UNIVERZITET METROPOLITAN BEOGRAD

IT250 - BAZE PODATAKA

Upravljanja informacijama; Arhitektura organizacije podataka

Lekcija 01

PRIRUČNIK ZA STUDENTE

IT250 - BAZE PODATAKA

Lekcija 01

UPRAVLJANJA INFORMACIJAMA; ARHITEKTURA ORGANIZACIJE PODATAKA

UVOD

Šta ćemo naučiti u ovoj lekciji?

U ovoj lekciji ćete naučiti:

• Razliku između podataka i informacija
• Šta je baza podataka, različite vrste baza podataka i zašto su važna sredstva za donošenje odluka
• Važnost dizajna baze podataka
• Kako su se savremene baze podataka razvile iz sistema datoteka
• O manama upravljanja podacima u sistemima datoteka
• Glavne komponente sistema baze podataka
• Glavne funkcije sistema upravljanja bazom podataka (DBMS)

Dobre odluke zahtevaju dobre informacije koje se dobijaju iz sirovih činjenica. Te sirove činjenice poznate su kao podaci. Podaci se najefikasnije upravljaju kada se skladište u bazi podataka. U ovoj lekciji ćete naučiti šta je baza podataka, šta ona radi i zašto daje bolje rezultate od drugih metoda upravljanja podacima. Takođe ćete saznati o različitim vrstama baza podataka i zašto je dizajn baze podataka toliko važan.

Baze podataka su se razvile iz računarskih sistema datoteka. Iako je upravljanje podacima u sistemima datoteka danas uglavnom prevaziđeno, razumevanje karakteristika sistema datoteka je važno jer su sistemi datoteka izvor ozbiljnih ograničenja upravljanja podacima. U ovoj lekciji ćete takođe naučiti kako pristup sistema baze podataka pomaže u otklanjanju većine nedostataka upravljanja podacima u sistemima datoteka.

Poglavlje 1

Potreba za čuvanjem podataka

ZAŠTO TRAJNO ČUVATI PODATKE?

Da bi se koristili i kada programi završe sa svojim izvršenjem

Mnoge aplikacije imaju potrebu da između dva sukcesivna izvršenja programa podatke trajno sačuvaju. U slučaju informacionih sistema, trajno pamćenje podataka predstavlja primarni zahtev. Poslovne i druge organizacije svoje informacione sisteme zasnivaju na slogovima podataka o drugim organizacijama, ljudima i fizičkim objektima i transakcijama. Podaci koji se u informacioni sistem unesu danas, se mogu upotrebiti u budućnosti; transakcije koje se u sistemu trenutno izvršavaju, se mogu odnositi na podatke koji su bili zapamćeni u prošlosti. Podaci koji se trenutno nalaze u operativnoj memoriji nekog računara, kao što je poznato nisu dostupni većem broju korisnika. Ukoliko veći broj korisnika treba da pristupi tim podacima, oni moraju biti sačuvani na nekoj vrsti deljivog sistema za pamćenje podataka tako da ih korisnici mogu pretraživati i pristupati im kad god je to potrebno. To je ono što se naziva trajnim čuvanjem podataka.

Podaci se trajno čuvaju na sekundarnim memorijama, tako da im se može pristupiti i kada programi završe sa svojim izvršenjem.

Trajni podaci mogu biti zapamćeni u datotekama ili bazama podataka koje su najčešće relacione ili objektno orijentisane. Korišćenje sistema za upravljanje bazama podataka (DBMS) ima niz prednosti nad datotekama.

Uzmimo za primer Google, poznat pretraživač na internetu. Iako Google ne želi da otkrije mnogo detalja o specifikacijama svog skladištenja podataka, procenjuje se da kompanija odgovara na preko 91 milion pretraga dnevno na kolekciji podataka koja ima nekoliko terabajta. Impresivno je da su rezultati ovih pretraga gotovo trenutno dostupni.

Kako ove kompanije mogu obraditi ovoliko podataka? Kako mogu sve to skladištiti, a zatim brzo pronaći samo činjenice koje donosioci odluka žele da znaju, baš kada žele da ih znaju? Odgovor je da koriste baze podataka. Baze podataka su specijalizovane strukture koje omogućavaju računarskim sistemima da veoma brzo skladište, upravljaju i pronalaze podatke. Gotovo svi moderni poslovni sistemi se oslanjaju na baze podataka; stoga je dobro razumevanje kako se ove strukture kreiraju i njihova pravilna upotreba od vitalnog značaja za svakog profesionalca informacionih tehnologija i sistema.

KAKO KRAJNJI KORISNIK VIDI PODATKE SAČUVANE U BAZAMA PODATAKA?

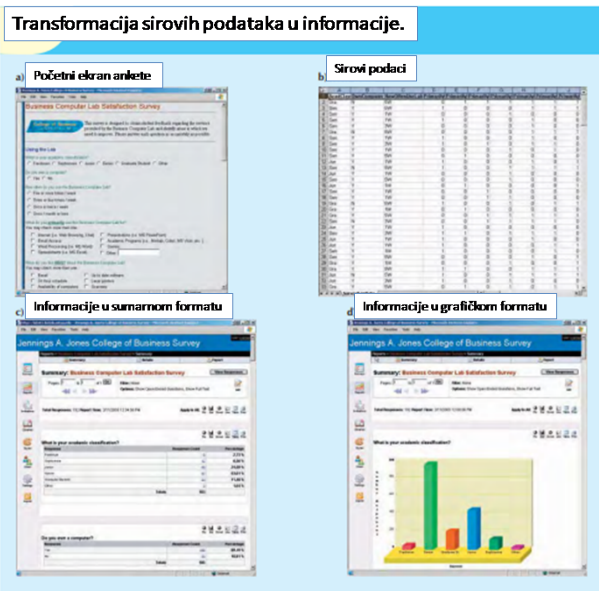
Ako su baze relacione, vidi tabele koje sadrže podatke, a ako su objektno orijentisane, vidi objekte i veze između njih. Korisnik ne vidi da su podaci zapamćeni u file-ovima

U principu, svi podaci se u računarima pamte u nekoj vrsti datoteka (engl. file).
Međutim, sistemi za upravljanje bazama podataka, obezbeđuju jedan sloj apstrakcije koji od korisnika skriva činjenicu da su podaci zapamćeni u file-ovima. Korisnik ne vidi da su podaci zapamćeni u file-ovima već ako su baze relacione, korisnik vidi tabele koje sadrže podatke, a ako su objektno orijentisane, vidi objekte i veze između njih. Način na koji baza podataka pamti tabele u file-ovima je važan samo za administratora baze podataka koji mora da zna gde su podaci zapamćeni, pravi back-up podataka itd.

Izbor sistema za upravljanje bazom podataka ima značajan uticaj na rad projektanata baze podataka. Ako se koristi objektna baza podataka, tada je za projektovanje načina na koji se objekti pamte potrebno relativno malo napora. Ako se koristi relaciona baza podataka koja se bazira na korišćenju tabela, potrebno je mnogo više rada.

TRANSFORMACIJA SIROVIH PODATAKA U INFORMACIJE

Grafički prikaz transformacije sirovih podataka u informacije



Slika 1.1 Transformacija sirovih podataka u informacije. Izvor: [Autor; 1]

Da biste razumeli šta pokreće dizajn baza podataka, morate razumeti razliku između podataka i informacija. Podaci su sirove činjenice. Reč "sirove" ukazuje na to da činjenice još

nisu obrađene da bi otkrile svoje značenje. Na primer, pretpostavimo da želite da saznate šta korisnici računarske laboratorije misle o njenim uslugama. Tipično, počeli biste anketiranjem korisnika kako biste ocenili performanse računarske laboratorije. Slika 1, Panel A, prikazuje veb obrazac ankete koji omogućava korisnicima da odgovore na vaša pitanja. Kada se obrazac ankete popuni, sirovi podaci forme se čuvaju u repozitorijumu podataka, kao što je prikazano na slici 1, Panel B. Iako sada imate činjenice u rukama, one nisu posebno korisne u ovom formatu - čitanje stranica nula i jedinica verovatno neće pružiti mnogo uvida. Stoga, transformišete sirove podatke u sažetak podataka kao što je prikazano na slici 1, Panel C. Sada je moguće brzo dobiti odgovore na pitanja poput "Koja je struktura naše korisničke baze računarske laboratorije?" U ovom slučaju, brzo možete utvrditi da je većina vaših korisnika treća godina (24,59%) i četvrta godina (53,01%). Budući da grafika može poboljšati vašu sposobnost brzog izvlačenja značenja iz podataka, prikazujete grafički prikaz sažetka podataka u obliku trakastog dijagrama na slici 1.1, Panel D.

Poglavlje 2

Čuvanje podataka u fajlovima

KAKO SE PODACI ČUVAJU U FILE-OVIMA?

File je podeljen na individualne slogove od kojih svaki predstavlja grupu izvesnog broja polja u kojima su smešteni podaci. Slogovi mogu imati različite forme

Najjednostavniji način za trajno čuvanje podataka u računarskim informacionim sistemima je korišćenje file-ova. Mnogi korisnici personalnih računara su familijarni s idejom i file-ovima. Word procesori na primer, svoje podatke pamte u file-ovima; browser-i ih download-uju sa web site-ova i takođe čuvaju u fajlovima. Na najnižem nivou, file predstavlja niz bajtova zapamćenih podataka na nekom fizičkom medijumu.

Programski jezici i u nekim slučajevima operativni sistemi, takođe koriste strukture file-ova. U ovakvoj strukturi, file je podeljen na individualne slogove od kojih svaki predstavlja grupu izvesnog broja polja koji predstavljaju podatke koje treba zapamtiti u file-u. Na isti način kao što svaki objekat sadrži određeni broj atributa, od kojih svaki sadrži određenu vrstu podataka o tom objektu, svako polje u slogu sadrži određenu vrstu podataka kojim se opisuje slog.

Slogovi u datoteci mogu imati različite forme:

- Slogovi fiksne dužine: svaki slog se sastoji od više polja, od kojih svako polje ima fiksnu dužinu u bajtovima. Svaki slog je iste, fiksne dužine, a sa prethodnog na sledeći slog je moguće doći preskakanjem fiksnog broja bajtova

- Slogovi promenljive dužine – svaki slog se sastoji od izvesnog broja polja od kojih svako može imati maksimalnu dužinu ali i minimalnu dužinu od nula bajtova. Polja su često razdvojena delimiterima ili specijalnim karakterima. Dužina svakog sloga može biti zapamćena na početku sloga, pa se na početak sledećeg sloga može doći preskakanjem navedenog broja karaktera.
- Zaglavlja i detalji – postoje dve vrste slogova: svaki slog transakcije se sastoji od sloga zaglavlja koji je praćen izvesnim brojem slogova detalja. Ovaj pristup se može koristiti u mnogim poslovnim dokumentima kao što su porudžbenice, fakture. U svakom slogu je zapamćen tip sloga. Broj slogova detalja se može pamtiti u zaglavlju tako da je moguće reći gde počinje sledeći slog zaglavlja.
- Tagovani podaci – podaci mogu imati složenu strukturu, kao što su na primer objektno orijentisani sistemi – i ponekad je neophodno objekte različitih klasa pamtiti u istom file-u. Svaki objekat i atribut može imati tag koji je opisan i na neki način saopštava programeru koji čita file o kakvom se podatku radi. Ovaj pristup se koristi za podatke u file-ovima koji koriste HTML i XML.

