Baze podataka

Uvod

Modeliranje podataka (*Data modeling*) je tehnika organiziranja i dokumentiranja podataka sustava.

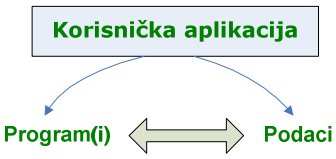
rezultat → **Baza podataka** sustava

Organiziranje podataka
Obrada podataka
Rukovanje podacima
Sigurnost podataka

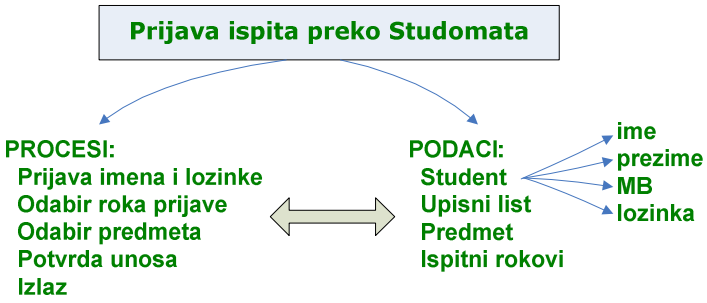
} sve to osigurava **Baza podataka**

450 - Baze podataka *uvod*

Osnova svake korisničke aplikacije su programi i podaci koje programi obrađuju:



Primjer 1.:



3

450 - Baze podataka *uvod*

Baza podataka je skup međusobno povezanih podataka, pohranjenih zajedno bez štetne ili nepotrebne (nekontrolirane) zalihosti (redundancije), s ciljem da ih koriste različite aplikacije.

Podaci su pohranjeni u obliku neovisnom od programa koji ih koriste.

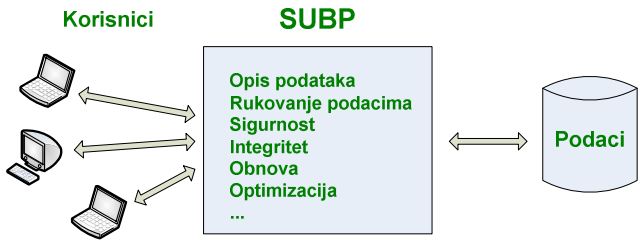
Ubacivanje, promjena, brisanje i čitanje podataka obavlja se posredstvom zajedničkog softvera.

Pri tome, korisnici i aplikacije ne moraju poznavati detalje fizičkog prikaza podataka, već se referenciraju na logičku strukturu baze.

4

450 - Baze podataka *uvod*

SUBP - Sustav za upravljanje bazom podataka (*DBMS – Database Management System*) je programski sustav koji osigurava osnovne funkcije rada sa bazom podataka.



Zahtjevi koji se postavljaju nad suvremenim **SUBP**:

- Opis i rukovanje podacima pomoću posebnog jezika (SQL-Structured Query Language)
- Zaštita integriteta podataka
- Visok nivo sučelja prema korisniku (bez obzira na strukturu podataka)

5

450 - Baze podataka *uvod*

Osnovne funkcije koje moraju biti ugrađene u svaki **SUBP**:

- Definicija podataka (SUBP mora biti u stanju definiciju podataka izrečenu u nekom jeziku za definiranje podataka pretvoriti u interni oblik pohrane podataka)
- Rukovanje podacima (dodavanje novih podataka, izmjena postojećih podataka, brisanje, …)
- Integritet i zaštita podataka (SUBP čuva suvislost podataka a zahtjeve koji narušavaju integritet podataka DBMS treba prepoznati i odbaciti)
- Oporavak u slučaju pogreške (SUBP mora imati mogućnost ponovne uspostave konzistentnosti podataka)
- Efikasnost (sve nabrojane funkcije SUBP mora osigurati uz što bolje performanse).

6

450 - Baze podataka *uvod*

Administrator baze podataka (*DBA – Database Administrator*)

- Zadužen za izvedbu i održavanje baze podataka
- Ima najveću razinu korisničkih prava u pristupu i rukovanju
- Dodaje ostale korisnike
- Ostalim korisnicima dozvoljava ili brani pristup pojedinim podacima.

7

450 - Baze podataka *Arhitektura BP*

Arhitektura (nivoi) baze podataka

Interni nivo (fizička razina):

- način fizičkog spremanja podataka i rukovanja podacima
- odnosi se na raspored podataka na vanjskim jedinicama

Koncepcijski nivo (globalna logička razina):

- logička struktura cijele baze
- oblik koji vidi projektant baze odnosno administrator
- zapis logičke definicije naziva se **shema** tj. tekst ili dijagram koji definira logičku strukturu

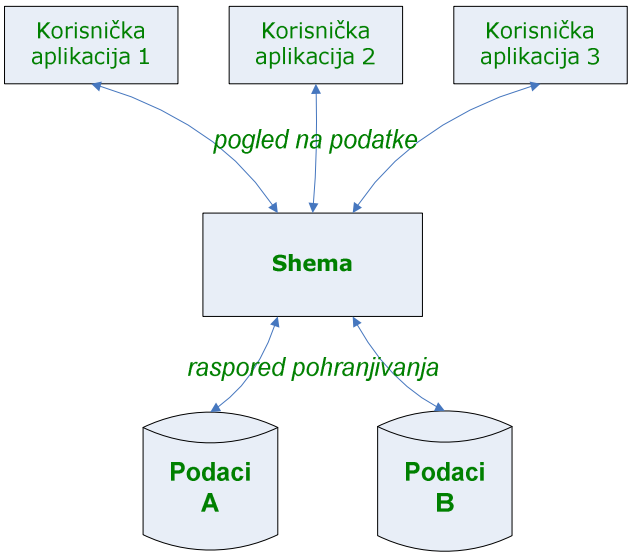
8

450 - Baze podataka *Arhitektura BP*

Eksterni nivo (lokalna logička razina):
- orijentiran prema korisniku
- logička predodžba o dijelu baze kojeg koristi pojedina aplikacija
- podrazumjeva izradu front-end korisničke aplikacije za rad sa bazom
- izbor programskog jezika zavisi o mogućnostima pojedinih alata kao i potrebama krajnje aplikacije

9

450 - Baze podataka *Arhitektura BP*



Eksterni nivo

Koncepcijski nivo

Interni nivo

10

Tipovi i strukture baze podataka

Centralna baza podataka – podaci su smješteni na jednom mjestu, zahtjevi i obrada podataka vrši se na jednom računalu, terminalske stanice daju zahtjeve i dobivaju rezultate operacija obrade podataka.

Client - Server pristup – podaci su smješteni na središnjem računalu server), korisnici putem mreže pristupaju bazi, PC računala na strani klijenta djelomično sudjeluju u obradi podataka

Paralelna struktura baze podataka – baza se formira u okružju računala međusobno povezanih u mrežu, istovremeno se koristi više resursa (najčešće za obradu podataka, dok je baza smještena na jednom računalu)

11

Distribuirana baza podataka – podaci su smješteni na više računala koja su mrežno povezana. Korisnik ne 'vidi' tu 'raspršenost' podataka, on u svom radu ima osjećaj da pristupa jednoj, središnjoj bazi.

Razlozi distribuiranosti su brojni:

- korisnici su smješteni na više udaljenih lokacija (multinacionalne kompanije)
- distribucijom se povećava raspoloživost i pouzdanost sustava
- u distribuiranim sustavima se koristi tehnika repliciranja istih podataka na više lokacija u mreži.

12

450 - Baze podataka *Tipovi BP*

Da bi se u potpunosti iskoristile prednosti distribuiranih sustava, oni moraju omogućiti:

- pristup udaljenim računalima u mreži
- prijenos upita i podataka među računalima
- održavanje konzistentnosti podataka koji se repliciraju
- izradu strategije za izvođenje pretraživanja i obrada koje dohvaćaju podatke iz više lokalnih baza
- oporavak sustava u slučaju ispada pojedinog računala iz mreže.

13

450 - Baze podataka *Entiteti*

Entiteti

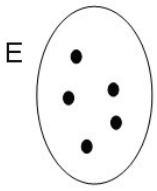
Baza podataka je slika stvarnog procesa iz okoline.

Definicija: **Entitet** (*Entity*) je skup objekata realnog svijeta koji imaju neka zajednička svojstva.

Definicija: Svojstva entiteta se nazivaju **atributima**.

$E = \{e_1, e_2, e_3, e_4, \ldots, e_n\}$

$e_1, \ldots, e_n$ - elementi entiteta (*entity instance, entity occurence*)



Entitet kao skup

Elementi entiteta imju neka zajednička svojstva koja ih opisuju.

14

450 - Baze podataka *Entiteti*

***Primjer*:**

Za opis procesa studiranja osnovni entitet je STUDENT, tj. skup svih studenata sa nekim zajedničkim svojstvima. Svaki student – pojedinac – predstavlja jedan element skupa STUDENT.

Mogući atributi studenata su: ime, prezime, adresa, JMBG, matični broj, datum rođenja itd.

Još primjera entiteta:

- osoba, npr. Karmen Klarin
- objekt, npr. DVD Mr. and Mrs. Smith
- apstraktni pojam, npr. hrvatski jezik
- ustanova (FESB), organizacija (hotel Lav)
- događaj – prošli, sadašnji ili budući – rođenje, školovanje, radni vijek, smrt
- povezanost različitih objekata, npr. zaposlenost

15

450 - Baze podataka *Entiteti*

Za prikaz entiteta, kao skupa objekata potrebno je utvrditi:

Selekciju atributa - utvrditi koji atributi (svojstva) su potrebni za opisivanje entiteta.

Integritet atributa – odrediti tip podataka, ograničenja i pravila vezana za pojedini atribut.

Definiranjem tipa podataka i ograničenja osigurava se suvislost podataka pojedinog atributa.

Primjer: Entitet STUDENT i atribut DATUM ROĐENJA.

Potrebno osigurati da podatak o datumu rođenja u standardnom formatu datuma *dan.mjesec. godina.*, te da po svojoj vrijednosti bude logičan.

Kardinalitet atributa – podatak o zastupljenosti pojedinih atributa.

16

Definicija: Kardinalitet atributa je broj koji kazuje koliko vrijednosti pojedini atribut daje za opis jednog elementa entiteta.

Primjer: Promatra se entitet STUDENT i njegov atribut Ime.

Kardinalitet atributa Ime u entitetu STUDENT kazuje koliko imena može imati student-pojedinac.

Pri utvrđivanju kardinaliteta određuje se donja i gornja granica – minimalni i maksimalni kardinalitet,

Kardinalitet atributa A u entitetu E opisuje se uređenim parom

c**ard(A,E)=(min card(A,E), max card(A,E))**

gdje je: **A** - atribut, **E**-entitet,

min card(A,E) – min. kardinalitet atributa A u entitetu E,

max card(A,E) – max. kardinalitet atributa A u entitetu E.

17

Primjer: Za entitet STUDENT i atribut IME vrijedi:

min card(IME, STUDENT)=1, jer svaki student ima najmanje i obavezno 1 ime.

max card(IME, STUDENT)=1, jer svaki student ima samo jedno ime

Iz ovoga slijedi card(IME,STUDENT)=(1,1)

Primjer: Pogledajmo atribut ZAVRŠENA ŠKOLA u entitetu STUDENT. Ovim atributom prikazujemo naziv srednje škole koju su završili studenti prije upisa na fakultet.

min card(ZAVRŠENA ŠKOLA, STUDENT)=0,

max card(ZAVRŠENA ŠKOLA, STUDENT)=1

Iz ovoga slijedi card(ZAVRŠENA ŠKOLA,STUDENT)=(0,1).

min card(ZAVRŠENA ŠKOLA, STUDENT)=0 kazuje da kod prikaza pojedinog studenta ne moramo obavezno imati podatak o njegovoj završenoj srednjoj školi.

max card(ZAVRŠENA ŠKOLA, STUDENT)=1 znači da je pojedini student završio jednu srednju školu.

18

Viševrijednosni atributi (multi-valued attributes)

To su atributi koji mogu imati više vrijednosti za pojedini element entiteta.

Primjer: Pri opisu studenata potrebno je znati kojim se sportom bave studenti.

Ovaj atribut može imati više različitih vrijednosti, jer se pojedini student može baviti sa više sportova, a neki drugi student se možda uopće ne bavi sportom.

Vrijedi:

min card(SPORT,STUDENT)=0, jer je moguće da se pojedini student ne bavi sportom

max card(SPORT, STUDENT)=n, jer se student može baviti sa više sportova.

Iz ovoga slijedi da je card(SPORT,STUDENT)=(0,n). Atributi koji imaju ovako definirani kardinalitet nazivaju se **viševrijednosni atributi**.

19

Opisni atributi i identifikatori

Postoje dvije vrste atributa: opisne i identifikacijske.

Primjer: Entitet STUDENT i atributi: Ime, Prezime, Mjesto rođenja, Datum rođenja, JMBG, Završena škola, Sport.

Definicija: Identikator (Identifikacijski atribut) je atribut koji jedinstveno određuje pojedine elemente entiteta.

Vrijedi: u entitetu (skupu) ne mogu postojati dva elementa sa istom vrijednošću identifikatora.

U primjeru atribut Ime ne može biti identifikator, jer je realna mogućnost da dva studenta imaju isto ime.

Isto vrijedi i za sve ostale atribute, osim atributa JMBG. Ovaj atribut je jedinstven za svaki element entiteta. Stoga je atribut JMBG identifikator, a ostali atributi su opisni.

Za svaki identifikator vrijedi da je njegov maksimalni kardinalitet jednak 1.

20

450 - Baze podataka *Entiteti*

Tablični prikaz entiteta

Redovi tablice odgovaraju elementima entiteta, a stupci pojedinim atributima.

Primjer: Svaki redak predstavlja jednog studenta.

STUDENT

Ime	Prezime	Dat.rođenja	Mjesto rod.	JMBG	Zavr. škola
Ante	Rožić	11.10. 1980	Osijek	1110980370071	-
Stipe	Anić	03.07. 1980	Split	0307980380025	Sr. teh. šk.
....	....	....	....		

U entitetu mora postojati identifikator koji jednoznačno definira elemente entiteta.

Jedinstveni identifikator za svaki element entiteta naziva se **Primarni ključ (Primary key).**

21

450 - Baze podataka *Entiteti*

Popisani su kardinaliteti atributa u entitetu STUDENT:

card(Ime, STUDENT)=(1,1)

card(Prezime, STUDENT)=(1,1)

card(Datum rođenja, STUDENT)=(1,1)

card(Mjesto rođenja, STUDENT)=(1,1)

card(JMBG, STUDENT)=(0,1)

card(Završena škola, STUDENT)=(0,1)

samo jedan od ovih atributa može biti identifikator, a to je atribut JMBG, jer ne postoje dva studenta sa istim JMBG

Atribut JMBG ima definiran kardinalitet (0,1), tj. student se može evidentirati i bez tog podatka (nije obvezan).

Iz toga proizlazi da JMBG ne može biti primarni ključ ovog entiteta.

Ovaj problem može se riješiti na način da se uvede novi atribut - jedinstveni redni broj studenta.

STUDENT

Student id	Ime	Prezime	Dat.rođenja	Mjesto rod.	JMBG	Zavr. škola
1	Ante	Rožić	11.10. 1980	Osijek	1110980370071	-
2	Stipe	Anić	03.07. 1980	Split	0307980380025	Sr. teh. šk.
3	....	....	....	....		

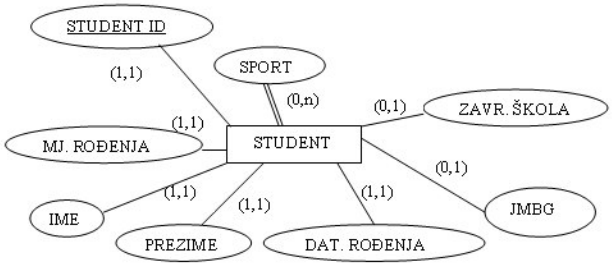
22

450 - Baze podataka *Entiteti*

Dijagram entiteta predstavlja grafički prikaz entiteta i njegovih atributa.

Osnovna pravila grafičkog prikaza entiteta:

- Entitet se prikazuje pravokutnim znakom u koji se upisuje naziv entiteta
- Pojedini atributi se prikazuju ovalnim znakovima i povezani su s entitetom. Unutar oznake atributa upisuje se naziv atributa.
- Viševrijednosni atributi imaju dvostruku poveznicu sa entitetom, koja podrazumijeva kardinalitet (0,n).
- Naziv atributa koji predstavlja primarni ključ se potcrtava.
- Kardinalitet atributa upisuje se na njegovu poveznicu sa entitetom



dijagram entiteta STUDENT

23

450 - Baze podataka *Entiteti*

Složeni atributi

Atribute koji opisuju logički povezana svojstva entiteta, može se povezati u tzv. **složene atribute (composite attributes)**.

Složeni atributi sastavljeni su od atributa koji u procesu ljudskog razmišljanja i modeliranju podataka čine logičku cjelinu.

Primjer: Proširimo entitet STUDENT, kao skup svih studenata, atributom ADRESA, koji opisuje adresu stanovanja studenata.

Uobičajeni način razmišljanja, podrazumijeva da se pod adresom smatraju podaci o ulici (uključujući i broj) i mjestu stanovanja.

Dakle za opis adrese stanovanja, kao složenog atributa, potrebna su dva atributa (svojstva): ULICA i MJESTO.

Svakom od ovih atributa potrebno je definirati odgovarajući kardinalitet u skladu sa definicijom kardinaliteta.

24

450 - Baze podataka *Entiteti*

nastavak primjera:

min card(ULICA, STUDENT)=0, jer ne smatramo nužnim da se zna ulica u kojoj student živi

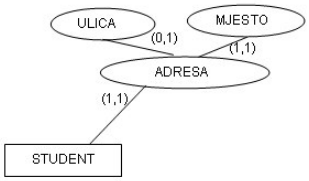
max card(ULICA, STUDENT)=1, jer student pojedinac može stanovati samo u jednoj ulici

min card(MJESTO, STUDENT)=1, jer podatak o mjestu stanovanja smatramo obveznim

max card(MJESTO, STUDENT)=1, jer student može živjeti samo u jednom mjestu

dakle vrijedi card(ULICA, STUDENT)=(0,1) i
card(MJESTO, STUDENT)=(1,1)

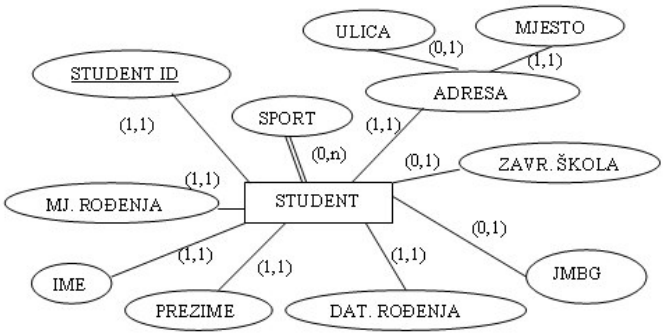
U dijagramu entiteta, složeni atributi se prikazuju na način da se postave između entiteta i atributa koje oni obuhvaćaju.



25

450 - Baze podataka *Entiteti*

Primjer: Dijagram entiteta STUDENT sa prikazom složenog atributa



26

450 - Baze podataka *Entiteti*

Kardinalitet složenog atributa određuje se na temelju kardinaliteta atributa koji su obuhvaćeni složenim atributom, principom maksimuma.

Uz definirani složeni atribut A, koji obuhvaća atribute A1, A2,... An, vrijedi:

min card (A,E) = max (min card(A1,E), min card(A2,E),..., min card (An,E))

max card (A,E) = max (max card(A1,E), max card(A2,E),..., max card (An,E))

U primjeru složeni atribut ADRESA obuhvaća atribute ULICA i MJESTO, pa je njegov kardinalitet

min card (ADRESA,STUDENT) = max(0, 1)=1

max card (ADRESA,STUDENT) = max(1, 1)=1

card (ADRESA,STUDENT)=(1, 1)

27

450 - Baze podataka *Entiteti*

Složeni atributi bitni su kod razrade modela baze podataka, budući su oni uobičajeni međukorak za definiranje svih potrebnih atributa pojedinog entiteta.

Složeni atributi također mogu biti identifikatori, pod uvjetom da svi atributi koji čine složeni atribut imaju definiran kardinalitet (1,1), te da je složeni atribut jedinstven za svaki element entiteta.

U gornjem primjeru složeni atribut ADRESA ne može biti identifikator u entitetu STUDENT, budući on obuhvaća atribut ULICA za koji vrijedi card(ULICA, STUDENT)=(0,1)

28

Relacije

U sustavu baze podataka skupovi – entiteti su međusobno povezani logičkim vezama tj. relacijama.

Primjer: STUDENT i UPISNI LIST opisuju proces studiranja svakog studenta.

Prilikom upisa u pojedini semestar svaki student predaje upisni list (dokument o upisu semestra).

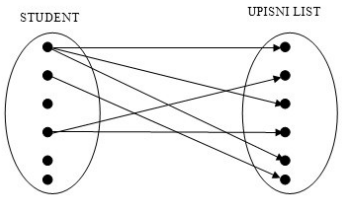
Atributi entiteta UPISNI LIST su npr. SEMESTAR, ŠK.GODINA, OBRAZOVNI PROGRAM.

UPISNI LIST je povezan sa entitetom STUDENT, budući svaki upisni list pripada nekom studentu.

