Projektovanje informacionih sistema

- Faze projektovanja IS
- Pojam baze podataka
- Sistem za upravljanje bazom podataka
- Normalizacija relacione baze podataka

Faze projektovanja IS

- **Faza konfiguracije:**
  - Identifikuje alternativna rešenja, analizira njihova izvodljivost i preporučuje globalno rešenje sistema.
  - Za svako alternativno rešenje treba da se odrede zahtevani hardver i softver.
  - Kod analize izvodljivosti mnogi analitičari imaju u vidu sledeće kriterijume:
    - **Tehnička izvodljivost**. Da li je rešenje tehnički praktično? Da li je osoblje tehnički osposobljeno da projektuje i izgrađuje to rešenje?
    - **Operaciona izvodljivost**. Da li će rešenje ispuniti korisničke zahteve? Do kog stepena? Kako će rešenje promeniti korisničko radno okruženje? Šta korisnici misle o takvom rešenju?
    - **Ekonomska izvodljivost**. Da li je rešenje troškovno efektivno?
    - **Izvodljivost programa.** Da li rešenje može biti projektovano i implementirano u toku prihvatljivog vremenskog perioda?
  - Analiza troškova i koristi se diskutuje sa vlasnicima i korisnicima rešenja.
  - Rangiraju se sva alternativna rešenja prema kriterijumu izvodljivosti.
  - Pismeni izveštaj koji sadrži analize i preporuke najboljeg rešenja.

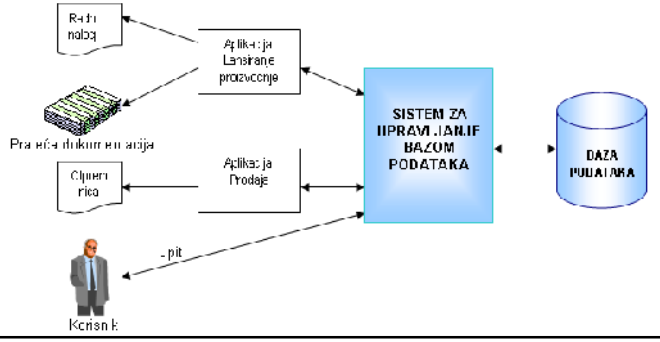
2

Faze projektovanja IS *(nastavak)*

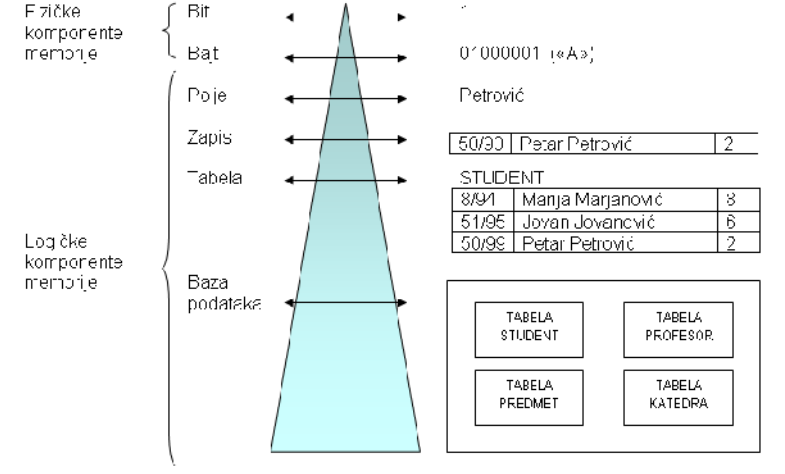
- U toku **faze nabavke** vrši se izbor odgovarajućeg hardvera i/ili softvera za novi sistem. Osnovni ciljevi ove faze su:
  - Identifikovati i ispitati hardver i softver koji može da podrži preporučeno rešenje.
  - Potražiti, proceniti i rangirati ponude dobavljača.
  - Izabrati i preporučiti najbolju ponudu dobavljača.
  - Odrediti zahteve za integracijom dodeljenih proizvoda dobavljača sa drugim proizvodima sistema.
- U okviru **faze projektovanja i integracije** razvija se tehničko rešenje sa kojim će moći da se konstruiše sistem. Projektant sistema mora da obavi sledeće aktivnosti:
  - Analizira i distribuira podatke.
  - Analizira i distribuira procese.
  - Projektuje bazu podataka.
  - Dizajnira računarske inpute (ulaze) i autpute (izlaze).
  - Dizajnira korisničke interfejse.
  - Prezentuje dizajn.

Pojam baze podataka

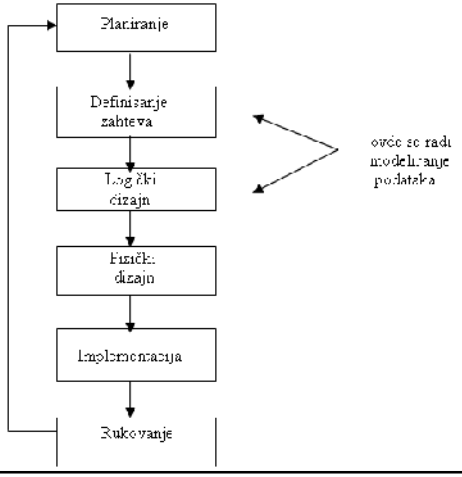
- **Baza podataka (BP)** je kolekcija međusobno povezanih podataka, uskladištenih sa minimumom redudanse, koje koriste, zajednički, svi procesi obrade u sistemu.
- Sa aspekta implementacije, **Baza podataka** predstavlja skup tabela međusobno povezanih putem spoljnog ključa.



Organizacija podataka u bazama podataka



Koraci kod projektovanja baze podataka

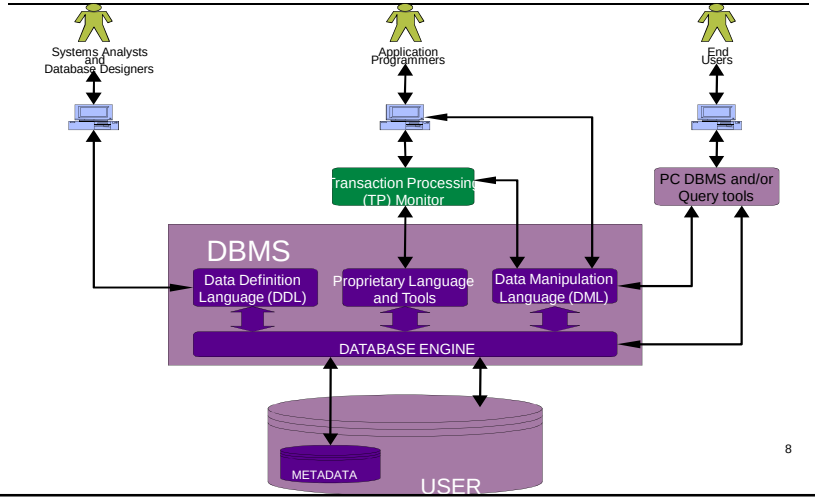


Sistem za upravljanje bazom podataka

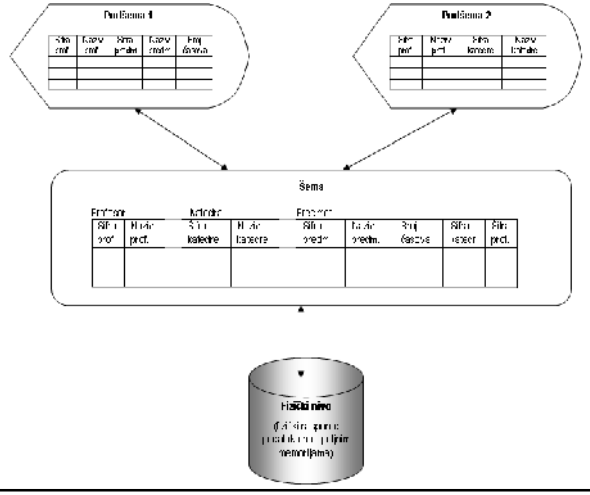
- **Sistem za upravljanje bazom podataka** (kraće DBMS, od početnih slova engleskih reči *Database Management Systems*) je softverski sistem koji kreira, pristupa, upravlja i kontroliše podacima (bazama podataka) i služi kao veza (interfejs) između podataka i aplikativnih programa.

- Sistem za upravljanje bazom podataka je jedan složeni softverski sistem koji treba da omogući:
  - Skladištenje podataka sa minimumom redudanse.
  - Korišćenje zajedničkih podataka od strane svih ovlašćenih korisnika.
  - Logičku i fizičku nezavisnost programa od podataka. Bez obzira što se podaci fizički pamte, po pravilu, samo jednom, u jedinstvenoj fizičkog organizaciji, svaki korisnik dobija svoju sopstvenu logičku sliku podataka kakva njemu najviše odgovara.
  - Jednostavno komuniciranje sa bazom podataka preko jezika bliskih korisniku, kako bi se neprofesionalni korisnici neposredno uključili u razvoj informacionog sistema, a profesionalnim programerima značajno povećala produktivnost.

Tipična DBMS arhitektura



Tronivovska struktura DBMS kroz primer



Jezici DBMS-a

- **Data Definition Language (DDL)** se koristi za izradu rečnika podataka, inicijalizaciju ili kreiranje baze podataka, opisivanje logičkog pogleda za svakog individualnog korisnika ili programera i specifiranje bilo kog ograničenja nad poljima ili tabelama baze podataka.
- **Data Manipulation Language (DML)** se koristi za održavanje podataka, koja uključuje takve operacije koje ažuriraju, ubacuju i brišu delove baza podataka.
- **Data Query Language (DQL)** se koristi za ispitivanje baze podataka. S obzirom na činjenicu da se DML koristi da menja sadržaj baze podataka, DQL samo sortira, uređuje, skladišti i predstavlja podskupove baze podataka prema odzivu na korisničke upite.
- Mnogi DBMS takođe uključuju i jezik za **pisanje izveštaja**, koji pojednostavljuje kreiranje izveštaja. Standardni upitni jezik koji omogućava komunikaciju sa bazom podataka jeste **SQL** (od početnih slova engleskih reči: *Structured Query Language*).