KAKO FAJLE-OVI MOGU BITI ORGANIZOVANI?

Serijski, sekvencijalno i random

Postoje tri načina na koje file-ovi mogu biti organizovani: serijski, sekvencijalni i random:

- Serijska organizacija file-a: Svaki sledeći slog u file-u se upisuje na kraj prethodnog sloga. Ako se slog izbriše, file mora biti kopiran od početka do izbrisanog sloga, koji se preskače, a tada se kopira i ostatak file-a.
- Sekvencijalna organizacija file-a: U osnovnoj formi sekvencijalne organizacije, svaki slog se upisuje u file po nekom predefinisanom redosledu, koji se najčešće bazira na vrednosti jednog polja u slogu, kao što je na primer broj fakture. Slogovi se brišu na isti način kao kod serijskih file-ova. Svaki slog mora biti dodat file-u na svoje odgovarajuće mesto, i ako je neophodno uneti novi slog u file, file se kopira do tačke gde je slog insertovan, novi slog se upisuje u file i tada se kopira ostatak file-a.
- Random organizacija file-a: Slogovi se dodaju u file korišćenjem preciznog algoritma koji dozvoljava da se slogovi upisuju i čitaju direktno bez obaveze da se čita ostatak file-a. To znači da ako ste vi izabrali bilo koji slog random (slučajno), moguće je da njemu pristupite manje ili više direktno ne prolazeći kroz ostatak file-a. Algoritam obično konvertuje polje ključa svakog sloga u adresu file-a kojoj se može direktno pristupiti.

SERIJSKI I DIREKTAN PRISTUP PODACIMA

Serijski pristup podrazumeva čitanje file-ova slog po slog; Kod direktnog pristupa se koristi algoritam za konverziju vrednosti ključa u adresu sloga

Zavisno od izabrane organizacije file-ova, postoje različiti načini za pristup podacima u file-ovima: serijski, indeks-sekvencijalni i direktan:

1. Serijski pristup podacima: Serijski se može pristupiti serijskim i sekvencijalnim file-ovima. Da bi se pronašao određeni slog, neophodno je čitati file slog po slog sve dok se određeni slog ne locira.
2. Direktan pristup podacima: metoda direktnog pristupa se zasniva na korišćenju algoritama za konverziju vrednosti polja ključa u adresu sloga u file-u.
    - Prvi i najjednostavniji je relativna adresa. Ovaj algoritam pristupa zahteva korišćenje slogova fiksne dužine i pozitivne sukcesivne vrednosti ključeva. Ako je svaki slog dugačak 200 bajtova, tada će prvi slog početi na prvom bajtu, drugi na bajtu 201, treći na bajtu 401 itd.
    - Drugi mogući algoritam je hashing adresiranje gde se mogu koristiti ključevi u bilo kojoj formi. Kao i kod indeks sekvencijalnih file-ova, file-u se dodeljuje fiksan broj blokova. Tada, da bi se odredilo kojem će se bloku dodeliti dati slog, se vrši heširanje ključa. Funkcija heširanja je algoritam koja uzima ASCII string i konvertuje ga u integer. Postoji više pristupa.

Najjednostavniji pristup je uzeti karaktere u stringu i konvertovati ih u njihove ASCII vrednosti (na primer A je 65). Te ASCII vrednosti se sumiraju i suma podeli sa brojem blokova u file-u. Ostatak deljenja daje broj bloka u file-u u kojem treba staviti slog, počevši od bloka 0. Ako je taj blok pun, koristi se dodatni blok kao overflow a njegova adresa se čuva u bloku u kojem je slog trebalo da bude zapamćen. Slika 2.1. prikazuje organizaciju file sa direktnim hashing pristupom.


Khan -> Kha

ASCII Values = 75, 104, 97

75+104+97 = 276

276 divided by 7 leaves a modulo of 3

So Khan will be added in Block 3.

Block no. in file	Records
0	Hao
1	Ford Farmer
2	Fimin Firth
3	Harris
4	Hastings Gomez Jacobson
5	Ibrahim Finch Fern
6	Hanson

Slika 2.1 Primer hešing adresiranja kojim se ASCII string konvertuje u intedžer [Izvor: NM IT350-2020/2021]

INDEKS-SEKVENCIJALNI PRISTUP PODACIMA

Pronalaženje sloga se vrši sekvencijalnim pretraživanjem područja indeksa, dok se ne nađe traženi ključ, a zatim se na osnovu adrese ključa direktno pristupa traženom slogu

Pristup sekvencijalnim file-ovima se može unaprediti održavanjem indeksa nad poljem koje se koristi za uređivanje podataka u file-u (polje ključa). Indeks sekvencijalni file-ovi se koriste tamo gde je neophodno čitati file sekvencijalno, slog po slog, i korišćenjem ključa otići pravo do određenog sloga. Mehanizam indeksiranja koji se koristi u indeks sekvencijalnim file-ovima nas vraća u vreme mainframe operativnih sistem.

U ovakvom pristupu se izdvajaju tri područja podataka:

- Glavno područje u koje se upisuju slogovi prilikom unosa ili ažuriranja slogova.
- Područje indeksa koje obično formira operativni sistem prilikom unosa ili ažuriranja slogova
- Prelazno područje koje služi da se u njega smeste slogovi koji ispadaju iz glavog područja zbog ubacivanja novih slogova.

Pronalaženje nekog sloga se vrši sekvencijalnim pretraživanjem područja indeksa, dok se ne nađe traženi ključ, a zatim se na osnovu adrese koja odgovara nađenom ključu brzo pronalazi

traženi slog. Vreme potrebno za pronalaženje slogova se uglavnom troši na sekvencijalno pretraživanje područja indeksa.

Da bi se podržali veliki file-ovi potrebno je imati dva ili više nivoa indeksa. Na primer može postajati master indeks i serija blok indeksa. U master indeksu se čuvaju vrednosti polja ključa zadnjeg sloga u svakom blok indeksu. Svaki blok indeksa čuva polja ključa u zadnjem slogu svakog bloka iz skupa blokova. Da bi se pronašao slog po njegovom ključu, master indeks se čita sve dok se ne pronađe vrednost ključa koja je veća ili jednaka ključu sloga koji se traži. To omogućava da se ode do bloka u kojem je sačuvan slog. Slogovi u bloku se nadalje čitaju sekvencijalno, sve dok se ne pronađe željeni slog. Sličan pristup se koristi i za dodavanje slogova indeks sekvencijalnim file-ovima. Slika 2.2. prikazuje organizaciju podataka i indeksa u indeks sekvencijalnim file-ovima.

Master Index

Record Key	Block address
Feng	1
Patel	2
Zarzycki	3

Block Index

Record key	Block address
Finlayson	55
Gomez	56
Hanson	
Jacobson	
...	...
Patel	

Data records in blocks (samo ključevi su prikazani)

Block address	Records
55	Fern Finch Finlayson
56	Finn Firmin Ford Gangar Gomez
57	Gordon Govan Hamer Hanson
58	Ho Ibrahim Jacobson
...	...

Slika 2.2 Organizacija indeks sekvencijalnih file-ova [Izvor: NM IT350-2020/2021]

PROBLEMI ČUVANJA PODATAKA U FAJLOVIMA

Redudantnost podataka, nemogućnost njihovog sinhronizovanog ažuriranja, zavisnost programskog koda od organizacije fajlova

Čuvanje podataka u file-ovima može da stvori mnogo problema:

- Kako broj aplikacija raste, raste i broj različitih file-ova. Neki od tih file-ova mogu da sadrže iste podatke za različite aplikacije u različitim formama. Na taj način se podaci dupliraju, zauzimajući bespotrebno prostor na diskovima. To je poznato kao redundantnost podataka.
- Postoji rizik da ažuriranje podataka u različitim aplikacijama neće biti sinhronizovano. Na primer, adresa kupca se može promeniti u jednom file-u ali ne i u drugom, što podatke može učiniti nekonzistentnim.
- Svaka aplikacija mora da sadrži svoje sopstvene mehanizme za čuvanje podataka u svom skupu file-ova, i ukoliko se podaci promene ili se promeni način na koji su oni zapamćeni, tada svaki program u okviru aplikacije koji pristupa tim podacima mora biti prilagođen izmenama. Zbog toga je dodavanje novih programa aplikaciji koja koristi isti podatke i istovremeno zahteva čuvanje dodatnih podataka, veoma teško.

- Prilagođavajući se promenama poslovnih zahteva, korisnici mogu želeti da podacima pristupe na novi način, na primer pravljenjem izveštaja u kojem se kombinuju podaci iz različitih aplikacija. To se ne može implementirati bez značajnih napora programera.
- Još jedna mana skladišta podataka u sistemima datoteka je nedostatak bezbednosti i ograničeno deljenje podataka. Deljenje podataka i bezbednost su tesno povezani. Deljenje podataka među više geografski raspršenih korisnika nosi sa sobom mnogo rizika po pitanju bezbednosti. Čak i kada se pokuša poboljšati bezbednost sistema i podataka, sigurnosni mehanizmi obično imaju ograničen obim i efikasnost.

Prvi korak u rešavanju ovih problema je analizirati zahteve za pamćenjem podataka različitih aplikacija u okviru organizacije i napraviti korporativnu bazu podataka koja će sadržati podatke iz različitih aplikacija.

Poglavlje 3

Čuvanje podataka u BP

SISTEMI ZA UPRAVLJANJE BAZOM PODATAKA (ENG. DATA BASE MANAGEMENT SYSTEM - DBMS)

Cilj korišćenja sistema za upravljanje bazom podataka je razdvojiti detalje o načinu na koji se podaci fizički pamte od načina na koji se oni koriste u aplikativnim programima.

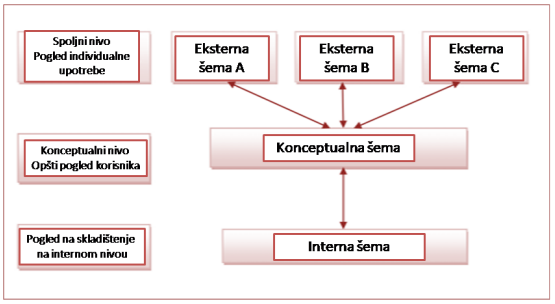
Baze podataka su nastale u pokušaju da se reše problemi sa čuvanjem podataka u file sistemima.
Prvi korak u rešavanju ovih problema je analizirati zahteve za pamćenjem podataka različitih aplikacija u okviru organizacije i napraviti korporativnu bazu podataka koja će sadržati podatke iz različitih aplikacija. Svaka aplikacija tada za zadovoljenje svojih zahteva koristi podskup podataka iz zajedničke baze.