Između ova dva entiteta postoji logička veza – relacija (eng. *Relationship*).

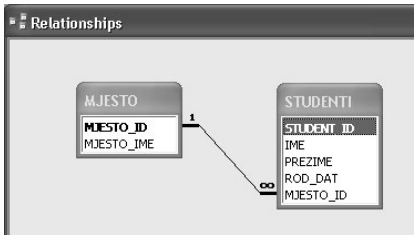
29

Grafički prikaz relacije između entiteta STUDENT i UPISNI LIST:

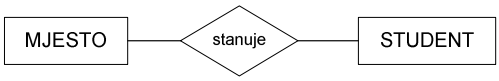


Prikaz relacije
jedan prema više

Primjer sa vježbi:



Veza između skupova tj. entiteta ostvaruje se vezom između pojedinih elemenata tih skupova



30

450 - Baze podataka *Relacije*

***Definicija*: *Kardinalitet entiteta* u relaciji definiran je brojem veza pojedinog elementa tog entiteta sa elementima entiteta s kojim je relacijski povezan.**

Primjer: Relacijska veza entiteta STUDENT i UPISNI LIST. Jedan student-pojedinac može imati više upisnih listova (za svaki semestar po jedan), ali ne mora nužno imati upisni list. Sa druge strane, svaki upisni list mora pripadati nekom studentu, i to samo jednom određenom.

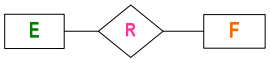
Vrijedi

min card (STUDENT, PREDAJE)=0	min card (UPISNI LIST, PREDAJE)=1
max card (STUDENT, PREDAJE)=n	max card (UPISNI LIST, PREDAJE)=1
odnosno	
card (STUDENT, PREDAJE)=(0,n)	card (UPISNI LIST, PREDAJE)=(1,1)

Ovakav tip relacije je relacija **1×n** odnosno **jedan prema više** (eng. *one-to-many*).

31

450 - Baze podataka *Relacije*



Definicija:

Ako dva entiteta E i F ostvaruju relaciju R, pri čemu vrijedi:

min card (E,R)=0	min card (F,R)=1
max card (E,R)=n	max card (F,R)=1
card(E,R)=(0,n)	card(F,R)=(1,1)

takva relacija je tipa 1×n (jedan prema više).

Napomena: Pri određivanju tipa relacije, ključan je maksimalni kardinalitet entiteta u relaciji.

To znači da su entiteti E i F povezani relacijom R sa definiranim kardinalitetima, card(E,R)=(0,n) i card(F,R)=(0,1) također u vezi jedan prema više.

32

450 - Baze podataka *Relacije*

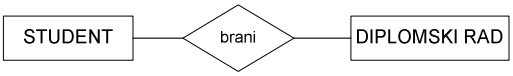
Relacija **1×1** odnosno **jedan-na-jedan** (eng. *one-to-one*)

***Definicija**:*

Za dva entiteta E i F koji sudjeluju u relaciji R jedan-na-jedan vrijedi:

min card (E,R)=0	min card (F,R)=0
max card (E,R)=1	max card (F,R)=1

Svaki element skupa E može biti povezan samo sa jednim elementom skupa F, ali i ne mora. Isto vrijedi za elemente skupa F.



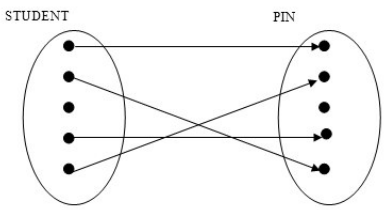
33

450 - Baze podataka *Relacije*

Primjer:
Studenti imaju mogućnost prijavljivanja ispita elektronskim putem. Za pristup sustavu za prijavljivanje ispita svaki student mora dobiti PIN (osobni identifikator sa šifrom). Pri tome svaki student može dobiti samo jedan PIN, ali je realna mogućnost da student ne dobije PIN odmah po upisu na fakultet. To znači da trenutno mogu postojati studenti kojima nije dodijeljen PIN. U isto vrijeme, svaki PIN može biti dodijeljen samo jednom studentu, ali mogu postojati i već pripremljeni PIN-ovi koji još nisu raspoređeni studentima.

Za relaciju PRIJAVA koja povezuje entitete STUDENT i PIN vrijedi

card(STUDENT,PRIJAVA)=(0,1) i
card(PIN,PRIJAVA)=(0,1),
dakle radi se o relaciji 1×1.



34

450 - Baze podataka *Relacije*

Relacija **n×n** odnosno **više-na-više** (eng. *many-to-many*)

Definicija:

Za dva entiteta E i F koji sudjeluju u relaciji R više-na-više vrijedi:

min card (E,R)=0	min card (F,R)=0
max card (E,R)=n	max card (F,R)=n

Svaki element skupa E može biti povezan sa više elemenata skupa F, ali i ne mora. Isto vrijedi za elemente skupa F.

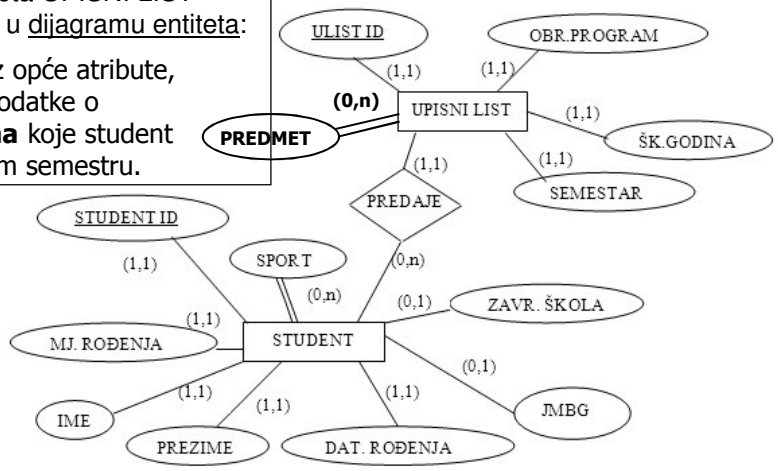
35

450 - Baze podataka *Relacije*

Primjer: U tijeku procesa studiranja svaki student predaje UPISNI LIST kojim potvrđuje upis pojedinog semestra, a koji je vezan za studenta. Eniteti STUDENT i UPISNI LIST povezani su relacijom jedan prema više, kako je prethodno opisano.

Atributi entiteta UPISNI LIST prikazani su u dijagramu entiteta:

Upisni list uz opće atribute, sadržava i podatke o **predmetima** koje student upisuje u tom semestru.



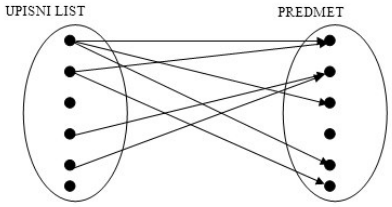
450 - Baze podataka *Relacije*

Primjer *(nastavak)*:
Da bi se definirao proces upisivanja predmeta potrebno je definirati novi entitet PREDMET, kao skup svih predmeta koji se predaju.
Entitet predmet definiran je atributima: PREDMET_ID, NAZIV PREDMETA, SATI PREDAVANJA, SATI VJEŽBE itd.

PREDMET

Predmet id	Ime predmeta	Sati pred	Sati AV	Sati LAB
100	Matematika	3	2	0
101	Fizika	3	1	0
....	....	....	....	

Odnos između entiteta UPISNI LIST i PREDMET:
Svaki student na jednom upisnom listu (u jednom semestru) upisuje više predmeta, ali u nekim slučajevima ne mora upisati niti jedan (npr. ponavljanje godine).
Sa druge strane, svaki predmet može upisati više studenata, ali se može dogoditi da neki od ponuđenih predmeta (izbornih) ne upiše niti jedan student.



37

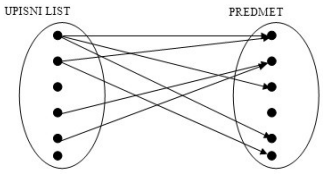
450 - Baze podataka *Relacije*

Za relaciju UPISUJE koja definira odnos između entiteta UPISNI LIST i PREDMET, vrijedi:

min card (UPISNI LIST,UPISUJE)=0
max card (UPISNI LIST,UPISUJE)=n

min card (PREDMET,UPISUJE)=0
max card (PREDMET,UPISUJE)=n

pa se radi o relaciji ***više-na-više (n×n).***

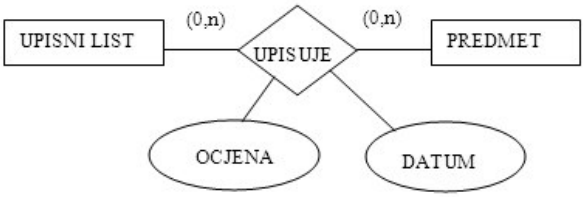


38

450 - Baze podataka *Relacije*

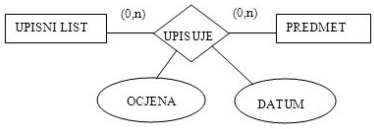
Za relacije više-na-više svojstveno je da ovakav tip relacije, osim povezivanja entiteta, sa sobom može donijeti određene **atribute** koji su posljedica relacijske veze.

U našem primjeru, kao posljedica činjenice da pojedini student u danom semestru upiše neki predmet, mogu se javiti atributi OCJENA i DATUM (datum polaganja).
Ovi atributi nisu dio niti jednog od entiteta koji sudjeluju u relaciji, već su posljedica relacijske veze, u ovom slučaju entiteta UPISNI LIST i PREDMET.



39

450 - Baze podataka *Relacije*



Kada se grafički prikaz entiteta i pripadajućih atributa (dijagram entiteta) proširi sa prikazom relacija dobija se **E-R dijagram**
(dijagram ENTITET-RELACIJA od eng. *Entity-Relationship* ili E-V dijagram od hrvatskog ENTITET-VEZA).

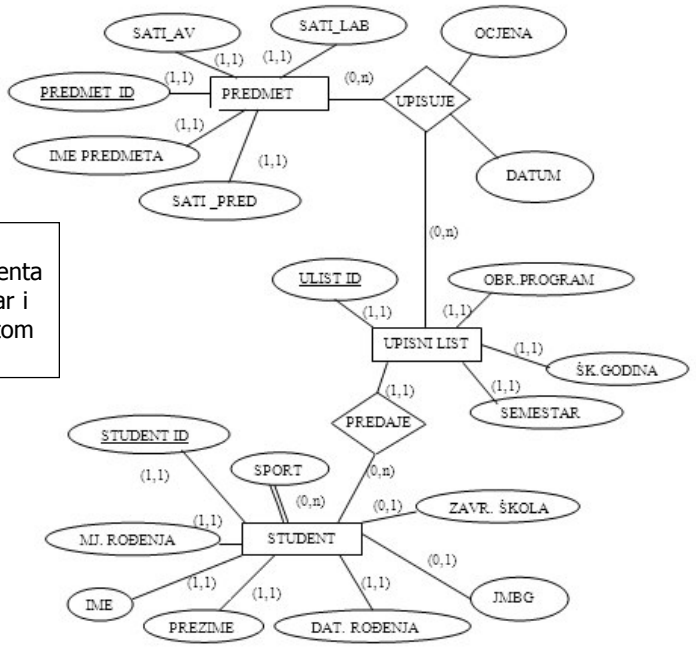
Osnovni princip prikaza relacija u dijagramu vidljiv je već iz gornjeg primjera:

- Relacija se simbolički ucrtava deltoidnim znakom (◊) između entiteta koji sudjeluju u relaciji, te se povezuje s njima.
- Na poveznicu entiteta i relacije upisuje se kardinalitet entiteta u relaciji u obliku uređenog para.
- Atributi koji su posljedica relacije ucrtavaju se ovalnim simbolom kao i atributi entiteta, te se povezuju sa relacijom iz koje proizlaze.

40

450 - Baze

E-R dijagram za proces upisa studenta u naredni semestar i predmeta koje u tom semestru upisuje



41

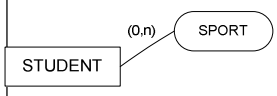
450 - Baze podataka — *Relacije*

Zadatak:

Jedan od atributa entiteta STUDENT je atribut SPORT. Prepoznat je kardinalitet atributa SPORT card(STUDENT, SPORT) = (0,n).

Kako podaci o sportu predstavljaju poseban entitet, zadatak je napraviti E-R dijagram za ovaj problem.

U zadatku treba definirati vezu između entiteta STUDENT i SPORT, te osim entiteta i veze u E-R dijagramu prikazati i pripadajuće atribute.



42

RELACIJSKI MODEL PODATAKA

E-R dijagram je osnova i uvod za prikaz podataka u relacijskom modelu.

Iz izvedbe E-R dijagrama vrši se rekonstrukcija relacijskog modela baze podataka.

Taj model podataka je vremenom usavršavan i danas je de facto najraširenij, općeprihvaćeni model podataka.

Najveća prednost relacijskog modela podataka jest u tome da on počiva na matematičkoj teoriji relacijske algebre.

Transformacijska pravila

Transformacijska pravila omogućavaju razvoj relacijskog modela baze podataka temeljem E-R dijagrama

1. pravilo transformacije

Entitet kao skup objekata prikazuje se tablicom.

Svaki redak u tablici odgovara jednom elementu entiteta, a svaki stupac je odgovarajući atribut. Ime tablice jednako je nazivu entiteta.

STUDENT

Student id	Ime	Prezime	Dat.rođenja	Mj. rođenja	JMBG	Zavr. škola
1	Ante	Rožić	11.10. 1980	Osijek	1110980370071	
2	Stipe	Anić	03.07. 1980	Split	0307980380025	Sr. teh. šk.
3	....	....	....	....	....	....

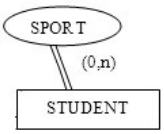
Simbolički prikaz **tablice** tj. njezina definicija izražava se u obliku
head(IME TABLICE) = {atribut1, atribut2, ...},

a **primarni ključ** **PK(IME TABLICE) = naziv kolone**

U gornjem primjeru tablice STUDENT vrijedi
head(STUDENT) = {Student_id, Ime,Prezime, Dat.rođenja, Mj.rođenja, JMBG, Zavr.škola}
PK(STUDENT) = Student_id

450 - Baze podataka *Relacijski model podataka*

2. pravilo transformacije
Viševrijednosni atributi ne prikazuju se preko jedne kolone, kao obični atributi, jer unos više vrijednosti u jedno polje nije dozvoljen. Umjesto toga **viševrijednosni atribut prikazuje se posebnom tablicom.**



Nova tablica sastoji se od dva stupca:
-**primarnog ključa entiteta i**
-**viševrijednosnog atributa.**

Primarni ključ nove tablice, složen je od obje vrijednosti.

Za viševrijednosni atribut S u entitetu E, vrijedi prikaz preko nove tablice S uz head(S)={E_id, S} i
PK(S)={E_id,S},
gdje je E_id=PK(E).

45

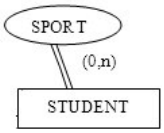
450 - Baze podataka *Relacijski model podataka*

Primjer:

Prethodno je opisan entitet STUDENT i viševrijednosni atribut Sport, kojim se opisuje bavljenje studenata sportom.

Prema tranformacijskom pravilu ovaj atribut prikazuje se posebnom tablicom, koja sadržava primarni ključ entiteta STUDENT (Student_id) i viševrijednosni atribut (Sport).

Složeni primarni ključ **nove tablice** čine oba atributa tj. PK(SPORT)={Student_id, Sport}.



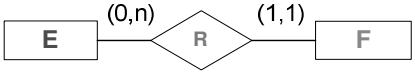
SPORT

Student_id	Sport
1	odbojka
2	nogomet
2	košarka

Objašnjenje: studenti iz relacije na slajdu 2 su Ante (koji ima PK=1) i Stipe (PK=2). Ante se bavi jednim sportom (odbojka) a Stipe se bavi dvama sportovima (nogomet i košarka).

46

450 - Baze podataka *Relacijski model podataka*



3. pravilo transformacije

Prikaz relacije jedan-na-više (*one-to-many*)

Ako su entiteti E i F u relaciji R tipa jedan-na-više i card(E,R)=(0,n) , card(F,R)=(1,1) , relacija se ostvaruje preko stranog ključa.

Pripadajuća tablica entiteta F proširuje se kolonom E_id, gdje je E_id = PK(E), a ta kolona u tablici F naziva se strani ključ (*foreign key* – FK).

Primjer:
Promotrimo već opisane entitete STUDENT i UPISNI LIST.
Ova dva entiteta povezani su relacijom PREDAJE i vrijedi
card (STUDENT, PREDAJE)=(0,n)
card (UPISNI LIST, PREDAJE)=(1,1)

47

450 - Baze podataka *Relacijski model podataka*

Relacija *jedan-na-više* ostvaruje se na način da se, prema transformacijskom pravilu, tablici UPISNI LIST dodaje **nova kolona (atribut)**, kojom se označava povezanost (pripadnost) pojedinog upisnog lista studentu.

Budući da je svaki student jednoznačno određen svojim primarnim ključem, nova kolona sadržava podatke o primarnom ključu studenta, te predstavlja **strani ključ u tablici UPISNI LIST**.

STUDENT

Student id	Ime	Prezime	Dat.rođenja	Mj. rođenja	JMBG	Zavr. škola
1	Ante	Rožić	11.10. 1980	Osijek	1110980370071	
2	Stipe	Anić	03.07. 1980	Split	0307980380025	Sr. teh. šk.
3	....	....	....	....	....	....

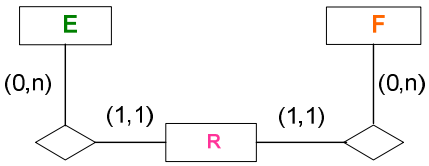
UPISNI LIST

Ulist id	Šk.godina	Semestar	Obr.program	Student_id
1	1999/00	2	Elektronika	2
2	1999/00	4	Strojarstvo	1
3	....	....	....	....

450 - Baze podataka — *Relacijski model podataka*

4. pravilo transformacije
Prikaz relacije više-na-više (*many-to-many*)

Ako su entiteti E i F u relaciji R tipa više-na-više i
card(E,R)=(0,n) card(F,R)=(0,n)
relacija se ostvaruje formiranjem **nove** tzv. relacijske tablice R, te vrijedi



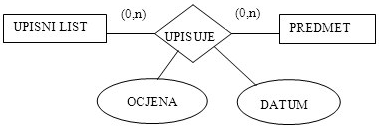
E_id=PK(E)=FK(R)
F_id=PK(F)=FK(R)
PK(R)=(E_id,F_id).

Nova relacijska tablica, uz strane ključeve entiteta koji sudjeluju u relaciji (a koji zajedno čine **primarni ključ nove tablice**), proširuje se atributima koji su posljedica relacije među entitetima.

49

450 - Baze podataka — *Relacijski model podataka*

Primjer:
U modelu baze podataka koja opisuje studiranje, entiteti UPISNI LIST i PREDMET nalaze se u relacijskoj vezi više-na-više



UPISNI LIST

Ulist id	Šk.godina	Semestar	Obr.program	Student_id
1	1999/00	2	Elektronika	2
2	1999/00	4	Strojarstvo	1
3	....	....	....	....

PREDMET

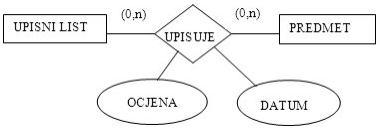
Predmet_id	Ime predmeta	Sati pred	Sati AV	Sati LAB
100	Matematika	3	2	0
101	Fizika	3	1	0
....	....	....	....	....

card (UPISNI LIST,UPISUJE)=(0,n)
card (PREDMET,UPISUJE)=(0,n)

50

450 - Baze podataka *Relacijski model podataka*

Primjer *(nastavak)*:
U skladu sa pravilom prikaza relacije više-na-više, ovakva relacija bit će prikazana novom relacijskom tablicom UPISUJE.



Vrijedi:

PK(UPISNI LIST)=FK(UPISUJE)=Ulist_id
PK(PREDMET)=FK(UPISUJE)=Predmet_id
PK(UPISUJE)=(Ulist_id,Predmet_id)

Tablica UPISUJE uz strane ključeve tablica koje sudjeluju u relaciji, a koji čine primarni ključ nove tablice,(Ulist_id,Predmet_id), sadržava i atribute koji su posljedica relacijske veze (Ocjena,Datum).

UPISUJE

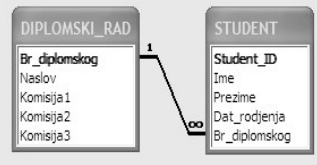
Ulist id	Predmet id	Ocjena	Datum
1	100	4	11.10.2000
2	101	-	-
3	....	....	

51

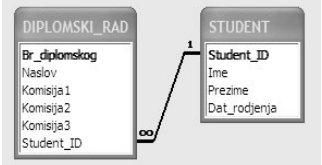
450 - Baze podataka *Relacijski model podataka*

5. pravilo transformacije
Prikaz relacije jedan-na-jedan (*one-to-one*)
Ako su entiteti E i F u relaciji R tipa jedan-na-jedan i card(E,R)=(0,1), card(F,R)=(0,1), E_id=PK(E), F_id=PK(F), relacija se može ostvariti na tri načina:

E_id=PK(E)=FK(F), stranim ključem u tablici F



F_id=PK(F)=FK(E), stranim ključem u tablici E



proširenjem tablice(entiteta) E sa kolonama (svojstvima) tablice F

52

450 - Baze podataka *Relacijski model podataka*

RELACIJSKA PRAVILA

Definiraju način prikaza i pristupa podacima

Pravilo normalnog oblika: zabranjuje prikaz viševrijednosnih atributa u jednoj tablici - identično transformacijskom pravilu broj 2.

