

Predavanje VI.

Sustavi fizičke pohrane

BAZE PODATAKA II

doc. dr. sc. Goran Oreški

Fakultet informatike,

Sveučilište Jurja Dobrile, Pula

Sadržaj

- ponavljanje prethodnih predavanja
 - trigeri
 - materijalizirani pogledi
 - autentifikacija
 - autorizacija
- mediji za pohranu
- sučelja za pohranu podataka
- RAID
- poboljšanje pouzdanosti i performansi
- organizacija podataka
- organizacija datoteka
- pohrana meta-podataka

Ponavljjanje

- trigeri su proceduralni izrazi (pohranjene procedure) koji se izvode automatski kada se baza podataka mijenja
- ključni elementi trigera
 - događaj (*engl. event*)
 - uvjet (*engl. condition*)
 - akcije (*engl. actions*)
- vrste trigera
 - DML trigeri
 - DDL operacije (create/alter/drop ...)
 - login/logout korisnika
 - database stratup, shutdown, errors...

Ponavljjanje

- materijalizirani pogledi stvaraju nove tablice koje se popunjavaju s rezultatima definicije pogleda
- ukoliko baza podataka ne podržava materijalizirane poglede
 - održavanje se može izvoditi pomoću trigeru na pogledom referenciranim tablicama
- baze podataka koje podržavaju materijalizirane poglede će same provoditi održavanje
 - puno jednostavnije od prethodne opcije
 - u pravilu pružaju više opcija
 - trenutno održavanje (*engl. immediate*) – obnavljanje sadržaja nakon svake promjene
 - odgođeno održavanje (*engl. deferred*) – obnavljanje sadržaja periodički

Ponavljjanje

- autentifikacija (*engl. authentication*)
 - "Ja sam onaj za kojeg se izdajem!"
- autorizacija (*engl. authorization*)
 - "Dozvoljeno mi je činiti ono što hoću!"
- SQL baza podataka provodi autentifikaciju korisnika
 - prilikom spajanja navode se korisničko ime i lozinka
 - baze podržavaju sigurno, enkriptirano povezivanje (npr. SSL) i slično
- SQL baza podataka pruža autorizacijske mehanizme za različite operacije
 - različite operacije zahtijevaju različite privilegije
 - korisnicima se mogu dodijeliti privilegije za specifične operacije
 - korisnicima se mogu opozvati privilegije za specifične operacije, da bi ograničilo njihovo pravo

Fizički razina

- do sada smo se u proučavanju baza posvetili modelima više razine:
 - konceptualna i logička razina (ER model i relacijski model)
 - „bitnije” razine za korisnika
- u slijedećih nekoliko predavanja ćemo istražiti implementaciju modela podataka i jezika na fizičkoj razini
- u ovom predavanju:
 - karakteristike medija za pohranu podataka
 - kako napraviti pouzdan sustav pohrane
 - organizacija podataka

Mediji za pohranu

- u većini računalnih sustava postoji nekoliko tipova pohrane
- mediji se klasificiraju prema:
 - brzini kojom se može pristupiti podacima
 - cijenom medija po jedinici podatka
 - i pouzdanosti
- tipično su dostupni slijedeći mediji:
 - cache, glavna memorija, flash memorija, magnetni diskovi, optički mediji, trake za pohranu
 - koje medije poznajete/koristite?

Cache

- najbrži i najskuplji tip pohrane
- relativno mala a njezinim korištenjem upravlja hardware sustava
 - fizički se nalazi u neposrednoj blizini, tj. na samom procesoru
 - djeluje kao posrednik između procesora i RAM-a
- nećemo se baviti upravljanjem cache memorijom prilikom analize sustava baze podataka
- međutim, timovi koji razvijaju baze podataka, paze na cache prilikom dizajniranja obrade upita
 - nešto više o tome u nastavku

Glavna memorija

- medij za pohranu podataka koji su dostupni za obradu
- strojne instrukcije osnovne namjene se izvode na glavnoj memoriji
 - dvije glavne komponente RAM i ROM
- veličina:
 - na osobnim računalima – desetke gigabajta
 - na velikim serverima – stotine ili čak tisuće gigabajta
 - u pravilu premala ili preskupa za pohranu cijele baze podataka
- sadržaj glavne memorije (RAMa) se gubi u slučaju da nestane struje ili se sruši sustav
 - stoga se glavna memorija naziva **nestalna** (*engl. volatile*)