Drugi korak je iskoristiti sistem za upravljanje bazom podataka koji će omogućiti da se podaci organizuju i njima upravlja tako da se mehanizmi za pamćenje podataka razdvoje od aplikativnih programa. To se postiže pravljenjem logičkog modela podataka koji je nezavistan od zahteva aplikacije i koji se u bazi podataka može čuvati na različite načine.

Danas je najčešće u upotrebi Tri-šeme arhitektura koja je prikazana na slici 3. 1 i koristi se pristup podacima u bazama podataka. Sastoji se od:

1. Eksterne šeme koja predstavlja način na koji se podaci iz baza podataka koriste u aplikativnim programima.
2. Konceptualne šeme koja predstavlja logički model podataka i nezavisna je i od eksterne šeme i od detalja o tome kako se podaci pamte.
3. Interne šeme, kojom se definišu file-ovi za pamćenje podataka.

Cilj ovog pristupa je izolovati aplikativne programe od detalja o tome kako je zapamćena bilo koja stavka podataka. To je osnovno za način na koji radi Sistem za upravljanje bazom podataka (DBMS).



Slika 3.1 Tri-šeme arhitektura. Izvor: [Autor]

ŠTA OBEZBEĐUJE DBMS?

Alate koji se mogu koristiti za upravljanje podacima kao što su: DDL, DML, upravljanje transakcijama, rešavanje konkuretnosti, oraničenje itegriteta itd.

DBMS obezbeđuje više od samog načina za pamćenje podataka koje mogu da koriste više aplikacija. On obezbeđuje alate koji se mogu koristiti za upravljanje podacima.

1. Data definition language (DDL): se koristi za specifikaciju podataka koji se drže u sistemu za upravljanje bazom podataka i za strukture koje se koriste za njihovo čuvanje
2. Data manipulation language (DML): se koristi da bi se specificirala ažuriranja i pretraživanja podataka u DBMS.
3. Ograničenja integriteta: se specificiraju da bi se održao integritet podataka
4. Upravljanje transakcijama: Ažuriranja nad bazama podataka se specificiraju kao transakcije u okviru kojih se moraju izvršiti sve komponente ažuriranja jer se u suprotnom mora izvršiti roll back transakcije, što znači da se ona neće trajno zapamtiti u bazi podataka (izvršiti commit transakcije).
5. Konkurentnost: Više korisnika može da simultano koristi bazu podataka i ažurira njen sadržaj.
6. Zaštita: pristup podacima u bazi podataka i dozvole koje su dodeljene različitim korisnicima za različite nivoe pristupa (na primer select, update, delete, može biti kontrolisan)

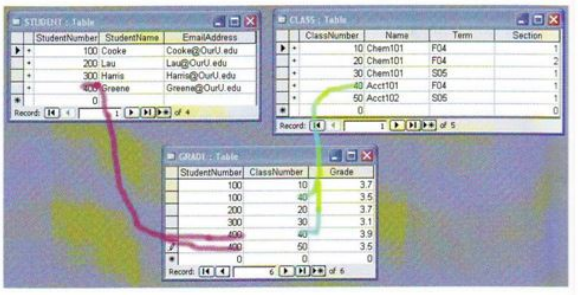
Poglavlje 4

Relacione baze podataka (RDBMS)

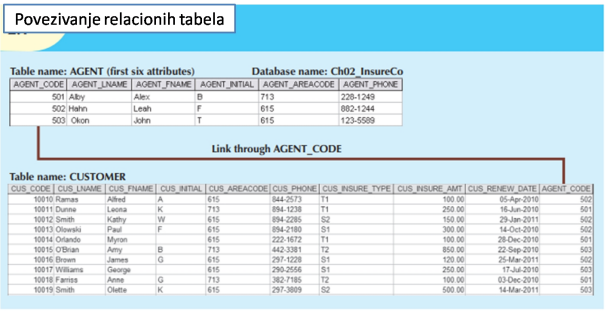
OD ČEGA SE SASTOJI RELACIONA BAZA PODATAKA?

Od podataka koji su smešteni u tabelama i relacijama koje postoje između redova tabela

Relacionu bazu podataka sačinjavaju svi podaci sa kojim radimo, smešteni u jednu ili više tabela. Slika 4.1. prikazuje bazu podataka sa tri tabele: STUDENT koja sadrži podatke o studentima, GRADE (USPEH) koja sadrži podatke o uspehu studenata i CLASS koja sadrži podatke o predmetima koje studenti izučavaju. Međutim, baza podataka na slici ne prikazuje samo podatke o studentima, uspehu i predmetima već pokazuje i relacije koje postoje među redovima tih tabela. Na primer, StudentNumber 400 ima ocenu 3.9 na predmetu 40 koji označava Acct101 i ocenu 3.5 na predmetu 50 koji označava predmet Acct102.
Slika zapravo prikazuje jednu veoma važnu karakteristiku relacionih baza podataka: u njima se ne pamte samo podaci već i relacije među podacima. Svaka tabela u relacionoj bazi podataka se sastoji od redova podataka, a svaki red sadrži vrednosti atributa koje su organizovane u kolone. Svaka kolona sadrži vrednost podataka istog tipa atributa.



Slika 4.1.1 Tabele relacione baze podataka [Izvor: NM IT350-2020/2021]



Slika 4.1.2 Povezivanje relacionih tabela. Izvor: [Autor, 1]

Zajednička veza između tabela KUPAC (CUSTOMER) i AGENT (slika 4.2) omogućava vam da povežete kupca sa njegovim prodajnim agentom, iako se podaci o kupcima čuvaju u jednoj tabeli, a podaci o prodajnim predstavnicima čuvaju u drugoj tabeli. Na primer, lako možete utvrditi da je agent za kupca Dunne Alex Alby, jer za kupca Dunne, polje AGENT_CODE u tabeli KUPAC ima vrednost 501, koja se poklapa sa vrednošću polja AGENT_CODE u tabeli AGENT za Alex Alby-a. Iako su tabele nezavisne jedna od druge, lako možete povezati podatke između njih. Relacioni model pruža minimalni nivo kontrolisane redundancije kako bi se eliminisala većina redundancija koje se često javljaju u sistemima datoteka.

TABELE META PODATAKA

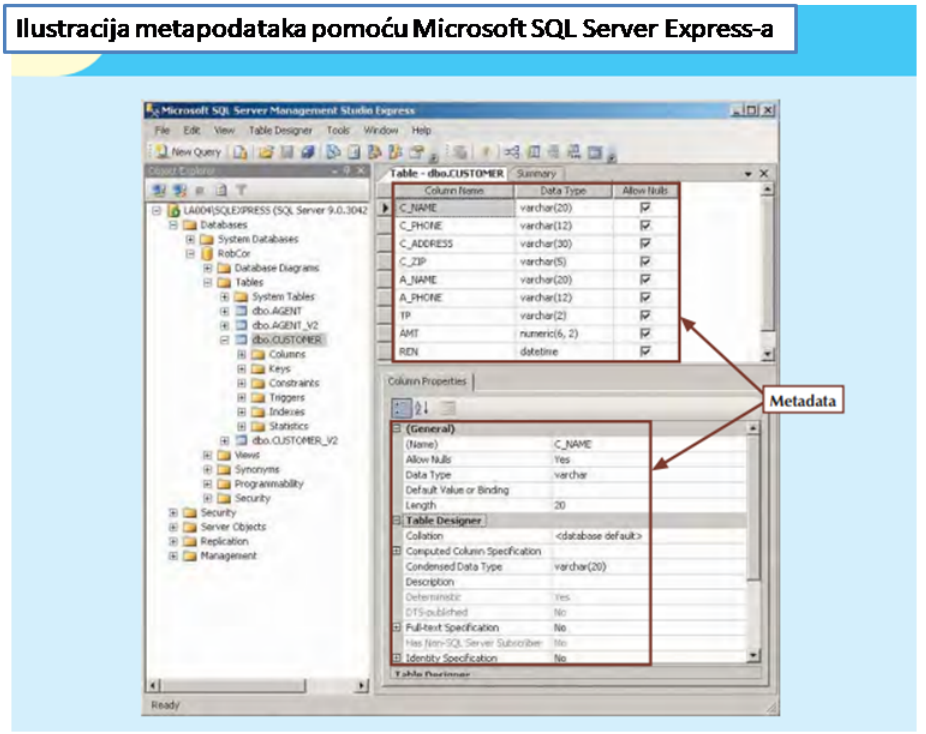
Opisuju tabele, kolone i ostale elemente baze podataka

Relaciona baza podataka pored tabela korisničkih podataka sadrže i tabele podataka koje opisuju te korisnike podataka. Ovi opisi se zovu metapodaci jer su to podaci o podacima. Format i forma metapodataka se razlikuje zavisno od DBMS-a. Slika 4.3 prikazuje generičku tabelu metapodataka koja opisuje tabele i kolone baze podataka, tj. prikazuje kako Microsoft SQL Server Express prikazuje definiciju podataka za tabelu KUPAC. Metapodaci se mogu pretraživati da bi se odredilo da li neka tabela, kolona, indeks ili druga struktura pripadaju bazi podataka.

Sistem upravljanja bazom podataka (DBMS) čuva definicije elemenata podataka i njihove veze (metapodaci) u rečniku podataka. Svi programi koji pristupaju podacima u bazi podataka rade preko DBMS-a. DBMS koristi rečnik podataka da pronađe potrebne strukture i veze komponenti podataka.

Da bi se iz baza podataka proizvodile različite vrste informacija, koristi se jezik koji se zove Structured Query Language (SQL).

Relacioni DBMS su trenutno najšire korišćeni DBMS-ovi. Najpoznatiji je svakako Access, a zatim Oracle, SQL-Server, DB2, Informix, Ingres, Sybase, MySQL itd. Zato, kada se govori o bazama podataka, misli se na relacione baze.



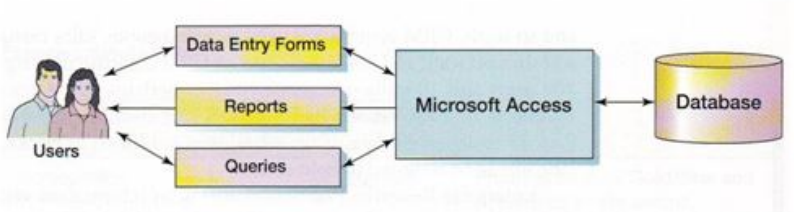
Slika 4.1.3 Metapodaci. Izvor: [Autor, 1]

PRIMER APLIKACIJE KOJA KORISTI RDBMS (MICROSOFT ACCESS)

Korisnici su u interakciji sa bazom preko aplikacije koja može da sadrži forme za unos podataka, da generiše izveštaje ili postavlja upite

Slika 4.4 može poslužiti kao primer za prikaz jedne jednostavne aplikacije koja koristi Microsoft Access sistem za upravljanje bazom podataka.