Primer DDL naredbe

Prikazuje se naredba koja kreira tabelu o dobavljačima:

```
CREATE TABLE    Dobavljač
(Šifra_dob          INTEGER (5) NOT NULL,
Ime_dob             CHARACTER (15)
Ulica_dob           CHARACTER (15)
Grad_dob            CHARACTER (15)
Poštanski_br_dob    INTEGER  (5)
Država_dob          CHARACTER (15)
Ostalo              CHARACTER (100)  )
```

Ova naredba kreira tabelu dobavljač sa sedam kolona. Šifra_dob i Poštanski_br_dob moraju da sadrže samo numeričnku vrednost, dok Šifra_dob ne sme da bude prazno (blanko) polje. Ostale kolone mogu da sadrže bilo brojeve bilo slova, a njihova maksimalna dužina je data u zagradama.

Rezultat naredbe:

Dobavljač

Šifra_dob	Ime_dob	Ulica_dob	Grad_dob	Poštanski _br_dob	Država_dob	Ostalo

Napomena: Ova naredba kreira praznu tabelu. Popunjavanje tabele zahteva korišćenje DML naredbi.

11

Primer DML naredbe

Ova naredba ubacuje novi red koji sadrži informacije o novom dobavljaču u tabelu dobavljač:

```
INSERT
INTO Dobavljač (Šifra_dob, Ime_dob, Ulica_dob, Grad_dob, Poštanski_br_dob, Država_dob, Ostalo)
VALUES (10004, 'COOL', 'Moše Pijade 8', 'Beograd', 11000, 'YU',
            'Dobavljač televizora, audio uređaja, mob. Telefona i bele tehnike')
```

Sledeća naredba menja ulicu dobavljača "COOL":

```
UPDATE Dobavljač
SET Ulica_dob = 'Dečanska 8'
WHERE Šifra_dob = 10004
```

Sledeća naredba briše iz tabele dobavljač, red koji sadrži informaciju o preduzeću COOL:

```
DELETE
FROM Dobavljač
WHERE Šifra_dob=10004
```

12

Primer DQL naredbe

```
SELECT Ime_potrošača, Ulica_potr, Grad_potr, Država_potr
FROM Potrošač, Račun, Artikal, Zaliha
WHERE  Date BETWEEN 10/01/96 and 10/31/96 AND
          Description='Televizor' AND
          Račun.Šifra_Potrošača=Potrošač.Šifra_potrošača AND
          Račun.Šifra_račun_prodato=Artikal.Šifra_račun_prodato AND
          Artikal.Šifra_artikla = Zaliha.Šifra_artikla
```


Relacioni DBMS

- Postoji nekoliko tipova sistema za upravljanje bazom podataka. Ono se mogu klasifikovati prema načinu struktuiranja zapisa. Prethodni sistemi za upravljanje bazom podataka su organizovali zapise u hijerarhije i mreže implementiranih sa indeksima i povezanim listama.
- Danas, najuspešniji sistemi za upravljanje bazom podataka se zasnivaju na relacionoj tehnologiji.
- **Relaciona baza podataka** implementira podatke u seriji dvodimenzionalnih tabela koje su u međusobnoj relaciji preko spoljnih ključeva.

Primer logičkog i fizičkog modela podataka

Tabela KLIJENT

Broj klijenta (primarni ključ)	Ime klijenta	Adresa klijenta	...
[illegible]	[illegible]	[illegible]	
[illegible]	Delta	[illegible]	
[illegible]	[illegible]	[illegible]	

Tabela PORUDŽBINA

Broj porudžbine (primarni ključ)	Broj klijenta (spoljašnji ključ)	...
[illegible]	[illegible]	
[illegible]	[illegible]	
[illegible]	[illegible]	

Tabela PORUČENI PROIZVODI

Broj porudžbine (spoljašnji ključ)	Broj proizvoda (spoljašnji ključ)	Naručena količina	...
[illegible]	[illegible]	1	
[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	

Tabela PROIZVODI

Broj proizvoda (primarni ključ)	Opis proizvoda	Količina na zalihama	...
[illegible]	[illegible]	800	
[illegible]	Sony TV	[illegible]	
[illegible]	Video rekorder	[illegible]	
[illegible]	[illegible]	20	
[illegible]	[illegible]	2	

Projektovanje informacionih sistema Prof. Dr Latinovic T.

Relacioni model

- Relacioni model poseduje semantički bogatije koncepte za opis strukture i znatno moćnije operacije.
- Tabela se može definisati kao matematička relacija i zatim iskoristiti bogata teorijska osnova odgovarajućeg matematičkog aparata.
- Svaka relacija ima svojstva skupa. (Osnovno svojstvo svakog skupa je da se elementi koje on sadrži međusobno razlikuju. Tako se i svi redovi tabele međusobno razlikuju.)
- Pošto je relacija skup, a svaka tabela nije, definišu se sledeći uslovi koje tabela mora da zadovolji da bi bila relacija:
  - Ne postoje duplikati vrsta tabele.
  - Redosled vrsta i redosled kolona nije značajan.
  - Nisu dozvoljni atributi ili grupe atributa sa ponavljanjem, odnosno nisu dozvoljene tabele u tabeli.

Ključevi

- Ključevi se definišu kao jedan ili više atributa čija vrednost jedinstveno identifikuje jednu n-torku u relaciji, tj. jedan red u tabeli.
- Ako je ključ definisan samo jednim atributom, onda je to prost ključ, odnosno ukoliko je definisan sa više atributa, onda je to složeni ključ.
- Ako se definiše relacija R, onda se može reći da ključ relacije R predstavlja kolekcija K njenih atributa koja zadovoljava :
  - osobinu jedinstvenosti i
  - osobinu neredundantnosti.
- Osobina *jedinstvenosti* je vezana za ograničenje da ne postoje bilo koje dve n-torke sa istom vrednošću K.
- Osobina *neredundantnosti* se odnosi na gubljenje osobina jedinstvenosti ako se bilo koji atribut izostavi iz K.
- U jednoj relaciji može postojati više različitih kolekcija K atributa koje zadovoljavaju definiciju ključa - sve se one nazivaju kandidati za ključ.

Ključevi

- **Primarni ključ** je jedan od izabranih kandidata za ključ koji služi za identifikaciju n-torke relacije. Ostali kandidati za ključ postaju alternativni ključevi.
- Atributi koji učestvuju u ključevima nazivaju se ključni atributi, dok se ostali nazivaju opisni atributi.
- **Spoljni ključ** ili preneseni ključ je atribut ili grupa atributa u relaciji R, čija se vrednost koristi za povezivanje sa vrednošću primarnog ključa u nekoj relaciji R2.
- Spoljni ključevi i njima odgovarajući primarni ključevi definisani su nad istim domenom.
- Spoljni ključevi služe da se uspostave veze između relacija u relacionoj bazi podataka. Na primer, u relaciji RADNIK spoljni ključ SIFRAODEL vezuje sa relacijom ODELJENJE.

Normalizacija relacione baze podataka

- Normalizacija je postupak projektovanja logičke strukture baze podataka.
- Pravilno projektovana baza podataka mora imati strukturu koja osigurava da u radu sa njom ne može biti neželjenih anomalija, kao što je na primer, uništavanje određenih podataka ili neusklađenost između memorisanih podataka kao posledica ažuriranja baze podataka itd.
- Dobra je ona struktura baze podataka u kojoj je logička redundansa minimalna.
- E. F. Codd je 1969. god. definisao postupak koji dovodi projektanta relacione baze podataka do željenog rezultata i nazvao ga “normalizacija” baze podataka.
- Codd je definisao tri nivoa normalizacije. Kasnije je Fagin dokazao da u nekim izuzetnim primerima relacije, koje su u trećoj normalnoj formi, još uvek zadržavaju neke neželjene osobine i zato je definisao četvrtu normalnu formu (4NF).

Funkcionalna zavisnost

- Ako je svakoj vrednosti atributa A u relaciji R priključena samo jedna vrednost atributa B u istoj relaciji, onda je atribut B **funkcionalno zavisan** od atributa A.
- Funkcionalna zavisnost se može definisati između složenog ključa (više atributa) i jednostavnog atributa.
- Ako je moguće svakom paru vrednost atributa A i B relacije R priključiti tačno jednu vrednost C iste relacije, tada je atribut C funkcionalno zavisan o sastavljenom atributu A i B.

Potpuna funkcionalna zavisnost

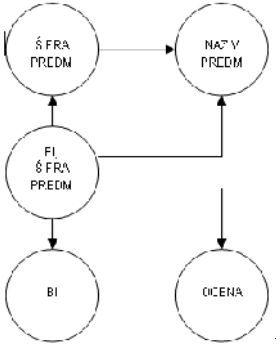
- Atribut B je **potpuno funkcionalno zavisan** od atributa A iste relacije, ako je funkcionalno zavisan od atributa A, a ne od nekog sastavnog dela atributa A (u slučaju kada je atribut A sastavljen).

Potpuna funkcionalna zavisnost se posmatra kada je ključ tabele isključivo sastavljen od više atributa.

