhijski model u odnosu na mrežni?

Do svakog čvora vodi tačno jedan put, pa umesto redundantnosti veza nastaje redundantnost podataka; složeni odnosi traže više hijerarhija.

3. Kakvu ulogu danas ima ER model, a kakvu ORM biblioteke?

ER je alat za modeliranje (projektovanje) baza. ORM programeru pruža objektni model na spoljašnjem nivou, nad relacionom bazom.

TEMA 4.1

Relacioni model: strukturni deo

Značaj i istorija · relacija i njen model · entiteti i atributi · shema · ključevi · modeliranje odnosa

Zašto je relacioni model najznačajniji?

Potpuna formalizacija

Neprotivrečan, nedvosmisleno precizan, pogodan za automatsko obrađivanje i optimizovanje.

Jedinstvena reprezentacija

I podaci i metapodaci predstavljaju se relacijama — npr. katalog SYSCAT u Db2.

Visoka standardizovanost

Relativno visoka prenosivost između sistema.

Velika zastupljenost

Veliki broj sistema i primena; izuzetno dobro proveren i dokumentovan.

Kratka istorija

> *„Relacioni model podataka obezbeđuje predstavljanje podataka samo njihovom prirodnom strukturom, tj. bez uvođenja bilo kakvih dodatnih pretpostavki o strukturi radi mašinske reprezentacije. … predstavlja osnovu za jezike višeg nivoa koji će omogućiti maksimalnu nezavisnost između programa i mašinske reprezentacije.”*
>
> E. F. Codd, A relational model of data for large shared data banks, Comm. ACM 13(6), 1970.

1970.	Coddov rad o relacionom modelu
1974.	IBM System R — prvi prototip
1979.	Oracle — prva komercijalna implementacija

Najrasprostranjeniji RSUBP (DB-Engines, nov. 2020):

Oracle, MySQL, MS SQL Server, PostgreSQL, IBM Db2, SQLite, MS Access, MariaDB, Teradata

Tri dela relacionog modela

TEMA 4.1

Strukturni deo

način modeliranja podataka

TEMA 4.2

Manipulativni deo

način rukovanja modeliranim podacima

TEMA 4.3

Integritetni deo

način obezbeđivanja valjanosti podataka

Osnovna ideja: sve se modelira relacijama — i entiteti i odnosi.

Entiteti su različiti postojeći elementi („objekti”) sistema koji modeliramo; **odnosi** su značajni međusobni odnosi dva ili više entiteta. Isto se iz jednog ugla može videti kao odnos, a iz drugog kao entitet.

Relacija i njen model — primer

Imamo crvenu, plavu, belu i zelenu kutiju, a u njima lopte, kocke i pločice. Relacija **kutijaSadrži(K, P)** je zadovoljena ako kutija K sadrži P.

```
Dom = {crvena, plava, bela, zelena} × {lopte, kocke, pločice}
model = {(plava, lopte), (plava, kocke),
         (zelena, pločice), (crvena, kocke)} ⊆ Dom
```

BOJA	SADRŽAJ
plava	lopte
plava	kocke
zelena	pločice
crvena	kocke

`(plava, kocke)`

tačno

torka pripada modelu

`(zelena, lopte)`

netačno

u domenu je, ali ne u modelu

`(žuta, kocke)`

nije definisano

argument nije u domenu

Entiteti, atributi i torke

Skup entiteta E karakteriše se atributima A(E) = {A1, …, An}: svaki atribut je **funkcija** Ai : E → Di sa jedinstvenim nazivom; Di je domen atributa.

```
φ(e) = (A1(e), …, An(e))    R = φ(E)
```

Ako je φ **injektivna** (različiti entiteti → različite slike), atributi dobro karakterišu E. Ako su još i svi atributi **atomični** (skalarni), R je relacija koja modelira E.

Torku (a1, …, an) = φ(e) često nazivamo i „entitetom e”; Ai(e) zapisujemo kao e.ti.

RADNIK: ime, prezime

IME	PREZIME
Dragana	Pantić
Marko	Marković
Dragana	Pantić
Jelena	Popović

Dve različite radnice, ista slika!

φ nije injektivna — zato uvodimo identifikator (npr. radnik_id, broj indeksa).

Rečnik pojmova

Termin	Značenje	Primer u bazi STUD2020
Relacija	matematički termin za tabelu	DA.DOSIJE
Torka	red (vrsta) tabele	podaci o jednom studentu
Atribut	kolona tabele	INDEKS, IME, PREZIME, …
Domen	skup dopustivih vrednosti (tip)	skup mogućih brojeva indeksa
Kardinalnost	broj torki	broj studenata u dosijeu
Stepen	broj atributa	broj kolona tabele DOSIJE
Primarni ključ	atribut(i) čije vrednosti jedinstveno identifikuju torku	INDEKS

„Relacija” ili „tabela”?

U teoriji i u konceptualnom i logičkom modeliranju kažemo **relacija**; u fizičkom modeliranju obično kažemo **tabela**. Ali relacija je **skup**, pa važi:

1. nema dve jednake torke;
2. redosled torki nije bitan;
3. redosled atributa nije bitan — atributi se referišu imenima, pa imena moraju biti jedinstvena;
4. vrednosti atributa su atomične.