Microsoft Access je popularan softver za upravljanje bazom podataka koji omogućava korisnicima da kreiraju, uređuju i upravljaju svojim relacijskim bazama podataka. Sa Microsoft Access-om, korisnici mogu da definišu tabele, veze između tabela, upite za pretragu i izveštaje za prikaz podataka na različite načine.



Slika 4.1.4 Arhitektura jednostavnih aplikacija koje koriste RDBMS [Izvor: NM IT350-2020/2021]

Forme za unos podataka su deo aplikacija koje rade nad bazama podataka. Aplikacije prihvataju podatke od korisnika, obrađuju ih prema zahtevima po kojima je aplikacija napravljena i koriste SQL za transfer podataka prema i od baze podataka.

U slučaju Access-a, aplikacije kreiraju i obrađuju forme, izveštaje i upite. U drugim slučajevima, aplikacije rade mnogo više.

Primer: Pretpostavimo da imate malu prodavnicu i želite da kreirate aplikaciju za praćenje inventara i upravljanje prodajom. Možete koristiti Microsoft Access kao RDBMS za izgradnju ove aplikacije.

U Microsoft Access-u možete kreirati tabelu "**Proizvodi**" koja će sadržati informacije o svakom proizvodu u prodavnici, kao što su naziv, cena, količina na stanju, itd. Takođe možete kreirati tabelu "**Prodaja**" koja će sadržati informacije o svakoj prodaji, kao što su datum, proizvod koji je prodat, količina, itd. Zatim možete definisati odgovarajuće veze između tabela, na primer, vezu između tabele "**Proizvodi**" i "**Prodaja**" na osnovu polja "ID proizvoda". To će omogućiti da se poveže svaka prodaja sa odgovarajućim proizvodom.

Nakon što ste definisali strukturu baze podataka, možete koristiti korisnički interfejs Microsoft Access-a za unos novih proizvoda, praćenje inventara, unos prodaja i generisanje izveštaja o prodaji. Možete izvršavati upite nad bazom podataka kako biste dobili odgovore na razna pitanja, na primer, koliko je proizvoda prodato u određenom periodu ili koji proizvodi imaju najviše prodaja.

VIDEO

What is Database?

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

4.1 Fizička organizacija podataka u RDBMS

SMEŠTANJE PODATAKA NA FIZIČKIM UREĐAJIMA ZA SKLADIŠTENJE

Način na koji se podaci čuvaju na spoljašnjim memorijama, metode pristupa podacima, vrste upita itd. imaju veliki uticaj na dobijanje rezultata.

Kao što smo već videli, baza podataka se sastoji od tabela koje predstavljaju logičnu formu pogleda na podatke. Ali stvarni podaci se čuvaju u fizičkoj memoriji. Baza podataka sadrži veliku količinu podataka koji se nalaze na fizičkim uređajima za skladištenje podataka - diskovima računara. Na fizičkim memorijskim uređajima, podaci se ne mogu čuvati onakvi kakvi jesu već se pretvaraju u binarni format. Svaki memorijski uređaj sadrži mnogo blokova podataka, od kojih će svaki moći da sačuva određenu količinu podataka. Da bi se podaci sačuvali na spoljašnjem disku, moraju se mapirati blokove.

Svaki korisnik koji želi da podatke pregleda ili ih izmeni, jednostavno aktivira SQL upit i dobija rezultat na ekranu. Bilo koji od ovih upita bi trebalo da da rezultate što je brže moguće. Postavlja se pitanje: kako se ovim podacima pristupa na fizičkoj memoriji? Da li jednostavnije čuvanje podataka na memorijskim uređajima daje bolje rezultate prilikom izvršenja upita? Sigurno ne. Način na koji se podaci čuvaju na spoljašnjim memorijama, metode pristupa podacima, vrste upita itd. imaju veliki uticaj na dobijanje rezultata. Stoga je organizovanje podataka u bazi podataka, a samim tim i na spoljašnjim memorijama, jedna od važnih tema o kojima treba razmišljati.

TIPOVI ORGANIZACIJE FILE-OVA

Za čuvanje podataka na fizičkim diskovima računara kada su oni smešteni u bazi podataka mogu se koristiti različite organizacije podataka

U bazi podataka je smešteno mnogo podataka. Svaki podatak je deo neke od grupa podataka koje se nazivaju tabele i koje sadrže veliki broj slogova. Svaki korisnik će ove slogove na ekranu videti u obliku tabela mada se oni na diskovima računara čuvaju u obliku file-ova. Obično se svi slogovi tabele čuvaju u jednom file-u.

Kao što smo videli, nije tako lako pristupiti sadržaju file-ova tj. zapisima u fizičkoj memoriji. Oni nisu sačuvani kao tabele i naši SQL upiti neće raditi. Potrebno je primeniti odrešene metode pristupa, tj. da bismo pristupili file-ovima, slogove moramo čuvati u određenom redosledu tako da se oni mogu lako preuzeti.

Čuvanjeslogova u određenom redosledu u okviru file-a se naziva organizacija file-ova. Glavni cilj organizacije file-ova je:

- optimizacija načina izbora slogova u pogledu brzine pristupa:
- svaka transakcija unošenja, ažuriranja ili brisanja slogova mora biti laka, brza i ne sme biti štetna za druge slogove.
- kao rezultat unošenja, ažuriranja ili brisanja slogova ne smeju se pojaviti nikakvi duplikati podataka
- slogovi treba da se efikasno čuvaju tako da troškovi skladištenja budu minimalni.

Neke od organizacije file-ova su:

- Sekvencijalna organizacija: Slogovi se čuvaju i sortiraju u redosledu u kojem dolaze. Nedostatak ove organizacije je što se sortiranje podataka vrši svaki put kada se vrši za unošenje / brisanje / ažuriranje slogova što sistem čini sporim.
- Organizacija gomile (engl.heap organization): Novi slogovi se čuvaju na kraju file-a ali se adrese za njihovo smeštanje na disku generišu random. Ovakva organizacija je najprikladnija za skupno unošenje slogova i male datoteke / tabele. Nedostaci ove organizacije su što su zapisi raspršeni po disku te se zbog toga povećava veličina potrebne memorije.
- Heš/direkna organizacija: Slogovi se čuvaju na adresama koje se generišu nekim hash algoritmom. Prednosti ove organizacije su brži pristup slogovima; nema potrebe za sortiranjem; može se obraditi više transakcija; pogodna je za online transakcije.

- Indeks sekvencijalni pristup: Slogovima se dodaju indeksi sa adresama. Prednosti: pretraživanje slogova je brže; ovakva organizacija je pogodna za velike baze podataka jer bilo koja od kolona se može koristiti kao kolona ključa.
- Organizacija B+ stabla: Slogovi se čuvaju u strukturi stabla, efikasna je za pretraživanje podatka i performanse se ne umanjuju u slučaju unošenja/brisanja/ažuriranja slogova.

4.2 Komponente RDBMS-a

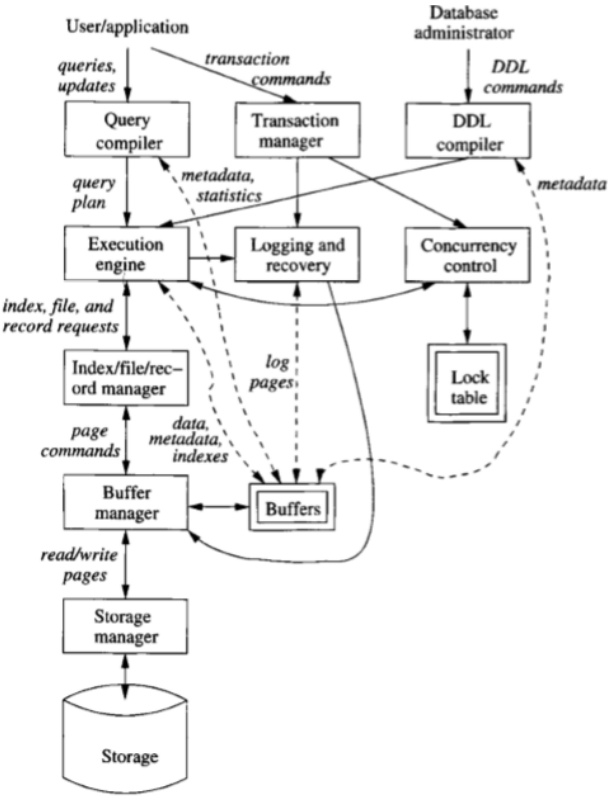
ŠEMA RDBMS

Na početku se mogu razdvojiti dve različite vrste komandi RDBMS-a: za obične korisnike i administratore

Na slici 5.1. data je potpuna šema RDBMS-a. Kvadratima su predstavljene komponente sistema, duplim kvadratima su predstavljene unutar memorijske strukture podataka, punim linijama je predstavljena kontrola unutar sistema i tok podataka u sistemu dok je isprekidanim linijama predstavljen samo tok podataka.

Dijagram RDBMS-a se na osnovu šeme sa slike 1. može raščlaniti na nekoliko nivoa. Prvo, na vrhu se izdvajaju dve različite vrste komandi DBMS-a:

1. Obični korisnici i aplikativni programi (eng. User/Application): koji samo pristupaju ili menjaju podatke u bazi podataka
2. Administratori baze podataka (eng. Database administrator): osobe koje su odgovorne za strukturu i šemu baze podataka.



Slika 4.2.1 Komponente RDBMS, Izvor:[2]

OBRADA DDL KOMANDI

DDL komande predstavljaju komande koje su jednostavnija za obradu i izvršavaju ih administratori sistema

DDL komande predstavljaju komande koje su jednostavnije za obradu, i prikazane su u gornjem desnom uglu slike 1. Npr. administrator baze podataka DDL komandama za bazu podataka univerziteta kreira tabele za studente, smerove i kurseve koje student pohađa i ocene na kursevima, takođe ovim komandama on može da ograniči da studenti jedino mogu da dobiju ocene 5, 6, 7, 8, 9 ili 10. Ova struktura i ograničenja su delovi šeme baze podataka. Na slici 1. je takođe prikazano da se za izvršenje komandi kojima se menja struktura baze podataka, zahtevaju specijalna ovlašćenja i uglavnom ovakve komande izvršavaju administratori baza podataka, jer izmene nastale ovim komandama mogu da imaju veliki uticaj na samu bazu i podatke. Komande DDL-a se parsiraju kroz DDL procesor i izvršavaju na execution engine-u koji kroz index/file/record menadžer dodaje meta podatke kako bi se dobila šema informacija unutar baze podataka.