Flash memorija

- ukoliko nestane struje ili se dogodi pad sustava podaci ostaju sačuvani
 - spada u **stalnu pohranu** (engl. non-volatile)
- ima nižu cijenu po bajtu pohrane od glavne memorije ali višu od magnetnog diska
- široko se koristi u uređajima kao što su kamere, mobiteli i za pohranu podataka „USB flash diskovi”
- sve češće se SSD koristi kao zamjena za magnetne diskove
 - cijena: 1TB SSD ~ 1500kn
 - imaju slično sučelje koje omogućava pohranu i dohvat podataka u blokovima; (engl. *block-oriented interface*)
 - veličina bloka: 512B - 8 KB

Magnetni disk

- osnovni medij za dugotrajnu pohranu podataka je magnetni disk
 - HDD - hard disk drive
 - spada u **medij stalne pohrane**
 - disk se može pokvariti; to je rijetko u odnosu na pad sustava ili nestanak struje
- da bi obradili podaci s diska se moraju prebaciti u glavnu memoriju, a nakon obrade se pohranjuju na disk
- veličina diska: 500GB – 14TB
- cijena: 1TB ~ 500kn
- cjenovno jeftiniji od SSD-a ali lošiji po performansama

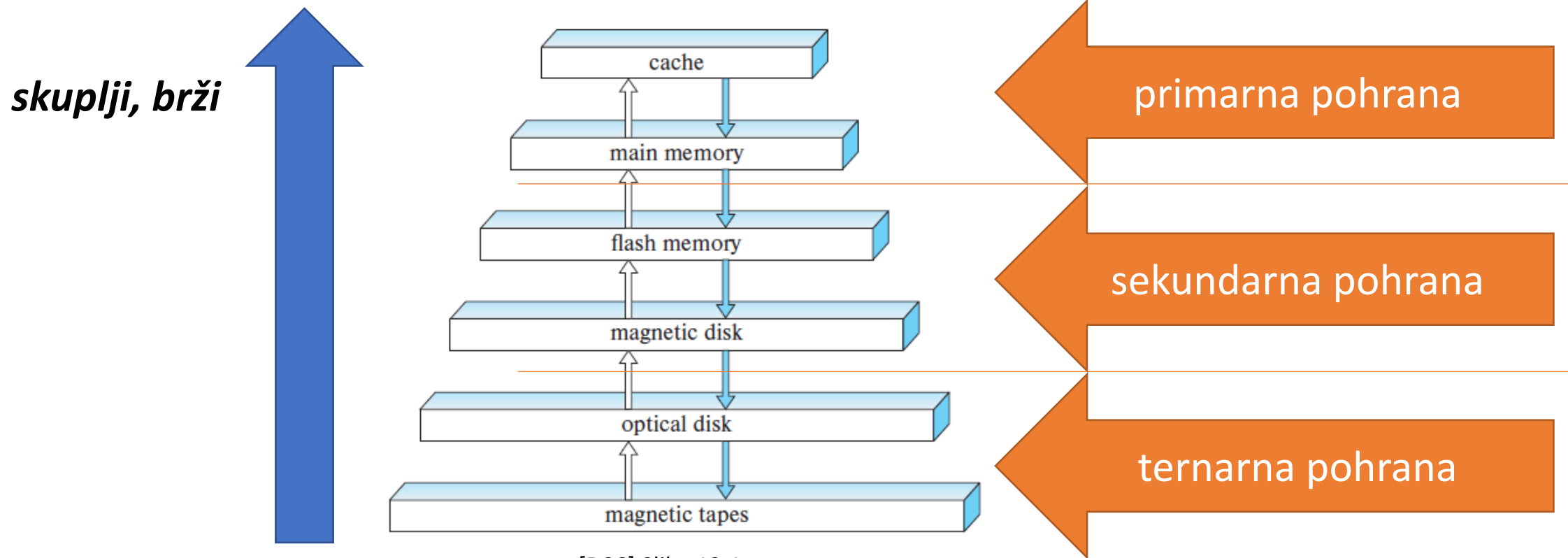
Optički mediji

- ***digital video disk (DVD)*** je optički medije za pohranu podataka
 - podaci se čitaju i upisi pomoću lasera kao izvora svjetlosti
- Blu-ray DVD imaju kapacitet od 27GB – 128GB, zavisno o broju slojeva
- iako je inicijalna namjena bila pohrana video podataka, mogu se koristiti za bilo kakve digitalne podatke
- nisu prikladni za aktivnu bazu podataka jer vrijeme pristupa podacima može biti značajno duže od magnetnog diska
- vrste: read-only, write-once, rewritable

Trake/vrpce za pohranu

- primarno se koriste za arhiviranje podataka
 - podaci koji se sigurno moraju pohraniti na duži period vremena
 - često zakonska obveza
- magnetne trake su jeftinije od HDD i mogu sigurno pohraniti podatke
- sporije od HDD-a jer je pristup sekvencijalan od početka trake
- magnetne trake -> pohrana sekvencijalnog pristupa
 - (engl. sequential-access)
- HDD i SSD -> pohrana direktnog pristupa
 - (engl. direct-access)

Hijerarhija medija za pohranu



[DSC] Slika 12.1

Sučelja za pohranu podataka

- diskovi u pravilu podržavaju SATA (*Serial ATA*) ili SAS (*Serial Attached SCSI*) sučelje