> ⚠ **SQL odstupa od teorije:** tabela bez ključa i rezultat upita mogu da sadrže duplikate. Zato u SQL-u postoji DISTINCT, a za određen redosled — ORDER BY.

Relaciona shema i relaciona baza

Opis relacije čine domen relacije i nazivi atributa.

Relaciona shema je skup opisa relacija koje čine bazu.

Relaciona baza podataka je skup relacija.

Domeni: N(a) — celi brojevi do a cifara; N(a,b) — a cifara levo i b desno od decimalnog zareza; C(a) — niske do a znakova; D — datumi.

```
Kol(RADNIK) = ('radnik_id', 'org_jed_id', 'ime', 'prezime', 'pol',
               'dat_zaposlenja', 'osnova_plate')
Dom(RADNIK) = N(4)×N(3)×C(30)×C(50)×C(1)×D×N(10,2)
```

Isto u SQL-u, za relaciju PROJEKAT:

```
create table projekat (projekat_id decimal(3), proj_bonus decimal(10,2),
naziv varchar(60))
```

Natključ i ključ

Natključ

Podskup atributa K ⊆ A(R) koji jednoznačno određuje torke relacije R.

Ključ (ključ-kandidat)

Natključ K takav da nijedan pravi podskup od K nije natključ — **minimalan** natključ.

Primer — relacija DOSIJE:

`{indeks}`	**ključ**	svaki student ima jedinstven broj indeksa
`{indeks, ime}`	**natključ, nije ključ**	nije minimalan: {indeks} je dovoljan
`{ime, prezime}`	**nije natključ**	dve osobe mogu imati isto ime i prezime

Modeliranje odnosa

Odnosi se modeliraju na isti način kao entiteti — relacijama. Neka su e ∈ E i f ∈ F modelirani kao e = (x1, ..., xn) i f = (y1, ..., ym).

Odnos bez atributa

$\omega(e, f) = (x1,\ldots,xn, y1,\ldots,ym)$
$Dom(\omega) = E \times F$

Ako su KE i KF ključevi, KR = KE ∪ KF je ključ od ω — dovoljna je „sužena" relacija samo sa atributima iz KR.

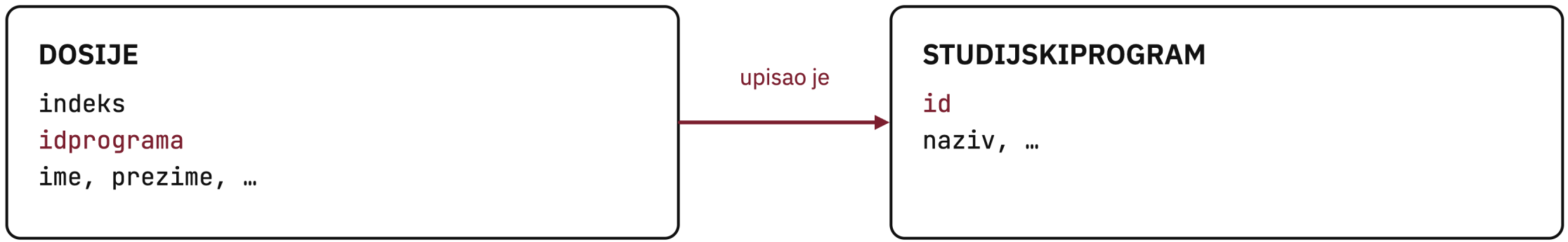
Odnos sa opisnim atributima

Pozajmica u biblioteci: knjiga, član, **datum pozajmice**, **datum vraćanja**.

U STUD2020: **ISPIT** povezuje studenta (indeks) i predmet (idpredmeta), uz opisne atribute kao što je **ocena**.

Integrisani odnosi

Odnos između E i F može da se **integriše** u relaciju koja modelira E samo ako svakom entitetu iz E odgovara **tačno jedan** entitet iz F (ili najviše jedan, ako su dopuštene nedefinisane vrednosti).



Može: student je upisao tačno jedan studijski program, pa DOSIJE sadrži ključ programa.

Ne može: student polaže više predmeta, a predmet više studenata — potrebna je posebna relacija ISPIT.

Proverite razumevanje

1. Koja je razlika između natključa i ključa? Navedite primer.

2. Zašto relacija ne može da sadrži dve jednake torke, a rezultat SQL upita može?

3. Kada se odnos može integrisati u relaciju koja modelira jedan od entiteta?

Proverite razumevanje — odgovori

1. Koja je razlika između natključa i ključa? Navedite primer.

Natključ jednoznačno određuje torke; ključ je minimalan natključ. U DOSIJE je {indeks, ime} natključ, a {indeks} ključ.

2. Zašto relacija ne može da sadrži dve jednake torke, a rezultat SQL upita može?

Relacija je skup. SQL radi sa multiskupovima: duplikati ostaju dok ih ne uklonimo sa DISTINCT.