OBRADA UPITA I AŽURIRANJA

Korisnik ili aplikativni program pokreće neki upit, koristeći jezik za upravljanje podacima (eng. data-manipulation language (DML)

Korisnik ili aplikativni program pokreće neku akciju, koristeći jezik za upravljanje podacima (engl. Data Manipulation Language - DML). Ovakve komande ne menjaju strukturu baze podataka već samo menjaju sadržaj podataka unutar same baza (ukoliko je upit za izmenu podataka) ili samo prikazuju podatke iz baze (ukoliko je upit za prikaz podataka). DML komande se izvršavaju na sledeći način:

Upit se prvo parsira i optimizuje u kompajleru upita. Rezultat ovoga je plan izvršenja upita, odnosno koraci koje DBMS treba da preuzme kako bi izvršio upit, koji se predaje execution engine-u.

Execution engine izdaje niz zahteva menadžeru skladišta za zapise podataka ili torke relacija. Menadžer skladišta sadrži podatke o skladištenim podacima, veličini podataka i indeksima, kako bi ih što brže našao i kako bi se upit izvršio brže. Zahtev za dostavljanje podataka se predaje menadžeru bafera.

Zadatak menadžera bafera je da dostavi podatke iz sekundarnog skladišta gde su oni trajno sačuvani (disk) trenutnoj memoriji bafera. Jedinica podataka koja se razmenjuje između bafera i diska je “disk blok”.

OBRADA TRANSAKCIJA

Svaka transakcija mora da bude trajna što znači da i ukoliko sistem padne odmah nakon izvršenja transakcije posledice izvršenja transakcije moraju da budu vidljive.

Upiti i ostale DML akcije su grupisane u transakcije, koje moraju da budu izvršene atomično i izolovano jedna od druge. Svaki upit ili bilo koja akcija modifikacije može biti transakcija. Svaka transakcija mora da bude trajna što znači da i ukoliko sistem padne odmah nakon izvršenja transakcije, posledice izvršenja transakcije moraju da budu vidljive. Izvršavanje transakcija možemo podeliti na dva dela:

1. *Menadžer paralelnog rada, ili raspoređivač, koji obezbeđuje atomičnost i integritet transakcija i*
2. *Sistem za logovanje i oporavak transakcija, koji je zadužen za trajno izvršenje istih.*

U bazama podataka podaci se uglavnom skladište u sekundarno skladište, koje je u današnje vreme hard disk. Da bi se izvršila neka izmena podataka, podaci moraju biti smešteni u glavnu memoriju. Posao menadžera skladišta je da upravlja podacima između glavne memorije i sekundarnog skladišta.

U jednostavnim sistemima za upravljanje, menadžer skladišta ne mora biti ništa drugo već file sistem operativnog sistema, ali iz razloga povećanja efikasnosti, DBMS upravlja podacima direktno na disku. Menadžer skladišta vodi evidenciju lokacija svakog fajla na disku kako bi lakše izvršio naredbe menadžera bafera.

Menadžer bafera je odgovoran za deljenje skladišta glavne memorije u bafere, koji su veličine bloka diska koji se prenose između diska i bafera. Tako da sve komponente DBMS-a koje koriste informacije sa diska će vršiti interakciju sa menadžerom bafera, bilo direktno ili preko execution engine-a.

TRANSAKCIJE I NJIHOVA OBRADA

Transakcija predstavlja jedinicu posla koja mora biti izvršena atomično i potpuno izolovana od drugih transakcija, a DBMS obezbeđuje da će rezultat izvršene transakcije biti sačuvani

U praksi je uobičajeno da se nekoliko operacija nad bazom podataka grupišu u transakciju, koja predstavlja jedinicu posla koja mora biti izvršena atomično i potpuno izolovana od drugih transakcija, a DBMS obezbeđuje da će rezultat izvršene transakcije biti sačuvan u potpunosti. Menadžer transakcija dobija transakcione komande od aplikativnih programa, koje određuju početak i kraj samih transakcija kao i određene izuzetke. Uloga procesora transakcija je da vrši sledeće operacije:

1. Logovanje: kako bi se obezbedila dugotrajnost same baze podataka svaka promena koja je nastala nad njom se beleži u poseban fajl na disku. Menadžer logovanja je dizajniran tako da bude otporan na bilo kakve prekide ili pad sistema, i da na osnovu log fajla bude omogućen povratak baze podataka u konzistentno stanje. Menadžer logovanja uglavnom vrši logovanja na mestima gde najčešće dolazi do pada sistema, a to je uglavnom u baferima, jer mora da bude siguran da je svaki podatak iz bafera uspešno upisan na disk.

2. Kontrola paralelnog rada: Transakcija se mora izolovano izvršiti, ali će se u mnogim sistemima više transakcija izvršavati istovremeno. Menadžer paralelnog rada mora da obezbedi da se svaka akcija iz više transakcija izvrši do kraja, samo jedna u istom trenutku nad jednim entitetom, i ovo se uglavnom postiže zaključavanjem nekih delova baze u toku njenog izvršenja, odnosno zabranjivanjem drugim transakcija da koriste entitet u okviru kog se izvršava trenutna akcija određene transakcije.

3. Blokada transakcija: kako se transakcije međusobno nadmeću za resurse kako bi bile izvršene, može se doći u situaciju da svakoj treba neki resurs koji druga transakcija trenutno drži zaključanim, odnosno ostali su zaključani neki delovi baze, pa na red dolaze druge transakcije kojima će možda trebati resursi koji nisu trenutno zaključani od strane drugih transakcija.

ULOGA PROCESORA UPITA (ENG. QUERY PROCESSOR)

Sastoji se od dve komponente: kompajlera upita i execution engine

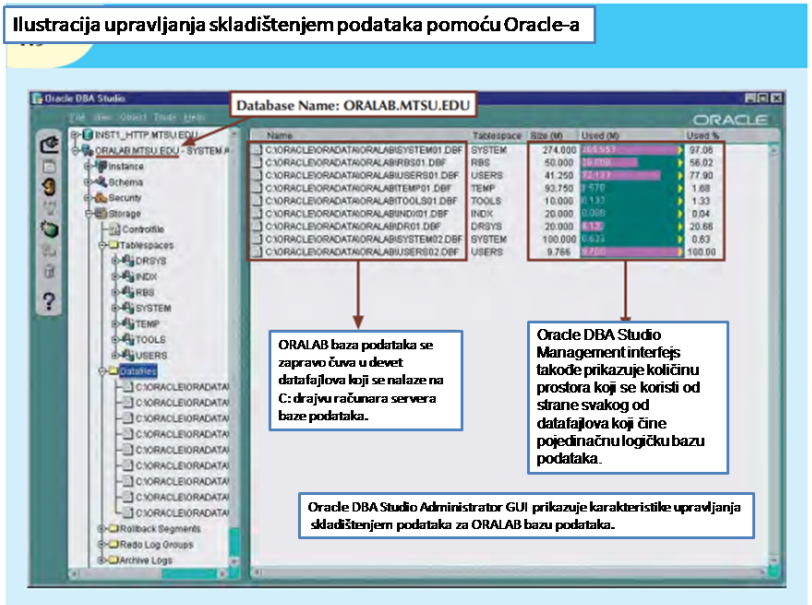
Deo DBMS-a koji najviše utiče na performanse sistema koje korisnici najčešće vide je procesor upita. Predstavljen je sa dve komponente:

1. Kompajler upita, koji prevodi upit u internu formu nazvanu plan upita, odnosno niz akcija koje treba izvršiti nad podacima. Kompajler upita koristi meta podatke i statističku analizu podataka kako bi odredio koji segment operacija je najbrži. Na primer, prisustvo indeksa, strukture koja olakšava pristup podacima, jedan plan može učiniti mnogo bržim za izvršenje od onog koji ne poseduje indeks. U planu upita operacije se najčešće zasnivaju na relacionoj algebri. Kompajler upita se sastoji iz sledećih delova:

- **Parser upita** – koji pravi strukturu stabla od tekstualne forme upita,

- **Procesor upita** – koji izvršava semantičku proveru upita (npr. da li sve relacije naznačene u upitu postoje) i vrši transformaciju strukture stabla u strukturu stabla algebarskih operacija koji predstavljaju polazni plan upita
- **Optimizer upita** – koji transformiše inicijalni plan upita u najbolji mogući niz operacija koji se izvršava nad podacima.

2. Execution engine: je odgovoran za izvršenje svakog koraka definisanim planom upita. On vrši interakciju sa većinom drugih komponenti DBMS-a, bilo direktno ili preko bafera. Execution engine mora da dovede podatke u bafer kako bi se manipulisalo njima, takođe, vrši i interakciju sa menadžerom raspoređivanja kako bi se izbegao pristup podacima koji su zaključani i sa log menadžerom kako bi se vršilo logovanje svih promena nastalih nad bazom.



Slika 4.2.2 Ilustracija upravljanja skladištenjem podataka pomoću Oracle-a. Izvor: [Autor,1]

Poglavlje 5

Objektno-orjentisane baze podataka (ODBMS)

CILJ OBJEKTNO ORIJENTISANIH BAZA PODATAKA

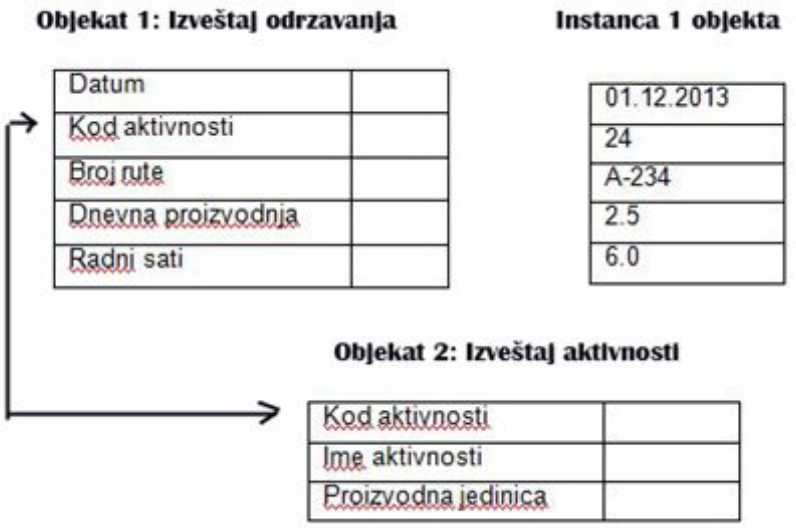
Da omoguće prirodniju manipulaciju podacima u aplikacijama koje se razvijaju nad bazom

Objektna orjentacija je pristup u kome se neki sistem organizuje kao kolekcija međusobno povezanih objekata koji ostvaruju postavljene ciljeve. Poslednja decenija u softverskom inženjerstvu jeste decenija objektne orjentacije. Ona danas preovladava u programiranju i programskim jezicima, metodološkim pristupima razvoju softvera i informacionih sistema, a sve se više koristi i u sistemima za upravljanje bazom podataka, bilo preko relacionih sistema za upravljanjem bazom podataka, bilo preko objektno-relacionih DBMS-a koji predstavljaju proširenje relacionih sa objektnim konceptima. Sistemi za upravljanje OO baza podataka (ODBMS ili OODBMS) imaju za cilj da omoguće prirodniju manipulaciju podacima u aplikacijama koje se razvijaju nad bazom.