 - SAS se koristi na serverima
 - SATA-3 verzija SATA-e 6 Gb/s
 - SAS-3 12 Gb/s
- Non-Volatile Memory Express (NVMe) sučelje je logičko sučelje razvijeno za bolju podršku SSD-a
 - koristi PCIe sučelje
- diskovi koji nisu direktno povezani:
 - storage area network (SAN), network attached storage (NAS), cloud storage

RAID

- količina podataka za pohranu u mnogim aplikacijama brzo rastu
 - javlja se potreba za velikim brojem diskova
- to je prilika za poboljšanje vremena potrebnog za upis i čitanje podataka, ukoliko se diskovima upravlja paralelno
- povrh toga, takva postava omogućava povećanje pouzdanosti pohrane podataka
 - redundantne informacije mogu biti spremljene na više diskova
 - gubitak jednog diska ne dovodi do gubitka podataka
- različite tehnike organizacije diskova, pod kolektivnim nazivom **RAID**
 - *engl. redundant arrays of independent disks*

Poboljšanje pouzdanosti kroz redundantnost

- vjerojatnost kvara: jedan od N diskova $>$ jedan disk
- pretpostavimo da je prosjek kvara diska 100.000 sati ili 11 godina
- ako je $N = 100$; srednja vrijednost pada jednog diska $100.000/100$ tj. 1000 sati ili 42 dana
 - jako loše ako su podaci pohranjeni na jednom mjestu
- rješenje – redundantnost
 - pohranimo viška podatke koji nisu potrebni u normalnom radu, ali se koriste u slučaju pada diska za povrat podataka
 - ako disk padne, podaci nisu izgubljeni, efektivna srednja vrijednost vremena kvara se povećava (ako brojimo samo kvar koji dovodi do gubitka podataka)

Poboljšanje pouzdanosti kroz redundantnost

- najjednostavnije rješenje je zrcaljenje diskova (engl. mirroring)
 - svi podaci se zapisuju na dva fizička diska
 - gubitak podataka nastaje samo ukoliko se drugi disk pokvari prije nego što je prvi popravljen
- srednja vrijednost gubitka podataka zrcaljenih diskova se sastoji od:
 - srednjeg vrijednosti vremena pada
 - srednje vrijednosti vremena popravka
- u ranije navedenom slučaju uz prosječno vrijeme popravka 10 sati;
 $100.000^2 / (2 * 10) \sim 57.000$ godina
 - neke pretpostavke u navedenom slučaju nisu ispravne ali danas postoje zrcaljeni diskovi sa s.v.g. od 55 – 110 godina