3. Kada se odnos može integrisati u relaciju koja modelira jedan od entiteta?

Kada svakom entitetu te relacije odgovara tačno jedan (ili najviše jedan) entitet druge — npr. DOSIJE.IDPROGRAMA. Odnos više-više traži posebnu relaciju.

TEMA 4.2

Relacioni model: manipulativni deo

Pojam upita · relaciona algebra · NULL i proširena algebra · relacioni račun n-torki i domena

Upit i formalni upitni jezici

Upit je definicija (ili izračunavanje) nove relacije na osnovu već poznatih relacija baze podataka.

Relaciona algebra

Operacije nad relacijama čiji su rezultati relacije. Opisuje **kako** se rezultat izračunava.

Relacioni račun

Primena predikatskog računa na relacije. Opisuje **šta** rezultat treba da zadovolji.

Relaciono zatvorenje: rezultat svake operacije je relacija, pa operande mogu biti i drugi izrazi — kao podupiti u SQL-u.

Relaciona algebra — pregled operacija

Relaciona algebra je **proširenje skupovne algebre**. Rezultat svake operacije je nova relacija.

SKUPOVNE OPERACIJE

Unija	Presek	Razlika	Proizvod
`R ∪ Q`	`R ∩ Q`	`R \ Q`	`R × Q`

SPECIFIČNE RELACIONE OPERACIJE

Projekcija	Restrikcija	Prirodno spajanje	Slobodno spajanje	Deljenje
`R[X]`	`R[p]`	`R * Q`	`R[p]Q`	`R[Y%Z]Q`

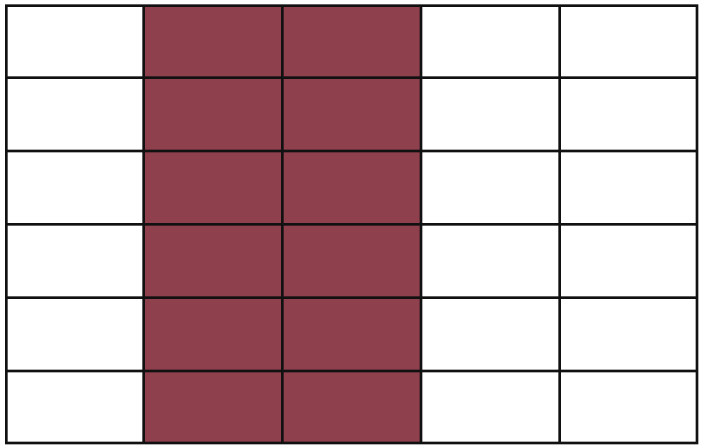
Projekcija — vertikalno „sečenje”

Izbor kolona X relacije R (Y = A(R) \ X):

$$R[X] = \{ x \mid x \in Dom(X) \land (\exists y \in Dom(Y))\ (x, y) \in R \}$$

Rezultat je skup, pa **ne sadrži ponovljene torke**; može imati manje elemenata od R.

Broj torki ostaje isti **ako i samo ako je X natključ** relacije R.



biraju se kolone, ne redovi

Relaciona algebra

```
dosije[ime, prezime]
```

SQL

```
select distinct ime, prezime
from da.dosije
```

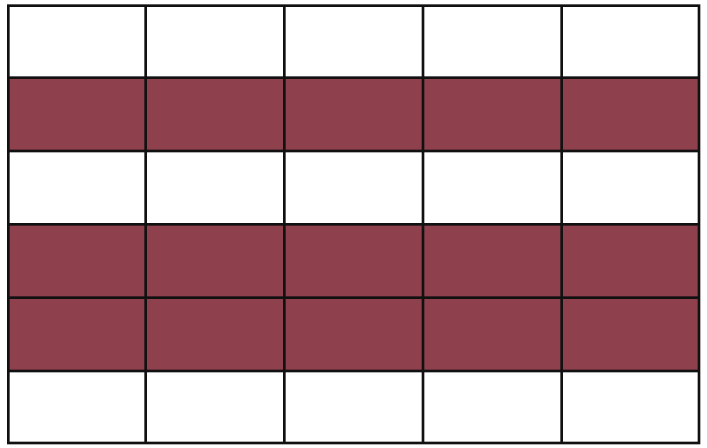
Restrikcija — horizontalno „sečenje”

Izbor torki relacije R koje zadovoljavaju predikat $p : R \to \{T, \bot\}$:

$$R[p] = \{ e \mid e \in R \wedge p(e) \}$$

Najjednostavniji oblik: atribut ima datu vrednost, R[X = x]. Rezultat ima isto zaglavlje kao R.

Operacije se kombinuju — rezultat je opet relacija.



biraju se redovi, ne kolone

Relaciona algebra

```
(dosije where mestorodjenja = 'Beograd')
  [ime, prezime]
```

SQL

```
select distinct ime, prezime from da.dosije
where mestorodjenja = 'Beograd'
```

Proizvod i prirodno spajanje

Dekartov proizvod R1 × R2

Svaka torka iz R1 uparuje se sa svakom torkom iz R2.