Pod objektom se podrazumeva entitet koji je sposoban da čuva svoja stanja i koji stavlja okolini na raspolaganje skup operacija preko kojih se ta stanja prikazuju ili menjaju. Objekat je korisnički definisani, kompleksni tip podataka koji ima svoju strukturu i stanje (atributi i svojstva), kao i ponašanje (metode i operacije).

Objekti programskog jezika se bez ikakve izmene čuvaju u bazi. Mogu se opisati sa četiri karakteristike:

1. *Identifikator* – univerzalni identiifikator objekta na nivou sistema
2. *Ime* – objekat bi trebalo da ima jedinstveno ime u bazi, ali nije neophodno
3. *Životni vek* – objekat može biti privremen ili postojan
4. *Struktura* – objekat se kreira uz pomoć konstruktora tipa podataka



Slika 5.1 Šema OO baza podataka [Izvor: NM IT250-2020/2021]

KARAKTERISTIKE ODBMS

Ne postoji standard za ODBMS, kao ni univerzalno prihvaćeni dogovor o modelima podataka, sastoje se od objektnih modela, objektnih jezika, objektnih upitnih jezika, programskih jezika

U OO modelu baze podataka ne postoji koncept primarnog ključa, već ODBMS za svaki objekat pamti jedinstveni i neprimenljivi identifikator - Object Identifier (skraćeno OID) koji se koristi za referenciranje objekta u okviru baze podataka. Ovaj identifikator ne može se ručno definisati ili menjati jer ga sistem sam definiše.

Cilj je da se objekti baze podataka i objekti aplikacija napisanih u nekom OO programskom jeziku izjednače. Ne eliminiše se mogućnost da više programa koriste istu bazu podataka.

ODBMS na početku nije bio zasnovan na jedinstvenom standardu i ciljevi su se ostvarivali na više načina sa manje ili više uspeha. Radilo se na intenzivnom razvoju standard za objektne baze podataka u okviru ODMG (Object Database Management Group). Verzija 1.2 ODMG standarda se pojavila 1996, a značajno je izmenjena i dopunjena verzija 2.0, 1997 godine.

Komponente ODBMS su: objektni modeli, objektni jezici, objektni upitni jezici, programski jezici kao što su: C++, Smalltalk, Java Language Binding. Objektni specifikacioni jezici služe kao skup međusobno povezanih objekata.

ODL je objektno specifikaconi jezik koji ima istu ulogu koju JOP (DDL) ima u konvencionalnim jezicima. C++, Smalltalk, Java Language Binding, predstavljaju nezavisne komponente arhitekture koje pokazuju kako se u navedenim programskim jezicima (npr. C++-u) ugrađuju mogućnosti manipulacije perzistentnim objektima, mehanizmi za povezivanje sa OQL-om, upravljanje transakcijama i sl.

Za razliku od relacionih modela, u svetu objektno-orijentisanih baza podataka ne postoji univerzalno prihvaćeni dogovor o modelima podataka. Svaki OOMP mora minimalno posedovati sledeće osobine:

- *Mora biti obezbeđena funkcionalnost baze podataka*
- *Mora podržavati identifikaciju objekata*
- *Mora postojati mogućnost enkapsulacije*
- *Mora podržavati objekte sa kompleksnim stanjem.*

KORIŠĆENJE I PREDNOSTI ODBMS

Koriste se kada je potrebno podržati princip nasleđivanja i prednosti su korišćenje kompleksnih tipova podataka, primena principa nasleđivanja i enkapsulacije itd.

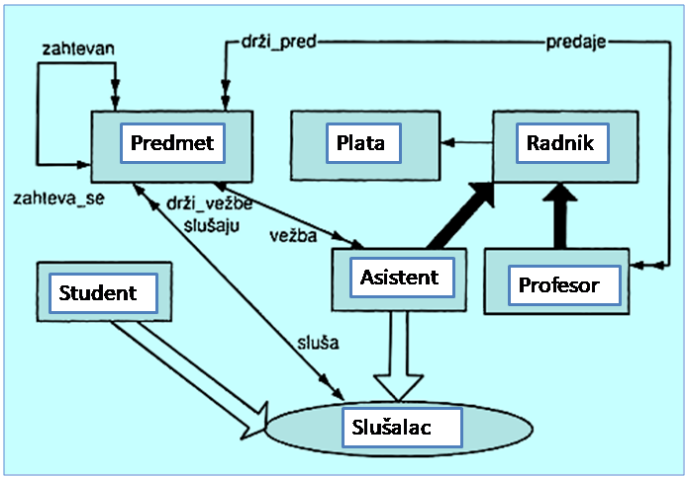
OO model baze podataka se koristi:

- *Kada je nasleđivanje kao osobina OO modela bitna*
- *Kada je potrebna besprekorna integracija operativnih sistema baze podataka, tabela, i sl.*
- *Ukoliko je potrebna bilo kakvo deljenje informacija, proizvoda, softvera.*

Objektne baze podataka se sve više koriste i van industrije. Sve više je kvalifikovanih stručnjaka, ali još uvek nedovoljno. Alati za ODBMS su relativno laki za korišćenje. Neki smatraju da su objektno orjentisane baze lakše za učenje od relacionih baza podataka i jeftinije. Doduše postoji i dobra većina koja misli da je ipak lakše upravljati relacionim bazama podataka. Objektno-orjentisani pristup je bio prirodna alternativa za rastuće kompanije koje nemaju novca za kupovinu relacionih baza i može se kreirati vrlo intenzivna obuka svojim zaposlenima. Zbog niske cene i jednostavnosti, objektne baze podataka su takođe savršen izbor za naučne zajednice i univerzitetska istraživanja.

Prednosti objektnih baza su:

1. *Visoke performanse*
2. *Navigacioni interfejs za operacije (kao što je obilazak grafa)*
3. *Objektni upitni jezik preuzima objekte deklarativno*
4. *Kompleksni tipovi podataka*
5. *Klase i hijerarhije (nasleđivanje i enkaspulacija)*
6. *Semantičko modeliranje*



Slika 5.2 Objektni model. Izvor: [Autor]

NEDOSTACI ODBMS

Mnogi postojeći problemi sa objektnim bazama su rešeni i nedostaci proizilaze iz nepostojanja iskustva u radu sa njima

Problemi sa objektnim sistemima za upravljanje bazom su:

1. *Ne postoji pravi standard za objektno-orjentisane baze podatka*

2. DBMS se koristi samo u industriji

3. Nedostatak alata za OODB u prošlosti

4. ODBMS zaostaje u performansama i skalabilnosti za sistemom relacionih baza podataka

5. Nedostaje im matematička formalizacija.

6. Nepostojanje univerzalnog modela podataka – kao što je već rečeno, ne postoji univerzalni dogovor oko izgleda modela podataka za OO DBMS-ove. Nedostaje i teorijska podloga za modele. Ipak, udruženje ODMG (Object Database Management Group) predložilo je objektni model koji predstavlja de facto standard.

7. Nedostatak iskustva – ne postoji iskustvo u radu sa njima

8. Nedostatak standarda – osim što ne postoji univerzalni dogovor oko izgleda modela podataka za OO DBMS-ove, ne postoje ni standardi za OO upitne jezike. Ipak udruženje ODMG (Object Database Management Group) predložilo je objektni upitni jezik (OQL) koji predstavlja standard de facto.

9. Složenost – uvođenjem dodatnih funkcionalnosti, povećala se kompleksnost OO DBMS-ova. Ovakva složenost dovodi do toga da su aplikacije skuplje i teže za korišćenje

10. Nedostatak podrške za poglede i bezbednost – trenutno, većina OO DBMS-ova nema ugrađene mehanizme za poglede i bezbednost pristupa podacima

Poglavlje 6

Objektno-relacione baze podataka (ORDBMS)

KARAKTERISTIKE

Tradicionalni relacioni model se proširuje osnovnim objektnim konceptima kao što su: apstraktni tipovi podataka, enkapsulacija, polimorfizam, nasleđivanje i sl.

Objektno-relacioni model i objektno-relacione baze podataka reprezentuju takozvani evolucijski pristup koji se ogleda u integraciji objektne paradigme u relacioni model podataka i njegovim proširenjem objektno-orjentisanim karakteristikama. Objektno-relacione baze se mogu nazvati i hibridne baze.
Osnovna postavka objektno-relacionih baza podataka je upravljanje objektima i pravilima uz očuvanje kompatibilnosti sa relacionim modelom i relacionim bazama podataka. SQL 2003 standardom ostvareno je potpuno slaganje nad objektno-relacionim modelom i svi glavni proizvođači sistema za upravljanje bazama nastoje u što je moguće većem obimu da implementiraju taj standard. Objektno-relacioni sistem za upravljanje bazom podataka (ORDBMS) je sličan sistemu za upravljanje relacionim bazama podataka, ali se bazira na objektno-orjentisanom modelu baze podataka: objekti, klase i nasleđivanje su direktno podržane u šemama baza podataka i upitnim jezicima.
Tradicionalni relacioni model se proširuje osnovnim objektnim konceptima kao što su: apstraktni tipovi podataka, enkapsulacija, polimorfizam, nasleđivanje i sl. Nastaju hibridni DBMS-ovi koji zadržavaju prednosti relacionog modela i istovremeno pružaju programerima objektno-orijentisan pogled na osnovne podatke. Ovi tipovi baza podataka zadržavaju performanse relacionog modela i semantički bogatu programsku podršku objektno-orijentisanog modela.

OSNOVNI KONCEPTI

Korisnički definisani (apstraktni) tipovi podataka i prošireni koncept relacije koji dopušta da atributi n-torki relacija kao domene ne moraju imati samo proste podataka

Osnovni koncepti su:

1. Korisnički definisani tipovi podataka - omogućavaju kreiranje aplikacijskih specifičnih, kompleksnih tipova podataka. Prekoračuje se jedno od osnovnih ograničenja relacionog modela koji podrazumeva i zahteva da domeni atributa

relacije budu prosti. Korisnički definisani tip utemeljen je na konceptu apstraktnih tipova podataka.
2. Relacije – su i dalje jedan od osnovnih koncepata objektno-relacionog modela. Koncept relacije je proširen dopuštanjem da atributi kao domene ne moraju imati samo proste, nego i korisničke tipove podataka. Ukoliko su elementi relacije primeri korisničkog tipa podataka ona se naziva objektna relacija.
3. Korisnički tipovi podataka mogu biti, kao i standardni jednostavni tipovi, domeni atributa standardnih relacija. Ne postoji formalna definicija objektnog modela (ODMG specifikacija).Objektno-relacioni model bazira se na konceptima objektne paradigme.