BI, ŠIFRA_PREDMETA → OCENA
BI ⟶ OCENA
ŠIFRA_PREDMETA ⟶ OCENA

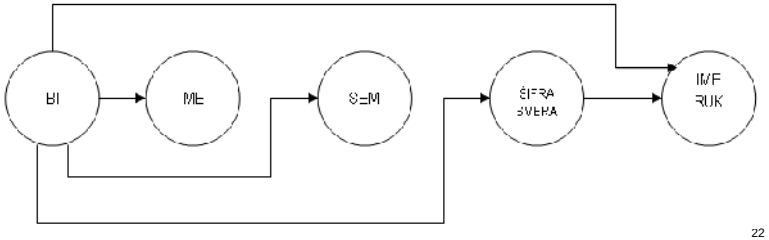
Atribut OCENA je potpuno funkcionalno zavisan od složenog atributa BI, ŠIFRA_PREDMETA.

BI, ŠIFRA_PREDMETA → NAZIV_PREDMETA
BI ⟶ NAZIV_PREDMETA
ŠIFRA_PREDMETA ⟶ NAZIV_PREDMETA



Tranzitivna funkcionalna zavisnost

- Atribut C je tranzitivno funkcionalno zavisan od atributa A, ako je funkcionalno zavisan od A i ako je funkcionalno zavisan od nekog atributa B koji je i sam funkcionalno zavisan od A.
- Tranzitivna zavisnost dovodi do veće redundanse.
- STUDENT(BI, IME, SEM, ŠIFRA_SMERA, IME_RUK).



22

Prva normalna forma

- Proces normalizacije predstavlja transformaciju početne tabele u jednu ili više korektnih tabela (relacija) u kojima su svi atributi potpuno funkcijski zavisni od ključa, a međusobno funkcijski nezavisni.
- Cilj normalizacije je da eliminiše redudansu u atributima i sa time omogući lakše održavanje integriteta podataka i jednostavniju manipulaciju.
- Osnovne operacije održavanja baze podataka su: *dodavanje* nove n-torke u relaciju, *izbacivanje* neke n-torke iz relacije i *izmena* (ažuriranje) vrednosti nekog atributa u relaciji.
- Problemi pri izvođenju ovih operacija kao što su ponavljanje ovih operacija više puta ili da se logičko dodavanje ne može izvršiti, odnosno da izbacivanje jednog logičkog skupa podataka dovodi do neželjenog izbacivanja drugih podataka, nazivaju se *anomalije* u održavanju baze podataka.
- **Relacija je u Prvoj normalnoj formi ukoliko su sve vrednosti njenih atributa atomske ili drugim rečima nisu dozvoljeni atributi ili grupe atributa «sa ponavljanjem», odnosno nisu dozvoljene «tabele u tabeli».**

23

Projektovanje informacionih sistema Prof. Dr Latinovic T.

Prva normalna forma

STUDENT

BI	IME STUDENTA	SEMESTAR	SIFRA SMERA	IME RUK.	SIFRA PREDMETA	NAZIV PREDMETA	OCENA
10	Slađana	5	01	Miloš	111	Menadžment	8
					222	Informatika	9
					333	Baze podataka	10
20	Pera	7	01	Miloš	222	Informatika	10
					333	Baze podataka	9
30	Mika	4	02	Maja	111	Menadžment	8
					444	Matematika	6

STUDENT

BI	IME STUDENTA	SEMESTAR	SIFRA SMERA	IME RUK.	SIFRA PREDMETA	NAZIV PREDMETA	OCENA
10	Slađana	5	01	Miloš	111	Menadžment	8
10	Slađana	5	01	Miloš	222	Informatika	9
10	Slađana	5	01	Miloš	333	Baze podataka	10
20	Pera	7	01	Miloš	222	Informatika	10
20	Pera	7	01	Miloš	333	Baze podataka	9
30	Mika	4	02	Maja	111	Menadžment	8
30	Mika	4	02	Maja	444	Matematika	6

Prva normalna forma

STUDENT1 (BI, IME_STUDENTA, SEMESTAR, ŠIFRA_SMERA,
IME_RUK, SIFRA_PREDMETA, NAZIV_PREDMETA, OCENA)

- Ova relacija ima sledeće anomaliije u održavanju baze podataka:
  - *anomalije u dodavanju*: ukoliko je po novom nastavnom planu definisan novi predmet, ne mogu se ubaciti podaci o tom predmetu dok ga neki student ne položi.
  - *Anomalije u izbacivanju*: ukoliko je jedan predmet položio samo jedan student i ako se on ispiše sa fakulteta, gube se i sve informacije o tom predmetu. Ako je bio jedini na smeru, gube se sve informacije o tom smeru.
  - *Anomalije u ažuriranju*: ukoliko se promeni naziv predmeta ili rukovodioc nekog smera, to se mora učiniti na onoliko mesta koliko je studenata položilo taj predmet, odnosno koliko je studenata upisano na dati smer.
  - *Problemi i u izveštavanju:* zahtev tipa: «Prikaži listu predmeta, imena svh studenata koji su ga položili i prosečnu ocenu na predmetu», bi zahtevao složeniji program ili bi trebalo samu relaciju prestruktuirati i dobiti dve relacije sa istim skupom podataka za dva različita zahteva, čime bi se redundansa podataka pa samim tim i problemi održavanja baze podataka umnožili.

25

Prva normalna forma

- Da bi se smanjila redundansa do koje bi ovakva normalizacija dovela, možemo ovako dobijenu relaciju, operacijom projekcije, dekomponovati na sledeće dve:

STUDENT1 (BI, IME_STUDENTA, SEMESTAR, ŠIFRA_SMERA, IME_RUK)
PRIJAVA (BI, ŠIFRA_PREDMETA, NAZIV_PREDMETA, OCENA)

- Relacija R se dekomponuje u svoje projekcije bez gubljenja informacija ako prirodno spajanje tako dobijenih projekcija dovodi do polazne relacije.
- *Heath*-ova teorema daje uslove pod kojima se može izvršiti dekompozicija relacije bez gubljenja informacija:

  Relacija R(A,B,C) gde su A,B i C podskupovi atributa, u kojoj važi R.A $\rightarrow$ R.B može se uvek dekomponovati u svoje projekcije R1(A,B) i R2(A,C) bez gubljenja informacija.

26

Druga normalna forma

- **Relacija R je u Drugoj normalnoj formi (2NF) ako i samo ako je u 1NF i svi njeni neključni atributi potpuno funkcionalno zavise od primarnog ključa.**
- Neključni atributi su atributi koji nisu kandidati za ključ, niti deo kandidata za ključ.
- Svođenje na 2NF vrši se dekompozicijom na taj način što se u jednoj projekciji ostavlja primarni ključ i svi atributi koji su potpuno funkcionalno zavisni od njega, a u drugim projekcijama se realizuju one funkcionalne zavisnosti koje su prouzrokovale nepotpune funkcionalne zavisnosti.

  PRIJAVA (BI, ŠIFRA_PRED, NAZ_PRED, OCENA),

  - Redundansa podataka: naziv predmeta se pojavljuje onoliko puta koliko je studenata položilo taj predmet. Ponovo postoje sve iste anomalije u održavanju baze podataka:
  - Ne mogu se dodati podaci o novom predmetu, ako ga neki student nije položio.
  - Ako se iz baze podataka izbaci n-torka studenta koji je jedini položio neki predmet, tada se gube sve informacije i o tom predmetu.
  - Ako se promeni naziv predmeta, to se mora učiniti na onoliko mesta koliko je studenata položilo taj predmet itd.

Druga normalna forma

- S obzirom da je relacija R u 2NF, ukoliko svi njeni atributi daju jednoznačne činjenice samo o celom ključu, onda relacija PRIJAVA nije u 2NF, jer NAZIV_PREDMETA daje jednoznačnu informaciju o delu ključa i to ŠIFRA_PREDMETA.
- Svaka relacija sa prostim primarnim ključem je u 2NF, jer prosti ključ nema semantički moguć pravi podskup.
- Svođenje na 2NF vrši se dekompozicijom na taj način što se u jednoj projekciji ostavlja primarni ključ i svi atributi koji su potpuno funkcionalno zavisni od njega, a u drugim projekcijama se realizuju one funkcionalne zavisnosti koje su prouzrokovale nepotupune funkcionalne zavisnosti.

  PRIJAVA1(BI, ŠIFRA_PREDMETA, OCENA)
  PREDMET(ŠIFRA_PREDMETA, NAZIV_PREDMETA)

Treća normalna forma

- **Relacija R je u Trećoj normalnoj formi (3NF) ako i samo ako je u 2NF i ako svi njeni neključni atributi netranzitivno funkcionalno zavise od primarnog ključa.**

  STUDENT1 (BI, IME, SEM, ŠIFRA_SMERA, IME_RUK)

- Da bi smo datu relaciju sveli na 3 NF, vršimo dekompoziciju (bez gubljenja informacija) relacije u njene projekcije, na taj način što u jednoj projekciji ostavljamo primarni ključ i sve netranzitivno zavisne atribute, a u drugim projekcijama se realizuju funkcionalne zavisnosti koje su dovele do tranzitivnih zavisnosti:

  STUDENT2 (BI, IME, SEM, ŠIFRA_SMERA)
  SMER (ŠIFRA_SMERA, IME_RUK)

Boyce-Codd-ova normalna forma (BCNF)

PRIJAVA (BI, ŠIFRA_PREDMETA, NAZIV_PREDMETA, OCENA)

- Pretpostavimo da važe sledeće funkcionalne zavisnosti:

  BI, ŠIFRA_PREDMETA → NAZIV_PREDMETA
  BI, ŠIFRA_PREDMETA → OCENA
  ŠIFRA_PREDMETA → NAZIV_PREDMETA
  NAZIV_PREDMETA → ŠIFRA_PREDMETA

- U ovom slučaju postoje dva složena i "preklapajuća" kandidata za ključ: BI, ŠIFRA_PREDMETA i BI, NAZIV_PREDMETA. Jedini neključni atribut je OCENA, pa pošto on potpuno funkcionalno zavisi od primarnog ključa, relacija je u 2 NF.