X1	X2
a	b
c	d

×

Y1	Y2
1	2
3	4

=

X1	X2	Y1	Y2
a	b	1	2
a	b	3	4
c	d	1	2
c	d	3	4

Prirodno spajanje R1 * R2

Uparuju se torke sa jednakim vrednostima zajedničkih atributa; rezultat ih sadrži samo jednom.

X1	X2
a	b
c	d

*

Y1	X2
1	b
3	f

=

X1	X2	Y1
a	b	1

Relaciona algebra

```
dosije * ispit
```

SQL

```
select * from da.dosije d join da.ispit i
  on d.indeks = i.indeks
```

Slobodno (Θ) spajanje

```
R[p]Q = { (e, f) | e ∈ R ∧ f ∈ Q ∧ p(e, f) }
```

= proizvod, pa restrikcija. Ako je p jednakost — **jednakosno spajanje**. Opštije je od prirodnog.

Primer: parovi studenata rođenih u istom mestu. Ista relacija se koristi dva puta, pa uvodimo **aliase**.

Relaciona algebra

```
define alias d1 for dosije
define alias d2 for dosije
((d1 × d2)
  where d1.mestorodjenja = d2.mestorodjenja
  and d1.indeks < d2.indeks)
[d1.indeks, d2.indeks]
```

SQL

```
select d1.indeks, d2.indeks
from da.dosije d1
  join da.dosije d2
  on d1.mestorodjenja = d2.mestorodjenja
  and d1.indeks < d2.indeks
```

Deljenje — „uparen sa svim”

$$R[Y\%Z]Q = \{\ x \mid x \in R[X] \wedge \{x\} \times Q[Z] \subseteq R\ \}$$

X = A(R) \ Y. Rezultat je **najveći** podskup od R[X] čiji je proizvod sa Q sadržan u R: izdvajaju se oni x koji su u R upareni **sa svim** torkama iz Q.

X1	X2
a	1
a	2
b	1
c	1
c	2
d	2

÷

X2
1
2

=

X1
a
c

Primer: studenti koji su polagali sve predmete

Relaciona algebra

```
ispit[indeks, idpredmeta]
  [idpredmeta % id]
predmet[id]
```

SQL — dvostruki NOT EXISTS

```
select indeks from da.dosije d where not exists
  (select * from da.predmet p where not exists
    (select * from da.ispit i
     where i.indeks = d.indeks and i.idpredmeta = p.id))
```

Relaciona algebra i SQL — podsetnik

Operacija relacione algebre	Odgovarajući SQL
Projekcija R[X]	`select distinct X from R`
Restrikcija R[p]	`select * from R where p`
Proizvod R × Q	`select * from R, Q`
Prirodno / jednakosno spajanje	`R join Q on R.a = Q.a`
Slobodno spajanje R[p]Q	`R join Q on p`
Unija, presek, razlika	`union, intersect, except`
Deljenje	`dvostruki not exists`

Proširena algebra: nedefinisane vrednosti

Svaki domen se proširuje **nedefinisanom vrednošću** (NULL), bez obzira na razlog zbog kog vrednost nedostaje.

Istinitosne vrednosti se proširuju **nepoznatom** vrednošću n; svako poređenje sa NULL daje n.

∧ (AND)

∧	T	F	n
T	T	F	n
F	F	F	F
n	n	F	n

∨ (OR)

∨	T	F	n
T	T	T	T
F	T	F	n
n	T	n	n

¬ (NOT)

x	¬x
T	F
F	T
n	n

U SQL-u

`null = null` → n

`ocena > 5` → n (za `ocena = null`)

Zato se piše IS NULL.

Proširena restrikcija i spajanje

Sa tri istinitosne vrednosti postoje tri vrste restrikcije. Primer: ispit[ocena > 5]

Tačna

`R[p]T : p(x) = T`

Odgovara originalnoj restrikciji. Ispiti sa poznatom ocenom većom od 5.

SQL: klauzula WHERE

Možda

`R[p]n : p(x) = n`

Nova operacija. Ispiti kod kojih ocena nije upisana.

Za ovaj uslov u SQL-u: where ocena is null

Ne-lažna

`R[p]Tn : p(x) ≠ F`

Unija prethodne dve. Sve osim ispita sa poznatom ocenom 5 ili manjom.

SQL: uslovi CHECK (Tema 4.3)

Analogno se proširuju i spajanja: tačno, možda i ne-lažno prirodno i slobodno spajanje.

Spoljašnje operacije

Spoljašnje spajanje zadržava i torke koje nemaju par, dopunjene nedefinisanim vrednostima (n). **Spoljašnja unija** spaja relacije koje nemaju sve iste atribute.

R

A	B
1	x
2	y

Q

B	C
x	p
z	q

levo

A	B	C
1	x	p
2	y	n

desno

A	B	C
1	x	p
n	z	q

puno

A	B	C
1	x	p
2	y	n
n	z	q

Svi predmeti i, ako postoje, njihovi ispiti

```
select p.naziv, i.indeks, i.ocena
from da.predmet p left join da.ispit i
  on p.id = i.idpredmeta
```

Spoljašnji presek je uvek prazan, a spoljašnja razlika uvek jednaka prvoj relaciji — zato nemaju smisla.