Poboljšanje performansi kroz paralelizam

- kod zrcaljenih diskova stopa čitanja zahtjeva je dvostruko veća
 - brzina transfera je ista ali je broj čitanja po jedinici vremena dvostruko veći
- moguće je poboljšati i brzinu transfera dijeljenjem podataka (*engl. striping data*)
- u najjednostavnijem obliku *dijeljenje na razini bita* (*engl. bit-level striping*)
 - broj pristupa je jednak kao jednom disku ali svaki pristup može pročitati 8 puta više podataka
- *dijeljenje na razini bloka*
 - koristi se u bazama podataka jer ima prednosti u performansama nad podjelom na razini bita

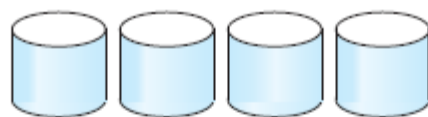
RAID razine

- korištenje više diskova
 - zrcaljenje pruža veliku pouzdanost, ali je skupo
 - dijeljenje na razini bita ili bloka pruža brzi transfer podataka, ali ne poboljšava pouzdanost
- RAID - sheme koje pružaju redundantnost po nižoj cijeni kombinirajući dijeljenje s paritetnim blokovima (*engl. parity blocks*)
 - blokovi u RAID sustavu se dijele u skupove
 - za svaki skup blokova računa se paritetni blok i pohranjuje na disku
 - i-ti bit paritetnog bloka se računa kao XOR i-tih bitova svih blokova u skupu
 - ako se bilo koji blok u skupu izgubi, njegov sadržaj se može vratiti računajući bitwise-XOR ostalih blokova u skupu i paritetnog bloka

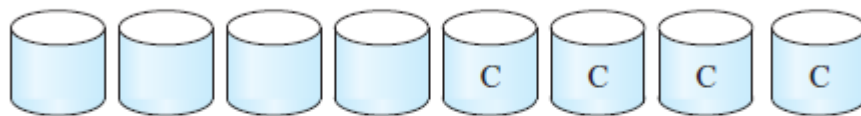
RAID razine

- kada se blok zapiše, paritetni blok njegovog skupa se mora izračunati i zapisati na disk
- nova vrijednost se može izračunati:
 - čitajući sve ostale blokove skupa i računajući paritetni blok
 - računajući XOR stare vrijednosti paritetnog bloka i stare i nove vrijednosti promijenjenog bloka
- RAID sheme predstavljaju različite kompromise cijene i performansi
- sheme se klasificiraju u **RAID razine**

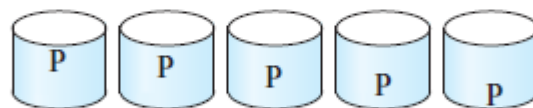
RAID razine



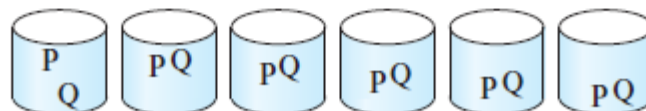
(a) RAID 0: nonredundant striping



(b) RAID 1: mirrored disks



(c) RAID 5: block-interleaved distributed parity



(d) RAID 6: P + Q redundancy

P – bitovi za ispravljanje greški
C – kopija podataka

[DSC] Slika 12.4

RAID razine

- RAID 0
 - niz diskova s dijeljenjem na razini blokova, bez redundancije
- RAID 1
 - zrcaljenje s dijeljenjem na razini blokova
 - negdje RAID 1+0 ili RAID 10 podrazumijeva zrcaljenje s dijeljenjem na razini blokova a RAID 1 samo zrcaljenje
- RAID 5

P0	0	1	2	3
4	P1	5	6	7
8	9	P2	10	11
12	13	14	P3	15
16	17	18	19	P4

RAID razine

- RAID 6
 - proširuje RAID 5
 - P+Q shema redundancije
 - koristi dva bloka pariteta, otporan na kvar dva diska
- postoje i druge varijacije na osnovne, opisane RAID sheme
- odabir RAID razine ovisi:
 - cijene dodatnog disk prostora
 - performansi I/O po sekundi
 - performansama kada se disk pokvari
 - performansama za vrijeme oporavka