Pravilo pristupa podacima - definira da se podacima u tablici može pristupati isključivo preko vrijednosti atributa (kolona).

Pravilo jedinstvenosti reda u tablici (elementa u entitetu): u tablici ne postoje dva potpuno ista reda, sa jednakim vrijednostima svih atributa.

53

450 - Baze podataka *Relacijski model podataka*

PRIMARNI KLJUČ

Primarni ključ je atribut ili skup atributa koji jedinstveno identificiraju svaki element entiteta (redak u tablici). Primarni ključ mora zadovoljavati tri osnovna uvjeta:

Jedinstvenost
U tablici ne mogu postojati dva redka s istom vrijednošću primarnog ključa.

Student_ID	Prezime	Ime	Datum_rodj
1	Matković	Mate	12.9.1985
2	Bulić	Majana	2.11.1984
3	Jurić	Mate	12.9.1885
4	Jurić	Ana	27.3.1987

Minimalnost
Ako je primarni ključ složen tj. sastoji se od više atributa, tada se niti jedna njegova komponenta ne može ukloniti a da se ne naruši pravilo jedinstvenosti.

FAX	Student_MB	Prezime	Ime	Datum_rodj
FESB	2003-025	Matković	Mate	12.9.1985
FESB		Bulić	Majana	2.11.1984
FER	AB2-2/2003	Jurić	Mate	12.9.1885
	JK-043-01	Jurić	Ana	27.3.1987

54

450 - Baze podataka *Relacijski model podataka*

Pravilo integriteta primarnog ključa

Niti jedna komponenta primarnog ključa **ne smije imati null** vrijednost tj. atribut koji je sastavni dio primarnog ključa mora imati definiran kardinalitet (1,1).

Pravilo integriteta posljedica je pravila minimalnosti. Kada bi bilo dopušteno da neki atribut kao dio primarnog ključa poprimi null vrijednost, to bi značilo da se elementi entiteta mogu identificirati i bez poznavanja vrijednosti tog atributa. Iz toga proizlazi da takav atribut uopće ne treba biti dio primarnog ključa, tj. narušilo bi se pravilo minimalnosti.

FAX	Student_MB	Prezime	Ime	Datum_rodj
FESB	2003-025	Matković	Mate	12.9.1985
FESB		Bukić	Majana	2.11.1984
FER	AB2-2/2003	Jurić	Mate	12.9.1885
	JK-043-01	Jurić	Ana	27.3.1987

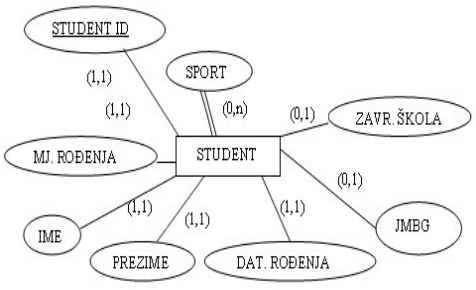
55

450 - Baze podataka *Dekompozicija atributa*

DEKOMPOZICIJA ATIBUTA

Entitet STUDENT definiran je sa pripadajućim atributima koji opisuju neko od svojstava zajedničko svim elementima skupa studenata.

Atribut MJESTO ROĐENJA za svakog studenta daje podatak gdje je rođen. Pod tim atributom podrazumijeva se naziv mjesta u kojem je student rođen.



Realna je mogućnost da postoji više različitih mjesta, koja imaju isti naziv. Stoga bi se, za studente koji su rođeni u dva potpuno različita mjesta pojavila ista vrijednost atributa MJESTO ROĐENJA.

Očito je da opis ovog atributa nije ispravno postavljen, tj. za opis jednog mjesta nije dovoljan samo naziv (ime) mjesta.

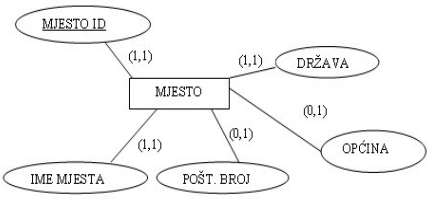
MJESTO_ID	MJESTO_IME	POST_BR
8469	Starigrad	53286
8470	Starigrad	48000
865	Starigrad	23244

MJESTO_ID	MJESTO_IME	POST_BR
8655	Dubrava	21252
8654	Dubrava	10342
8656	Dubrava	20248
3370	Dubrava kod Šibenika	22000
3371	Dubrava kod Tisnoga	22240
3372	Dubrava Križovljanska	42208
3373	Dubrava Pušćanska	10294
3374	Dubrava Samoborska	10430
3375	Dubrava Zabočka	49210

450 - Baze podataka *Dekompozicija atributa*

Rješenje problema naziva mjesta je u redefiniciji atributa MJESTO kao posebnog entiteta.

Taj entitet, kao i svaki drugi predstavlja skup elemenata (mjesta) sa zajedničkim svojstvima



Pri definiciji novog entiteta, za njegov opis uzimamo sve bitne atribute koji pobliže definiraju svojstva pojedinog mjesta. U skladu sa 1. transformacijskim pravilom novi entitet bi se u relacijskom modelu baze prikazao tablicom:

Posebno mjesto zauzima atribut općina, kojim se opisuje pripadnost mjesta pojedinoj općini.

MJESTO

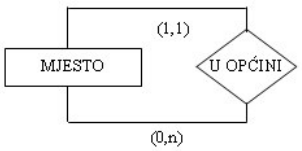
MJESTO_ID	IME MJESTA	POŠT. BROJ	OPĆINA	DRŽAVA
1	Split	21000	Split	Hrvatska
2	Smokvica	23249	Pag	Hrvatska
3	Pag	23250	Pag	Hrvatska
4	Osijek	31000	Osijek	Hrvatska
5	Bonn	-	-	Njemačka

U praksi je općina opisana mjestom koje predstavlja općinsko središte, tj. pojam općine moguće je povezati sa pojmom mjesta.
To je logička veza gdje je jedan entitet povezan sa samim sobom.

57

450 - Baze podataka *Dekompozicija atributa*

E-R dijagram opisuje logičku vezu entiteta sa samim sobom.

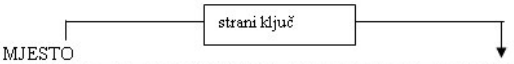


MJESTO

MJESTO_ID	IME MJESTA	POŠT. BROJ	OPĆINA	DRŽAVA
1	Split	21000	Split	Hrvatska
2	Smokvica	23249	Pag	Hrvatska
3	Pag	23250	Pag	Hrvatska
4	Osijek	31000	Osijek	Hrvatska
5	Bonn	-	-	Njemačka

Ovakva struktura u odnosu entiteta naziva se **ring struktura** (prsten), gdje je jedan entitet povezan sa samim sobom.

Prema pravilima transformacije, budući se radi o obliku relacije **jedan-prema-više**, ona se ostvaruje preko **stranog ključa**.



MJESTO

MJESTO_ID	IME MJESTA	POŠT. BROJ	OPĆINA_ID	DRŽAVA
1	Split	21000	1	Hrvatska
2	Smokvica	23249	3	Hrvatska
3	Pag	23250	3	Hrvatska
4	…	…	…	…

58

INTEGRITET PODATAKA

Integritetom podataka osigurava se njihova suvislost i postiže se da podaci odgovaraju točno zadanim pravilima i formatima u okviru baze podataka.

Svaka je baza podataka više ili manje vjerna slika svijeta koji nas okružuje, tj. onog njegovog dijela o kojem želimo obrađivati podatke.

Baza podataka sastoji od podataka koji su međusobno povezani na različite načine i njihove vrijednosti predstavljaju dio realnog okružja.

Potrebno je definirati određena pravila kojima je zadatak postavljanje ograničenja:

A) na pojavljivanje vrijednosti pojedinih atributa,
B) na njihovo međusobno povezivanje.

Ta se pravila nazivaju pravila **integriteta (integrity rules)** i od ogromnog su značenja za ispravno funkcioniranje sustava i zaštitu informacija.

59

A) INTEGRITET ENTITETA – DOMENA PODATAKA

To su pravila tj. ograničenja na pojavljivanje vrijednosti pojedinih atributa.

Domena podataka predstavlja skup vrijednosti koje određeni atribut može poprimiti.

Pojedinačna vrijednost atributa se smatra najmanjom nedjeljivom semantičkom jedinicom podataka.

Domena se definira za svaki atribut i predstavlja podatke koji pripadaju istom tipu podataka.

Miješanje više tipova podataka unutar jedne domene nije dopušteno.

60

450 - Baze podataka *Integritet podataka*

Primjer:

Promatrajmo entitet STUDENT i njegov atribut DATUM ROĐENJA. Ovaj atribut daje podatak o tome kada je rođen student.

Radi se o podatku koji mora udovoljavati standardnom tipu podataka koji prikazuje datum u uobičajenom formatu **dan.mjesec.godina**.

Logično je da se za ovaj atribut veže tip podataka **datuma**, koji osigurava prikaz podataka u željenom formatu.

Međutim definiranje tipa podataka za pojedini atribut najčešće nije dovoljno da bi se osigurala suvislost podataka.

Pretpostavimo da u tablicu STUDENT želimo unijeti podatak o novom studentu, za kojeg upisujemo datum rođenja 11.05.2006.
Ovaj podatak je sa stajališta formata podataka potpuno ispravan, ali nije logičan, budući je student svakako morao biti rođen prije.

Stoga je uz svaki atribut osim tipa podataka potrebno definirati i ograničenja, koja osiguravaju realnost podataka.

61

450 - Baze podataka *Integritet podataka*

B) REFERENCIJALNI INTEGRITET

To su pravila tj. ograničenja na međusobno povezivanje podataka.

Osigurava logičnu vezu i pravila odnosa među podacima u tablicama koje su relacijski povezane.
U tablici ne može postojati vrijednost stranog ključa za koju ne postoji ista vrijednost primarnog ključa u osnovnoj tablici.

STUDENT

Student id	Ime	Prezime	Dat.rođenja	Mjesto rođ.	JMBG	Zavr. škola
1	Ante	Rožić	11.10. 1980	Osijek	1110980370071	-
2	Stipe	Anić	03.07. 1980	Split	0307980380025	Sr. teh. šk.
3	....	....	....	....		

FK

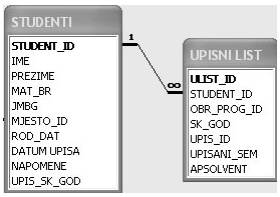
UPISNI LIST

Ulist id	Student_id	Sem	Školska god.	Obr.prog
1	2	1	1999/00	elektronika
2	2	2	1999/00	elektronika
3	....	....	....	....

62

450 - Baze podataka *Integritet podataka*

Između tablica STUDENT i UPISNI LIST postoji relacija one-to-many koja je ostvarena stranim ključem u tablici UPISNI LIST (Student_id).



Referencijalni integritet definira pravila unosa, brisanja i promjene (ažuriranja) podataka, kako bi se osigurala konzistentnost podataka u bazi.

Na dizajneru baze podataka je da definira ponašanje baze prilikom pojedinih promjena. Pri tome postoje dvije mogućnosti ponašanja sustava:

- Tražena akcija se odbija i u bazi se na događaju nikakve promjene.
- Akcija se dopušta, ali se uz nju pokreću još neke akcije koje imaju svrhu uspostaviti konzistentnost baze podataka.

63

450 - Baze podataka *Integritet podataka*

Unos podataka

Zabranjen je unos podatka u tablicu, sa nekom vrijednost stranog ključa, ako u osnovnoj tablici ne postoji ista vrijednost primarnog ključa. (***Restricted***).

Brisanje podataka

Ograničeno (***Restricted***) - Brisanje reda sa određenom vrijednošću primarnog ključa dozvoljeno je samo ako se ta vrijednost ne pojavljuje u drugoj tablici kao strani ključ.

Primjer: Ne može se ukloniti pojedinog studenta iz tablice STUDENT, ako u tablici upisni list postoje njegovi upisni listovi.

Stupnjevano – Kaskadno (***Cascade***) - Brisanje podatka sa određenom vrijednosti primarnog ključa izaziva brisanje svih podataka u drugoj tablici gdje se ta vrijednost primarnog ključa pojavljuje kao strani ključ.

Primjer: Brisanjem pojedinog studenta iz tablice STUDENT, automatski se brišu svi njegovi upisni listovi iz tablice UPISNI LIST.

64

450 - Baze podataka *Integritet podataka*

Brisanje podataka *- nastavak*

Nuliranje (***Nullifies***) – Brisanjem određene vrijednosti primarnog ključa, najprije se sve iste vrijednosti stranog ključa postavljaju na null vrijednost, a onda se iz osnovne tablice briše ta vrijednost primarnog ključa.

Primjer: Brisanjem pojedinog studenta iz tablice STUDENT, najprije se za sve upisne listove koji pripadaju tome studentu, atribut student_id, koji je u tablici UPISNI LIST strani ključ postavlja na null vrijednost, a potom se uklanja željeni podatak iz tablice STUDENT.

Ima li to smisla?

65

450 - Baze podataka *Integritet podataka*

Ažuriranje podataka

Ograničeno (***Restricted***) - Ažuriranje vrijednosti primarnog ključa dozvoljeno je samo ako se ta vrijednost ne pojavljuje u drugoj tablici kao strani ključ.

Stupnjevano – Kaskadno (***Cascade***) - Ažuriranje vrijednosti primarnog ključa izaziva ažuriranje svih podataka u drugoj tablici gdje se ta vrijednost primarnog ključa pojavljuje kao strani ključ.

Nuliranje (***Nullifies***) – Ažuriranjem određene vrijednosti primarnog ključa, najprije se sve iste vrijednosti stranog ključa postavljaju na null vrijednost, a onda se u osnovnoj tablici mijenja ta vrijednost primarnog ključa.

66

450 - Baze podataka *Relacijska algebra*

Relacijska algebra podrazumijeva definirane **operacije** nad entitetima (tablicama) i podacima koji im pripadaju.

Operacije teorije skupova (*Set-theory operations*)
Unija (T:=R ⋃ S) Presjek (T:=R ⋂ S)
Razlika (T:=R – S) Produkt (T:=R × S)

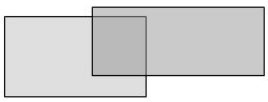
Prirodne relacijske operacije (*Native-relation operations*)
Projekcija (T:=R [a]) Selekcija (T:=R where a=12)
Join (T:=R⋈ S) Dijeljenje (T:=R÷S)

Definicija
Kompatibilnim tablicama smatramo tablice koje
- Sadrže atribute jednakog naziva i isti broj atributa (isti broj kolona)
- Atributi istog naziva definirani su nad jednakim domenama.

67

450 - Baze podataka *Relacijska algebra*

UNIJA (T:=R ⋃ S)



Operacija unije može se provoditi samo nad kompatibilnim tablicama.

Po matematičkoj teoriji unija dvaju skupova R i S je skup T koji se sastoji od svih elemenata koji pripadaju bilo kojem od skupova R ili S.

Primijenjeno na tablice, tablica T je unija tablica R i S, koja ima isto zaglavlje (header) kao i tablice R i S, a sadrži sve redove koji se nalaze bilo u tablici R ili S.

R

A	B	C
s	1	4
d	1	5
c	2	5

S

A	B	C
d	1	5
c	2	4
f	4	4

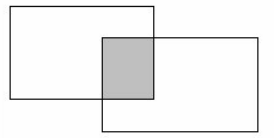
T:=R ⋃ S

A	B	C
s	1	4
d	1	5
c	2	5
c	2	4
f	4	4

68

450 - Baze podataka *Relacijska algebra*

PRESJEK (T:=R ⋂ S)



Operacija presjeka može se provoditi samo nad kompatibilnim tablicama.

Po matematičkoj teoriji presjek dvaju skupova R i S je skup T koji se sastoji od svih elemenata koji pripadaju i skupu R i skupu S.

Primijenjeno na tablice, tablica T je presjek tablica R i S, koja ima isto zaglavlje (header) kao i tablice R i S, a sadrži sve redove koji se nalaze u oba dvije tablice.

R

A	B	C
s	1	4
d	1	5
c	2	5

S

A	B	C
d	1	5
c	2	4
f	4	4

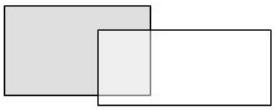
T:=R ⋂ S

A	B	C
d	1	5

69

450 - Baze podataka *Relacijska algebra*

RAZLIKA (T:=R - S)



Operacija razlike može se provoditi samo nad kompatibilnim tablicama.

Po matematičkoj teoriji razlika dvaju skupova R i S je skup T koji se sastoji od svih elemenata koji pripadaju skupu R i ne pripadaju skupu S.

Primijenjeno na tablice, tablica T je razlika tablica R i S, koja ima isto zaglavlje (header) kao i tablice R i S, a sadrži sve redove koji se nalaze u R, a ne nalaze se u S.

R

A	B	C
s	1	4
d	1	5
c	2	5

S

A	B	C
d	1	5
c	2	4
f	4	4

T:=R- S

A	B	C
s	1	4
c	2	5

70

450 - Baze podataka *Relacijska algebra*

PRODUKT (T:=R × S)

Operacija produkta nad entitetima (tablicama), temelji se na skupovnoj operaciji Kartezijevog produkta.

Po matematičkoj teoriji Kartezijev produkt dvaju skupova R i S je skup T koji se sastoji od uređenih parova, pri čemu je prvi element uređenog para iz skupa R, a drugi iz skupa S.

Primijenjeno na tablice, tablica T je produkt tablica R i S, čije je zaglavlje (header) definirano:
head(R × S)=head(R) ⋃ head(S), a elementi (redovi) te tablice nastaju spajanjem redova iz tablice R i redova iz tablice S.

R

A	B	C
s	1	4
d	1	5
c	2	5

S

D	E	F
d	1	5
c	2	4

T:=R× S

A	B	C	D	E	F
s	1	4	d	1	5
s	1	4	c	2	4
d	1	5	d	1	5
d	1	5	c	2	4
c	2	5	d	1	5
c	2	5	c	2	4

71

450 - Baze podataka *Relacijska algebra*

PRODUKT - nastavak

Općenito za produkt tablica vrijedi:
Ako je **R (m × n)** m redova i n kolona
S (k × l) k redova i l kolona,
onda je
T:=R × S tablica sa (m+k) redova i (n+l) kolona

Za operatore unije, presjeka i produkta vrijede pravila asocijativnosti i komutativnosti:

(R ⋃ S) ⋃ T= R ⋃ (S ⋃ T)= R ⋃ S ⋃ T
(R ⋂ S) ⋂ T= R ⋂ (S ⋂ T)= R ⋂ S ⋂ T
(R × S) × T= R × (S × T)= R × S × T

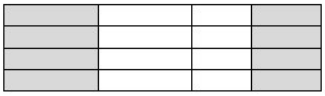
Međuzavisnost operacija:

R ⋂ S = R - (R - S) = S - (S - R)
R – S = R – (R ⋂ S)

72

450 - Baze podataka *Relacijska algebra*

PROJEKCIJA T:=R[a]



Operacijom projekcije tablice nad atributima izvajaju se atributi tablice na kojima se vrši projekcija.

Projekcija tablice R nad atributima X,Y,Z jest tablica T sa zaglavljem head(T)= {X,Y,Z} koja sadržava sve redove koji su sadržani u tablici R.

Kao rezultat operacije projekcije dobija se nova tablica koja predstavlja vertikalni podskup zadane tablice.

R

A	B	C
s	1	4
d	1	5
c	2	5

T:=R[A]

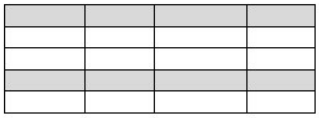
A
s
d
c

Operacijom projekcije nad zadanom tablicom R dobija se rezultat koji sadrži isti broj redova kao i tablica R, ali samo one atribute (kolone) po kojima se vrši projekcija.

73

450 - Baze podataka *Relacijska algebra*

SELEKCIJA - IZDVAJANJE
(T:=R where a=12)



Operacijom izdvajanja (selekcije) nad zadanom tablicom R izdvaja se skup redova koji zadovoljavaju uvjet po kojem se selekcija vrši.

Tablica koja se dobija kao rezultat operacije selekcije sadrži sve atribute (kolone) kao i izvorna tablica, ali samo one redove koji zadovoljavaju traženi uvjet.
Dobijena tablica predstavlja horizontalni podskup izvorne tablice.

R

A	B	C
s	1	4
d	1	5
s	4	6
c	2	5

R where A=s

A	B	C
s	1	4
s	4	6

74

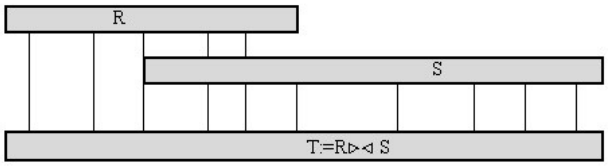
SPAJANJE - JOIN

Operacija spajanja (join) ima nekoliko podvrsta.

Inner join - unutrašnja veza T:=R⋈ S

Inner join operacijom povezuju se tablice na način da se spajaju redovi tablica po istim vrijednostima zajedničkog atributa.