Relacioni račun n-torki

```
{ (t1, t2, …, tk) : f }
```

t1, ..., tk su n-torne promenljive — **ciljna lista**; f je **kvalifikacioni izraz**, čije su slobodne promenljive tačno one iz ciljne liste.

Atomi

pripadnost R(s); **poređenje** s[i] Θ p[j] ili s[i] Θ a

Formule

atomi, AND, OR, NOT, (∃s)(f), (∀s)(f) i zagrade — ništa drugo

Radnici koji zarađuju bar 40000 — dva zapisa istog upita:

Formalni zapis

```
{ (x[3], x[4], x[6]) :
  RADNIK(x) AND x[7] ≥ 40000 }
```

Zapis prilagođen računaru

```
range of x is radnik
x.ime, x.prezime, x.dat_zaposlenja
where x.osnova_plate ≥ 40000
```

Kvantifikatori: EXISTS i FORALL

`exists x (uslov)`

tačno ako uslov važi za **neku** torku promenljive x

`forall x (uslov)`

tačno ako uslov važi za **sve** torke promenljive x

`forall = not exists not`

zato je dovoljno podržati samo exists

Primer: studenti koji su položili sve predmete

Relacioni račun n-torki

```
range of d is dosije
range of i is ispit
range of p is predmet
d.ime, d.prezime
where forall p (exists i (
  p.id = i.idpredmeta
  and d.indeks = i.indeks
  and i.ocena > 5))
```

SQL

```
select ime, prezime
from da.dosije d
where not exists (
  select * from da.predmet p
  where not exists (
    select * from da.ispit i
    where p.id = i.idpredmeta
    and d.indeks = i.indeks
    and i.ocena > 5))
```

Relacioni račun domena

Promenljive se ne vezuju za relacije, već za **domene atributa** (domenske promenljive). Umesto liste atributa navodi se lista domenskih promenljivih.

Uslov pripadnosti

R(A1 x1, A2 x2, ...) je tačan ako u R postoji torka u kojoj je svaki atribut Ai jednak xi (promenljivoj ili konstanti).

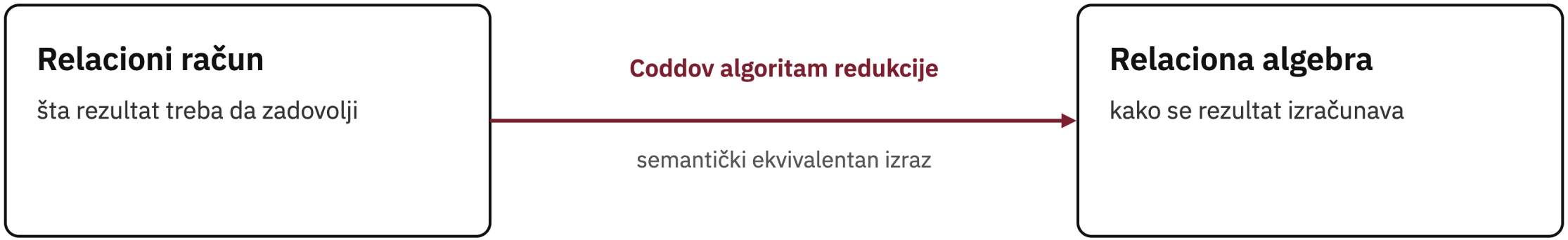
Imena studenata iz Beograda

```
imex
where dosije(ime imex,
      mestorodjenja 'Beograd')
```

Imena i indeksi studenata koji su polagali Analizu 1

```
imex, indeksx
where exists idx (
  predmet(id idx, naziv 'Analiza 1')
  and ispit(idpredmeta idx, indeks indeksx)
  and dosije(indeks indeksx, ime imex))
```

Algebra ili račun?



Relaciona kompletnost

Jezik je relaciono kompletan ako je moćan bar koliko i relacioni račun, odnosno relaciona algebra — osnovna mera izražajnosti.

A SQL?

Ima osobine oba: FROM i JOIN su „algebarski”, a EXISTS i uslovi u WHERE „računski”.

Proverite razumevanje

1. Kada projekcija ima isti broj torki kao polazna relacija?

2. Kojom operacijom algebre se izražava „polagao je sve predmete” i kako se to piše u SQL-u?

3. Da li upit sa WHERE ocena > 5 vraća ispite čija je ocena NULL? Zašto?

Proverite razumevanje — odgovori

1. Kada projekcija ima isti broj torki kao polazna relacija?

Ako i samo ako je skup atributa projekcije natključ. Inače se ponovljene torke uklanjaju, pa ih može biti manje.

2. Kojom operacijom algebre se izražava „polagao je sve predmete” i kako se to piše u SQL-u?

Deljenjem. U SQL-u dvostrukim NOT EXISTS: ne postoji predmet za koji ne postoji ispit tog studenta.

3. Da li upit sa WHERE ocena > 5 vraća ispite čija je ocena NULL? Zašto?

Ne. Uslov je tada nepoznat (n), a WHERE je tačna restrikcija — zadržava samo torke za koje je uslov tačan.