- **Relacija je u Boyce-Codd-ovoj normalnoj formi (BCNF) ako i samo ako su sve determinante u relaciji i kandidati za ključ.**

Boyce-Codd-ova normalna forma (BCNF)

- Determinanta relacije R je bilo koji atribut, prost ili složen, od koga neki drugi atribut u relaciji potpuno funkcionalno zavisi.
- Ukoliko označimo sve determinante (D) i sve kandidate za ključ (KK) u relaciji PRIJAVA:

BI, ŠIFRA_PREDMETA → NAZIV_PREDMETA, OCENA (D) (KK)
BI, NAZIV_PREDMETA → ŠIFRA_PREDMETA, OCENA (D) (KK)
ŠIFRA_PREDMETA → NAZIV_PREDMETA (D)
NAZIV_PREDMETA → ŠIFRA_PREDMETA (D),

sve determinante nisu kandidati za ključ pa relacija nije u BCNF.

- Dekompozicijom, pri kojoj se iz relacije izvlače projekcije sa onim determinantama koje nisu kandidati za ključ, relacije se svodi na BCNF.

PRIJAVA1 (BI, ŠIFRA_PREDMETA, OCENA)
PREDMET (ŠIFRA_PREDMETA, NAZIV_PREDMETA)

Dekomozicija na zavisne i nezavisne projekcije

STUDENT1 (BI, IME, SEM, ŠIFRA_SMERA, IME_RUK)

se može svesti na 3NF na sledeća dva načina:

(a)	(b)
STUDENT2a (BI, IME, SEM, ŠIFRA_SMERA)	STUDENT2 (BI, IME, SEM, ŠIFRA_SMERA)
STUDSMER (BI, IME_RUK)	SMER (ŠIFRA_SMERA, IME_RUK)

- Projekcije (a) pokazuju izvesne anomalije u održavanju, jer da bi se dodali ili izbacili podaci o jednom studentu, to se mora uraditi u obe projekcije, ili da bi ažurirali atribut IME_RUK, za svaku vrednost BI koja odgovara datoj vrednosti ŠIFRA_SMERA iz relacije STUDENT2a, mora se u relaciji STUDSMER izvršiti ažuriranje atributa IME_RUK.
- *Rissansen*-ova teorema daje sledeće uslove pod kojima se neka relacija može dekomponovati na nezavisne projekcije:
  - Projekcije R1 i R2 relacije R su nezavisne tada i samo tada ukoliko se:
    - Svaka funkcionalna zavisnost u R može logički dedukovati iz funkcionalnih zavisnosti u R1 i R2.
    - Zajednički atribut relacija R1 i R2 je KK barem u jednoj od njih.

Četvrta normalna forma

PROGRAM (PREDMET, NASTAVNIK, KNJIGA)

- ukoliko pretpostavimo da jedan predmet predaje više nastavnika, i da se za jedan predmet koristi više knjiga, odnosno postoje sledeće višeznačne zavisnosti:

PREDMET $\rightarrow\rightarrow$ NASTAVNIK PREDMET $\rightarrow\rightarrow$ KNJIGA

- i pretpostavimo da ne postoji veza između nastavnika i knjiga odnosno da se ne zna koji nastavnik koristi koje knjige i da li koristi jednu ili više.
- Data relacija PROGRAM je u BCNF, ali anomalije u ažuriranju su očigledne, jer da bi ubacili knjigu za jedan predmet, to bi trebalo da uradimo na onoliko mesta koliko profesora drži taj predmet. Razlog anomalijama jeste postojanje dve nezavisne višeznačne činjenice.

Četvrta normalna forma

- U relaciji R(A,B,C) postoji **višeznačna zavisnost** A $\rightarrow\rightarrow$ B, ako za datu vrednost A, postoji skup od nula, jedne ili više vrednosti B, a taj skup vrednosti ni na koji način ne zavisi od vrednosti atributa C. Atributi A, B i C mogu biti složeni.
- **Relacija R je u četvrtoj normalnoj formi (4NF) ukoliko u njoj nisu date dve ili više nezavisne višeznačne činjenice.**
- *Fagin* navodi:

  Relacija R (A,B,C) može se dekomponovati bez gubljenja informacija na projekcije R1(A,B) i R2 (A,C) ako i samo ako važi A $\rightarrow\rightarrow$ B (što uključuje i A$\rightarrow\rightarrow$C, jer ako u relaciji postoji višeznačna zavisnost A$\rightarrow\rightarrow$ B, tada postoji i višeznačna zavisnost A$\rightarrow\rightarrow$ C).

RASPORED (PREDMET, NASTAVNIK)
UDŽBENIK (PREDMET, KNJIGA)

- Relacije RASPORED i UDŽBENIK su u 4NF jer u njima ne postoje višeznačne zavisnosti samim tim što su obe binarne relacije, odnosno obe sadrže samo po jednu višeznačnu činjenicu.

Peta normalna forma

- U relaciji R(X,Y, ... Z) postoji **zavisnost spajanja** ako i samo ako relacija R rezultuje iz prirodnog spajanja njenih projekcija po X, Y, ..., Z, gde su X,Y, .. Z podskupovi atributa relacije R.
- **Relacija R je u Petoj normalnoj formi (5NF) ako i samo ako se svaka zavisnost spajanja može pripisati kandidatu za ključ.**

  STUDENT2 (BI, IME, SEM, ŠIFRA_SMERA)
  R1 (BI, IME)
  R2 (BI, SEM)
  R3 (BI, ŠIFRA_SMERA)

- Relacija STUDENT2 se može rekonstruisati prirodnim spajanjem preko BI, koji je njen ključ, pa je stoga relacija STUDENT2 u 5 NF.

Normalna forma ključeva i domena (DK/NF)

- 1981. - R. Fagin je dao najopštiju definiciju normalne forme ključeva i domena i pokazao da je relacija koja je u DK/NF ne prouzrokuje anomalije u održavanju.
- **Relacija je u DK/NF:**
  - **Ako su sve zavisnosti u njoj posledica definicije ključeva i domena.**
  - **Ako svaki atribut daje informaciju isključivo o ključu, o celom ključu i njegovim delovima.**

  PRIJAVA(BI, ŠIFRA_PREDMETA, NAZIV_PREDMETA, OCENA)

- Nije u DK/NF jer je definisano ograničenje da jedan predmet ima jedan naziv (ŠIF_PREDMETA →NAZIV_PREDMETA), a to nije posledica definicije ključa ove relacije.
- Direktna primena ove definicije nije moguća, jer otkrivanje ograničenja u relacijama zahteva otkrivanje funkcionalnih, višeznačnih i zavisnosti spajanja, a to praktično znači primenu svih navedenih definicija.

Integritet baze podataka

- Zaštita baze podataka
- Pravila integriteta baze podataka
- Sigurnost baze podataka

Zaštita baze podataka

- Zaštitu baze podataka tretiramo kroz dva aspekta i to:
  - *Integritet* – zaštita od slučajnog pogrešnog ažuriranja i
  - *Sigurnost* – zaštita od neovlašćenog ažuriranja i korišćenja podataka.
- Termin **integritet** podataka označava tačnost, korektnost ili konzistentnost.
- **Integritet baze podataka** podrazumeva problem zaštite baze podataka od pogrešnog ažuriranja, odnosno od pogrešnih ulaznih podataka, greški operatera i programera, sistemskih otkaza i dr.
- Termin **sigurnost** podataka podrazumeva mehanizme zaštite baze podataka od neovlašćenog korišćenja.

38

Pravila integriteta

- Pravila integriteta definišu koje uslove podaci u BP treba da zadovolje, kada se vrši provera i koje akcije treba preduzeti kada definisani uslovi nisu zadovoljeni.
- Pravila integriteta se dele u dve klase:
  - *Pravila integriteta domena* – definišu se za vrednosti pojedinih atributa nezavisno od vrednosti ostalih atributa u BP (karakterističan primer je integritet objekta), i
  - *Pravila integriteta relacija* – odnose se na relaciju kao celinu, odnosno definišu kada neka n-torka može da se ubaci u relaciju ili kako n-torke jedne relacije zavise od n-torki druge relacije (karakterističan primer je referencijalni integritet).

Pravila integriteta domena

- Definišimo domen STAR, nad kojim je definisan atribut STAROST relacije RADNIK:

Domen: STAR
Skup od INTEGER
Operacije:
Proveri-domen-STAR
Ulaz: vrednost_atributa
Izlaz: Boolean
Post: **if** vrednost_atributa between 15 and 65
then izlaz = true **else** izlaz = false.

Pravila integriteta relacije

- Najznačajnija pravila integriteta relacija su *referencijalni integriteti*.
- Referencijalni integritet zahteva da za sve spoljne ključeve postoje vrednosti primarnog ključa bazne tabele. Ovo pravilo zahteva da se spreče sledeći tipovi transakcija:
  - Dodavanje zapisa na strani "više" relacije tipa jedan-prema-više, a da ne postoji odgovarajući zapis na strani "jedan" relacije.
  - Brisanje zapisa na strani "jedan" relacije tipa jedan-prema-više, a da se prvo ne obrišu svi odgovarajući zapisi na strani "više" relacije.
  - Brisanje ili dodavanje zapisa tabeli koja je u relaciji tipa jedan-prema-jedan sa drugom tabelom, a da se ne obriše ili doda odgovarajući zapis u povezanoj tabeli.
  - Menjanje vrednosti polja primarnog ključa bazne tabele od koje zavise zapisi u povezanoj tabeli.
  - Menjanje vrednosti polja spoljnjeg ključa u relacionoj tabeli u vrednost koja ne postoji u polju primarnog ključa bazne tabele.