Spajaju se redovi koji u kolonama istog naziva u obje tablice imaju istu vrijednost.



75

R

A	B	D
s	1	4
d	1	5
c	2	5
s	3	3

S

D	E	F
5	1	5
1	2	4
3	6	7

T:=R⋈ S

A	B	D	E	F
d	1	5	1	5
c	2	5	1	5
s	3	3	6	7

U navedenom primjeru tablice R i S imaju zajednički atribut D, te se spajanje odvija po tom atributu.

Prvi red tablice R ima vrijednost atributa D=4, te ne sudjeluje u vezi, jer u tablici S ne postoji red sa D=4.

Drugi i treći red tablice R imaju vrijednost D=5, te se spajaju sa prvim redom u tablici S.

Četvrti red tablice R preko vrijednosti D=3 povezuje se sa trećim redom tablice S, gdje također imamo D=3.

76

Left outer join - lijeva vanjska veza T:=R⟕$_{LO}$ S

Lijeva vanjska veza je proširenje unutrašnje veze, kojoj se dodaju oni elementi tablice, koja je u relacijskom izrazu sa lijeve strane, koji ne sudjeluju u vezi.

R

A	B	D
s	1	4
d	1	5
c	2	5
s	3	3

S

D	E	F
5	1	5
1	2	4
3	6	7

T:=R⟕$_{LO}$S

A	B	D	E	F
d	1	5	1	5
c	2	5	1	5
s	3	3	6	7
s	1	4	null	null

U stvaranju R⟕$_{LO}$ S najprije se realizira R ⟕ S (time se dobiju prva tri reda tablice T).
Potom se dodaju svi redovi tablice R, koji ne sudjeluju u inner-join vezi. U ovom slučaju jedini red u tablici R koji nije obuhvaćen inner-join vezom je prvi red, a budući on ne sudjeluje u inner join-u vrijednosti atributa E i F su null vrijednosti.

77

Right outer join - desna vanjska veza T:=R⟖$_{RO}$ S

Desna vanjska veza je proširenje unutrašnje veze, kojoj se dodaju oni elementi tablice, koja je u relacijskom izrazu sa desne strane, koji ne sudjeluju u vezi.

R

A	B	D
s	1	4
d	1	5
c	2	5
s	3	3

S

D	E	F
5	1	5
1	2	4
3	6	7

T:=R⟖$_{RO}$S

A	B	D	E	F
d	1	5	1	5
c	2	5	1	5
s	3	3	6	7
null	null	1	2	4

U stvaranju R⟖$_{RO}$ S najprije se realizira R ⟕ S (time se dobiju prva tri reda tablice T).
Potom se dodaju svi redovi tablice S, koji ne sudjeluju u inner-join vezi. U ovom slučaju jedini red u tablici S koji nije obuhvaćen inner-join vezom je drugi red (D=1), a budući on ne sudjeluje u inner join-u vrijednosti atributa A i B su null vrijednosti.

78

450 - Baze podataka *Relacijska algebra*

Outer join - vanjska veza T:=R⟕$_O$ S

Vanjska veza je proširenje unutrašnje veze, kojoj se dodaju elementi obje tablice koji ne sudjeluju u unutrašnjoj vezi.

R

A	B	D
s	1	4
d	1	5
c	2	5
s	3	3

S

D	E	F
5	1	5
1	2	4
3	6	7

T:=R⟕$_O$S

A	B	D	E	F
d	1	5	1	5
c	2	5	1	5
s	3	3	6	7
null	null	1	2	4
s	1	4	null	null

Vrijedi **T:=R⟕$_O$ S= (R⟕$_{LO}$ S) ∪ (R⟕$_{RO}$ S)**

Dijeljenje T:=R ÷ S

Tablica T koja se dobije dijeljenjem R i S je najveća tablica za koju vrijedi da se svi redovi produkta T × S nalaze u tablici R.

R

A	B	D
s	1	4
s	1	5
c	2	5

S

A	B
s	1

T:=R÷S

D
4
5

79

450 - Baze podataka *Relacijska algebra*

Prioritet operacija:

1. Projekcija
2. Selekcija
3. Produkt
4. Join, Dijeljenje
5. Razlika
6. Unija, Presjek

LOGIČKE OPERACIJE

Logičke operacije često se primijenjuju u relacijskoj algebri.
Posebno se primjenjuju kod operacije selekcije tj. postavljanja složenih uvjeta za selekciju pojedinih redova iz tablice.

Logički operatori su AND, OR i NOT. Operatori AND i OR su funkcije koje se primijenjuju nad dva argumenta, a funkcija NOT nad jednim argumentom.

AND	T	F
T	T	F
F	F	F

OR	T	F
T	T	T
F	T	F

NOT	
T	F
F	T

80

450 - Baze podataka *Relacijska algebra*

Primjer sa ispita 13.11.2006.g.

2. Zadana je tablica Z:

Z

A	B	C	D
m	3	1	a
m	7	2	b
n	9	1	b
m	9	5	null

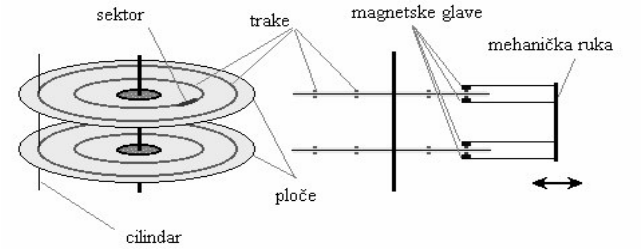
Neka je: X = Z[A,B] where D = 'b'
Y = Z[B,C,D] where D <> null

Treba odrediti $Z - X \triangleright\!\triangleleft Y$

81

450 - Baze podataka *Indeksiranje BP*

Podaci u bazi podataka (tablice i ostale strukture) moraju biti trajno smješteni na mediju koji omogućava sačuvanje podataka i njihovo normalno održavanje.
Radna memorija računala, iako pruža mogućnost brzog i efikasnog pristupa podacima, ne može biti medij za trajni smještaj podataka.
Stoga se podaci smještaju na disku računala.



Disk je rotirajući magnetski medij, uobičajeno sastavljen od nekoliko ploča (površina, *platters*) i mehaničkom rukom (*arm*).
Na vrhu ruke nalaze se magnetske glave (*heads*) preko kojih se vrši učitavanje i zapis.ivanje podataka.
Disk je sastavljen od niza traka (*tracks*), sektora (*sector*) i cilindara (*cylinder*).

82

450 - Baze podataka

Indeksiranje BP

Za pristup određenom podatku na disku, potrebno je pristupiti točno određenoj poziciji.
Ukupni ciklus pristupa podacima na disku može se prikazati kroz tri faze:
1. Pozicioniranje ruke
2. Rotiranje diska
3. Transfer – učitavanje podataka u memoriju.

Vremenski gledano prve dvije faze su znatno trajnije i čine daleko najveći dio ukupnog vremenskog ciklusa pristupa podacima, što pristup podacima na disku ukupno čini razmjerno sporim, što čini glavni problem smještaja podataka na disku.

Za isti kapacitet memorije troškovi radne memorije su oko 60 puta veći od cijene diskovnog prostora.
Ovo je posebno značajno kod baza podataka, gdje baratamo sa velikim količinama podataka.
Dodatni problem bio bi prebacivanje podataka sa jednog računala na drugo i stvaranje rezervnih kopija (backup).
Imajući sve ovo u vidu računalni disk ostaje kao jedini prikladni medij za pohranu podataka.

83

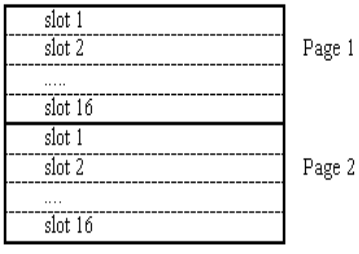
450 - Baze podataka

Indeksiranje BP

Diskovni prostor za smještaj baze podataka podijeljen je na stranice (*pages*). Uobičajene veličine stranica su 2Kb (2048 byte) i 4Kb (4096 byte).
Podaci iz tablice u bazi podataka smještaju su na stranice tvrdog diska u slotove. Broj slotova na pojedinoj stranici zavisi od veličine podatka u jednom redu tablice.

Primjer: Ako je memorijska stranica veličine 2Kb, a za smještaj jednog reda tablice treba 128 byte-a, onda jedna stranica za smještaj podataka iz navedene tablice sadrži 16 slotova. (128*16=2048)

Stranica predstavlja osnovnu jedinicu za učitavanje podataka sa diska.
Najveći dio vremena za učitavanje podataka čini smještaj magnetske glave na točno određenu poziciju, gdje se nalaze podaci.
Stoga se pri učitavanju podataka, učitavaju čitave stranice kako bi se što više smanjila potreba za ponovnim pozicioniranjem glave.



84

450 - Baze podataka

Indeksiranje BP

Uvođenje indeksa u bazu podataka ima za osnovni cilj ubrzavanje pretraživanja, tj. smanjenje broja pristupa disku. **Indeks** definiran nad tablicom vezuje se za određeni atribut (kolonu) u tablici.

Koncept indeksiranja može se usporediti sa katalogizacijom knjiga u biblioteci.
Za svaku knjigu postoji odgovarajuća kartica sa osnovnim podacima o knjizi i podatkom gdje je smještena.
Pri tome može postojati nekoliko odvojenih kataloga, u jednom su knjige sortirane prema naslovu, u drugom prema imenu autora itd.
Kad korisnik zatraži knjigu, npr. od željenog autora, potrebno je pogledati u katalog gdje su knjige sortirane po autoru, te se nađe pripadajuća kartica i pogleda u kojem je dijelu knjižnice ta knjiga.

85

450 - Baze podataka

Indeksiranje BP

Indeks za određeni atribut jest tablica sa dva stupca i jednakim brojem redova, kao i tablica nad kojom je indeks definiran.

INDEX (Prezime)

Prezime (Key)	D.P.
Bilić	P1
Milić	P2

STUDENT

id	Ime	Prezime	JMBG	Datum rođ.
1	Marko	Milić	xxxxx	
2	Ante	Bilić	xxxxx	

Prva kolona u indeksu je atribut za koji je indeks definiran i naziva se ključ indeksa (*index key*).
Druga kolona sadrži *disk pointer* (pokazivač na mjesto na disku na kojem se nalazi podatak sa određenom vrijednošću indeksnog ključa).

Važna osobina indeksa je da je tablica indeksa uvijek striktno uređena po vrijednostima ključa, bilo u rastućem redoslijedu (*ascending*) ili padajućem (*descending*).

86

450 - Baze podataka

Indeksiranje BP

Kod pristupa podacima po vrijednostima atributa nad kojima postoji indeks, sustav ne pretražuje cijelu polaznu tablicu, već umjesto toga, pretražuje se tablica indeksa, koja je po veličini mnogo manja.
Pri tome nije nužno pretražiti cijelu tablicu indeksa, već samo do zadane vrijednosti indeksnog ključa.
Kada se u pretraživanju indeksa dođe do zadane vrijednosti indeksnog ključa i lociraju svi podaci, pretraživanje indeksa se završava, budući je indeks uređen po vrijednostima ključa, pa ostatak tablice indeksa sigurno ne sadrži više podatke sa zadanom vrijednošću ključa.

Negativna strana indeksiranja ogleda se u činjenici da se smanjuje brzina ažuriranja i dodavanja novih podataka. Kod ovih operacija, uz promjene u samoj tablici, potrebno je promijeniti i tablicu indeksa.

87

450 - Baze podataka

Indeksiranje BP

VIŠESTRUKI INDEKSI

Višestruki indeksi predstavljaju indeksiranje bazne tablice po većem broju atributa. Indeksna tablica zadržava istu strukturu, tj. sastoji se od već opisanih kolona ključ indeksa (*key*) i pokazivač – *disk pointer*. Ključ indeksa složen je od vrijednosti svih atributa koji su uključeni u indeks.

INDEX (Prezime,JMBG)

(Key)	D.P.
Bilić;2222	P1
Milić;1111	P2

STUDENT

id	Ime	Prezime	JMBG	Datum rod.
1	Marko	Milić	1111	
2	Ante	Bilić	2222	

Ovakvi indeksi pogodni su za pretraživanje po jednoj ili obadvije vrijednosti atributa. Pretraživanje po jednoj vrijednosti atributa identično je pretraživanju indeksa sa jednim atributom. Pretraživanje po više vrijednosti daje rezultat samo one vrijednosti gdje oba atributa zadovoljavaju tražene uvjete.

88

CLUSTERED INDEKS

Uobičajeni način popunjavanja tablice prilikom unosa novih redova podrazumijeva slijedno unošenje, tj. jedan red iza drugog, po redu.
Uvođenjem indeksa formira se nova, indeksna tablica, koja sadržava podatke o položaju podataka u tablici prema indeksnom ključu, ali redoslijed podataka ostaje nepromijenjen.
Pretraživanjem tablice indeksa dobije se podatak o smještaju traženih podataka, ali su oni u osnovnoj tablici raspršeni.

Ukoliko se indeks definira svojstvom **CLUSTERED**, podrazumijeva se smještaj podataka u osnovnoj tablici u uređenom redoslijedu prema vrijednostima ključa.

INDEX (Prezime)

Prezime (Key)	D.P.
Antić	P1
Bilić	P2
Katić	P3
Milić	P4

STUDENT

id	Ime	Prezime	JMBG	Datum rod.
1	Ivo	Antić	xxxxx	
2	Ante	Bilić	xxxxx	
3	Jure	Katić		
4	Marko	Milić		

U jednoj tablici može postojati samo jedan CLUSTERED indeks.

89

JEDINSTVENI INDEKS (UNIQUE INDEX)

Definiranjem indeksa sa svojstvom jedinstvenosti (UNIQUE), određuje se da u indeksnoj tablici ne mogu postojati dvije iste vrijednosti indeksnog ključa, tj. ključ indeksa je jedinstven.
Budući je ključ indeksa vrijednost atributa iz tablice, time se definira da u tablici ne mogu postojati dva redka sa istom vrijednošću atributa po kojoj se stvara indeks.

Jedinstvenost indeksa, a time i atributa u tablici podrazumijeva da vrijednost atributa ne može biti null, tj. atribut po kojem se stvara jedinstveni indeks, mora imati minimalni kardinalitet 1.

Za višestruke indekse, koji predstavljaju kombinaciju više atributa, jedinstveni indeks podrazumijeva jedinstvenost kombinacije atributa koji su obuhvaćeni indeksom. Npr. ukoliko se definira jedinstveni indeks na tablici STUDENT po atributima ime i prezime, to podrazumijeva da u tablici ne mogu postojati dva studenta koji imaju jednaka oba atributa (ime i prezime).

90

NORMALIZACIJA BAZE PODATAKA

Normalizacija baze podataka predstavlja primjenu određenih matematičkih i formalnih pravila kojima se osigurava ispravno postavljanje modela podataka i njihova logička povezanost.

Definicija:
Funkcijska zavisnost atributa
Za tablicu R koja sadrži atribute X i Y koji mogu biti i složeni vrijedi funkcijska zavisnost atributa Y o atributu X , tj. X→Y, ako je svaka pojedina vrijednost atributa X povezana sa samo jednom vrijednošću atributa Y.
ILI
Vrijedi X→Y ako u tablici R ne postoje dva reda sa istom vrijednošću atributa X, a različitim vrijednostima atributa Y.

Definicija funkcijske zavisnosti (ovisnosti):
Vrijedi X→Y ako u tablici R ne postoje dva reda sa istom vrijednošću atributa X, a različitim vrijednostima atributa Y.

Primjer:

A	B	C
a	100	6
b	200	7
c	300	8
d	200	9

vrijedi A→B, A→C, C→A, C→B, B↛A, B↛C.

Definirane su 4 normalne forme, koje sve tablice u bazi moraju zadovoljavati da bi struktura podataka bila ispravna.

1. NORMALNA FORMA

Tablica se nalazi u 1. normalnoj formi ako su svi neključni atributi funkcijski ovisni o primarnom ključu.

Ovo pravilo je jednostavno i proizlazi iz definicije primarnog ključa.
Budući je primarni ključ jedinstven, tj. ne mogu se pojaviti dvije iste vrijednosti primarnog ključa u jednoj tablici, onda se podrazumijeva da su svi ostali atributi funkcijski ovisni o ključu.

Baza podataka je u 1. normalnoj formi, ako su sve tablice u 1. normalnoj formi.
Pravilo prve normalne forme naglašava transformacijsko pravilo prikaza viševrijednosnih atributa.

Primjer: Prikaz viševrijednosnog atributa sport za entitet STUDENT.

head(STUDENT) =
{Student_id, Ime, Prezime,
Dat.rođenja, Mj.rođenja,
JMBG, Sport}

card(Ime,STUDENT)=(1,1),
card(Prezime,STUDENT)=(1,1),
card(Dat.rođenja,STUDENT),
card(Mj.rođenja,STUDENT)=(1,1),
card(JMBG,STUDENT)=(1,1),
card(Sport,STUDENT)=(0,n)

STUDENT

Student id	Ime	Prezime	Dat.rođenja	Mj. rođenja	JMBG	Sport
1	Ante	Rožić	11.10. 1980	Osijek	1110980370071	-
2	Stipe	Anić	03.07. 1980	Split	0307980380025	Odbojka
2	Stipe	Anić	03.07. 1980	Split	0307980380025	Košarka
3	....	....	....	....		

Navedeni prikaz u skladu je sa pravilom da nije dozvoljen unos više vrijednosti pojedinog atributa u jednom retku. Dobijena tablica nije u prvoj normalnoj formi, jer nije zadovoljeno pravilo da su svi neključni atributi funkcijski zavisni o ključu.
U primjeru vrijedi Student_id ↛ Sport.
Posljedice ovakvog prikaza su razne anomalije, problemi koji se javljaju pri unosu i manipuliranju podacima.
1. Povrijeđena je jedinstvenost primarnog ključa
2. Anomalija unosa: za svaki sport kojim se bavi pojedini student, potrebno je ponovno unijeti vrijednost svih atributa vezanih za tog studenta: Ime, Prezime, itd.
3. Anomalija ažuriranja/promjene podataka: Kod promjene vrijednosti nekog atributa potrebno je taj atribut promijeniti u svim redovima.

2. NORMALNA FORMA

Tablica se nalazi u 2. normalnoj formi ako se nalazi u 1. normalnoj formi, i ako su svi neključni atributi potpuno funkcijski zavisni o ključu.
Pravilo druge normalne forme vrijedi za složeni primarni ključ koji se sastoji od više atributa (kolona).

Definicija:
Potpuna funkcijska zavisnost atributa
U tablici R koja sadrži atribute X i Y koji mogu biti i složeni, vrijedi da je Y potpuno funkcijski zavisan o atributu X , ako je Y funkcijski zavisan o X i nije funkcijski zavisan niti o jednom manjem podskupu atributa X. Drugim rječima, kada vrijedi $X \rightarrow Y$, tada ne smije postojati niti jedan podskup Z ($Z \subset X$), za koji bi vrijedilo $Z \rightarrow Y$.

95

Primjer:

Viševrijednosni atribut prikazan je novom tablicom u kojoj je primarni ključ složen od stranog ključa entiteta STUDENT i viševrijednosnog atributa, odnosno PK(SPORT)=(Student_id,Sport).

STUDENT

Student id	Ime	Prezime	Mj. rođenja	JMBG
1	Ante	Rožić	Osijek	1110980370071
2	Stipe	Anić	Split	0307980380025
3	....	....	....	

SPORT

Student_id	Sport	Dat.rođenja
2	odbojka	03.07. 1980
2	košarka	03.07. 1980
...	...	...

Tablica sadrži i atribut Dat.rođenja. S obzirom da je datum rođenja vezan za studenta, neovisno o tome da li se bavi sportom ili ne, vrijedi Student_id→Dat.rođenja.
Student_id je podskup primarnog ključa u tablici SPORT.
Ovakva tablica dakle nije u drugoj normalnoj formi.
Nepravilnosti koje prizlaze iz ovakvog prikaza:
1. Anomalija unosa: Ne može se unijeti dat.rođ. za studenta koji se ne bavi sportom.
2. Anomalija promjene: Kod promjene datuma rođenja, potrebno je promijeniti taj podatak u svim redovima koji su vezani za tog studenta.
3. Anomalija brisanja: Brisanjem posljednjeg sporta kojim se student bavi, gubi se i podatak o njegovom datumu rođenja.

Problem se riješava tako da se atribut koji je funkcijski zavisan o podskupu primarnog ključa (student_id), prebaci u drugu tablicu u kojoj je taj atribut-podskup primarni ključ. To je u ovom slučaju tablica STUDENT.

96

3. NORMALNA FORMA

Tablica se nalazi u 3. normalnoj formi ako se nalazi u 2. normalnoj formi, i ako niti jedan neključni atribut nije tranzitivno funkcijski zavisan o primarnom ključu.

Definicija:
Tranzitivna funkcijska zavisnost atributa

U tablici R koja sadrži atribute X, Y i A , vrijedi da je A tranzitivno funkcijski zavisan o atributu X , ako je X→Y, Y↛X, Y→A i A↛Y.
Simbolički tranzitivna zavisnost se prikazuje X→Y→A.

Primjer:

STUDENT

Student id	Ime	Prezime	JMBG
1	Ante	Rožić	1110980370071
2	Stipe	Anić	0307980380025
3	….	….	