Pristup disk blokovima

- niz zahtjeva za blokovima spremljenim na disku mogu biti klasificirani kao:
 - sekvencijalni pristup (engl. sequential access)
 - slučajni pristup (engl. random access)
- razvijene su brojne tehnike za poboljšanje brzine pristupa blokovima
 - pogotovo bitno kod magnetnih diskova
- pohrana u međuspremnik (*engl. buffering*), čitanje unaprijed, planiranje (*engl. scheduling*), organizacija datoteka, stalni međuspremnik za pisanje (NVRAM)

Organizacija podataka

- trajni podaci su pohranjeni na medijima stalne pohrane
 - magnetni disk ili SSD koriste blokove kao jedinice pohrane, čitaju i pišu u blokovima
 - ali baze podataka rade sa zapisima, koji su u pravilu mnogo manji od blokova
- baza podataka se mapira u datoteke koje su održavane od operativnog sustava na kojem se nalaze
 - datoteke su trajno pohranjene na disku
 - logički organizirane kao slijed zapisa
 - svaka datoteka je logički podijeljena u jedinice fiksne veličine: blokove
 - u pravilu veličine od 4-8KB

Organizacija datoteka

- blok može sadržavati nekoliko zapisa
 - točan broj ovisi o obliku organizacije fizičkih podataka koji se koristi
 - pretpostavit ćemo da niti jedan zapis nije veći od bloka
 - također, da niti jedan zapis ne može biti podijeljen na više od jednog bloka
- u relacijskoj bazi podataka n-troke različitih relacija su u pravilu različite veličine
 - pohraniti zapise jednake veličine u jednu datoteku (jednostavnije rješenje)
 - ili omogućiti da datoteke pohranjuju zapise varijabilne veličine
 - analizirat ćemo oba slučaja

Zapisi fiksne veličine

- uzmimo datoteku koja pohranjuje zapise *nastavnika*:
 - zapis je definiran (u pseudo kodu) kao:

```
type nastavnik = record  
    ID varchar(5) ;  
    ime varchar(20) ;  
    fakultet name varchar(20) ;  
    primanja numeric(8,2) ;  
end
```

- ako alociramo prostor za maksimalni broj bajtova koje svaki atribut može primiti, potreban je prostor veličine **53 bajta**

Zapisi fiksne veličine

- jednostavan pristup: koristimo prva 53 bajta za prvi zapis, druga za drugi, itd.

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Zapisi fiksne veličine

- postoje dva problema s ovakvim jednostavnim pristupom:
 - ako veličina bloka nije višekratnik broja 53 (vrlo vjerojatno), može se dogoditi da se neki zapis pohrani u dva bloka
 - što znači dva blok pristupa za čitanje i pisanje
 - vrlo je teško pobrisati zapis iz takve strukture
 - prostor mora biti popunjen s nekim drugim zapisom, ili moramo označavati obrisane zapise
- rješenje?
 - spremamo samo zapise koji mogu stati u cijelosti, jedan dio ostaje neiskorišten
 - pomičemo sve zapise za jedno mjesto nakon pobrisanog zapisa

Zapisi fiksne veličine

- pobrisan zapis 3...

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

bolje rješenje?

Zapisi fiksne veličine

- pomičemo samo zadnji zapis na mjesto obrisano

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 11	98345	Kim	Elec. Eng.	80000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000

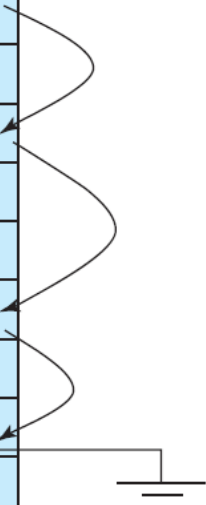
Zapisi fiksne veličine

- nije poželjno premiješati zapise na mjesto obrisano, jer to zahtjeva dodatnu operaciju pristupa
- umetanje zapisa je u pravilu mnogo češća operacija od brisanja, stoga je prihvatljivo ostaviti prazno mjesto do slijedećeg umetanja zapisa
 - za označavanje obrisanih zapisa, zbog brzine, jednostavni marker nije dovoljan
 - potrebno je definirati dodatnu strukturu podataka
- na početku datoteke alocira se nekoliko bajtova kao zaglavlje datoteke
 - zaglavlje će sadržavati različite informacije o datoteci
 - jedan element zaglavlja je *slobodna lista*, pokazivač na prvi slobodni zapis u datoteci, koji pokazuje na slijedeći... čime se formira *vezana lista*