41

Referencijalni integritet

- Za svaki spoljni ključ u relaciji mora se definisati jedno pravilo integriteta koje se sastoji od uslova, trenutka kada se uslov ispituje i akcija koje se preduzimaju kada uslov nije zadovoljen.
- Akcije koje se preduzimaju ako uslov integriteta nije zadovoljen su sledeće:
  - RESTRICTED – Kada se uslov ispituje pre operacije održavanja i ako nije zadovoljen, operacija se odbija, uz odgovarajuću poruku.
  - NULLIFIES – Spoljni ključ dobija nula vrednost, ako je nula vrednost dozvoljena.
  - DEFAULT – Spoljni ključ dobija neku difoltnu vrednost koju treba unapred predvideti u definiciji odgovarajućeg domena.
  - CASCADES – Operacija se prenosi na relaciju koju referiše spoljni ključ, da bi se u njoj izvršile promene koje će zadovoljiti uslov integriteta.

42

Pravila integriteta relacija kod povezivanja objekata

- Ukoliko posmatramo dva objekta koja su u vezi (prvi objekat nazvaćemo *domen*, a drugi objekat koji je sa njim u vezi nazvaćemo *kodomen*), i želimo da ih povežemo (*connect*). Akcije koje se mogu preduzeti su sledeće:
  - RESTRICTED – operacija se odbija jer ne postoji objekat kodomen.
  - NULLIFIES – umesto sa navedenim objektom kodomena, ako to pojavljivanje ne postoji, povezivanje se izvršava sa specijalnim pojavljivanjem kodomena, tzv «nula objektom», čije je značenje «nepoznati objekat».
  - DEFAULT – umesto sa navedenim objektom kodomena, ako to pojavljivanje ne postoji, povezivanje se izvršava sa «default» objektom kodomena. Pretpostavlja se da se difolt objekat kodomena definisan.
  - CASCADES – ako objekat kodomena sa kojim dati objekat domena treba da se poveže ne postoji moguće ga je prvo kreirati operacijom *insert (ubaci)*, pa zatim izrvšiti odgovarajuće povezivanje.

Pravila integriteta relacija kod razvezivanja objekata

- Operacija *disconnect* (razveži) raskida vezu objekta domena sa nekim kodomenom i na taj način, ako je posmatrano preslikavanje obavezno, narušava odgovarajuće strukturno ograničenje.
  - RESTRICTED – odbija se operacija.
  - NULLIFIES – razvezuje se pojavljivanje objekta domena od navedenog pojavljivanja objekta kodomena i vezuje za «nula objekat» kodomena.
  - DEFAULT – razvezuje se pojavljivanje objekta domena od navedenog pojavljivanja objekta kodomena i vezuje za «default» objekat kodomena.
  - CASCADES – izbacuje se pojavljivanje objekta domena koji posle operacije disconnect ostaje da «visi».
- Pravila integriteta treba da budu definisana u okviru opisa baze podataka, a ne u okviru aplikacionih programa. U opisu svake relacije, za svaki spoljni ključ, treba navesti odgovarajuće pravilo integriteta.

Sigurnost baze podataka

- Termin **sigurnost** podataka podrazumeva mehanizme zaštite baze podataka od neovlašćenog korišćenja.
- Opšti model zaštite podataka treba da definiše koji subjekat zaštite, može nad kojim objektom zaštite da izvrši neku operaciju i pod kojim uslovima.

SUBJEKT	OBJEKAT	OPERACIJA	USLOV
Ana	Radnik	Prikaz, dodavanje	STAROST < 40
Branko	Raspored	Prikaz	ŠIFRA PROJEKTA = pr1
…	…	…	…

SQL naredbe za dodelu ovlašćenja pristupa bazi podataka

- Ovlašćenje korisnicima, prema SQL standardu, se daje pomoću naredbe GRANT.

  GRANT privilegije [ON objekat] TO subjekat [WITH GRANT OPTION]

- Primer:
  GRANT SELECT, INSERT ON RADNIK TO Ana
  GRANT SELECT, INSERT, DELETE ON RASPORED TO Branko WITH GRANT OPTION
  GRANT DBA TO Pera.
- Opis privilegija:
  - SELECT - selektovanje
  - UPDATE - promeniti
  - INSERT - ubaciti
  - DELETE – obrisati
  - DBA – za administratora baze podataka kome su dozvoljene sve operacije nad njegovom bazom.
  - WITH GRANT OPTION – znači da autorizovani subjekt može da prenosi svoju autorizaciju i drugim subjektima.

Zadaci implementacione faze

- Implementacija sistema podrazumeva puštanje tog sistema u rad odnosno u svakodnevno poslovanje.
- Zadaci implementacione faze su:
  - Testiranje ponašanja sistema
  - Priprema prelaznog plana
  - Instaliranje baze podataka
  - Obuka korisnika
  - Prelazak na novi sistem
- Testiranje sistema je test koji osigurava da aplikacioni programi koji su izolovano pisani i testirani rade adekvatno kada su integrisani u celokupan sistem.
- Test prihvatanja sistema je finalni test sistema koji se obavlja od strane krajnjih korisnika koji koriste realne podatke u jednom dugotrajnom vremenskom periodu.

... sledeći slajdovi koji se bave operacijama relacionog modela, oporavkom baze podataka i o osnovnim funkcijama MS Access-a nisu neophodni za ispit iz ovog predmeta, već služe kao dopuna vašem znanju.

Relacioni modeli

Teoretske postavke relacionih modela

- Tabela se može definisati kao matematička relacija i nad njom razmatrati sledeće:

- **Dekartov proizvod** od n skupova D1 x D2 x...Dn je skup svih mogućih uređenih n-torki <d1,d2,....dn> tako da je d1 ∈ D1, d2 ∈ D2 ...dn ∈ Dn.
  Na primer data su dva skupa brojeva:
  D1 = {1,2,3,4} i D2 = {4,6}
  D1 x D2 = {<1,4>,<1,6>,<2,4>,<2,6>,<3,4>,<3,6>,<4,4>,<4,6>}.

- **Relacija** se definiše kao podskup Dekartovog proizvoda nad n-skupova R ⊆ D1 x D2 x ...x Dn, tj. podskup sadrži one n-torke Dekartovog proizvoda koje zadovoljavaju zadatu relaciju.
  Primer: Definisati za prethodni primer nad skupovima D1 i D2 relaciju:
  R: D1 x D2 ={ <d1,d2> | d1 = d2/2}
  R = {<2,4>, <3,6>}.

50

Operacije relacionog modela

- Postoje dva opšta načina iskazivanja operacija relacionog modela:
  - **Relaciona algebra** u kojoj se definiše skup operacija pomoću kojih je moguće dobiti željenu relaciju iz skupa datih relacija;
  - **Relacioni račun** koji predstavlja neproceduralan način iskazivanja operacija, gde se pomoću konstrukcija predikatskog računa prvog reda definišu osobine relacije koja se želi dobiti.

Relaciona algebra

- Relaciona algebra definiše skup operacija pomoću kojih se, na proceduralan način, može dobiti željena relacija, tj.tabela iz skupa datih relacija.
- Relacija se definiše kao podskup Dekartovog proizvoda i može se tretirati kao skup n-torki.
- Za nivo relacione algebre operacije: unija, presek i diferencije nad relacijama R1 i R2 moraju zadovoljiti uslov kompatibilnosti, tj. relacije R1 i R2 moraju imati isti broj atributa (isti stepen), a odgovarajući atributi su definisani nad istim domenom.
- Operacije nad relacijama mogu da se grupišu u:
  - Konvencionalne skupovne operacije (unija, presek, razlika i Dekartov proizvod),
  - Specijalne relacione operacije (projekcija, selekcija, spajanje i deljenje),
  - Dodatne operacije relacione algebre (operacije koje su se kasnije dodavale originalnoj relacionoj algebri da bi se povećala njena moć kao upitnog jezika),
  - Operacije ažuriranja baze,
  - Operacije sa null vrednostima.

Operacije relacione algebre

- Unija
  - Ako su date relacije R1 i R2 koje zadovoljavaju uslove kompatibilnosti, onda je rezultat operacije unije R3 = R1 U R2 relacija R3 koja sadrži sve n-torke koje se pojavljuju bilo u R1, bilo u R2.
  - Primer:
    - Definišimo relaciju NovoOdel (OdleID, NazivOdel)
      Odel4, Razvoj
      Odel5, Skladište
    - Relacija RR=Odel **U** NovoOdel (OdleID, NazivOdel)
      Odel1, Proizvodnja
      Odel2, Računovodstvo
      Odel3, Marketing
      Odel4, Razvoj
      Odel5, Skladište

Operacije relacione algebre

- Diferencija
  - Ako su date relacije R1 i R2 koje zadovoljavaju uslov kompatibilnosti, onda rezultat operacije diferencije R3 = R1 - R2 predstavljaju n-torke relacije R1, koje nisu istovremeno i n-torke relacije R2.
  - Primer:
    - RRR = RR – Odel (OdleID, NazivOdel)
      Odel4, Razvoj
      Odel5, Skladište

Operacije relacione algebre

- Presek
  - Ako su date relacije R1 i R2 koje zadovoljavaju uslov kompatibilnosti, onda je rezultat operacije preseka R3 = R1 ∩ R2 relacija R3 koja sadrži n-torke koje se pojavljuju u obe relacije R1 i R2.
  - Operacije unije i diferencije su pogodne za ažuriranje baze podataka.
  - Operacijom unije se mogu u neku relaciju dodati nove n-torke, a operacijom diferencije se mogu iz neke relacije izbaciti neželjene n-torke.
  - Izmena vrednosti pojedinih atributa u nekoj relaciji vrši se uzastopnom primenom operacije diferencije, pomoću koje se izbaci cela n-torka u kojoj se nalazi posmatrani atribut, a zatim se operacijom unije "vraća" izbačena n-torka sa promenjenom vrednošću atributa.