TEMA 4.3

Relacioni model: integritetni deo

Konzistentnost i integritet · domen · ključevi · strani ključ · specifični uslovi · okidači

Konzistentnost i integritet

Konzistentnost

Saglasnost sadržaja baze sa stvarnim prostorom koji se modelira — tačnost, ispravnost. Baza nije konzistentna ako sadrži netačne ili nepotpune podatke.

Integritet

Logička ispravnost sadržaja baze. Ostvaruje se proveravanjem ispunjenosti logičkih pravila integriteta.

Ocena 11 — narušen integritet

Vrednost je van dozvoljenog opsega; SUBP to **može** da spreči.

Upisana 8 umesto 9 — baza nije konzistentna

Logički je ispravno, ali netačno; SUBP to **ne može** da otkrije.

Vrste uslova integriteta

Uslovi integriteta se u relacionom modelu opisuju **predikatima** (istinitosnim formulama) nad relacijom ili bazom; formalnost modela olakšava njihovo formulisanje.

IMPLICITNI

Opšti uslovi

Ugrađeni su u model i implementaciju SUBP; ne definišu se posebno za svaku relaciju — pri određivanju strukture zadaju se samo njihovi parametri.

- integritet domena
- integritet ključa
- integritet stranog ključa

EKSPLICITNI

Specifični uslovi

Odnose se na pojedinu relaciju, atribut ili bazu; određuju se pri određivanju strukture baze, a odgovarajući logički uslovi se eksplicitno navode.

- uslovi na atributu i torki
- uslovi na relaciji
- uslovi na bazi podataka

Integritet domena

> *„Svaka relacija mora da ima tačno određen domen.”*

Svaka vrednost atributa u bazi mora da pripada **domenu tog atributa** — posledica strukturnog dela modela.

Tehnički se ostvaruje određivanjem **tipa kolone**, koji može da obuhvata:

- dužinu podatka;
- da li su dopuštene nedefinisane vrednosti;
- podrazumevanu vrednost;
- (opciono) jedinstvenost.

Db2 — tipovi kolona (pojednostavljen primer)

```
create table ispit_primer (
  indeks     integer     not null,
  idpredmeta integer     not null,
  status     char(1)     not null
             default 'p',
  ocena      smallint,
  datum      date
)
```

Vrste ključeva

Vrsta	Značenje	Primer
Primarni	jedan izabran ključ; preko njega SUBP proverava jedinstvenost torki	DOSIJE.INDEKS
Alternativni (jedinstveni)	svaki ključ osim primarnog	npr. JMBG uz broj indeksa
Strani	skup kolona koji predstavlja ključ povezane relacije	ISPIT.INDEKS → DOSIJE
Složeni	sastoji se od više atributa	ključ tabele ISPIT
Objedinjeni	složeni ključ sastavljen samo od dva ili više stranih ključeva	(indeks, idpredmeta) u vezi student–predmet
Surogat	veštački uveden atribut sa ulogom ključa	PREDMET.ID

Primarni i jedinstveni ključ

Integritet primarnog ključa

„Svaka relacija ima bar jedan ključ” — jer je relacija skup. Za primarni se proglašava tačno jedan ključ, koji ne sadrži nedefinisane vrednosti.

Integritet jedinstvenosti (entiteta)

Primarni ključ ne sme da sadrži NULL, niti dve torke smeju imati iste vrednosti primarnog ključa.

Jedinstveni ključevi

Ostali (alternativni) ključevi se deklarišu kao jedinstveni; koriste se ređe za referisanje.

Db2

```
create table student (
  indeks integer  not null,
  jmbg   char(13) not null,
  ime    varchar(50),
  constraint pk_student
    primary key (indeks),
  constraint uk_student_jmbg
    unique (jmbg)
)
```

Ako NULL mora biti dopušten, a poznate vrednosti jedinstvene: create unique index … exclude null keys

Referencijalni integritet

Uslovi o međusobnim odnosima koje moraju da zadovoljavaju torke dveju relacija. Ostvaruje se **integritetom stranog ključa**.

Ne sme se obrisati ni izmeniti...

...torka na koju se odnosi neka torka, tako da referenca postane neispravna.

Npr. ne briše se student koji ima ispite.

Ne sme se dodati...

...torka sa neispravnom referencom, tj. takvom da ne postoji torka na koju se odnosi.

Npr. ne upisuje se ispit za nepostojeći indeks.