Zapisi fiksne veličine

- slobodna lista nakon brisanja zapisa 1, 4 i 6

header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000



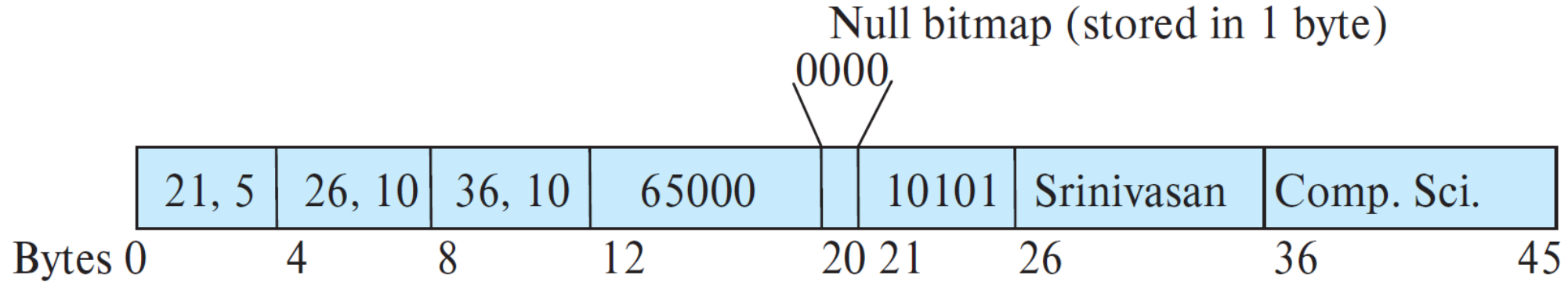
Zapisi varijabilne veličine

- zašto varijabilna veličina:
 - prisutnost polja varijabilne veličine, znakovni nizovi (VARCHAR)
 - polja, multiskupovi
 - različiti zapisi u istoj datoteci
- dva problema koji se pojavljuju:
 - kako zapisati zapis tako da je jednostavno dohvatiti atribut, čak i ako su varijabilne veličine
 - kako pohraniti varijabilne zapise unutar bloka, tako da se zapisi mogu jednostavno dohvatiti

Zapisi varijabilne veličine

- reprezentacija takvih zapise se najčešće sastoji iz dva dijela:
 - početni dio s podacima fiksne veličine, čija struktura je ista u svim zapisima
 - potom atributi varijabilne veličine
- za attribute fiksne veličine se alocira broj bajtova potrebnih za pohranu njihove vrijednosti
- atributi varijabilne veličine, u inicijalnom dijelu, su predstavljeni uređenim parom (zamak, dužina) (*engl. offset, length*)
 - zamak je broj bajta na kojem počinje atribut mjereno od početka zapisa
 - dužina je veličina atributa varijabilne veličine u bajtovima
 - varijabilni atributi se pohranjuju slijedno