Objektni aspekti su:

1. Apstraktni tipovi podataka - može se definisati klasa podataka koja se sastoji od atributa i operacija uz skrivanje detalja njihove implementacije
2. Nasleđivanje - mogućnost da više tipova podataka dele atribute i operacije
3. Identitet objekata - svaki primerak abstraktnog tipa podataka poseduje jedinstveni ID, nevidljiv korisnicima
4. Polimorfizam - egzistencija više operacija sa istim imenom, a sa različitom semantikom tj. mogućnost rada nad različitim tipovima podataka.

Objektno relacione baze podataka kombinuju jednostavnost i efikasnost relacionih baza podataka sa mogućnostima objektnih baza podataka da pamte složene objekte. Standardni SQL je izmenjen tako da relacionom modelu dozvoljava da inkorporira mnoge karakteristike objektno orijentisanih sistema kao što su apstraktni tipovi podataka, nasleđivanje i operacije.

Open source proizvod PostgreSQL je verovatno najpoznatiji hibridni DBMS. Oracle takođe uključuje neke hibridne karakteristike.

VIDEO

The differences between relational, object, no-SQL, and data warehouse databases

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 7

Pokazna vežba

NAČIN ORGANIZACIJE POKAZNIH VEŽBI

Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata

Vežba je organizovana kroz uvodni deo i deo za samostalni rad studenata.

1. U uvodnom delu pokaznih vežbi se daju se pokazni primeri koji studentima treba da pomognu u samostalnom rešavanju zadataka.
2. Zadatke koji su zadati za samostalni rad, student samostalno rešava uz pomoć asistenta.

7.1 Pokazni primer

VRSTE SISTEMA ZA UPRAVLJANJE BAZAMA PODATAKA - (5 MIN.)

Hijerarhijski, mrežni, relacioni, objektno orijentisan itd.

Baza podataka je kolekcija podataka ili zapisa. Sistemi za upravljanje bazama podataka dizajnirani su za upravljanje bazama podataka. Sistem za upravljanje bazama podataka (DBMS) je softverski sistem koji koristi standardnu metodu za skladištenje i organizovanje podataka. Podaci se mogu dodavati, ažurirati, brisati ili prenositi korišćenjem različitih standardnih algoritama i upita.

Postoji nekoliko vrsta sistema za upravljanje bazama podataka. Sedam uobičajenih sistema za upravljanje bazama podataka su:

1. Hijerarhijske baze podataka
2. Mrežne baze podataka
3. Relacione baze podataka
4. Objektno orijentisane baze podataka
5. Graf baze podataka
6. Dokument baze podataka
7. NoSQL baze podataka

U ovoj vežbi će biti više reči o mrežnim i hijerarhijskim bazama podataka kao pretečama danas najčešće korišćenih relacionih i objektno orijentisanih baza podataka. Fokus je stavljen

na poslednje dve vrste baza podataka kao i razlike koje između njih postoje. U okviru predmeta će se takođe govoriti i o graf i NoSql bazama podataka.

HIJERARHIJSKE BAZE PODATAKA - (7 MIN.)

Iako je hijerarhijska struktura jednostavna, ona je nefleksibilna zbog relacija roditelj-dete tipa jedan prema više.

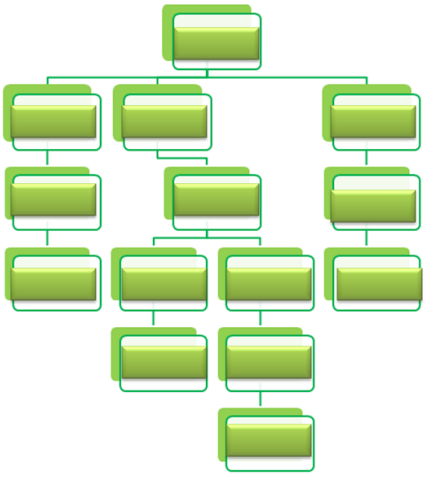
U hijerarhijskom sistemu upravljanja bazama podataka (hijerarhijski DBMS-ovi) podaci se čuvaju u čvorovima koji su povezani relacijama roditelj-deca. U hijerarhijskoj bazi podataka, pored stvarnih podataka, u slogovima se nalaze i informacije o njihovim relacijama roditelj / dete.

U hijerarhijskom modelu baze podataka podaci su organizovani u strukturu stabla. Podaci se čuvaju u obliku kolekcije polja u kojima svako polje sadrži samo jednu vrednost. Slogovi se međusobno povezuju preko linkova koji odslikavaju relacije roditelj-deca. U hijerarhijskom modelu baze podataka, svaki slog deteta ima samo jednog roditelja. Roditelj može imati više dece.
Da bismo pretražili polja sa podacima, moramo da prolazimo kroz svako stablo dok se ne nađe slog.

IBM je početkom 1960-ih razvio strukturu hijerarhijske baza podataka. Iako je hijerarhijska struktura jednostavna (slika 8.1.), ona je nefleksibilna zbog relacija roditelj-dete tipa jedan prema više. Hijerarhijske baze podataka se široko koriste za izgradnju aplikacija visokih performansi i raspoloživosti, obično u bankarskoj i telekomunikacionoj industriji. IBM-om Information Management System (IMS) i Windows Registry su dva popularna primera hijerarhijskih baza podataka.

Prednost: Hijerarhijskoj bazi podataka se može brzo pristupiti i ažurirati. Kao što je prikazano na slici 8.1, njena struktura modela je poput stabla i relacije između zapisa su unapred definisane. Ova karakteristika je mač sa dve oštrice.

Nedostatak: Ova vrsta strukture baze podataka omogućava da svako dete na stablu može imati samo jednog roditelja. Veze između dece nisu dozvoljene, čak i ako imaju smisla sa logičkog stanovišta. Dodavanje novog polja ili zapisa zahteva da se celokupna baza podataka redefiniše.



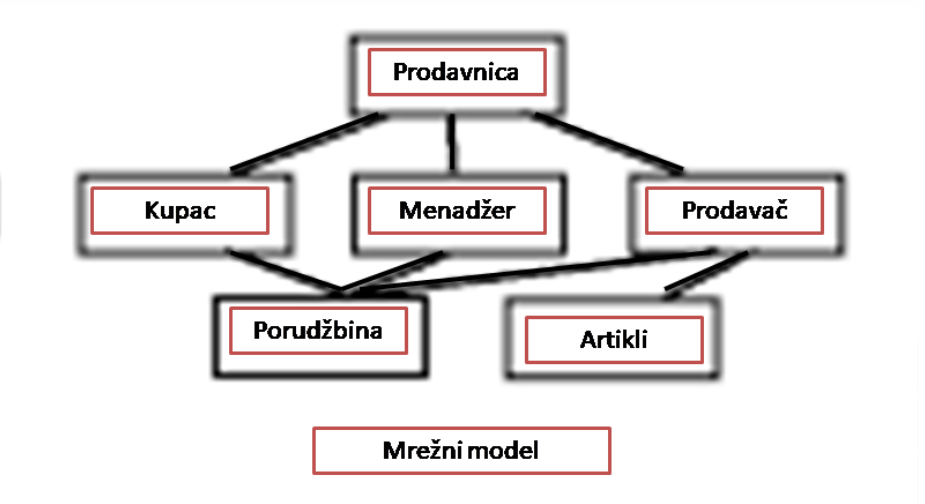
Slika 7.1.1 Hijerarhijski model. Izvor: [Autor]

MREŽNE BAZE PODATAKA - (7 MIN.)

Za razliku od hijerarhijskih baza podataka u kojima jedan čvor može imati samo jednog roditelja, mrežni čvor može imati relaciju s više entiteta

Mrežni sistemi za upravljanje bazama podataka (Mrežni DBMS) koriste mrežnu strukturu za kreiranje relacija između entiteta. Mrežne baze podataka se uglavnom koriste na velikim digitalnim računarima. Mrežne baze podataka su hijerarhijske baze podataka, ali za razliku od hijerarhijskih baza podataka u kojima jedan čvor može imati samo jednog roditelja, mrežni čvor može imati relaciju s više entiteta. Mrežna baza podataka više liči na paukovu ili međusobno povezanu mrežu slogova.

U mrežnim bazama podataka deca se nazivaju članovima. Razlika između svakog deteta ili člana je u tome što može imati više od jednog roditelja.
Mrežni modela podataka je sličan hijerarhijskom modelu podataka. Podaci u mrežnoj bazi podataka su organizovani relacijama više na prema više. Strukturu mrežne baze podataka je prvi kreirao Charles Bachman. Neke od popularnih mrežnih baza podataka su Integrated Data Store (IDS), IDMS (Integrated Database Management System), Raima Database Manager, TurboIMAGE, and Univac DMS-1100.



Slika 7.1.2 Mrežni model. Izvor: [Autor]

RELACIONE BAZE PODATAKA - (7 MIN.)

U sistemima za upravljanje relacionim bazama podataka (RDBMS) veze između podataka su relacione i podaci se čuvaju u tabelarnom obliku (u kolonama i redovima).

U sistemima za upravljanje relacionim bazama podataka (RDBMS) veze između podataka su relacione i podaci se čuvaju u tabelarnom obliku (u kolonama i redovima). Svaka kolona tabele predstavlja atribut, a svaki red u tabeli predstavlja slog (torku). Svako polje u tabeli predstavlja vrednost podataka.
Strukturirani jezik upita (SQL) je jezik koji se koristi za postavljanje upiti nad RDBMS, uključujući unošenje, ažuriranje, brisanje i pretraživanje zapisa. Relacione baze podataka rade sa tabelama koje imaju polje ključa koje jedinstveno označava svaki red (slog). Ova polja ključa se mogu koristiti za povezivanje jedne tabele podataka sa drugim tabelama.
Relacione baze podataka su najpopularnije i najčešće korišćene baze podataka. Neki od popularnih DDBMS su Oracle, SQL Server, MySQL, SQLite i IBM DB2.
Relaciona baza podataka ima dve glavne prednosti:

1. *Relacione baze podataka mogu se koristiti sa malo ili nimalo obuke.*
2. *Unosi u bazu podataka se mogu modifikovati bez navođenja celog sloga tabele*

Svojstva relacionih tabela: U relacionoj bazi podataka moramo da sledimo sledeće osobine:

- *Vrednosti su atomske*
- *Svaki red je jedinstven.*
- *Vrednosti svih kolona su istog tipa.*
- *Kolone se ne razlikuju.*
- *Redosled redova je beznačajan.*
- *Svaka kolona ima zajedničko ime.*

OBJEKTNO ORIJENTISANI MODEL - (7 MIN.)