Operacije relacione algebre

- Dekartov proizvod
  - Može se primeniti na bilo koje dve relacije.
  - Rezultat operacije R3 = R1 x R2 je relacija R3 čije su n-torke svi "parovi" koje čine jedna n-torka relacije R1 i jedna n-torka relacije R2.
- Projekcija
  - Unarna operacija koja iz neke relacije selektuje skup navedenih atrubuta, odnosno "vadi" vertikalni podskup iz odgovarajuće tabele.
  - Operacijom projekcija eliminišu se duplikati n-torki.
  - Primer:
    - R1

A	B
a	b
a	?
?	b
?	?

Građanin (MLB, Ime, Starost, MestoRod)

1111	Ana	19	Beograd
2222	Mirko	21	Valjevo
3333	Zoran	20	Beograd
4444	Ana	19	Niš
5555	Mirko	21	Beograd

    - P[A]R1

A
a
?

Pr1:=π Ime, Starost (Građanin)

Ime	Starost
Ana	19
Mirko	21
Zoran	20

Operacije relacione algebre

- Selekcija
  - unarna operacija koja iz date relacije selektuje n-torke koje zadovoljavaju zadati uslov ("vadi" horizontalni podskup tabele).
  - Primeri:
    - S[A=a]R1

      A B
      a b
      a ?
    - RR = S[Starost>24 AND MestoRodj = 'Beograd'] RADNIK

      RadnikID MLB Ime Starost MestoRodj OdelID
      110 11971717... Mika 33 Beograd odel1
      111 22975718... Pera 29 Beograd odel1

Operacije relacione algebre

- Spajanje
  - binarna operacija koja spaja dve relacije na taj način da se u rezultatu pojavljuju oni parovi n-torki jedne i druge relacije koji zadovoljavaju uslov zadat nad njihovim atributima.
  - Primer:
    Student (BrInd, MLB, Smer)
    155/01 1111 Kompjuterski
    255/02 2222 Izvršno upravljanje
    322/01 3333 Međunarodno poslovanje

    Pr_2= Građanin [Građanin.MLB=Student.MLB] Student

    Pr2 (MLB, Ime, Starost, MestoRođ, MLB, BrInd, Smer)
    1111 Ana 19 Beograd 1111 155/01 Kompjuterski
    2222 Mirko 21 Valjevo 2222 255/02 Izvršno upravljanje
    3333 Zoran 20 Beograd 3333 322/01 Međunarodno poslovanje

Operacije relacione algebre

- Deljenje
  - pogodna za upite u kojima se javlja reč "svi".
  - Primer:
    - Iz relacija Predmet i Prijava prikazati brojeve indeksa studenata koji su položili sve predmete.
    - **Predmet** (PredmetID, NazivPredmeta)

p1	Matematika
p2	Marketing
p3	Informatika

      Prijava (BrInd, PredmetID)

155/01	p1
155/01	p3
255/01	p2
322/01	p1
155/01	p2

    - Prijava [Prijava.PredmetID ÷ Predmet.PredmetID] Predmet

      BrInd
      155/01

59

Oporavak baze podataka

Oporavak baze podataka i upravljanje izvršenjem transakcija

- Baza podataka je zajednički resurs koga konkurentno (istovremeno) koriste veći broj programa.
- Kod istovremenog korišćenja može doći do mnogih neželjenih efekata, kao na primer:
  - Otkaz sistema u toku izvršavanja nekog programa ili obrade neke transakcije (na primer skinute su pare sa računa nekog komitenta, ali usled otkaza u tom trenutku se nije izvršio prenos na drugi račun).
  - Gubljenje rezultata ažuriranja (na primer, ukoliko se izvršavaju dve transakcije istovremeno, transakcija A podiže, dok transakcije B ulaže novac na isti račun. Ažuriranje koje transakcija A obavlja je izgubljeno, odnosno prebrisano ažuriranjem koje je izvršila transakcija B u istom trenutku).
  - Problem nekorektne analize podataka (na primer, za vreme nekog sračunavanja koje se obavlja u jednoj transakciji, druga promeni vrednost argument koji je prva već obradila).
- Osnovni cilj baze podataka je da omogući efikasnu obradu transakcija (jedno izvršenje neke logičke jedinice posla ili programa).

Transakcije

- Transakcija je operacija kojom se izvodi serija izmena nad jednom ili više tabela, tj. transakcija je izvršenje neke logičke jedinice rada korisnika baze podataka.
- Osnovni cilj baze podataka je da omogući efikasnu obradu transakcija.
- Transakcija mora da poseduje sledeći skup osobina:
  - Atomnost – skup aktivnosti nad bazom podataka koje se izvršavaju po principu “sve ili ništa” (ili su sve aktivnosti uspešno obavljene ili je BP ostala nepromenjena) je atomski skup aktivnosti.
  - Konzistentnost – izvršenje transakcije treba da prevede BP iz jednog u drugo konzistentno stanje.
  - Izolacija – Transakcija ne treba da svoje promene BP učini vidljivim drugim transakcijama pre nego što se ona okonča.
  - Trajnost – Kada su u BP potvrđene promene koje je izvršila neka transakcija, ove promene se više ne mogu izgubiti.
- Da bi se obezbedile ove osobine, skup instrukcija koje predstavlja transakciju počinje sa specijalnom instrukcijom BEGIN TRANSACTION, a završava se bilo sa instrukcijom COMMIT (potvrđuju se promene u BP) ili ROLLBACK (poništavaju se promene u bazi).
- U savremenim DBMS transakcije mogu biti “ugnježđene”, odnosno dozvoljeno je da se jedna transakcija smesti u drugu. Naredba COMMIT ugnježđene transakcije stvarno ne potvrđuje promene u bazi dok se uspešno ne okonča i transakcija na najvišem nivou ugnježđenja.
- Ukoliko se u bilo kojoj transakciji ugnježđene strukture pokrene instrukcija ROLLBACK, poništavaju se promene koje su proizvele sve ostale transakcije strukture.

Problem nepotvrđenih promena (čitanja)

- Ovaj problem se javlja kada se jednoj transakciji dozvoli da čita ili da menja rekord koji druga transakcija ažurira, a promene koje je ona učinila još nisu potvrđene naredbom COMMIT.
- Da bi se mogle poništiti promene koje je transakcija izvršila nad BP, DBMS poseduje i održava specijalnu memorijsku lokaciju (na nekoj spoljnoj memoriji) koja se naziva log ili žurnal.
- Na ovoj memorijskoj lokaciji, za svaku transakciju i svaki objekat baze koji je ona ažurirala čuvaju se vrednosti pre (before-image) i vrednosti posle ažuriranja (after-image).
- Kada se izda instrukcija ROLLBACK sistem, koristeći vrednosti pre za datu transakciju, ažurira odgovarajuće objekte baze podataka.

Konkurentna obrada transakcija

- Transakcija se u sistemima zasnovanim na bazi podataka ne obavlja u izolaciji, već konkuretno (uporedo) sa drugim transakcijama u sistemu.
- Više transakcija mogu istovremeno zahtevati iste resurse, isti rekord baze podataka.
- Nekontrolisana međusobna interferencija transakcija može da dovede bazu u nekonzistentno stanje.
- Komponenta DBMS-a koja vodi računa o redosledu izvršavanja akcija nad bazom u skupu transakcija koje se konkuretno izvršavaju se zove Planer.
- Komponenta koja upravlja celokupnim izvršenjem transakcija naziva se Menadžer transakcije.

Protokoli zaključavanja

- Za razliku od konkuretnog , serijsko izvršenje transakcija je izvršenje u kome se transakcije ne prepliću, već se prvo potpuno izvrši jedna, pa onda druga transakcija.
- Velika je verovatnoća da ničim ograničeno izvršenje jednog skupa transakcija neće biti serijabilno.
- Zadatak planera izvršenja je da proveri serijabilnost nekim i da spreči da neserijabilna izvršenja naruše integritet baze podataka.
- S obzirom da proveru serijabilnosti i preduzimanje odgovarajućih akcija planer izvršenja teško može da obavi u realnom vremenu primenjuje se mehanizam zaključavanja (locking).
- Ovaj mehanizam upravljanja izvršenjem skupa transakcija omogućava da transakcija "zaključa" (postavi lokot) na objekat baze kome je pristupila, da bi onemogućila da druge transakcije nekorektno operišu sa istim objektom.

Protokoli zaključavanja

- Postoje više različitih vrsta zaključavanja, neke od njih su:
  - Ekskluzivno zaključavanje (exclusive (XL) lock ili write lock) – ako neka transakcija postavi ekskluzivni lokot na objekat baze, nijedna druga transakcija ne može da taj objekat postavi bilo koji drugi lokot.
  - Deljivo zaključavanje (shared (SL) lock ili read lock) – ukoliko neka transakcija postavi deljivi lokot na objekat baze, neka druga transakcija takođe može da postavi deljivi lokot na isti objekat baze, ali nijedna druga ne može da postavi eksluzivni lokot na taj objekat.
- Protokol zaključavanja koji bi mogao da reši prikazane probleme može se iskazati na sledeći način:
  1. Transakcija koja želi da pročita neki objekat baze mora prvo da postavi deljivi lokot na taj objekat;
  2. Transakcija koja želi da ažurira neki objekat baze mora prvo da postavi eksluzivni lokot na taj objekat. Ako je ta transakcija ranije postavila S lokot, ona treba da taj lokot transformiše u X lokot;
  3. Ako transakcija nije uspela da postavi lokot na željeni objekat baze, tada ona prelazi u stanje čekanja;
  4. Transakcija oslobađa E lokot obavezno, a po pravilu i S lokot na kraju, sa COMMIT i ROLLBACK naredbom.