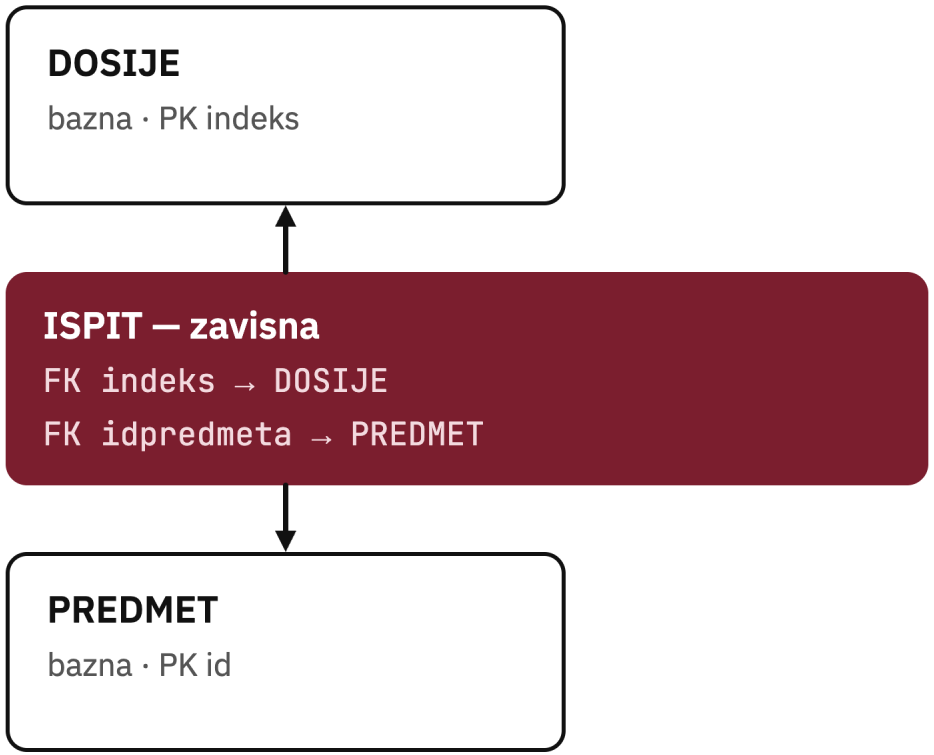
Uz NULL vrednosti Referenca je ispravna i ne referiše ništa ako je **u potpunosti** nedefinisana; u većini implementacija dovoljno je da je **bar jedan** njen deo nedefinisan.

Integritet stranog ključa

Skup FK atributa relacije R je njen **strani ključ** koji se odnosi na **baznu relaciju** B akko:

1. B ima primarni ključ PK;
2. domen ključa FK je identičan domenu ključa PK;
3. svaka vrednost FK u torkama R jednaka je PK bar jedne torke iz B.

R je **zavisna** relacija, a B **bazna** ili **roditeljska**.



Pravila brisanja i ažuriranja

Šta se dešava pri pokušaju da se obriše (ili promeni primarni ključ) torke bazne relacije B na koju se odnosi neka torka zavisne relacije R? Bira se tačno jedno pravilo.

`RESTRICT`	**Aktivna zabrana** — promena torke iz B se odbija.
`NO ACTION`	**Pasivna zabrana** — kao RESTRICT, ali se provera odlaže do kraja naredbe, jer neko kaskadno pravilo možda menja zavisnu torku.
`CASCADE`	**Kaskadno** — brišu se zavisne torke, odnosno na isti način im se menja strani ključ.
`SET NULL`	Strani ključ u zavisnim torkama dobija nedefinisane vrednosti.
`SET DEFAULT`	Strani ključ u zavisnim torkama dobija podrazumevane vrednosti.

Db2: ON DELETE — RESTRICT, NO ACTION, CASCADE, SET NULL; ON UPDATE — RESTRICT, NO ACTION. Podrazumevano: NO ACTION.

Strani ključ u Db2

Sintaksa

```
create table ime_tabele (
  …
  [constraint ime_kljuca]
    foreign key (lista_kolona)
    references bazna_tabela
      [(lista_kolona)]
    [on delete pravilo]
    [on update pravilo]
  …
)
```

Primer (pojednostavljen)

```
create table ispit_primer (
  indeks     integer not null,
  idpredmeta integer not null,
  ocena      smallint,
  constraint fk_ispit_student
    foreign key (indeks)
    references student
    on delete cascade,
  constraint fk_ispit_predmet
    foreign key (idpredmeta)
    references predmet (id)
    on delete restrict
)
```

Specifični uslovi: od atributa do baze

Logička pravila koja uvek moraju da budu ispunjena, prema opsegu podataka koji se proveravaju:

LOKALNI

Atribut

Zavisi samo od vrednosti jednog atributa; sužava domen ili proverava oblik vrednosti.

`CHECK uz kolonu`

podržava većina SUBP

LOKALNI

Torka

Zavisi od svih atributa jedne torke; npr. početak intervala nije posle kraja.

`CONSTRAINT … CHECK`

podržava većina SUBP

GLOBALNI

Relacija

Zavisi od svih torki jedne relacije; npr. broj torki ili suma po atributu.

`CHECK sa podupitima`

ne podržava većina SUBP

GLOBALNI

Baza podataka

Odnosi se na sadržaj različitih relacija, i to šire od referencijalnog integriteta.

`CREATE ASSERTION`

ne podržava većina SUBP

Specifični uslovi — primeri

Na atributu

```
ocena smallint
  check (ocena between 5 and 10)
```

Na torki

```
god_od int, god_do int,
constraint opseg check (god_od ⩽ god_do)
```

Na relaciji

```
constraint c1 check (
  (select sum(a) from t1) <
  (select sum(b) from t1))
```

Na bazi podataka

```
create assertion asrt_1 check (
  (select count(*) from t1) <
  (select count(*) from t2))
```

⚠ Uslov mora da bude **ne-lažan**: ako je ocena NULL, uslov ocena between 5 and 10 je nepoznat — i red se prihvata. Za zabranu NULL vrednosti potreban je NOT NULL.