Objektno orijentisane baze podataka dodaju funkcionalnost baze podataka objektno orijetisanim programskim jezicima i kreiraju standardno okruženje za razvoj aplikacija.

Ovaj model je nastao kao način za prevazilaženje problema vezanih za funkcionalnost objektno orijentisanog programiranja kome je potrebno obezbediti mnogo više od skladištenja objekata programskih jezika. Objekt DBMS povećava semantiku C ++ i Java.

Objektno orijentisane baze podataka dodaju funkcionalnost baze podataka objektno orijetisanim programskim jezicima i kreiraju standardno okruženje za razvoj aplikacija. Za pisanje aplikacije je potrebno manje koda, koriste se prirodnije modeliranje podataka, a kod se lakše održava.

Moć objektno orijentisanih baza podataka potiče od cikličnog tretiranja kako konzistentnih podataka, koji se nalaze u bazama podataka, tako i privremenih podataka, koje nema potrebe trajno pamtiti, kakvi se nalaze u izvršnim programima.

Objektno orijentirane baze podataka koriste objekte koji sadrži dva elementa:

1. *Deo podataka (npr. Zvuk, video, tekst ili grafika).*
2. *Uputstva ili softverske programe koji se nazivaju metode, šta treba učiniti sa podacima.*

ODBMS (sistem za upravljanje objektnim bazama podataka) kombinuju

mogućnosti baze podataka sa objektno orijentisanim programskim jezikom. OODBMS omogućavaju objektno orijentisanim programerima da razvijaju proizvod, skladište ih kao objekte i kopiraju ili modifikuju postojeće objekte kako bi napravili nove objekte unutar OODBMS.
Objektne baze podataka su se pojavile početkom 1980-ih.
Neke objektno orijentisane baze podataka dizajnirane su tako da dobro funkcionišu sa objektno orijentisanim programskim jezicima kao što su Delphi, Rubi, Python, JavaScript, Perl, Java, C #, Visual Basic .NET, C ++, Objective-C i Smalltalk; drugi poput JADE imaju svoje programske jezike. OODBMS koriste potpuno isti model kao i objektno orijentisani programski jezici.

Nedostaci objektno orijentisanih baza podataka:

1. *Objektno orijentirane baze podataka skuplje su za razvoj.*
2. *Većina organizacija ne želi da se upusti u njihov razvoj.*

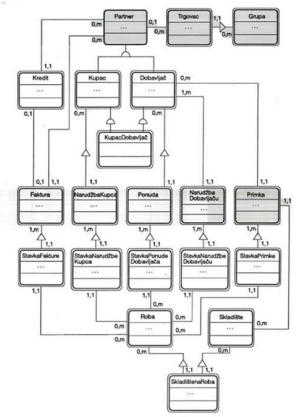
Prednosti objektno orijentisanih baza podataka su upečatljive. Mogućnost mešanja i podudaranja objekata koji se mogu ponovo (rejuzibilno) koristiti pruža neverovatne mogućnosti za multimedije.

PRIMER MODELA OBJEKTNO ORIJETISANE BAZE PODATAKA (7 MIN.)

Postoje dve vrste objektno orijentisane baze podataka: Multimedijalna baza podataka i Baza podataka hiperteksta

Objektno orijentisani model podataka može da čuva audio, video i grafičke fileove. Oni se sastoje od podataka i metoda. Postoje dve vrste objektno orijentisane baze podataka:

1. Multimedijalna baza podataka čuva medije, poput slika koje relaciona baza podataka ne može da smešta.
2. Baza podataka hiperteksta omogućava povezivanje bilo kojeg objekta sa bilo kojim drugim objektom. Pomaže u organizovanju različitih podataka.



Slika 7.1.3 Objektni model nekog preduzeća. Izvor: [Autor]

RAZLIKA IZMEĐU RELACIONIH I OBJEKTNO ORIJETISANIH BAZA PODATAKA -(5 MIN.)

Upotreba ODBMS-a je usmerena na multimedijalnu prezentaciju ili CAD sisteme za razliku od RDBMS-a koje se koriste u poslovnim aplikacijama

Objektne baze podataka se razlikuju od relacionih baza podataka koje su orijentisane na tabele.

Budući da je objaktno orijentisana baza podataka integrisana s programskim jezikom, programer može održavati doslednost unutar jednog okruženja, tako da će i OODBMS i programski jezik koristiti isti model reprezentacije.

Relacioni DBMS projekti, nasuprot tome, održavaju jasniju podelu između modela baze podataka i aplikacije

Kako se upotreba veb-bazirane tehnologije povećava sa primenom Intraneta i ekstraneta, kompanije imaju veliko interesovanje za OODBMS-ove za prikazivanje njihovih složenih podataka. Upotreba DBMS-a koji je posebno osmišljen za čuvanje podataka kao objekata

daje prednost onim kompanijama koje su usmerene na multimedijalnu prezentaciju ili organizacijama koje koriste računarski dizajn (CAD). Za razliku od njih relacione baze podataka se koriste kada postoji potreba za čuvanjem struktuiranih podataka najčešće u poslovnim informacionim sistemima.

7.2 Pokazna vežba - samostalni rad

ZADACI ZA SAMOSTALNI RAD - (45 MIN.)

Uraditi sledeće zadatke

ZADATAK 1: Imajući u vidu sledeća svojstva relacionih tabela:

1. Vrednosti su atomske
2. Svaki red je jedinstven
3. Vrednosti svih kolona su istog tipa
4. Kolone se ne razlikuju
5. Redosled redova je beznačajan
6. Svaka kolona ima zajedničko ime

koristeći exel ili word ili bilo koji drugi tekstualni editor, kreirajte tabelu koja sadrži podatke o studentima Fakulteta Informacionih tehnologija na Univerzitetu Metropolitan. **(Vreme izrade 10 minuta)**

ZADATAK 2**:** Imajući takođe u vidu svojstva relacionih tabela na isti način kreirajte tabelu koja sadrži podatke o studijskim grupama koje studenti Fakulteta informacionih tehnologija mogu da upišu. **(Vreme izrade 10 minuta)**

ZADATAK 3: Koristeći primer iz predavanja, pokušajte da povežete ove dve tabele tako da se zna koji student studira na kojoj studijskoj grupi. (**Vreme izrade 10 minuta)**

ZADATAK 4: Pokušajte da za navedeni primer skicirate model objektno orijentisane baze podataka. **(Vreme izrade 15 minuta)**

Poglavlje 8

Domaći zadatak

DOMAĆI ZADATAK 1 - (90 MIN.)

Tekst i uputstvo za slanje domaćeg zadatka 1

ZADATAK 1: Izaberite dva proizvoljna entiteta između kojih postoji relacija jedan prema više npr. (RAČUN i KORISNIK- jedan KORISNIK može imati više RAČUNA) ili RODITELJ i DETE (jedan RODITELJ može imati više DECE) ili RADNIK i ORGANIZACIONA JEDINICA (jedana ORAGANIZACIONA JEDINICA može imati više radnika). Za izabrane entitete korišćenjem exel-a ili nekog tekstualnog editora nacrtajte odgovarajuće tabele.

1. Kolone tabela treba da odgovaraju atributima entiteta (npr. entitet RAČUN može imati atribute: BrojRacuna, TipRacuna i StanjeRacuna a entitet KORISNIK atribute: ID, Ime, Prezime, adresa)
2. Redovi tabela treba da sadrže vrednosti definisanih atributa za svako pojedinačno pojavljivanje entiteta (popunite najmanje 3 reda tabela)
3. Rukom skicirajte objektno orijentisan model podataka za ovaj slučaj ali tako da jedan KORISNIK može imati više RAČUNA a jedan RAČUN više KORISNIKA.

Napomena: entitete navedene kao primer ne možete koristiti u zadatku.

Rešenje zadatka snimiti kao IT250-DZ01-BrojIndeksa-Ime_Prezime.cdm (gde su ime, prezime i broj indeksa vaši podaci a .cdm ekstenzija file-a se dobija kao izlaz iz konceptualnog modela PowerDesigner-a).

Prilikom slanja domaćih zadatka, neophodno je da ispunite sledeće:
Subject mail-a mora biti IT250-DZbr (u slučaju kada šaljete domaći za ovu nedelju to je IT250-DZ01)
U prilogu mail-a treba da se nalazi arhiviran projekat koji se ocenjuje imenovan na sledeći način: IT250-DZbr-BrojIndeksa-ImePrezime. Na primer, IT250-DZ01-1234-VeljkoGrkovic
Telo mail-a treba da ima pozdravnu poruku
Arhivu sa zadatkom poslati na adresu predmetnog asistenta:
milica.vlajkovic@metropolitan.ac.rs (studenti u Beogradu i online studenti) ili
tamara.vukadinovic@metropolitan.ac.rs (studenti u Nišu).
Svi poslati mail-ovi koji ne ispunjavaju navedene uslove NEĆE biti pregledani. Za sva pitanja ili nedoumice u vezi zadatka, možete se obratiti asistentu.

Zaključak

ŠTA SMO NAUČILI U OVOJ LEKCIJI?

Šta smo naučili u ovoj lekciji?

U ovojlekciji su date uvodne napomene o bazama podataka. Najpre je objašnjena potreba za čuvanjem podataka a zatim način smeštanja i čuvanja podataka u fajlovima. Baze podataka su nastale s ciljem da se prevaziđu problemi čuvanja podataka u fajlovima, pre svega problem redundantnosti (ponovljivost)i podataka.

Sistemi za upravljanje bazama podataka (DBMS) su razvijeni kako bi rešili inherentne slabosti sistema za čuvanje podataka u fajlovima. Umesto da podaci budu smešteni u nezavisnim fajlovima, DBMS predstavlja bazu podataka korisniku kao jedinstveno skladište podataka. Ova organizacija podataka omogućava deljenje podataka, čime se eliminiše potencijalni problem fragmentacije informacija. Pored toga, DBMS sprovodi integritet podataka, eliminiše redundanciju i promoviše bezbednost podataka.
U lekciji se objašnjavaju tri tipa baza podataka: relacione, objektne i relaciono objektne baze podataka kao i njihove međusobne razlike. Međutim, kako su danas u upotrebi uglavnom relacione baze podataka, opisane su karakteristike njihovih osnovnih komponenti.

LITERATURA

Za pisanje ove lekcije je korišćena sledeća literatura:

1. http://corpgov.crew.ee/Materjalid/Database%20Systems%20-%20Design,%20Implementation,%20and%20Management%20(9th%20
2. David M. Kroenke, Database Processing – fundamentals,design and implementation, Prentice Hall, 2004.
3. C. J. Date, An introduction to Database Systems, Addison-Wesley Publishing Company, 1990