"Živi" i "mrtvi" lokoti

- U toku izvršenja skupa transakcija, moguće je da dve ili više transakcija formiraju "mrtvi lokot", odnosno da lokoti koje su one postavile na objekte baze sve njih dovode u stanje čekanja.
- Pojam "živog lokota" se definiše kada je neka transakcija stalno u stanju čekanja na neki objekat baze zbog toga što druge transakcije uvek pre nje postave lokot na taj objekat.
- Problem "živog lokota" se jednostavno rešava uvođenjem nekog redosleda zaključavanja objekta (na primer FIFO).
- Za razrešavanje problema mrtvih lokota koriste se tri tehnike:
  - Prekidanje transakcije posle isteka intervala vremena – koristi se parametar timeout koji menadžer lokota dodeljuje transakciji pri pokušaju zaključavanja nekog objekta. Ukoliko istekne vreme transakcija se poništava i ponovo startuje;
  - Prevencija lokota – definisanje različitih protokola koji će sprečiti da dođe do mrtvog čvora. Na primer uređenje elemenata baze, gde se blokovi uređuju po njihovim adresama na spoljnoj memoriji (A1<A2<A3 ..). Na primer ako transakcija T1 zahteva A1 pa A2, transakcija T2 neće moći da traži lokote u redosledu A2 pa A1, koji bi doveo do mrtvog čvora, jer ovakvo zaključavanje nije po definisanom protokolu;
  - Detekcija "mrtvog čvora" – dozvoljava se da dođe do mrtvog lokota, ali transakcija koja ga je izazvala se ubija i njeni efekti na bazu se poništavaju.

Dugačke transakcije

- Postoje aplikacije u kojima su transakcije mnogo duže, reda desetine minuta, sati ili čak nekoliko dana. Prikazani mehanizmi zaključavanja u kome bi neka transakcija ovako dugo ekskluzivno zaključala elemente baze koje koristi je neprihvatljivo.
- Primeri aplikativnih sistema sa dugim transakcijama su:
  - Neke transakcije u poslovnim aplikacijama – na primer ispitivanje svih računa u nekoj banci kako bi se proverilo ukupno stanje računa banke.
  - Projektni sistemi – CAE, CAD, CASE i dr preko kojih se projektuju različiti inženjerski sistemi. Pretpostavlja se da više projektanata radi na istom projektu i da postoji zajednička baza projekta. Jedan projektant može da preuzme jedan deo podataka iz baze i da sa njim radi svoj deo projekta i nekoliko dana. Očigledno postoji potreba za istim podacima od strane i drugih projektanata.
  - Workflow sistemi – sistemi u kojima se jedna logička jedinica posla za korisnika, obavlja preko sekvence aktivnosti od kojih neke mogu biti programi, neke interaktivne aplikacije dok su neke realizovane i potpuno ručno.

Koncept “saga”

- Standardni skup čvorova u UML-u je proširen za čvor “Neuspešan kraj aktivnosti – abort” da bi se uveo osnovni koncept u teoriji upravljanja dugačkom transakcijama – koncept “sage”.
- Saga se može definisati kao graf koji ima, standardom UML-a, definisane tipove čvorova. Putanja od početnog čvora do bilo kog terminalnog čvora predstavlja moguću sekvencu akcija.
- Konkurentno upravljanje sagama se obavlja na sledeći način:
  - Svaka aktivnost se može tretirati kao posebna “kratka” transakcija koja se izvršava pod kontrolom konvencionalnih mehanizama za upravljanje transakcijama.
  - Celokupna dugačka transakcija, jedna od putanja od početnog čvora do uspešnog ili neuspešnog završetka, se obavlja na taj način što se u slučaju uspeštnog kraja zadržavaju sve promene koje su učinile pojedine aktivnosti, a u slučaju neuspešnog koristi se mehanizam kompenzacionih transakcija da bi se baza vratila u isto stanje u kome je bila pre otpočinjanja sage.

Oporavak baze podataka

- Oporavak baze podataka je vraćanje baze podataka u korektno stanje posle nekog otkaza.
- Da bi se oporavak baze mogao uspešno izvršiti neophodno je da DBMS obezbedi redundante podatke. Jedan skup redundantnih podataka se čuva u logu, a drugi skup u arhivskoj memoriji u koju se povremeno prenosi sadržaj cele baze.
- Log se koristi za otkaze koji fizički ne oštećuju memorijske jedinice (diskove) na kojima se čuva baza, dok arhivska memorija služi za oporavak posle ovakvih otkaza.
- Tipična strategija za oporavak baze podataka je:
  - Ukoliko su oštećenje memorijske lokacije (na primer, pad glave diska) koristi se poslednja arhivska kopija cele baze za njen oporavak.
  - Kada baza nije fizički oštećena, oporavak se vrši korišćenjem loga, preko posebno definisanih protokola oporavka. U ovom slučaju DBMS vodi računa o sledećim otkazima:
    - Softverski ili hardverski otkaz koji ne oštećuje bazu (nestanak napajanja);
    - Otkaz u samoj transakciji, na primer zbog deljenja sa nulom;
    - Akcije sistema za konkurentnu obradu transakcija, ubijanje transakcije koja je izazvala mrtvi lokot ili neserijabilnost izvršenja skupa transakcija i sl.

Ukratko o Microsoft Access-u

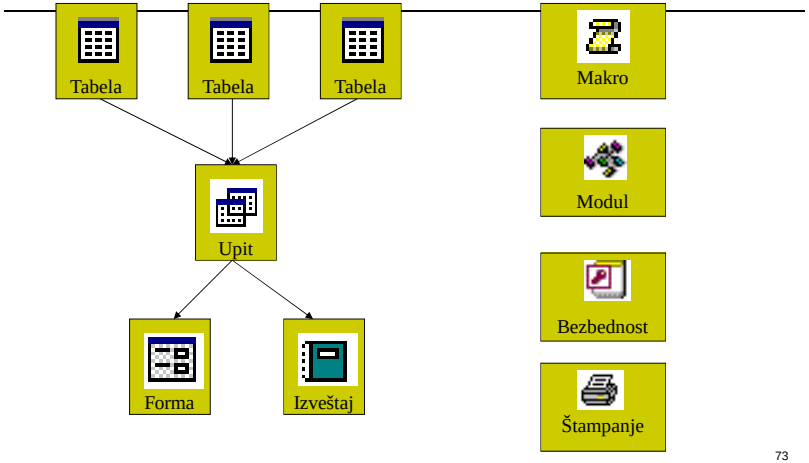
- MS Access kao DBMS
- Osnovne funkcije za podršku Access-a
- Režimi rada Access-a
- Bibliotečke baze podataka programa Access
- Postupak projektovanja baze podataka
- Tipovi relacija
- Normalizacija baze podataka
- Upiti
- Upravljanje grupama i korisnicima

MS Access kao DBMS

- Da bi se okvalifikovala kao potpun sistem za upravljanje relacionom bazom podataka, aplikacija mora da izvršava sledeće četiri osnovne funkcije, od kojih svaka ima sopstvenu prezentaciju za korisnika:
  - *Organizacija podataka* – obuhvata izradu i rukovanje tabelama koje sadrže podatke u konvencionalnom tabelarnom formatu koju Access naziva pogled (*Datasheet*).
  - *Povezivanje tabela i izdvajanje podataka* – povezuje više tabela prema relacijama između podataka radi izrade privremenih tabela, koje sadrže izabrane podatke. Access koristi upite da bi povezao tabele i izabrao podatke koji će se čuvati u privremenoj tabeli, koja se naziva objekat Recordset. Objekti Recordset nazivaju se virtuelne tabele, jer se čuvaju u memoriji računara umesto u datotekama baze podataka.
  - *Unos i uređivanje podataka* – zahteva projektovanje i implementaciju obrazaca za pregled, unos i uređivanje podataka kao alternativu tabelarnom prikazu. Obrasci su ti koji umesto aplikacije omogućavaju da kontrolišete prikazivanje podataka.
  - *Prikazivanje podataka* – zahteva izradu izveštaja koji mogu da sumiraju podatke u skupovima zapisa (Recordset). Njih možete da pregledate, štampate ili objavljujete na internetu ili intranetu.