UPISNI_LIST

ulist_id	student_id	semestar	šk.godina	obr.program	Mj.rođenja
1	1	1	1999/00	Elektronika	Osijek
2	1	2	1999/00	Elektronika	Osijek

Vrijedi: ulist_id→student_id (svaki upisni list vezan je samo za jednog studenta)
student_id↛ulist_id (jedan student ima više upisnih listova)
student_id→Mj.rođenja (student je rođen u jednom određenom mjestu)
Mj.rođenja↛student_id (u jednom mjestu može biti rođeno više studenata)

Budući vrijedi:
ulist_id→student_id →Mj.rođenja,
atribut mjesto rođenja je tranzitivno(posredno) funkcijski zavisan o primarnom ključu tablice (ulist_id).
Ovakva tablica dakle nije u trećoj normalnoj formi jer je Mj.rođenja funkcijski ovisno o student_id, a to polje nije primarni ključ, te ova tablica nije u 1. normalnoj formi.

450 – Baze podataka *Normalizacija BP*

Nepravilnosti koje proizlaze iz ovakve strukture podataka:
1. Anomalija unosa: Nije moguće unijeti mjesto rođenja za pojedinog studenta, dok se ne unese njegov upisni list.
2. Anomalija promjene/ažuriranja: Kod promjene mjesta rođenja pojedinog studenta, taj podatak treba mijenjati u svim upisnim listovima, koji su vezani za tog studenta.
3. Anomalija brisanja: Brisanjem upisnog lista briše se podatak o mjestu rođenja.

Problem svođenja tablice na treću normalnu formu rješava se na način da se atribut koji je tranzitivno zavisan o ključu, preseli u tablicu, u kojoj je atribut koji posreduje u tranzitivnoj vezi (u ovom slučaju student_id) primarni ključ.

STUDENT

Student id	**Ime**	**Prezime**	**Mj. rođenja**	**JMBG**
1	Ante	Rožić	Osijek	1110980370071
2	Stipe	Anić	Split	0307980380025
3	....	....	....	

UPISNI_LIST

ulist_id	**student_id**	**semestar**	**šk.godina**	**obr.program**
1	1	1	1999/00	Elektronika
2	1	2	1999/00	Elektronika

99

450 – Baze podataka *Normalizacija BP*

Primjer:

Zadana je relacijska tablica R sa svojstvom head(R)={A,B,C,D,E,F,G,H,I,J}.

Postoji skup funkcionalnih zavisnosti
F{ABCD→EFGHIJ, CD→GHJ, GH→J, E→F }

Treba razmotriti normalizaciju tablice R.

R
A
B
C
D
E
F
G
H
I
J

Prva normalna forma:

Po definiciji prve normalne forme svi neključni atributi moraju biti funkcijski zavisni o primarnom ključu.

S obzirom na funkcijsku zavisnost ABCD→EFGHIJ mora vrijediti da je primarni ključ PK(R)={A,B,C,D} .

	R
PK	**A**
PK	**B**
PK	**C**
PK	**D**
	E
	F
	G
	H
	I
	J

100

Druga normalna forma:

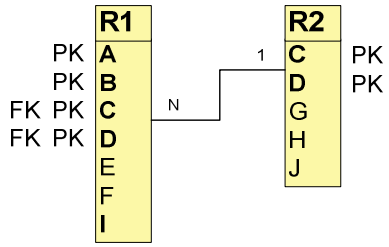
Svi neključni atributi moraju biti potpuno funkcijski zavisni o primarnom ključu.

Budući postoji funkcijska zavisnost CD→GHJ, atributi G, H i J su funkcijski zavisni o podskupu primarnog ključa.

Da bi se postigla druga normalna forma potrebno je tablicu R dekomponirati u dvije tablice.

R1| head(R1)={A, B, C, D, E , F, I}
PK(R1)={A,B,C,D}

R2| head(R2)= {C, D, G, H, J},
PK(R2)={C, D}



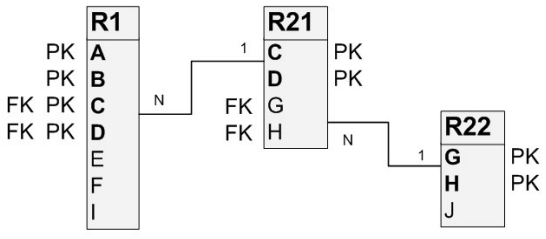
Treća normalna forma:

Uz definiranu funkcijsku zavisnost GH→J tablica R2 nije u trećoj normalnoj formi zbog tranzitivne funkcijske zavisnosti o primarnom ključu.

Tj. vrijedi CD→GH→J. Tablicu R2 treba dekomponirati po trećoj normalnoj formi:

R21 | head(R21)={C, D, G, H},
PK(R21)={C, D}

R22 | head(R22)={G, H, J },
PK(R22)={G, H}



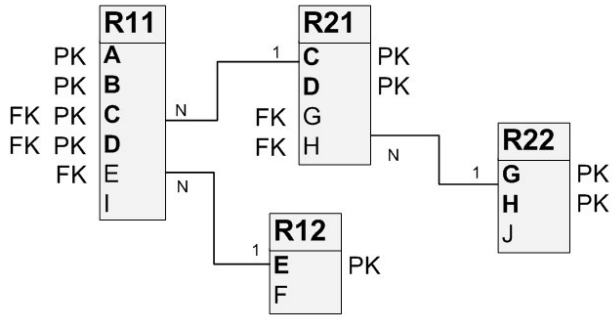
Treća normalna forma:

Slično razmatranje vrijedi za tablicu R1, koja uz definiranu funkcijsku zavisnost E→F, nije u trećoj normalnoj formi zbog tranzitivne funkcijske zavisnosti o primarnom ključu ABCD→E→F.

Tablica se dekomponira u dvije tablice:

R11 | head(R11)={ A, B, C, D, E , I},
PK(R11)={A,B,C,D}

R12 | head(R12)={ E, F},
PK(R12)={E}



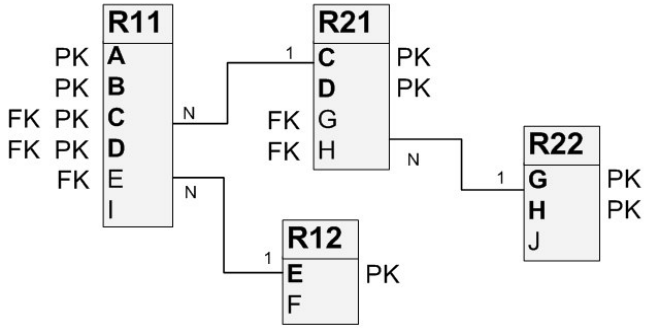
Konačno rješenje svođenja tablice R na normalne forme su tablice:

R11 | head(R11)={ A, B, C, D, E , I},
PK(R11)={A,B,C,D}

R21 | head(R21)={C, D, G, H},
PK(R21)={C, D}

R12 | head(R12)={ E, F},
PK(R12)={E}

R22 | head(R22)={G, H, J },
PK(R22)={G, H}



BOYCE-CODDOVA NORMALNA FORMA (BCNF)

Definicija:
Determinant je atribut o kojem je neki atribut potpuno funkcijski zavisan.

Tablica se nalazi u BCN formi ako svaki determinant ima jedinstvenu vrijednost u cijeloj tablici.

Primjer 2:
Tablice STUDENT i MJESTO.
PK(STUDENT)=Student_id,
PK(MJESTO)=Mjesto_ID
=FK(STUDENT)

MJESTO

Mjesto_ID	IME MJESTA	OPĆINA	DRŽAVA
1	Split	Split	Hrvatska
2	Smokvica	Pag	Hrvatska
3	Smokvica	Smokvica	Hrvatska
4	Osijek	Osijek	Hrvatska
5	Bonn	-	Njemačka

STUDENT

Student id	Ime	Prezime	Dat.rođenja	Mjesto_ID	Pošt.broj
1	Ante	Rožić	11.10. 1980	4	31000
2	Stipe	Anić	03.07. 1980	1	21000
3	....	....	....	....	

105

Za tablicu STUDENT vrijedi:

Student_id→Mjesto_ID , Mjesto_ID →Pošt.broj,
Mjesto_ID ↛Student_id, Pošt.broj → Mjesto_ID

Atribut Mjesto_Id je determinant jer je atribut Pošt.broj potpuno funkcijski zavisan o oznaci mjesta (Mjesto_id).
U tablici se ne mogu pojaviti dva reda sa istom vrijednošću Mjesto_id, a različitom vrijednošću atributa Pošt.broj.
Dakle Mjesto_ID jest determinant, ali nije jedinstven, tj. u tablici se mogu pojaviti studenti koji su rođeni u istom mjestu, tj. u raznim redovima može se pojaviti ista vrijednost atributa Mjesto_ID.
Tablica dakle nije u BCN formi.

Nepravilnosti koje proizlaze iz ovakve strukture podataka:
1. Anomalija unosa: Nije moguće unijeti poštanski broj za mjesto dok god nije unešen prvi student koji je rođen u tom mjestu.
2. Anomalija promjene: Ako se promijeni poštanski broj pojedinog mjesta, treba promijeniti sve zapise koji pripadaju studentima rođenim u tom mjestu
3. Anomalija brisanja: Ako se iz tablice ukloni zadnji student koji je rođen u nekom mjestu, gubi se podatak o poštanskom broju

106

Problem svođenja tablice na BCN formu rješava se na način da se atribut koji je potpuno funkcijski zavisan o determinantu koji nije jedinstven, preseli u tablicu, u kojoj je taj determinant (u ovom slučaju Mjesto_ID) primarni ključ.

U ovom primjeru to znači da se atribut Pošt.broj preseli u tablicu u kojoj je primarni ključ atribut Mjesto_ID.

MJESTO

Mjesto_ID	IME MJESTA	POŠT. BROJ	OPĆINA	DRŽAVA
1	Split	21000	Split	Hrvatska
2	Smokvica	23249	Pag	Hrvatska
3	Smokvica	20272	Smokvica	Hrvatska
4	Osijek	31000	Osijek	Hrvatska
5	Bonn	-	-	Njemačka

STUDENT

Student id	Ime	Prezime	Dat.rođenja	Mjesto_ID
1	Ante	Rožić	11.10. 1980	4
2	Stipe	Anić	03.07. 1980	1
3	....	....	....	....

107

4. NORMALNA FORMA

Tablica se nalazi u 4. normalnoj formi ako i samo ako vrijedi da postojanje višeznačne zavisnosti atributa A→» B povlači za sobom postojanje funkcijske ovisnosti svih atributa u toj tablici o atributu A.

Definicija:
Višeznačna zavisnost atributa
U tablici R koja sadrži atribute A,B i C , vrijedi da je B višeznačno zavisan o atributu A , ako vrijedi:
za svaki B koji odgovara vrijednostima atributa A i C,
B je ovisan samo o A, ali ne i o C.

108

Primjer: Tablica predavanja

Predavač	Predmet	Poglavlje
Ilić	Fizika	Optika
Ilić	Fizika	Mehanika
Ilić	Fizika	Toplina
Matić	Fizika	Optika
Matić	Fizika	Mehanika
Matić	Fizika	Toplina

U tablici se bilježe predmeti, predavači i poglavlja koja se predaju za pojedine predmete.
Vrijedi:
Predmet→>Poglavlje, budući da je sadržaj predmeta vezan isključivo za predmet, a ne zavisi od toga koji nastavnik predaje taj predmet.
Da bi tablica bila u 4. normalnoj formi treba vrijediti Predmet→Predavač, što ovdje ne vrijedi budući za jedan isti predmet postoji više predavača.

Tablica se svodi na 4. NF rastavljanjem postojeće tablice na dvije nove:

Predavač	Predmet
Ilić	Fizika
Matić	Fizika

Predmet	Poglavlje
Fizika	Optika
Fizika	Mehanika
Fizika	Toplina

Svođenje tablica na zadane normalne forme, često zahtjeva rastavljanje postojećih tablica (dekompoziciju).

Pri dekompoziciji tablica, mora vrijediti da se postupkom razdvajanja tablica ne izgubi niti jedna informacija, niti se pojave informacije koje ne postoje u polaznoj tablici.

U tu svrhu postoje dva pravila:

1. Svaka funkcijska zavisnost tablice T može biti logički izvedena iz funkcijskih zavisnosti u tablicama R i S, koje nastanu dekompozicijom tablice T.
2. Zajednički atribut tablica R i S mora biti ključ u barem jednoj od tih tablica.

450 - Baze podataka

Normalizacija BP

Primjer:

STUDENT1

Student id	Sport	Ime	Prezime	Mjesto_id	Ime_mjesta	Pošt.broj	JMBG
1	odbojka	Ante	Rožić	4	Osijek	31000	1110980370071
2	nogomet	Stipe	Anić	1	Split	21000	0307980380025
1	košarka	Ante	Rožić	4	Osijek	31000	1110980370071
3		....	....		....		

PK(STUDENT1) =(Student_id,Sport)
Ovako formirana tablica ne zadovoljava 2NF, budući su neki atributi funkcijski zavisni o podskupu primarnog ključa.

Vrijedi:
Student_id→Ime, Student_id→Prezime, Student_id→Mjesto_id itd.

U skladu sa opisanim načinom rješavanja 2NF, sve atribute koji su funkcijski zavisni o nekom atributu koji je podskup primarnog ključa treba prebaciti u tablicu u kojoj je taj atribut primarni ključ.

Dakle treba formirati novu tablicu u kojoj je primarni ključ atribut Student_id, u koju se zatim projiciraju svi atributi koji su funkcijski zavisni o Student_id (Ime,Prezime, Mjesto_id, Ime_mjesta, Pošt.broj, JMBG).

111

450 - Baze podataka

Normalizacija BP

Primjer:

STUDENT

Student id	Ime	Prezime	Mjesto_id	Ime_mjesta	Pošt.broj	JMBG
1	Ante	Rožić	4	Osijek	31000	1110980370071
2	Stipe	Anić	1	Split	21000	0307980380025
3	....	....		....		

Polazna tablica rastavljena je u dvije tablice, koje su sada u 2NF. Međutim tablica STUDENT nije u 3NF, budući unutar nje postoji tranzitivna zavisnost pojedinih atributa o primarnom ključu.

SPORT

Student id	Sport
1	odbojka
2	nogomet
1	košarka
3	

Vrijedi
Student_id→Mjesto_id→Ime_mjesta.
Razrješavanje tranzitivne zavisnosti podrazumijeva formiranje nove tablice u kojoj je atribut koji posreduje u funkcijskoj zavisnosti (Mjesto_id) primarni ključ.

MJESTO

Mjesto_ID	Ime_mjesta
1.	Split
2.	Smokvica
3.	Smokvica
4	Osijek
5.	Bonn

STUDENT

Student id	Ime	Prezime	Mjesto_id	Pošt.broj	JMBG
1	Ante	Rožić	4	31000	1110980370071
2	Stipe	Anić	1	21000	0307980380025
3	....	....			

112

450 - Baze podataka *Normalizacija BP*

Sada je tablica STUDENT u 3NF, ali ne zadovoljava BCN formu, zbog toga što je atribut Mjesto_ID determinant s obzirom na atribut Pošt.broj, ali nema jedinstvenu vrijednost u cijeloj tablici.
Razrješavanjem ove BCN forme dobijemo konačnu strukturu tablica:

STUDENT

Student id	Ime	Prezime	Mjesto_id	JMBG
1	Ante	Rožić	4	1110980370071
2	Stipe	Anić	1	0307980380025
3	....	....		

MJESTO

Mjesto_ID	Ime_mjesta	Pošt.broj
1	Split	21000
2.	Smokvica	....
3.	Smokvica.	....
4	Osijek	31000
5.	Bonn	-

113

450 - Baze podataka *SQL*

SQL jezik
(*Structured Query Language* – strukturni jezik za pretraživanje) jest srce relacijske tehnologije.
Razvoj SQL-a kao jezika za manipulaciju podacima tekao je usporedo s razvojem relacijskog modela podataka.
Osnove SQL-a je postavio 1971. godine E.F.Codd. Kada je Codd postavio svoj koncept relacijskog modela podataka, ustvrdio je da "... usvajanje relacijskog modela podataka... dopušta razvoj univerzalnog jezika podataka baziranog na primjeni relacijske algebre". Iako je odmah uočio zahtjeve koje bi takav jezik trebao ispunjavati, kao i njegove prednosti pri manipulaciji podacima, tada nije pokušao razviti takav jezik u praksi.

SQL je standardiziran po ISO (International Standardization Organization) i ANSI (American National Standards Institute).
Svi sustavi za upravljanje bazama podataka, kako oni najmanji na PC platformama, tako i veliki client-server sustavi, nastoje što je moguće više slijediti originalni standard SQL-a, ali ga i obogaćuju raznim dodatnim opcijama.

114

TIPOVI PODATAKA

Prilikom definiranja podataka u relacijskim sustavima, za sva svojstva entiteta (za sve atribute), potrebno je definirati tip podataka.
To je ključni segmenat određivanja domene nad kojom su definirani pojedini atributi.

Tipovi podataka (format) mogu se podijeliti u 4 ključne kategorije
- znakovni tipovi podataka
- binarni podaci
- numerički tipovi podataka
- podaci formata datum-vrijeme.

115

ZNAKOVNI TIPOVI PODATAKA

Character

Char (n) – string duljine točno n znakova.

Svaki znak prikazuje se jednim bajtom.

Znakovi mogu biti alfanumerički i specijalni.
Duljina znaka je uvijek n bajtova, bez obzira na veličinu podatka koji se unosi, a maksimalan broj znakova je 255.

Ako je stvarna duljina unesenog stringa manja od n, sustav dodaje prazne znakove do duljine n.

Primjer: Ako je n=6, a uneseni string 'ABC', on se u memoriji pohranjuje kao 'ABC###', tj. uvijek fiksne duljine n=6.

Pridjeljuje se alfanumeričkim atributima, za koje se pretpostavlja da su približno iste duljine.

Primjer: atribut *Post_broj char(5)*

116

Character varying

Varchar (n) – string promjenive duljine, maksimalno n karaktera.
Pri tome maksimalna duljina zavisi od mogućnosti procesora i operativnog sustava, a najčešće iznosi 255.

Pri unosu podataka, duljina stringa nije fiksna već je određena veličinom podatka.

Primjer:
Ako je n=25, a uneseni podatak 'Ivo', bilježi se duljine 4 bajta kao 'Ivo3', pri čemu zadnji bajt 3 označava stvarnu duljinu stringa.

Format varchar pridjeljuje se alfanumeričkim atributima promjenjive duljine.

Primjer: atributi: *Prezime_stud varchar(30), Ime_stud varchar(20), Adresa varchar (50)*

Text

Text - String neograničene veličine.

Primjer: atribut *Napomena text*

BINARNI PODACI

Bit

Bit – podatak koji predstavlja binarnu vrijednost (1 ili 0).

Veličina rezerviranog memorijskog prostora je 1 bajt.

Ukoliko se bit tipu podataka pokuša pridjeliti cjelobrojna vrijednost, ona se interpretira kao 1.

Primjer: atribut *Apsolvent bit*

NUMERIČKI TIPOVI PODATAKA

Binary(n)

Sadržava maksimalno 255 bajta binarnih podataka, te se koristi za spremanje binarnih i heksadecimalnih vrijednosti.

INTEGER – int

cjelobrojni format s predznakom ili bez njega.
Zauzima 4 bajta i sadržava cijele brojeve
od $\mathbf{-2^{31}}$ (-2,147,483,648) do $\mathbf{2^{31} - 1}$ (2,147,483,647).

SMALL INTEGER – smallint

cjelobrojni format s predznakom ili bez njega.
Zauzima 2 bajta i sadržava cijele brojeve
od $\mathbf{-2^{15}}$ (-32,768) do $\mathbf{2^{15}-1}$ (32,767).

Primjer: atribut *Broj_racuna smallint*

119

TINY INTEGER (BYTE) – tinyint

cjelobrojni format koji prikazuje brojeve od 0 do 255. Zauzima 1 bajt.

Primjer: atribut *Ocjena tinyint*

NUMERIC - numeric(p,q)

decimalni broj, ukupno p znamenki i predznak, q znamenki iza decimalnog zareza.

Primjer: atribut *Cijena_artikla numeric(12,2)*

DECIMAL - decimal(p,q)

decimalni broj, koji se interno bilježi sa m znamenki, pri čemu vrijedi 0<p<q<m.

Npr. *Decimal (7,2)* dopušta unos broja koji ima točno dvije znamenke iza decimalnog zareza, a ukupan broj znamenaka mu je najmanje 7.

120

450 - Baze podataka

SQL – tipovi podataka

FLOAT – Float(n)

Realni broj u formatu pomičnog zareza (floating point). Zauzima 4 bajta memorije.

DOUBLE – Double(n)

Realni broj dvostruke preciznosti u formatu pomičnog zareza (floating point). Zauzima 8 bajta memorije.

121

450 - Baze podataka

SQL – tipovi podataka

DATUM I VRIJEME FORMAT

Datum i vrijeme sastavljeni su od alfanumeričkih podataka koji predstavljaju podatke o datumu i vremenu.
Uobičajeni format prikaza je 'dan-mjesec-godina sat:minut:sekunda'

datum_posl_promj
2006-01-02 00:00:00

datetime

Tip podatka koji je predstavljen sa 8 bajtova ili dva 4-bajtna cjela broja: 4 bajta za prikaz datuma i 4 bajta za prikaz vremena (sati).
Moguće je prikazati datume od 1. siječnja 1753 do 31. prosinca 9999, sa točnošću od 3.33 millisekunde.

smalldatetime *- skraćeni format datum - vrijeme*

Tip podatka koji je predstavljen sa 4 bajta: 2 bajta za prikaz datuma i 2 bajta za prikaz vremena (sati).
Moguće je prikazati datume od 1. siječnja 1900 do 6. lipnja 2079, sa točnošću od minute.