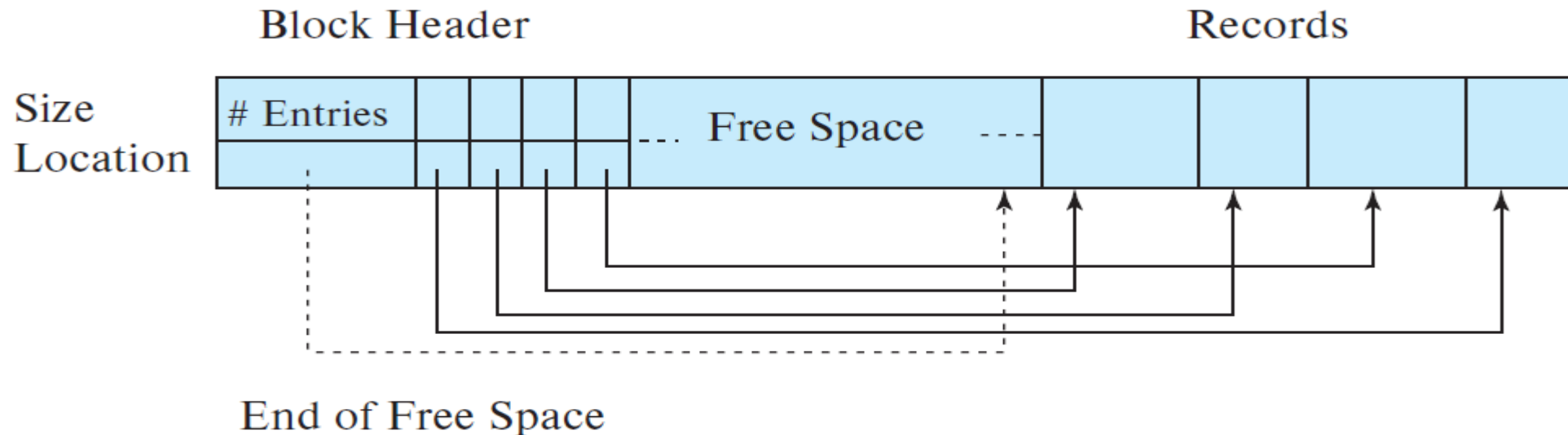
Zapisi varijabilne veličine



- **null bitmap** se koristi za označavanje atributa koji imaju null vrijednost
- u prikazanom primjeru, recimo da su primanja null, tada bi null bitmap bio 0001, a bajtovi od 12-19 ignorirani
- u nekim implementacijama null bitmap može biti na početku, a atributi s null vrijednosti izbačeni iz zapisa

Zapisi varijabilne veličine

- da bi se riješio problem pohrane zapisa varijabilne veličine u blok, dodaje se zaglavlje na početak svakog bloka:
 - broj elemenata zaglavlju
 - kraj slobodnog prostora u bloku
 - polje koje sadrži broj elemenata i njihovu veličinu u bloku



Organizacija zapisa u datotekama

- do sada smo govorili kako se zapisi reprezentiraju u strukturi datoteke
- slijedeće bitno pitanje je kako ih organizirati u datoteci
- postoji nekoliko načina:
 - organizacija pomoću hrpe (*engl. heap file organization*)
 - zapis može biti spremljen bilo gdje u datoteci gdje ima slobodnog prostora
 - sekvencijalna organizacija
 - zapisi su pohranjeni u sekvencijalnom rasporedu prema ključu pretrage za svaki zapis
 - više tablična organizacija (*engl. multitable clustering file organization*)
 - više tablica se pohranjuje u jednu datoteku, čak i u jedan blok, prednost kod join operacije

Organizacija zapisa u datotekama

- još načina za organizaciju datoteke:
 - B+tree organizacija
 - temelji se na B+tree indeks strukturi, korisna kada postoji jako puno brisanja i umetanja vrijednosti u datoteci sa sekvencijalnim pristupom
 - Hash organizacija
 - računa se hash na nekom atributu svakog zapisa, rezultati funkcije određuju u kojem bloku će zapis biti spremljen