Kada se uslovi proveravaju?

Podrazumevano

Pri svakoj promeni sadržaja baze podataka.

Odloženo

Ponekad tek na kraju transakcije — npr. kada se dve torke međusobno referišu, pa nijedna ne može da se upiše prva.

Većina RSUBP omogućava da se promeni podrazumevani način rada ili način rada u tekućoj transakciji. U SQL standardu:

```
set constraints all deferred
```

Aktivno održavanje integriteta: okidači

Na određene promene reaguje se pokretanjem eksplicitno definisanog programskog koda, koji pravi potrebna usklađivanja povezanih podataka. Mehanizam: **okidači** (engl. trigger).

Relacija	Vrsta promene	Trenutak
na kojoj se prate promene — ON tabela	`INSERT | UPDATE | DELETE`	BEFORE ili AFTER promene
Granularnost	**Referisanje torki**	**Programski kod**
FOR EACH ROW ili FOR EACH STATEMENT	OLD (nema kod INSERT), NEW (nema kod DELETE)	naredba SQL-a, blok BEGIN ... END, serverska procedura...

Okidači — sintaksa i primer

Sintaksa (pojednostavljena)

```
create [or replace] trigger ime
  {[no cascade] before | after}
  {insert | update | delete}
  on ime_tabele
  [referencing {old | new}
    as ime]
  for each {row | statement}
  [when (uslov)]
  programski_kod
```

Broj radnika u org. jedinici

```
create or replace trigger insert_radnik
  after insert on radnik
  referencing new as n
  for each row
begin
  update org_jedinica
  set broj_radnika = broj_radnika + 1
  where org_jed_id = n.org_jed_id;
end
```

Uz njega ide i simetričan okidač delete_radnik (before delete, referencing old as r, broj_radnika – 1).

Kada koristiti okidače?

Održavanje podataka u bazi

Najčešće loša ideja

Signal da u bazi postoje redundantni podaci — potencijalno ozbiljan problem. Obično se opravdava performansama (npr. prosečna ocena kao kolona), ali postoje i bolja rešenja, npr. materijalizovani pogledi.

Akcije van baze podataka

Može biti dobar izbor

Povezivanje baze sa drugim elementima informacionog sistema: kada stanje proizvoda padne ispod nekog nivoa, pošalje se porudžbina dobavljaču; ili se ažurira sadržaj neke druge baze.

Proverite razumevanje

1. Koja je razlika između konzistentnosti i integriteta baze podataka?

2. Šta se dešava pri brisanju studenta koji ima ispite uz ON DELETE CASCADE, a šta uz RESTRICT?

3. Da li CHECK (ocena BETWEEN 5 AND 10) dopušta NULL ocenu? Zašto?

Proverite razumevanje — odgovori

1. Koja je razlika između konzistentnosti i integriteta baze podataka?

Integritet je logička ispravnost (pravila koja SUBP proverava); konzistentnost je saglasnost sa stvarnošću. Baza može ispunjavati sva pravila, a sadržati netačan podatak.

2. Šta se dešava pri brisanju studenta koji ima ispite uz ON DELETE CASCADE, a šta uz RESTRICT?

CASCADE: brišu se i svi njegovi ispiti. RESTRICT: brisanje studenta se odbija.

3. Da li CHECK (ocena BETWEEN 5 AND 10) dopušta NULL ocenu? Zašto?

Dopušta: uslov je tada nepoznat, a CHECK traži samo da ne bude lažan. Za zabranu NULL potreban je NOT NULL.

Rezime relacionog modela

Puna matematička zasnovanost

Neprotivrečan i nedvosmislen model.

Jedinstven i potpun koncept relacije

Relacija modelira i entitete i odnose, tj. sav sadržaj baze.

Jednostavan integritetni deo

Stroga formalnost omogućava formalne i precizne uslove integriteta.

Univerzalan pristup na svim nivoima

Isti model od interne, preko konceptualne, do spoljašnje šeme; tehnike modela omogućavaju potpuno razdvajanje nivoa.

Literatura za teme 2–4

G. Pavlović-Lažetić	**Uvod u relacione baze podataka**, 2. izdanje, Matematički fakultet, 1999. `poincare.matf.bg.ac.rs/~gordana//urbp-2016.htm`
M. Maljković, I. Tanasijević	**Relacione baze podataka** (skripta), glave 2–5
R. Ramakrishnan, J. Gehrke	**Database Management Systems**, 2. izdanje, 2000.
A. Silberschatz i dr.	**Database System Concepts**, 6. izdanje, 2010.
E. F. Codd	**A relational model of data for large shared data banks**, Comm. ACM 13(6), 1970; **Extending the database relational model to capture more meaning**, ACM TODS 4(4), 1979; **The Relational Model for Database Management – Version 2**, 1990.
H. Darwen, C. J. Date	**The Third Manifesto**, 1995.
IBM	**Dokumentacija za Db2 11.5** — ibm.com/docs/en/db2/11.5