122

FORMIRANJE TABLICE

Prilikom definiranja tablice (entiteta) potrebno je definirati sve atribute (kolone) kao i ograničenja vezana za integritet podataka i referencijalni integritet.

Za formiranje tablice koristi se SQL instrukcija **CREATE TABLE**, koja ima sintaksu oblika:

CREATE TABLE *ime_tablice*
(
naziv_kolone1 | svojstva | column_constraint,
naziv_kolone2 | svojstva | column_constraint,
....,
table constraints
)

```
CREATE TABLE MJESTO
(
naziv_mjesta    VARCHAR(20),
post_broj       CHAR(5)
)
```


CREATE TABLE *ime_tablice*
(
naziv_kolone1 | svojstva | column_constraint,
naziv_kolone2 | svojstva | column_constraint,
....,
table constraints
)

```
CREATE TABLE MJESTO
(
naziv_mjesta    VARCHAR(20),
post_broj       CHAR(5)
)
```

ime_tablice
Ime svake tablice mora biti jedinstveno u bazi podataka. Dužina naziva tablice može imati maksimalno 128 znakova.

naziv_kolone
Predstavlja naziv pojedinačne kolone u tablici. Naziv pojedine kolone mora biti jedinstven u tablici.

svojstva
Određuju tip podataka, NULL vrijednosti, IDENTITY – svojstvo za kolonu.

124

450 - Baze podataka

SQL – formiranje tablice

IDENTITY[(*seed*, *increment*)]

Generira inkrementalnu vrijednost za svaki novi red podataka u tablici na osnovu *seed* i *increment* parametara.
Ako je navedena, *seed* vrijednost označava početak brojača i bit će dodijeljena prvom redu u tablici, a svaki slijedeći red dobit će jedinstvenu identity vrijednost, koja je jednaka posljednjoj identity vrijednosti uvećanoj za vrijednost parametra *increment*.
Ako vrijednosti *seed* i *increment* nisu navedene smatra se da je njihova vrijednost 1.
Pridjeljivanje IDENTITY svojstva koloni podrazumijeva njezino svojstvo NOT NULL.
Samo jedna kolona u tablici može biti definirana sa svojstvom identity.
Svojstvo identity može se dodijeliti onoj koloni koja je definirana kao numerički tip podataka.

```
CREATE TABLE MJESTO
(
MJESTO_id       INT IDENTITY,
naziv_mjesta    VARCHAR(20),
post_broj       CHAR(5)
)
```

450 - Baze podataka

```
CREATE TABLE MJESTO
(
MJESTO_id       INT IDENTITY,
naziv_mjesta    VARCHAR(20) NOT NULL,
post_broj       CHAR(5) NULL
)
```

NULL | NOT NULL
Svojstvo koje određuje da li je navedena kolona može sadržavati null vrijednosti. Ukoliko se u definiciji kolone ovo svojstvo ne navede izričito, smatra se NULL.

CONSTRAINT *ime_ograničenja*
Constraint – ograničenje, definira ograničenja i dodatna relacijska svojstva unutar tablice.
Ako *ime_ograničenja* nije navedeno u naredbi CREATE TABLE, sistem sam generira naziv za pojedina ograničenja.
Ako se zahtjevi na pojedina ograničenja promijene, ograničenje se uklanja, te potom kreira novo.
Pri tome je bitno razlikovati postavljanje ograničenja na pojedine kolone, od postavljanja ograničenja na nivou tablice (*table constraints*). Ukoliko su ograničenja vezana za pojedinu kolonu ona se pri definiranje tablice navode u definiciji kolone. Ako se radi o ograničenjima koja obuhvaćaju više kolona (npr. složeni primarni ključ), njih je potrebno definirati kao ograničenja na nivou tablice, nakon što su definirane sve kolone u tablici.

126

450 - Baze podataka

SQL – formiranje tablice

PRIMARY KEY [CLUSTERED | NONCLUSTERED] (*col_1*, *col_2*, ... , *col_n*)

Osigurava integritet i jedinstvenost podataka u određenom atributu (koloni), gdje su *col_1, ..., col_n* kolone (atributi) koje čine primarni ključ.

Sve kolone koje čine primarni ključ moraju imati svojstvo NOT NULL.

Ako to nije posebno navedeno, sustav sam postavlja NOT NULL, za sve kolone koje čine primarni ključ tablice.

Ukoliko se ograničenje primarnog ključa definira na nivou kolone, lista kolona *(col1)* se izostavlja.

Za osiguranje jedinstvenosti primarnog ključa, SQL automatski formira jedinstveni (unique) index na zadane kolone primarnog ključa.

Unutar tablice može postojati samo jedan primarni ključ.

```
CREATE TABLE MJESTO
(
MJESTO_id       INT IDENTITY
        CONSTRAINT PK_MJESTO PRIMARY KEY,
naziv_mjesta    VARCHAR(20) NOT NULL,
post_broj       CHAR(5) NULL
)
```

450 - Baze podataka

SQL – formiranje tablice

UNIQUE [CLUSTERED | NONCLUSTERED] (*col_1*, *col_2* , ... , *col_n*)

Osigurava jedinstvenost podataka u navedenim kolonama, gdje su *col_1, ..., col_n* kolone (atributi) obuhvaćene ograničenjem.

Ukoliko se ograničenje jedinstvenosti definira na nivou kolone, lista kolona se izostavlja.

Unutar tablice može se definirati više UNIQUE ograničenja.

SQL sustav automatski stvara jedinstveni (unique) index za kolonu ograničenu UNIQUE.

Za kolonu koja se definira sa ograničenjem UNIQUE, sistem automatski postavlja svojstvo NOT NULL.

```
CREATE TABLE MJESTO
(
MJESTO_id       INT IDENTITY
        CONSTRAINT PK_MJESTO PRIMARY KEY,
naziv_mjesta    VARCHAR(20) NOT NULL
        CONSTRAINT UQ_NAZIV UNIQUE,
post_broj       CHAR(5) NULL
)
```

128

450 - Baze podataka

SQL – formiranje tablice

DEFAULT *konstanta* | *izraz*

Određuje uobičajenu vrijednost (default) za kolonu, kada se prilikom unosa podataka u tablicu ne navede vrijednost za taj atribut.

DEFAULT constraint može se primijeniti na kolone bilo kojeg tipa podataka, osim one kolone kojoj je dodijeljeno IDENTITY svojstvo.

```
CREATE TABLE MJESTO
(
MJESTO_id       INT IDENTITY
        CONSTRAINT PK_MJESTO PRIMARY KEY,
naziv_mjesta    VARCHAR(20) NOT NULL
        CONSTRAINT UQ_NAZIV UNIQUE,
post_broj       CHAR(5) NULL,
drzava          VARCHAR(20) DEFAULT 'Hrvatska'
)
```

CHECK (*expression*)

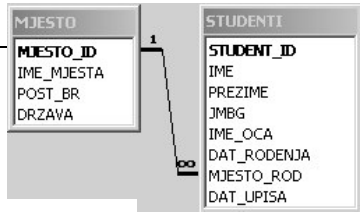
Ograničava moguće vrijednosti koje se unose u pojedine kolone tablice.

129

450 - Baze podataka

Primjer:

Definirane su tablice MJESTO i STUDENT iz baze podataka koja opisuje studiranje

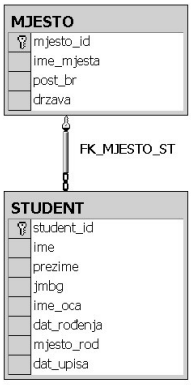


Access

```
CREATE TABLE MJESTO
(       mjesto_id       int IDENTITY CONSTRAINT PK_MJESTO PRIMARY KEY,
        ime_mjesta      varchar(30) NOT NULL,
        post_br         int NULL,
        drzava          varchar(25) NOT NULL
)

CREATE TABLE STUDENT
(       student_id      int IDENTITY CONSTRAINT PK_STUDENT PRIMARY KEY,
        ime             varchar(20) NOT NULL,
        prezime         varchar(30) NOT NULL,
        jmbg            char(13) NULL,
        ime_oca         varchar(20) NULL,
        dat_rođenja     smalldatetime NOT NULL,
        mjesto_rod      int NOT NULL
        CONSTRAINT FK_MJESTO_ST FOREIGN KEY REFERENCES MJESTO(mjesto_id),
        dat_upisa       smalldatetime NOT NULL DEFAULT getdate()
)
```

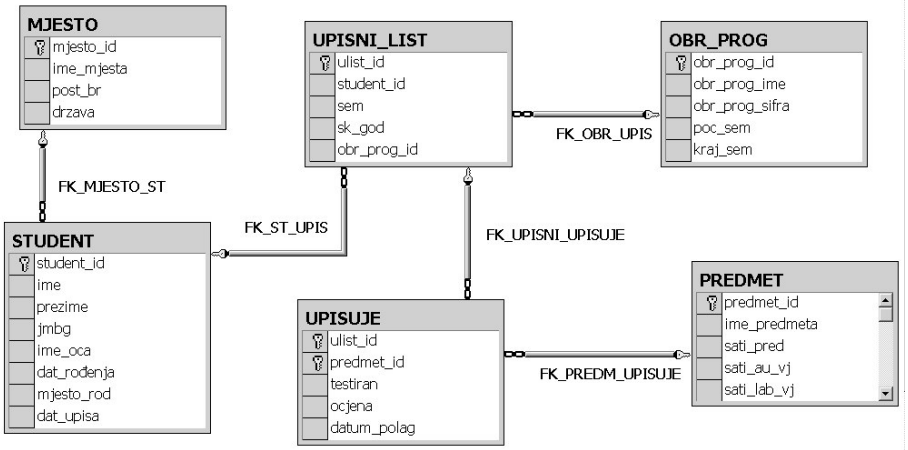
SQL Server



450 - Baze podataka

SQL – CREATE TABLE

Potrebno je još kreirati tablice OBR:PROG, PREDMET, UPISNI_LIST i UPISUJE:



131

450 - Baze podataka

SQL – CREATE TABLE

Potrebno je još kreirati tablice OBR:PROG, PREDMET, UPISNI_LIST i UPISUJE:

```
CREATE TABLE OBR_PROG
(       obr_prog_id      int IDENTITY CONSTRAINT PK_OBR_PROG PRIMARY KEY,
        obr_prog_ime    varchar(50) NOT NULL,
        obr_prog_sifra  int NOT NULL,
        poc_sem         int NOT NULL,
        kraj_sem        int NOT NULL
)

CREATE TABLE PREDMET
(       predmet_id      int IDENTITY,
        ime_predmeta    varchar(25) NOT NULL,
        sati_pred       tinyint NOT NULL,
        sati_au_vj      tinyint NOT NULL DEFAULT 0,
        sati_lab_vj     tinyint NOT NULL DEFAULT 0,
        CONSTRAINT PK_PREDMET PRIMARY KEY(predmet_id)
)
```

132

Potrebno je još kreirati tablice OBR:PROG, PREDMET, UPISNI_LIST i UPISUJE:

```
CREATE TABLE UPISNI_LIST
(       ulist_id        int IDENTITY CONSTRAINT PK_UPISNI PRIMARY KEY,
        student_id      int NOT NULL
        CONSTRAINT FK_ST_UPIS FOREIGN KEY REFERENCES student(student_id),
        sem             tinyint NOT NULL,
        sk_god          varchar(8) NOT NULL,
        obr_prog_id     int NOT NULL
        CONSTRAINT FK_OBR_UPIS FOREIGN KEY REFERENCES OBR_PROG(obr_prog_id),
        CONSTRAINT CHK_SEM CHECK (sem>0 AND sem<6)
)

CREATE TABLE UPISUJE
(       ulist_id        int
        CONSTRAINT FK_UPISNI_UPISUJE FOREIGN KEY REFERENCES upisni_list(ulist_id),
        predmet_id      int
        CONSTRAINT FK_PREDM_UPISUJE FOREIGN KEY REFERENCES predmet(predmet_id),
        testiran        bit NOT NULL DEFAULT 0,
        ocjena          tinyint,
        datum_polag     smalldatetime,
        CONSTRAINT PK_UPISUJE PRIMARY KEY (ulist_id,predmet_id)
)
```


Na ovim primjerima vidljive su razlike u definiranju ograničenja (constraints) na nivou kolone (column constraints) ili na nivou tablice (table constraints).

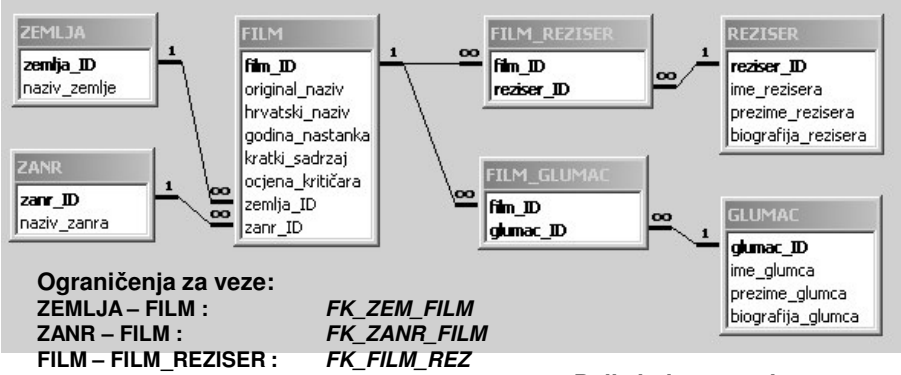
Kod deklariranja tablice ***STUDENT***, primarni ključ čini jedna kolona (*student_id*), te je ograničenje primarnog ključa (PRIMARY KEY) moguće primijeniti u sklopu definicije kolone koja čini primarni ključ (*student_id*).

Tablica ***UPISUJE*** je relacijska tablica koja je nastala kao posljedica relacije više-na-više između entiteta ***UPISNI_LIST*** i ***PREDMET***. Takva tablica ima složeni primarni ključ koji je sastavljen od stranih ključeva entiteta koji se nalaze u relacijskoj vezi (*ulist_id* i *predmet_id*).
Stoga ograničenje primarnog ključa nije moguće prikazati u sklopu deklaracije jedne kolone, već se mora primijeniti ograničenje na nivou tablice.

450 - Baze podataka

SQL – CREATE TABLE

Primjer:
Za zadani model podataka treba napisati SQL naredbe za kreiranje tablica:



Ograničenja za veze:

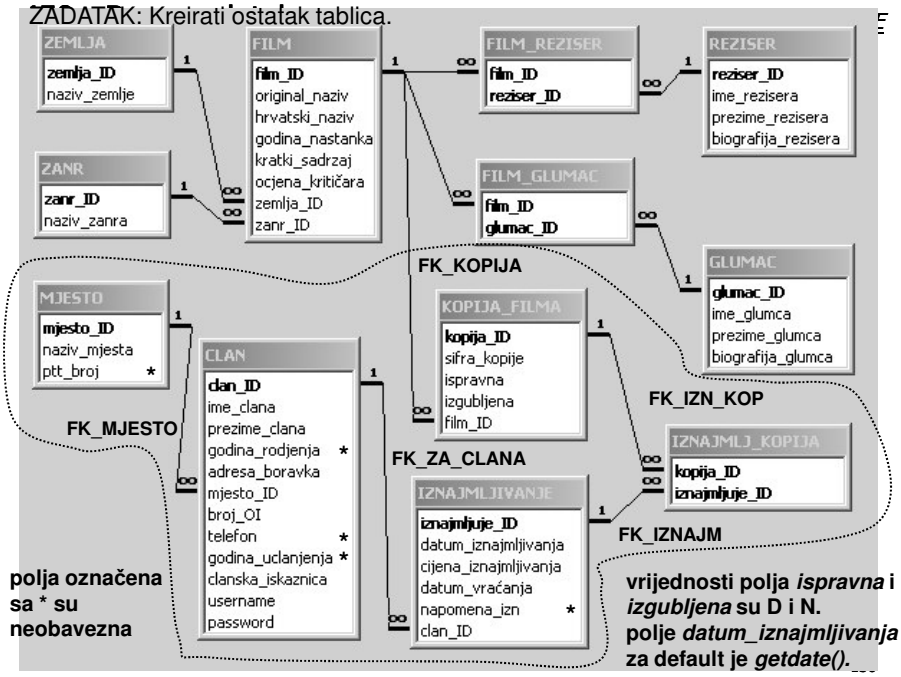
ZEMLJA – FILM :	***FK_ZEM_FILM***
ZANR – FILM :	***FK_ZANR_FILM***
FILM – FILM_REZISER :	***FK_FILM_REZ***
FILM_REZISER - REZISER :	***FK_REZISER***
FILM – FILM_GLUMAC :	***FK_FILM_GLUM***
FILM_GLUMAC - GLUMAC :	***FK_GLUMAC***

Dodatna ograničenja:
ocjena_kriticara: integer od 1 do 5
godina_nastanka: nakon 1950.

Polja koja su neobavezna:
hrvatski_naziv
kratki_sadrzaj
biografija_rezisera
biografija_glumca

135

ZADATAK: Kreirati ostatak tablica.



polja označena sa * su neobavezna

vrijednosti polja *ispravna* i *izgubljena* su D i N.
polje *datum_iznajmljivanja* za default je *getdate()*.

PROMJENE STRUKTURE TABLICE

Promjene na već formiranoj tablici obavljaju se naredbom **ALTER TABLE.**

A) Dodavanje kolona ili ograničenja (constraints) u tablicu:

ALTER TABLE *ime_tablice*
ADD *naziv_kolone | svojstva | constraints,*
naziv_kolone | svojstva | constraints.

Primjer:

ALTER TABLE student ADD CONSTRAINT UQ_JMBG UNIQUE (jmbg)

ALTER TABLE mjesto ADD CONSTRAINT DF_DRZAVA
DEFAULT 'Hrvatska' FOR drzava

ALTER TABLE student ADD spol char (1) NOT NULL
CONSTRAINT CK_SPOL check (spol='M' OR spol='Ž')

B) Brisanje kolona ili ograničenja (constraints) iz tablice:

ALTER TABLE *ime_tablice*
DROP *COLUMN naziv_kolone,*
CONSTRAINT naziv_ograničenja

Primjer:
ALTER TABLE student DROP COLUMN ime_oca

C) Promjene svojstava postojećih kolona u tablici:

ALTER TABLE *ime_tablice*
ALTER COLUMN *naziv_kolone | svojstva*

Primjer:
ALTER TABLE student ALTER COLUMN prezime varchar(20) NOT NULL

Promjene na nekoj već formiranoj tablici, kako je navedeno obavljaju se naredbom ALTER TABLE.

Međutim ovaj oblik promjena podliježe određenim pravilima.

Dodavanje novih kolona

Dodavanje novih kolona može se slobodno realizirati dok je tablica prazna, te ne sadrži podatke.

Međutim ako tablica već sadrži neke podatke, svaki od već postojećih redova u tablici proširuje se sa novom kolonom, za koju prethodno nije definirana njegova vrijednost.

U ovakvim slučajevima, moguća su dva rješenja:

1. Kolona koja se dodaje u tablicu treba se definirati sa svojstvom NULL.
   Na taj način svim već postojećim redovima dodaje se nova kolona, a vrijednosti u toj koloni za svaki redak u tablici bit će null.

Dodavanje novih kolona – *nastavak*

2. Ako se kolona ne želi definirati sa svojstvom NULL, za nju je potrebno navesti default vrijednost.
   U takvim slučajevima svi postojeći redovi u novostvorenoj koloni poprimaju vrijednost koja je određena kao default.

Primjer:

ALTER TABLE student ADD ime_majke varchar (20)
ili
ALTER TABLE student ADD ime_majke varchar (20) NOT NULL DEFAULT 'Nije poznato'

Dodavanje novih ograničenja

Kod dodavanja novih ograničenja treba voditi računa da svako ograničenje počinje važiti od trenutka kada je definirano.
Ukoliko se nekoj tablici, koja već sadrži određene podatke (određeni broj redova) želi dodati ograničenje tipa CHECK ili FOREIGN KEY, potrebno je definirati kako se novo ograničenje odnosi prema već postojećim podacima.

Neka je tablica upisni_list definirana kao:

```
CREATE TABLE upisni_list
(     ulist_id        int IDENTITY
      CONSTRAINT PK_UPISNI PRIMARY KEY,
      student_id      int NOT NULL
      CONSTRAINT FK_ST_UPIS FOREIGN KEY REFERENCES student(student_id),
      sem             tinyint NOT NULL,
      sk_god          varchar(8) NOT NULL,
      obr_prog_id     int NOT NULL
      CONSTRAINT FK_OBR_UPIS FOREIGN KEY REFERENCES obr_prog(obr_prog_id)
)
```


Pretpostavimo da su u tablicu uneseni neki upisni listovi, nakon čega primjenimo ograničenje na vrijednosti semestra izrazom:

ALTER TABLE upisni_list ADD CONSTRAINT CHK_SEM CHECK (sem>0 AND sem<6)

Novopostavljeno ograničenje vrijedit će za sve podatke koji se unesu nakon što je ograničenje definirano, ali je pitanje kako se odnosi prema onim redovima u tablici koji su prethodno unešeni.