72

Osnovne funkcije za podršku Accessa

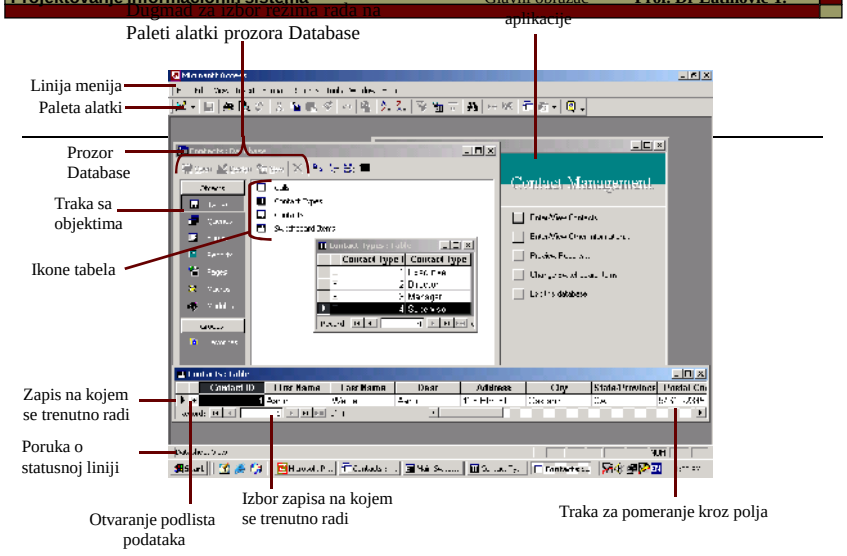


Osnovne funkcije za podršku Accessa *(nastavak)*

- ***Makroi*** su sekvence aktivnosti, koje automatizuju operacije nad bazom podataka koje se ponavljaju. Pri radu sa bazama podataka Access 2000, za automatizaciju se koristi Visual Basic for Application (VBA).
- ***Moduli*** su funkcije i procedure koje su napisane u programskom jeziku VBA. Funkcije VBA se koriste da bi se izvršavala složenija izračunavanja od onih koja se mogu lako izložiti pomoću niza konvencionalnih matematičkih simbola, ili za izračunavanja koja zahtevaju donošenje odluka. VBA potprogrami napisani su za izvršavanje operacije koje prevazilaze mogućnosti standardnih aktivnosti makroa što je jedan od razloga da se u Accessu napušta podrška makroima. VBA podprogrami se izvršavaju tako što se pridružuju odgovarajućim događajima, kao što je pritisak na dugme pomoću miša, koji se dešava kada je aktivni objekat neki obrazac ili izveštaj.
- ***Bezbednost*** sačinjavaju funkcije koje su dostupne kao stavke menija i preko VBA potprograma. Pomoću funkcija bezbednosti podataka može se dopustiti drugim osobama da koriste vašu bazu podataka, u višekorisničkom okruženju. Pristup možete dodeliti grupi korisnika ili pojedincima, ali i ograničite njihove mogućnosti za pregled ili modifikacije svih ili samo nekih tabela u bazi podataka.
- ***Štampanje*** dopušta da odštampate praktično sve što možete da pregledate u radnom režimu programa Accessa.
- Mogućnost ***objavljivanja*** unapređuju distribuciju informacija preko intranet korporacije i javne Internet mreže u obliku Word Wide Web strana. Access uvodi strane za pristup podacima (DAP – Data Access Page). One vam dopuštaju da napravite aplikaciju za prikazivanje i ažuriranje podataka na stranama, koje koriste prednosti jezika Dynamic HTML (DHTML) i Extensible Markup Language (XML).

Režimi rada Accessa

- Access ima tri osnovna radna režima:
  - ***Režim za pokretanje (Startup mode)*** omogućava da konvertujete, šifrujete, dešifrujete i popravite podatke iz baze, izborom komandi iz podmenija Database Utilities i Security, menija Tools, pre otvaranja baze podataka. Ove komande su dostupne samo ako baza podataka nije otvorena.
  - ***Režim projektovanja (Design mode)*** omogućava da napravite i modifikujete strukturu tabela i upita, razvijate obrasce za prikaz i uređivanje podataka, kao i da formatirate izveštaje za štampanje.
  - ***Režim izvršavanja (Run mode)*** prikazuje dizajn tabela, obrasca i izveštaja u posebnim prozorima za dokument. Makroe izvršavate tako što jedan od njih izaberete, a zatim izaberete režim izvršavanja. Ovaj režim se ne primenjuje na VBA module, jer se funkcije izvršavaju kada se pojave kao elementi upita, obrazaca ili izveštaja. Režim izvršenja za tabele i upite naziva se ***pogled Datasheet***, za obrasce ***pogled Form***, za strane za pristup podacima (DAP), ***pogled Page***, a za izveštaje ***pogled Print Preview***.



Bibliotečke baze podataka programa Access

- Još jedna kategorija datoteka, kod baza podataka u Accessu, pojavljuju se dopunski programi, koji se nazivaju i biblioteke.
- Dopunski programi predstavljaju bibliotečku bazu podataka Accessa, obično sa oznakama tipa .mde ili .mda, da bi se razlikovali od korisničkih baza podataka, a sa Accessom možete da ih povežete izborom alatke Add-In Manager (kojoj možete da pristupite izvorom opcije Tools, Add-Ins).
- Kada povežete neku biblioteku Accessa, svi elementi te bibliotečke baze podataka biće vam dostupni kada otvorite Access.

Postupak projektovanja sistema relacione baze podataka

1. Identifikujte objekte (izvori podataka) koje sistem baze podataka predstavlja.
2. Otkrijte veze između objekata.
3. Odredite značajna svojstva (atribute) i ponašanja objekata.
4. Ustanovite kako svojstva objekata utiču jedna na druge.
5. Izradite uvodni rečnik podataka da biste definisali tabele koje čine osnovu baze podataka.
6. Naznačite relacije između tabela baze podataka na osnovu veza između objekata koje se nalaze u njima i ove informacije uključite u rečnik podataka.
7. Uspostavite tipove ažuriranja i transakcija koji prave i menjaju podatke u tabelama, uključujući sve neophodne zahteve u vezi sa integritetom podataka.
8. Odredite način korišćenja indeksa kako biste ubrzali upite, s tim, da izrazito ne usporite dodavanja podataka u tabele ili da dodatno zauzimate velik prostor na disku.
9. Ako je potrebno da se obezbedi zaštita podataka, odredite ko može da pristupi i menja podatke u svakoj tabeli (zaštita podataka) i da promeni strukturu tabela.
10. Dokumentujte dizajn baze podataka, kao jedne celine, a i za tabele koje ona sadrži, napišite procedure za održavanje baze podataka, uključujući i one za izradu rezervne kopije datoteke i restauraciju.

Tipovi relacija

- RELACIJA JEDAN-PREMA-JEDAN

  Jednom redu u jednoj tabeli odgovara jedan red u drugoj tabeli. Ovakve tabele možete kombinovati u jednu tabelu koja se sastoji od svih kolona obe tabele.

- RELACIJA TIPA JEDAN-PREMA-VIŠE

  Povezuju jedan red iz jedne tabele sa više redova druge tabele preko relacije između primarnog ključa bazne tabele i odgovarajućeg spoljnjeg ključa u povezanoj tabeli.

- RELACIJE TIPA VIŠE-PREMA-JEDAN

  Povezuju više redova jedne tabele sa jednim redom druge tabele.

- RELACIJE TIPA VIŠE-PREMA-VIŠE I ČETVRTA NORMALNA FORMA

  Ne mogu da se izraze kao jednostavne relacije između dva sudelujuća entiteta. Njih ostvarujete tako što pravite tabelu koja ima relacije tipa više-prema-jedan sa dve bazne tabele.

Normalizacija baze podataka

- *Normalizacija* je formalizovani postupak za grupisanje atributa podataka u tabele i tabela u baze podataka.
- Ciljevi normalizacije:
  - Eliminisanje dupliranih informacija u tabelama.
  - Prilagođavanje budućim izmenama u strukturi tabela.
  - Umanjivanje uticaja strukturnih izmena baze podataka na korisničke aplikacije koje pristupaju podacima.
- Pravila normalizacije
  - Prva normalna forma zahteva da tabele budu ravne i da ne sadrže duplirane grupe.
  - Druga normalna forma zahteva da podaci u svim kolonama koje nisu deo ključa budu potpuno zavisni od primarnog ključa i svakog elementa (kolone) primarnog ključa kada je on složeni primarni ključ. Potpuno zavisni znači da je vrednost podatka u svakoj koloni koja nije deo ključa zapisa, na jedinstven način određena vrednošću primarnog ključa. Druga normalna forma uklanja veći deo nepotrebnih (redudantnih) podataka.
  - Treća normalna forma zahteva da sve kolone koje nisu deo ključa tabele budu zavisne od primarnog ključa tabele i nezavisne jedna od druge. Tabele moraju da odgovaraju prvoj i drugoj formi da bi bile sposobne za treću normalnu formu.

Upiti - Querys

- Access omogućava pravljenje četiri osnovna tipa upita, za postizanje različitih ciljeva:
  - *Upiti za izbor (Select Querys)* izdvajaju podatke iz jedne ili više tabela i prikazuju te podatke u tabelarnom obliku.
  - *Upiti unakrsnih tabela (Crosstab queries)* sumiraju podatke iz jedne ili više tabela u obliku radne tabele. Ovakvi upiti su korisni za analiziranje podataka i izradu grafika ili dijagrama, na osnovu sume vrednosti numeričkih polja većeg broja zapisa.
  - *Akcioni upiti (Action queries)* prave nove tabele iz tabela upita, ili prave velike izmene u nekoj tabeli. Takvi upiti dopuštaju da dodate ili obrišete zapise iz tabele, ili napravite izmene u zapisima na osnovu izraza koji unosite pri dizajnu upita.
  - *Parametarski upiti* (Parameter queries) čije se korišćenje ponavlja pri čemu se vrše samo jednostavne izmene njihovih kriterijuma. Kad izvršavate parametarski upit, Access prikazuje okvir za dijalog koji od vas zahteva da unesete novi kriterijum. Parametarski upiti zapravo nisu poseban tip upita, jer ove parametarske funkcije možete da dodate u upite za izbor, upite unakrsnih tabela i u akcione upite.

Upravljanje grupama i korisnicima

- Većina klijent-server baza podataka prepoznaje sledeće tri grupe korisnika baze podataka:
  1. *Administratori (Admins)* imaju ovlašćenja da pregledaju i ažuriraju postojeće tabele i dodaju ili obrišu tabele i druge objekte baze podataka iz baze podataka. Članovi grupe Admins obično imaju dozvolu da menjaju aplikacije sadržane u bazama podataka.
  2. *Obični članovi radnih grupa (Users)* imaju dozvolu da otvore bazu podataka, a po potrebi im se dodeljuje dozvola za pregledanje i menjanje baza podataka.
  3. *Povremenim korisnicima baza podataka (Guests)* često su dodeljena ograničena prava da koriste bazu podataka i objekte koje ona sadrži, ali im se ne dodeljuje korisnički nalog.

- Access ima dva nivoa bezbednosti:
  - **na nivou aplikacije** (zahteva da svaki korisnik Accessa unese korisničko ime i lozinku da bi mogao da pokrene Access) i
  - **na nivou datoteke** (uspostavio je mrežni operativni sistem, kao što je Windows NT Server i ona određenim korisnicima dozvoljava ili ne dozvoljava pristup zajedničkim omotnicama i/ili pojedinačnim datotekama).