Pohrana meta-podataka

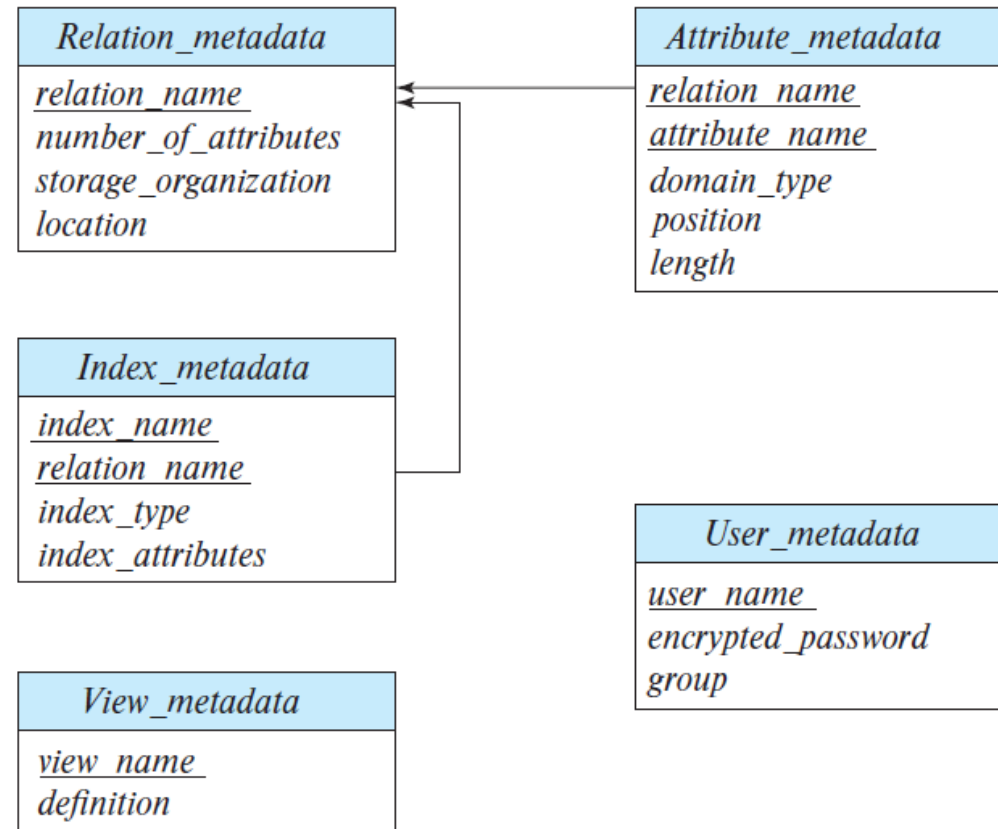
- relacijska baza podataka treba čuvati podatke o relacijama kao što su sheme
- relacijske sheme i ostali meta-podaci o relacijama se pohranjuju u rječniku ili katalogu, između ostalog pohranjuju se:
 - broj relacija
 - broj atributa u svakoj relaciji
 - domene i veličine atributa
 - imena pogleda definiranih u bazi podataka, i njihova definicija
 - ograničenja

Pohrana meta-podataka

- dodatno mnogi sustavi pohranjuju podatke o:
 - imenima korisnika, zadane sheme za korisnike, lozinke ili druge informacije za autentifikaciju korisnika
 - podatke o autorizaciji za korisnike
 - statističke i opisne podatke o relacijama i atributima
 - podatke o organizaciji pohrane (heap, sekvencijalna, hash...) i gdje se datoteke nalaze
 - podatke o indeksima
 - imena
 - imena relacija i atributa na kojima je definiran
 - tipe indeksa

Pohrana meta-podataka

- navedeni meta-podaci tvore malu bazu podataka
- neki sustavi pohranjuju te podatke u posebnim strukturama podataka
- dobro rješenje je pohraniti podatke o bazi podataka u relacije baze podataka
- pojednostavljen primjer sheme tablica ->



Buffer baze podataka

- veličina memorije servera kroz godine je sve veća
 - neke manje baze podataka mogu u potpunosti stati u memoriju
- veličina memorije koju server dodjeljuje bazi podataka može biti puno manja (u pravilu je) od same baze podataka
 - još uvijek se podaci primarno nalaze na diskovima, te se učitavaju u memoriju radi obrade, te potom ponovno pohranjuju na disk
- obzirom da je pristup puno sporiji nego unutar memorije, glavni cilj baze je što više smanjiti transfere između memorije i diska
 - tj. maksimizirati vjerojatnost da se traženi blok nalazi u memoriji
- **buffer** je dio memorije koji sadrži kopije blokova s diska

Buffer baze podataka

- podsustav za alokacijom prostora buffera se naziva **buffer upravitelj** (*engl. buffer manager*)
- ukoliko se blok već nalazi u bufferu, on prosljeđuje njegovu adresu iz glavne memorije
- u protivnom, upravitelj prvo alocira prostor za traženi blok
 - ako je potrebno izbacuje određeni blok da bi se napravilo prostora
 - odbačeni blok, ukoliko je promijenjen se zapisuje na disk
- potom učitava traženi blok i prosljeđuje njegovu adresu

Literatura

- Pročitati
 - [DSC] poglavlje 10.1. – 10.9
- Slijedeće predavanje
 - [DSC] poglavlje 11.1. – 11.4
 - korisno ([DSC] poglavlje 10.5)
 - Caltech CS121 - 11