Djelovanje novog ograničenja na već postojeće podatke određuje se opcijama WITH CHECK i WITH NOCHECK.

Ako se ograničenje definira sa opcijom WITH CHECK, automatski će biti provjereni i svi prethodno uneseni podaci da li zadovoljavaju pravilo.

Primjenom opcije WITH NOCHECK, novo ograničenje primjenit će se samo na one podatke koji se unesu nakon ograničenja, bez provjere prethodno unešenih podataka.

ALTER TABLE upisni_list WITH CHECK ADD CONSTRAINT CHK_SEM CHECK (sem>0 AND sem<6)

142

Brisanje postojećih kolona

Brisanje postojećih kolona u tablici primjenom instrukcije *ALTER TABLE DROP* može se primijeniti samo na one kolone koje ne podliježu nikakvim ograničenjima.

Drugim rječima, nije moguće ukloniti kolone:
- koje čine primarni ključ ili dio primarnog ključa
- koje predstavljaju strani ključ
- koje su vezane ograničenjem tipa CHECK
- jedinstvene kolone (UNIQUE)
- kolone koje imaju određenu default vrijednost
- kolone po kojima postoji indeks.

Za uklanjanje ovakvih kolona, potrebno je prethodno izbrisati sva ograničenja vezana za te kolone.

Promjena svojstava postojećih kolona

Promjena svojstava neke od postojećih kolona vrši se instrukcijom oblika:

ALTER TABLE *ime_tablice*
ALTER COLUMN *naziv_kolone | svojstva*

Na tablici je moguće promijeniti samo svojstva kolone, a ne i pojedina ograničenja.

Kod promjene pojedinih ograničenja, potrebno je najprije ukloniti postojeće ograničenje, a nakon toga definirati novo.

Stoga je instrukcijom ALTER TABLE moguće samo mijenjanje svojstava vezanih za pojedinu kolonu (tip podataka i NULL | NOT NULL svojstvo).

ALTER TABLE *ime_tablice*
ALTER COLUMN *naziv_kolone | svojstva*

Primjenom ove instrukcije nije moguće promijeniti svojstva kolona:
- koje čine primarni ključ ili dio primarnog ključa
- koje predstavljaju strani ključ
- koje su vezane ograničenjem tipa CHECK ili UNIQUE. Jedina promjena koja se može izvršiti jest ako su te kolone definirane kao niz znakova varchar, dozvoljeno je mijenjanje duljine niza.
- kolone po kojima postoji indeks.

d) Brisanje tablice

Tablica se briše naredbom
DROP TABLE *ime_tablice*

145

INDEKSI

Za formiranje indeksa koristi se instrukcija **CREATE INDEX** oblika:
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX ime_indeksa ON ime_tablice(kolona_1,..kolona_n)

UNIQUE – definira jedinstveni indeks,
CLUSTERED | NONCLUSTERED – određuje clustered indeks. Ukoliko se ovaj podatak ne navede indeks se smatra NONCLUSTERED

Primjer:
CREATE UNIQUE INDEX IX_JMBG ON student(JMBG)

Brisanje indeksa vrši se instrukcijom oblika:
DROP INDEX ime_tablice.ime_indeksa

146

UNOS PODATAKA

Za unos podataka u tablicu koristi se SQL izraz oblika:
INSERT [INTO] ime_tablice (kolone)
VALUES (vrijednosti)

INTO - opcijska ključna riječ, može se navesti, ali nije obavezno.

ime_tablice - opisuje tablicu u koju se unose podaci. Tablica prethodno mora biti definirana naredbom CREATE TABLE.

kolone - lista kolona u koje se unose vrijednosti.
Kolone se mogu navesti u proizvoljnom redoslijedu, ali tada i podaci koji se unose (vrijednosti) moraju biti u istom redoslijedu. Navođenje kolona potrebno je samo ako se novi red tablice popunjava sa samo nekim kolonama, a ne svim. Ako se ne navodi lista kolona, podrazumijeva se da se popunjavaju sve kolone.

VALUES - ključna riječ koja se koristi za navođenje vrijednosti koje se unose u novi red tablice.

vrijednosti - vrijednosti koje se unose u novi red tablice.
Tip podataka pojedinih vrijednosti mora odgovarati tipu podataka koji je definiran za pojedinu kolonu.
Za kolone koje imaju definirane default vrijednosti može se koristiti ključna riječ *default*.
Prilikom unosa novog redka u tablicu, ne unose se vrijednosti kojima rukuje sam sustav (IDENTITY kolone).

450 - Baze podataka

SQL – UNOS PODATAKA

Primjer: Unos podataka u tablicu STUDENT koja je definirana:

```
CREATE TABLE student
(    student_id      int IDENTITY
     CONSTRAINT PK_STUDENT PRIMARY KEY,
     ime             varchar(20) NOT NULL,
     prezime         varchar(30) NOT NULL,
     jmbg            char(13),
     spol            char(1),
     dat_rođenja     smalldatetime NOT NULL,
     mjesto_rod      int NOT NULL
     CONSTRAINT FK_MJESTO_ST FOREIGN KEY REFERENCES mjesto(mjesto_id)
)
```

Unos svih kolona tablice:
INSERT STUDENT
VALUES ('Marko', 'Bilić', '1110980370086', 'M', '11.10.1980',11)

Unos sa nazivima kolona:
INSERT STUDENT (spol, mjesto_rod, dat_rođenja, ime, prezime)
VALUES ('M' , 11 , '11.10.1980', 'Marko', 'Bilić')

149

450 - Baze podataka

SQL – UNOS PODATAKA

Prilikom unosa podataka u novi redak tablice, moguće su različite situacije i ponašanja sustava u zavisnosti o tome kako su definirane pojedine kolone, te njihova null i default svojstva.

svojstvo kolone	**nema ulaza, nema default**	**nema ulaza, postoji default**	**ulaz null nema default**	**ulaz null postoji default**
null	null	Default	null	null
not null	greška – error	Default	greška- error	greška- error

U tablici su prikazana ponašanja sustava za pojedine situacije.

150

SELEKCIJA PODATAKA

Selekcija – dobavljanje podataka iz baze podataka, vrši se SQL izrazom oblika**:**

SELECT [ALL | DISTINCT] imena_kolona
FROM imena_tablica|logičke veze
WHERE uvjetni_izraz

pri čemu ***ALL*** daje sve redove izlaznog rezultata prema zahtjevu izraza, a ***DISTINCT*** daje samo jedinstvene redove, eliminirajući duplikate.
Ukoliko se u select izrazu ne navede niti jedan od modifikatora ALL | DISTINCT, podrazumijeva se ALL.

151

imena_kolona
Definira kolone koje se prikazuju.
- Zvjezdica (*), predstavlja sve kolone po redoslijedu navedenom kod formiranja tablice (CREATE TABLE).
- Ako lista *imena_kolona* sadrži više kolona, nazivi kolona se međusobno odvajaju zarezom.

Primjer:

SELECT * FROM STUDENT

SELECT ime, prezime, jmbg FROM STUDENT

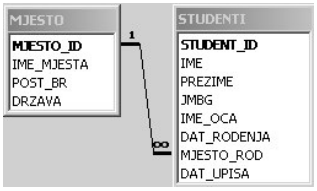
SELECT DISTINCT ime, prezime FROM STUDENT

152

imena_tablica* | *logičke veze
nazivi tablica iz kojih se dobavljaju podaci.
Tablice moraju prethodno biti definirane u bazi podataka.

Logičke veze opisuju operacije prirodnog spajanja među tablicama (*inner join, outer join* itd).

Realizacija logičkih veza podrazumijeva navođenje imena tablica i načina njihovog povezivanja (join) u obliku:
tablica1 operator_veze tablica2 ON tablica1.kolona1=tablica2.kolona1



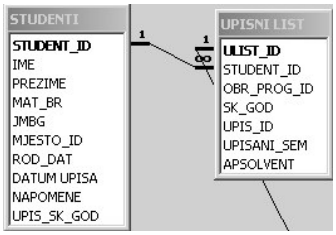
***Primjer**:*

```
SELECT  STUDENT.IME, STUDENT.PREZIME,
        IME_MJESTA
FROM    STUDENT INNER JOIN MJESTO
        ON STUDENT.MJESTO_ROD = MJESTO.MJESTO_ID
```

153

Prilikom definiranja select izraza koji obuhvaća atribute (kolone) iz više tablica, treba voditi računa o jedinstvenosti naziva izabranih atributa.

To znači da, ukoliko se navode kolone iz više tablica koje imaju isti naziv, uz naziv kolone treba definirati iz koje tablice se selektira tražena kolona.

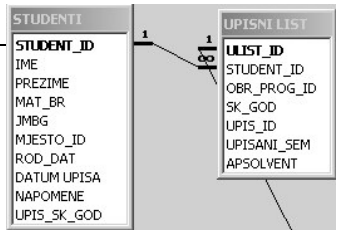


Primjer:

```
SELECT  STUDENT_ID, IME, PREZIME,
        SEM, SK_GOD
FROM    STUDENT INNER JOIN UPISNI_LIST
        ON STUDENT.STUDENT_ID = UPISNI_LIST.STUDENT_ID
```

154

450 - Baze podataka



Primjer sa prethodne str.:

```
SELECT  STUDENT_ID, IME, PREZIME,
        SEM, SK_GOD
FROM    STUDENT INNER JOIN UPISNI_LIST
        ON STUDENT.STUDENT_ID = UPISNI_LIST.STUDENT_ID
```

Ovako definirana naredba SELECT nije ispravna budući se atribut (kolona) *STUDENT_ID* nalazi u obje tablice (*STUDENT* i *UPISNI_LIST*), pa time nije jednoznačno određeno po kojem se atributu vrši selekcija.

Ispravna definicija izraza bila bi:

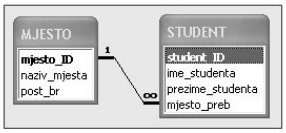
```
SELECT STUDENT.STUDENT_ID, IME, PREZIME, SEM, SK_GOD
FROM   STUDENT INNER JOIN UPISNI_LIST
       ON STUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID
```

155

450 - Baze podataka

SQL – SELEKCIJA PODATAKA

Primjer:



MJESTO : Table

	mjesto_ID	naziv_mjesta	post_br
+	1	Zagreb	10000
+	2	Split	21000
+	3	Zadar	23000
+	4	Dubrovnik	20000

STUDENT : Table

student_ID	ime_studenta	prezime_studer	mjesto_preb
1	Mate	Matić	1
2	Jure	Jurić	2
3	Maja	Majić	2
4	Nina	Ninić	

```
SELECT  STUDENT.ime_studenta,
        STUDENT.prezime_studenta,
        MJESTO.naziv_mjesta
FROM    MJESTO INNER JOIN STUDENT ON
MJESTO.mjesto_ID = STUDENT.mjesto_preb
```

inner join – unutarnja veza

Mate	Matić	Zagreb
Jure	Jurić	Split
Maja	Majić	Split

left join – lijeva veza

Mate	Matić	Zagreb
Jure	Jurić	Split
Maja	Majić	Split
		Zadar
		Dubrovnik

right join – desna veza

Nina	Ninić	
Mate	Matić	Zagreb
Jure	Jurić	Split
Maja	Majić	Split

outer join – vanjska veza

		Dubrovnik
		Zadar
Jure	Jurić	Split
Maja	Majić	Split
Mate	Matić	Zagreb
Nina	Ninić	

156

Uvjetni izrazi

SELECT lista_kolona FROM ime_tablice WHERE uvjetni izraz

Oblik uvjetnog izraza ima oblik:

Naziv_kolone *Predikat* Vrijednost

Predikati usporedbe (Primjenjivo na sve tipove podataka)

= jednako	<> različito od
> veće od	!> nije veće
< manje od	!< nije manje
>= veće ili jednako	
<= manje ili jednako	

Primjer:
SELECT * FROM STUDENT WHERE IME = 'Marko'

157

Predikati područja

BETWEEN x and y - atribut zauzima vrijednosti između x i y. Primjenjivo na sve tipove podataka.

NOT BETWEEN x and y - atribut zauzima vrijednosti izvan opsega između x i y. Primjenjivo na sve tipove podataka.

Primjer:
SELECT * FROM MJESTO
WHERE POST_BR between 20000 and 30000

SELECT * FROM MJESTO
WHERE POST_BR NOT between 20000 and 30000

158

Predikati liste

naziv_kolone IN (lista vrijednosti) – vrijednost atributa jednaka je nekoj od vrijednosti navedenoj u listi. Primjenjivo na sve tipove podataka.

Primjer:
SELECT * FROM MJESTO
WHERE IME_MJESTA IN ('Split', 'Trogir', 'Makarska')

što je jednako kao:

SELECT * FROM MJESTO
WHERE (IME_MJESTA= 'Split') OR (IME_MJESTA= 'Trogir')
OR (IME_MJESTA= 'Makarska')

naziv_kolone NOT IN (lista vrijednosti) – atribut ima bilo koju vrijednost osim vrijednosti navedenih u listi. Primjenjivo na sve tipove podataka.

Primjer:
SELECT * FROM MJESTO
WHERE IME_MJESTA NOT IN ('Split', 'Trogir','Makarska')

159

Predikati uzorka

naziv kolone LIKE uzorak - vrijednost atributa poklapa se sa navedenim uzorkom. Primjenjivo na tekstualne i datum tipove podataka.

Vrste uzoraka:
% - niz karaktera bilo koje duljine.
_ - Bilo koji pojedinačni karakter
[ ] - Bilo koji karakter u navedenim granicama

Primjeri:

SELECT * FROM STUDENT WHERE PREZIME Like 'A%'
Prikazuje sve studente kojima prezime počinje sa slovom A.

SELECT * FROM STUDENT WHERE IME Like '_arko'
Prikazuje sve studente kojima se u imenu nalazi 'arko', a ispred je bilo koje slovo.npr. 'Marko','Darko','Žarko'.

SELECT * FROM STUDENT WHERE PREZIME Like '[P-Z]ilić'
Prikazuje sve studente kojima prezime počinje sa slovom između P i Z, a u nastavku se nalazi 'ilić'.

160

Predikati nepoznate vrijednosti

IS NULL
IS NOT NULL

Primjer:

```
SELECT IME, PREZIME, SEM
FROM STUDENT LEFT OUTER JOIN UPISNI_LIST
        ON STUDENT.STUDENT_ID = UPISNI_LIST.STUDENT_ID
WHERE SEM IS NULL
```

LOGIČKI OPERATORI - AND, OR, NOT

Logički operatori AND i OR povezuju više uvjeta u uvjetnom izrazu.

Logički izraz AND povezuje dva ili više uvjeta i vraća rezultat samo ako su svi uvjeti zadovoljeni (istiniti).

Primjer:

```
SELECT IME, PREZIME, SEM
FROM STUDENT INNER JOIN UPISNI_LIST
ON STUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID
WHERE SK_GOD='2004/05' AND SEM=1
```


Logički izraz OR povezuje dva ili više uvjeta i vraća rezultat ako je bilo koji uvjet zadovoljeni (istinit).

Primjer:

```
SELECT * FROM MJESTO
WHERE IME_MJESTA= 'Split' OR IME_MJESTA= 'Trogir'
        OR IME_MJESTA= 'Makarska'
```

Logički izraz NOT negira uvjet koji mu slijedi.

Primjer:

```
SELECT * FROM MJESTO
WHERE NOT IME_MJESTA = 'Split'
```

PRIORITET LOGIČKIH OPERATORA

Kada se u uvjetnom dijelu izraza pojavljuje više logičkih operatora, logički izraz formira se prema prioritetu djelovanja operatora

NOT- najviši prioritet
AND -srednji prioritet
OR - najniži prioritet.

Primjer:

```
SELECT IME, PREZIME
FROM STUDENT INNER JOIN UPISNI_LIST
        ON STUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID
WHERE SK_GOD='2004/05' OR ŠK_GOD='2005/06' AND SEM=1.
```

Kao rezultat dobije se lista svih studenata koji su upisani u 1. semestar šk.godine '2005/06' i onih koji su upisani u šk.godinu '2004/05' (bilo koji semestar).

Prioritet logičkih operatora, može se promijeniti uvođenjem zagrada () u logički izraz.

Primjer:

```
SELECT IME, PREZIME
FROM STUDENT INNER JOIN UPISNI_LIST
        ON STUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID
WHERE (SK_GOD='1999/00' OR ŠK_GOD='1998/99') AND SEM=1.
```


OBLIKOVANJE IZLAZNIH REZULTATA

Preimenovanje kolona u rezultatu

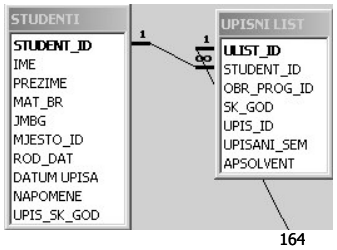
Kod postavljanja upita (*SELECT*) kojim se dohvaćaju podaci, izlazna lista (rezultat) formira se na način da naziv kolone koja se dohvaća postaje zaglavlje te kolone.

Primjer:

```
SELECT PREZIME, IME, UPISANI_SEM
FROM   STUDENT INNER JOIN UPISNI_LIST
    ON STUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID
WHERE SK_GOD='2004/05'
```

Rezultat:

PREZIME	IME	UPISANI_SEM
Matić	Ivo	1
Bilić	Ante	3
Alić	Josip	1



Za postizanje izlazne liste rezultata, koja je korisniku razumljivija, može se primijeniti preimenovanje izlaznih rezultata u obliku:
zaglavlje_kolone=naziv_kolone
u sklopu select izraza.

Primjer:

```
SELECT Prezime_studenta=PREZIME, Ime_studenta=IME,
        Upisani_semestar=UPISANI_SEM
FROM STUDENT INNER JOIN UPISNI_LIST
      ON STUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID
WHERE SK_GOD='2004/05'
```

Prezime_studenta	Ime_studenta	Upisani_semestar
Matić	Ivo	1
Bilić	Ante	3
Alić	Josip	1

165

Zamjena podataka u rezultatu upita

Proširimo tablicu STUDENT atributom koji opisuje status studenta (R- redovni student, I - izvanredni student)

```
ALTER TABLE student ADD tip_studija char (1) NOT NULL
     CONSTRAINT CK_2 CHECK (tip_studija='R' or tip_studija ='I')
```

U tablicu STUDENT uvodimo novi atribut (kolonu), koja označava tip studiranja, te može poprimiti samo vrijednosti 'R' ili 'I', kao oznaka za redovne ili izvanredne studente.

Upit
```
SELECT IME, PREZIME, TIP_STUDIJA
FROM STUDENT
```
daje rezultat u kojem se u koloni tip_studija nalaze znakovi 'R' ili 'I' kao oznaka tipa studiranja.

IME	PREZIME	TIP_STUDIJA
Ante	Matić	R
Josip	Matić	R
Željko	Celić	I
Tomislav	Pavlinović	R
Marino	Kaštelan	I

Kako bi se ove kratice učinile razumljivijim mogu se u izlaznom rezultatu dodijeliti puni nazivi korištenjem **CASE** izraza.

166

```
SELECT IME, PREZIME,
    STUDIJ=CASE TIP_STUDIJA
                WHEN 'R' THEN
'Redovni'
                WHEN 'I' THEN 'Izborni'
            END
FROM STUDENT.
```

IME	PREZIME	STUDIJ
Ante	Matić	Redovni
Josip	Matić	Redovni
Željko	Celić	Izvanredni
Tomislav	Pavlinović	Redovni
Marino	Kaštelan	Izvanredni

Proširimo prethodni upit:

```
SELECT IME, PREZIME, UPISANI_SEM, TIP_STUDIJA
FROM STUDENT INNER JOIN UPISNI_LIST
      ON TUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID
WHERE SK_GOD='2004/05' AND UPISANI_SEM IN (1,3,5)
```

Ovaj upit se može pomoću naredbe CASE prikazati u pogodnijem obliku *(iduća stranica).*

IME	PREZIME	UPISANI_SEM	TIP_STUDIJA
Lovre	Kapetanović	1	R
Ivica	Lovrić	1	I
Jakov	Pivac	3	I
Damir	Lučić	3	R
Damir	Krezić	5	R
Ante	Turudić	5	R

167

IME	PREZIME	STATUS
Lovre	Kapetanović	Redovni student 1. godine
Ivica	Lovrić	Izvanredni student 1. godine
Jakov	Pivac	Izvanredni student 2. godine
Damir	Lučić	Redovni student 2. godine
Damir	Krezić	Apsolvent
Ante	Turudić	Apsolvent

```
SELECT IME, PREZIME
    STATUS=CASE
            WHEN TIP_STUDIJA='R' AND UPISANI_SEM=1
                THEN 'Redovni student 1. godine'
            WHEN TIP_STUDIJA='R' AND UPISANI_SEM=3
                THEN 'Redovni student 2. godine'
            WHEN TIP_STUDIJA='I' AND UPISANI_SEM=1
                THEN 'Izvanredni student 1. godine'
            WHEN TIP_STUDIJA='I' AND UPISANI_SEM=3
                THEN 'Izvanredni student 2. godine'
            ELSE 'Apsolvent'
            END
    FROM STUDENT INNER JOIN UPISNI_LIST
        ON STUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID WHERE
SK_GOD='2004/05'AND SEM IN (1,3,5)
```

168

Izračunate kolone (Computed values)

SQL dozvoljava formiranje novih kolona u select izrazu, čije se vrijednosti izračunavaju na temelju podataka koji se nalaze u numeričkim kolonama postojećih tablica.

IME	PREZIME	UPISANI_SEM	GODINA_STUDIJA
Lovre	Kapetanović	1	1
Ivica	Lovrić	1	1
Jakov	Pivac	3	2
Damir	Lučić	3	2
Damir	Krezić	5	3
Ante	Turudić	5	3

Primjer:
Prikaz upisnih listova vezanih za pojedine studente uz označenu godinu studiranja.

```
SELECT IME, PREZIME, UPISANI_SEM,
        GODINA_STUDIJA= UPISANI_SEM/2+ UPISANI_SEM%2
FROM STUDENT INNER JOIN UPISNI_LIST
ON STUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID
        WHERE SK_GOD='2004/05' AND UPISANI_SEM IN (1,3,5)
```

Pri izračunavanju vrijednosti u pojedinim kolonama mogu se koristiti vrijednosti postojećih kolona za koje su definirani numerički tipovi podataka i neki od numeričkih operatora:
+,-,*,/, %(modulo operator – ostatak pri cjelobrojnom dijeljenju).

169

SORTIRANJE IZLAZNIH REZULTATA

Za sortiranje izlaznih rezultata koristi se izraz **ORDER BY**:

```
SELECT [ALL | DISTINCT] nazivi_kolona
FROM     nazivi_tablica|logičke veze
WHERE uvjetni izraz
ORDER BY kolone_sortiranja
```

Sintaksa:
ORDER BY naziv_kolone tip_sortiranja, naziv_kolone tip_sortiranja,...

Sortiranje može biti rastuće (ascending ASC) ili padajuće (descending DESC). Ako nije naveden tip sortiranja podrazumijeva se rastuće (ASC) sortiranje.

Primjer:

```
SELECT * FROM STUDENT ORDER BY PREZIME, IME
```

Umjesto naziva kolona u izrazu order by može se koristiti redni broj kolone u select listi.

STUDENT_ID	IME	PREZIME	JMBG	DAT_RODENJA
1	Ante	Ligutić	1310975380001	1975-10-13
2	Igor	Grbin	1802976380013	1976-02-18

Primjer:

```
SELECT * FROM STUDENT ORDER BY 3 ASC, 2
```

170

OGRANIČAVANJE ISPISA REZULTATA

Upit tipa:
SELECT [ALL | DISTINCT] ***imena_kolona***
FROM ***imena_tablica|logičke veze***
WHERE ***uvjetni_izraz***
prikazuje sve rezultate iz izabranih tablica koji zadovoljavaju uvjetni izraz.

U slučaju potrebe za prikazom samo dijela izlaznih rezultata ispis se može ograničiti primjenom sintakse:
SELECT [ALL | DISTINCT] [TOP n [PERCENT]]
imena_kolona
FROM ***imena_tablica|logičke veze***
WHERE ***uvjetni_izraz***
ORDER BY kolone_sortiranja
pri čemu je select izraz proširen sa **TOP n** ili **TOP n PERCENT**.

Modifikator TOP označava da će u ispisu rezultata biti prikazano:
- samo prvih n redova (TOP n)
- odnosno samo n% (TOP n PERCENT)

od ukupnog broja redova koji zadovoljavaju postavljeni uvjet.

Primjer:

```
SELECT TOP 50 PREZIME, IME, JMBG
FROM   STUDENT
ORDER BY PREZIME, IME
```

Iz ukupnog skupa studenata prikazuje samo prvih 50 sortirano po prezimenu i imenu.

```
SELECT TOP 50 PERCENT PREZIME, IME, JMBG
FROM   STUDENT
ORDER BY PREZIME, IME
```

Iz ukupnog skupa studenata prikazuje prvu polovicu liste (50%) u uređenom sortiranju po prezimenu i imenu.

ALIAS-i

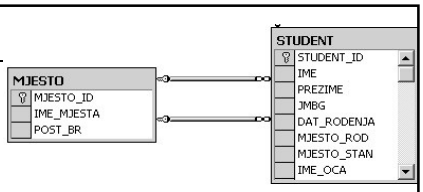
Alias (drugo ime) koristi se u upitima SQL jezika za promjenu naziva objekta (tablice) kojoj se pristupa.

Svakoj tablici može se dodijeliti alias, bilo zbog lakše upotrebe ili razlikovanja kod tzv. self-join query postupaka (*upit nad samim sobom*).

Sintaksa: ***Naziv_tablice alias***
ili ***Naziv_tablice AS alias***

Primjer:
Ispisati sve parove studenata koji su rođeni istog datuma:

```
SELECT STUDENT_1=STUDENT_1.PREZIME +' '+ STUDENT_1.IME,
       STUDENT_2=STUDENT_2.PREZIME +' ' + STUDENT_2.IME
FROM  STUDENT STUDENT_1 INNER JOIN STUDENT STUDENT_2
       ON STUDENT_1.DATUM=STUDENT_2.DATUM
WHERE STUDENT_1.STUDENT_ID>STUDENT_2.STUDENT_ID
```

Primjer:
U tablici STUDENT se vode podaci o mjestu rođenja i o mjestu stanovanja. Ta dva podatka (*mjesto_rod i mjesto_stan*) su FK iz tablice MJESTO.
Zadatak je ispisati nazive mjesta rođenja i mjesta stanovanja:

IME	PREZIME	MAT_BR	mjesto_stan	mjesto_rod
Joško	Akrap	548-2001	831	769
Robert	Alagušić	32	8384	790
Mladen	Alajbeg	11	785	785
Mijo	Alerić	10006	9252	813
Ante	Alfirević	10221	769	769

```
SELECT STUDENT.IME, STUDENT.PREZIME, STUDENT.MAT_BR,
       MJESTO.IME_MJESTA AS 'Stanuje u',
       MJESTO1.IME_MJESTA AS 'Roden u'
FROM STUDENT INNER JOIN MJESTO
       ON STUDENT.MJESTO_STAN = MJESTO.MJESTO_ID
       INNER JOIN MJESTO MJESTO1
       ON STUDENT.MJESTO_ROD = MJESTO1.MJESTO_ID
```

IME	PREZIME	MAT_BR	Stanuje u	Roden u
Joško	Akrap	548-2001	Stobreč	Split
Robert	Alagušić	32	Radošić	Sinj
Mladen	Alajbeg	11	Trogir	Trogir
Mijo	Alerić	10006	Runović	Imotski
Dragan	Alfirević	682-2001	Kaštel Sućurac	Split

AGREGATNE FUNKCIJE

Agregatne funkcije generiraju zbirne (skupne) vrijednosti nad pojedinim kolonama tablice.

Agregatne funkcije (SUM, AVG, MIN, MAX, COUNT) koriste se u SELECT izrazu, te mogu biti primjenjene nad svim redovima tablice ili samo nad onim redovima koji su definirani WHERE izrazom.

Značenje agregatnih funkcija

SUM – zbroj **AVG -** prosjek **MIN** - minimum **MAX** - maximum
COUNT – pobrojavanje

Agregatne funkcije SUM i AVG mogu se koristiti samo za numeričke tipove podataka.

Sintaksa:
SELECT agregate(naziv_kolone), agregate(naziv_kolone)
FROM tablice
WHERE uvjetni izraz

Primjeri za agregatne funkcije:

```
SELECT COUNT(ULIST_ID)
FROM UPISNI_LIST
WHERE šk_god='2004/05' and sem=1

SELECT MAX(UPISANI_SEM)
FROM UPISNI_LIST INNER JOIN STUDENT
        ON UPISNI_LIST.STUDENT_ID=STUDENT.STUDENT_ID
WHERE ime='Meri' and prezime='Merić'
```

SELECT INTO (MAKE TABLE QUERY)

SELECT [ALL | DISTINCT] nazivi_kolona
INTO nova_tablica
FROM nazivi_tablica
WHERE uvjetni izraz

Izraz **select into** stvara novu tablicu na osnovi:
- kolona navedenih u *nazivi_kolona* i
- redova koji se biraju uvjetnim izrazom iz postojećih tablica u bazi.

Podatak *nova_tablica* predstavlja ime nove tablice, koja će se formirati iz rezultata upita.
Ta tablica ne smije prethodno biti definirana u bazi.

SELECT INTO je operacija koja se odvija u dva koraka:
- u prvom koraku formira se nova tablica,
- drugim korakom u novu tablicu se unose redovi koji zadovoljavaju uvjetni izraz.

Ako se **drugi korak ne izvrši** iz bilo kojeg razloga, nova tablica ostaje ali u njoj nema nikakvih podataka.

Ovim izrazom omogućeno je **automatsko formiranje nove tablice** bez njezinog prethodnog definiranja naredbom CREATE TABLE.

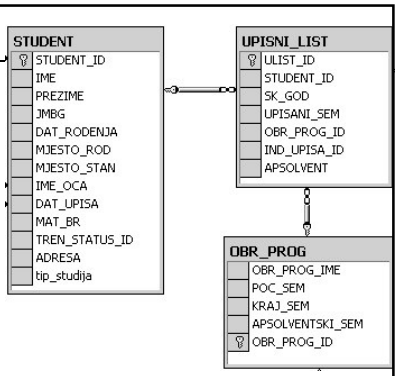
Definicija podataka u novoj tablici formira se automatski i ona je identična definiciji pojedinih atributa(kolona) u postojećim tablicama, koje su obuhvaćene SELECT izrazom.

450 - Baze podataka

Primjer:

Na temelju strukture podataka u bazi studentske službe potrebno je formirati novu tablicu koja će sadržavati podatke o studentima koji su upisani u šk. god. 2005/06.

Tablica mora sadržavati slijedeće podatke: Ime, Prezime i Matični broj studenta, upisani semestar, obrazovni program i školsku godinu.



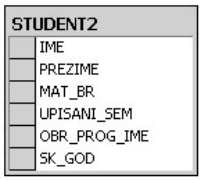

```
SELECT IME, PREZIME, MAT_BR, UPISANI_SEM, OBR_PROG_IME, SK_GOD
FROM STUDENT INNER JOIN UPISNI_LIST
        ON STUDENT.STUDENT_ID = UPISNI_LIST.STUDENT_ID
        INNER JOIN OBR_PROG ON OBR_PROG.OBR_PROG_ID = UPISNI_LIST.OBR_PROG_ID
WHERE SK_GOD='2001/02'
```

	IME	PREZIME	MAT_BR	UPISANI_SEM	OBR_PROG_IME	SK_GOD
1	Dejan	Varešković	360-2001	1	Elektrotehnika 6. stupanj	2001/02
2	Dejan	Varešković	360-2001	2	Elektrotehnika 6. stupanj	2001/02
3	Siniša	Spahija	364-2001	1	Elektrotehnika 6. stupanj	2001/02
4	Siniša	Spahija	364-2001	2	Elektrotehnika 6. stupanj	2001/02
5	Jure	Šerić	365-2001	1	Elektrotehnika 6. stupanj	2001/02

450 - Baze podataka

SQL – SELECT

```
SELECT IME, PREZIME, MAT_BR, UPISANI_SEM, OBR_PROG_IME,
        SK_GOD
INTO STUDENT2
FROM STUDENT INNER JOIN UPISNI_LIST
                ON STUDENT.STUDENT_ID = UPISNI_LIST.STUDENT_ID
        INNER JOIN OBR_PROG ON OBR_PROG.OBR_PROG_ID =
                UPISNI_LIST.OBR_PROG_ID
WHERE SK_GOD='2001/02'
```




```
SELECT * FROM STUDENT2
```

	IME	PREZIME	MAT_BR	UPISANI_SEM	OBR_PROG_IME	SK_GOD
1	Dejan	Varešković	360-2001	1	Elektrotehnika 6. stupanj	2001/02
2	Dejan	Varešković	360-2001	2	Elektrotehnika 6. stupanj	2001/02
3	Siniša	Spahija	364-2001	1	Elektrotehnika 6. stupanj	2001/02
4	Siniša	Spahija	364-2001	2	Elektrotehnika 6. stupanj	2001/02
5	Jure	Šerić	365-2001	1	Elektrotehnika 6. stupanj	2001/02

APPEND UPITI (proširivanje tablice podacima iz drugih tablica)

Upitima tipa Append (dodavanje) u postojeću tablicu u bazi podataka dodijeljuju se podaci iz drugih tablica prema zadanim uvjetima.

Standardna SQL sintaksa ovih izraza je oblika:

INSERT *ime_tablice*
SELECT *nazivi_kolona*
FROM *tablice*|*veze*
WHERE *uvjetni_izraz*

gdje *ime_tablice* predstavlja naziv neke od postojećih tablica u bazi podataka.

181

Primjer:

Dobijenu tablicu STUDENT2 u prethodnom zadatku treba proširiti sa podacima za školsku godinu 2002/03.

Nakon što je problem riješen za školsku godinu 2001/02 prethodnim zadatkom, u bazi smo dobili novu tablicu STUDENT2, koja je definirana: head(STUDENT2)= {IME, PREZIME,MAT_BR, UPISANI_SEM,OBR_PROG_IME, SK_GOD}

Postojeće podatke u tablici potrebno je proširiti sa onima koji se odnose na školsku godinu 2002/03.

```
INSERT STUDENT2
SELECT IME, PREZIME, MAT_BR, UPISANI_SEM, OBR_PROG_IME,
       SK_GOD
FROM STUDENT INNER JOIN UPISNI_LIST
                ON STUDENT.STUDENT_ID = UPISNI_LIST.STUDENT_ID
       INNER JOIN OBR_PROG ON OBR_PROG.OBR_PROG_ID =
                UPISNI_LIST.OBR_PROG_ID
WHERE SK_GOD='2002/03'
```

182

AŽURIRANJE PODATAKA U TABLICI

Ažuriranje podataka u tablici provodi se naredbom UPDATE prema sintaksi:

UPDATE *naziv_tablice*
SET
***naziv_kolone* = *nova vrijednost*,**
***naziv_kolone2* = *nova_vrijednost2*, …**
[FROM *tablica_1, tablica_2,...*]
WHERE *uvjetni izraz*

Primjeri:

```
UPDATE STUDENT
SET  ime = 'Marko'
WHERE ime = 'Darko' and prezime = 'Darkić'
```

```
UPDATE UPISNI_LIST
SET sem = 3
FROM STUDENT, UPISNI_LIST
WHERE  ime = 'Darko' AND  prezime = 'Darkić'
        AND STUDENT.student_id = UPISNI_LIST.student_id
        AND  sem = 1
```

```
UPDATE STUDENT
SET mjesto_id = MJESTO.mjesto_id
FROM MJESTO, STUDENT
WHERE MJESTO.mjesto_id = STUDENT.mjesto_id
        MJESTO.ime_mjesta = 'Trogir'
        and student_id = 21
```

BRISANJE PODATAKA IZ TABLICE

Brisanje podataka (zapisa, redova) iz tablice provodi se naredbom DELETE prema sintaksi:

DELETE *naziv_tablice*
[FROM *tablica_1, tablica_2,...*]
WHERE *uvjetni izraz*

Primjeri:

```
DELETE  STUDENT
WHERE ime='Marko' and prezime='Markić'

DELETE UPISNI_LIST
FROM  STUDENT INNER JOIN UPISNI_LIST
        ON STUDENT.STUDENT_ID=UPISNI_LIST.STUDENT_ID
WHERE  ime='Marko' AND  prezime= 'Markić'
        AND  sem=1
```

185

Grupni upiti

Određuje grupe po kojima se dijele izlazni rezultati:

SELECT [ALL|DISTINCT] nazivi kolona, agregatne kolone
FROM nazivi_tablica| join veze
[WHERE uvjetni izraz]
[GROUP BY nazivi kolona]
[HAVING uvjetni izraz]
[ORDER BY sort_izraz]

HAVING uvjetni izraz je ograničenje izlaznih rezultata po zadanom uvjetu nakon izvršenja GROUP BY operacije.

Osnovna razlika između WHERE i HAVING izraza:
- WHERE izraz ograničava izlazne rezultate prije njihovog grupiranja tj. uvjetuje broj redova koji ulaze u grupiranje
- HAVING izraz postavlja uvjet na izlazne rezultate nakon što je grupiranje završeno.

186

Primjeri:

```
SELECT STUDENT.student_id, ime, prezime,
       Broj_semestara=COUNT(upisani_sem), sk_god
FROM   STUDENT INNER JOIN UPISNI_LIST
       ON STUDENT.student_id = UPISNI_LIST.student_id
GROUP BY STUDENT.student_id, ime, prezime, sk_god
```

	student_id	ime	prezime	Broj_semestara	sk_god
1	3	Josip	Matić	1	2003/04
2	18789	Marin	Batina	2	2002/03
3	18790	Hrvoje	Bitanga	2	2002/03
4	18797	Ante	Matić	21	2002/03

```
SELECT STUDENT.student_id, ime, prezime,
       Broj_semestara=COUNT(upisani_sem), sk_god
FROM   STUDENT INNER JOIN UPISNI_LIST
       ON STUDENT.student_id = UPISNI_LIST.student_id
GROUP BY STUDENT.student_id, ime, prezime, sk_god
HAVING sk_god > '2002/03'
```

	student_id	ime	prezime	Broj_semestara	sk_god
1	3	Josip	Matić	1	2003/04

187

Ugnježđeni upiti - podupiti

Podupit (Subquery) je SELECT izraz uključen u glavni SELECT (INSERT, UPDATE ili DELETE) upit.
Podupit i glavni upit mogu djelovati ne istu tablicu ili na različite tablice.

Mnogi SQL izrazi koji uključuju podupite mogu se realizirati i stvaranjem upita sa povezivanjem (join).

SELECT [ALL | DISTINCT] nazivi kolona, agregatne kolone
FROM nazivi_tablica| join veze
[WHERE naziv_kolone predikat (subquery)]

Postoje tri osnovna oblika oblika podupita (subqueries):
-podupiti liste na koje se primijenjuje IN predikat - vraća niz rezultata (listu), koju glavni upit koristi u svom uvjetnom izrazu preko predikata liste (IN ili NOT IN).
- podupiti sa predikatom usporedbe - rezultat podupita može se u glavni upit uključiti preko jednog od predikata usporedbe (=,<>,<,....)
- podupiti sa predikatom postojanja - Operator EXISTS ispituje postojanje podataka uz navedene uvjete

188

Podupiti liste

Primjer: Prikazati studente upisane u 1. sem šk.god 2006/07:

```
SELECT student_id, ime, prezime
FROM   STUDENT
WHERE          student_id IN
       (SELECT student_id
       FROM   UPISNI_LIST
       WHERE sk_god = '2006/07'
              AND sem = 1)
ORDER BY prezime, ime
```

Izraz se izvršava u dva koraka.

Prvo podupit iz tablice upisni_list vraća student_id za sve studente koji imaju upisni list u 1. sem školske godine 2006/07. Rezultat je lista sa podacima student_id.

Potom glavni upit iz tablice student vraća podatke o svim studentima čiji se student_id nalazi u listi koristeći predikat liste (IN).

Isti problem može se realizirati i primjenom veze (join) između tablica STUDENT i UPISNI_LIST.

```
SELECT STUDENT.student_id, ime, prezime
FROM   STUDENT INNER JOIN UPISNI_LIST
       ON STUDENT.student_id = UPISNI_LIST.student_id
WHERE sk_godine = '2006/07'
       AND sem = 1
ORDER BY prezime, ime
```

189

Podupiti sa predikatom usporedbe

Podupit mora vratiti samo jednu vrijednost; ukoliko podupit vraća više od jedne vrijednosti javlja se greška u izvršavanju glavnog upita.

Primjer: Prikazati sve studente koji su rođeni u istom mjestu kao i student 'x y'.

```
SELECT student_id, ime, prezime
FROM   STUDENT
WHERE          mjesto_id =
       (SELECT mjesto_id
       FROM   STUDENT
       WHERE ime = 'X'
              AND prezime = 'Y')
ORDER BY prezime, ime
```

Primjer: Prikazati sve upisne listove za studenta 'x y' osim posljednjega.

```
SELECT  sk_god, sem
FROM    UPISNI_LIST
WHERE   student_id =
        (SELECT student_id
        FROM    STUDENT
        WHERE   ime = 'X'
                AND prezime = 'Y')
        AND sem <
        (SELECT MAX(sem)
        FROM    UPISNI_LIST
        WHERE   student_id =
                ( SELECT student_id
                FROM STUDENT
                WHERE ime = 'X'
                AND prezime = 'Y'));
```

190

Podupiti sa predikatom postojanja

Operator EXISTS ispituje postojanje podataka uz navedene uvjete. Podupit sa predikatom postojanja ne vraća nikakve podatke, već daje samo rezultat TRUE ili FALSE.

Primjer: Prikazati sve studente koji nemaju niti jedan upisni list.

```
SELECT student_id, ime, prezime
FROM   STUDENT
WHERE NOT EXISTS     (SELECT *
                      FROM   UPISNI_LIST
                      WHERE STUDENT.student_id =
                                          UPISNI_LIST.student_id)
```

Primjer: Prikazati sva mjesta u kojima je rođen neki student.

```
SELECT *
FROM   MJESTO
WHERE EXISTS (SELECT *
              FROM   STUDENT
              WHERE MJESTO.mjesto_id = STUDENT.mjesto_id)
```

191