Strukture za čuvanje podataka
Baze podataka 2

dr Miloš CVETANOVIĆ

Organizacija datoteka (file organization)

- Baza podataka se čuva kao kolekcija datotetka
  - Svaka datotetka je sekvenca zapisa (records)
  - Svaki zapis je sekvenca polja (fields)
- Jedan mogući pristup (najlakši za implementaciju)
  - Pretpostaviti da su svi zapisi fiksne veličine
  - Svaka datoteka sadrži zapise samo jednog određenog tipa (tj. jedne relacije)
  - Različite datoteke se koriste za različite relacije
- Pretpostaviti da su zapisi manji od veličine disk bloka

Zapisi fiskne veličine

- Jednostavan pristup:
  - Čuvati zapis *i* počevši od bajta $n \cdot (i-1)$, gde je *n* veličina svakog od zapisa
  - Pristup zapisima je jednostavan, ali neki od zapisa mogu biti na dva bloka
    Modifikacija: ne dozvoliti da zapisi prelaze granicu bloka
    (tj. zapis koji ne može da stane u tekući blok, ceo se prebacuje u naredni blok)

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 fiskne veličine

- Brisanje zapisa *i* može imati nekoliko pristupa:
  - **Premestiti zapise *i*+1, …,*n* na pozicije *i*, …,*n*-1**
  - Premestiti zapis *n* na poziciju *i*
  - Ne premeštati zapise, već održavati listu slobodnih zapisa - lista slobodnih (free list)
- Obrisan zapis 3, a zapisi od 4 do 11 pomereni

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

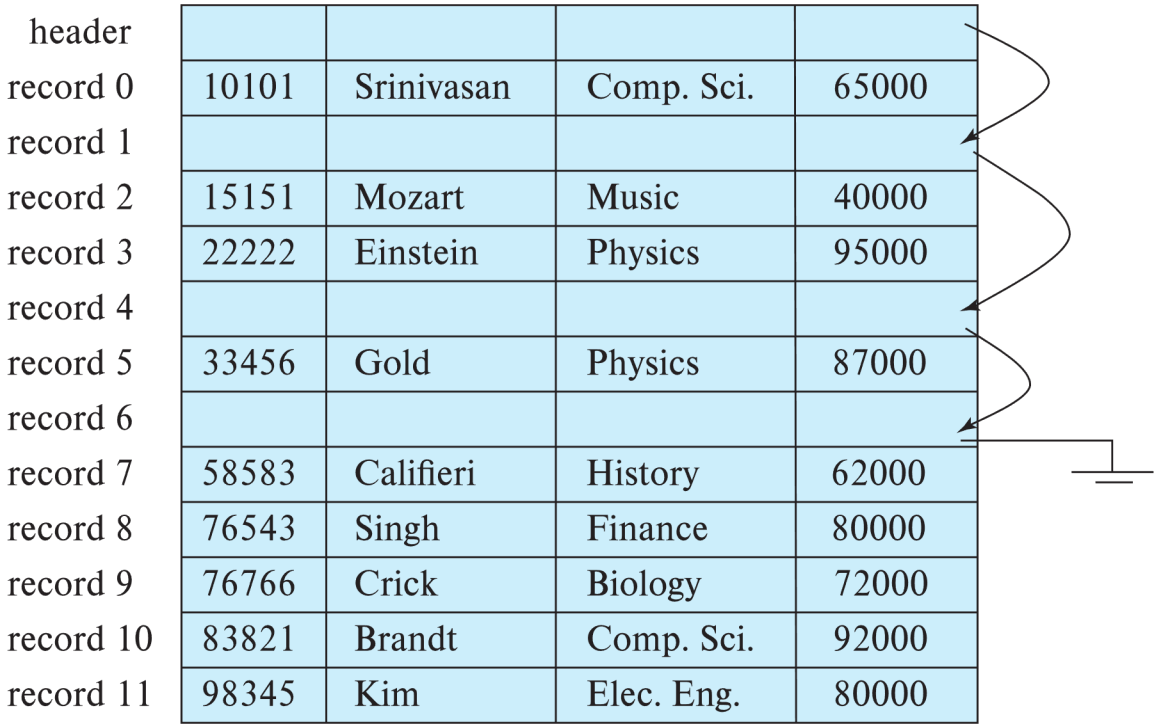
Zapisi fiskne veličine

- Brisanje zapisa *i* može imati nekoliko pristupa:
  - Premestiti zapise *i*+1, …,*n* na pozicije *i*, …,*n*-1
  - **Premestiti zapis *n* na poziciju *i***
  - Ne premeštati zapise, već održavati listu slobodnih zapisa - lista slobodnih (free list)
- Obrisan zapis 3 i zamenjen zapisom 11

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 fiskne veličine

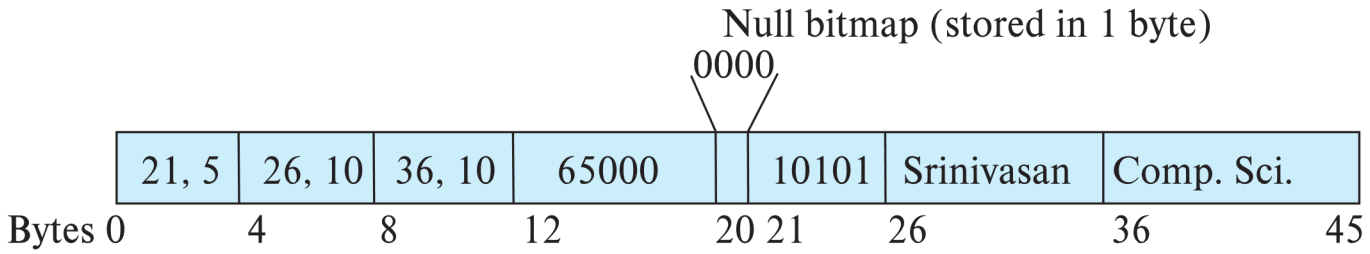
- Brisanje zapisa *i* može imati nekoliko pristupa:
  - Premestiti zapise *i*+1, …,*n* na pozicije *i*, …,*n*-1
  - Premestiti zapis *n* na poziciju *i*
  - **Ne premeštati zapise, već održavati listu slobodnih zapisa - lista slobodnih (free list)**
- Obrisani zapisi 1, 4 i 6



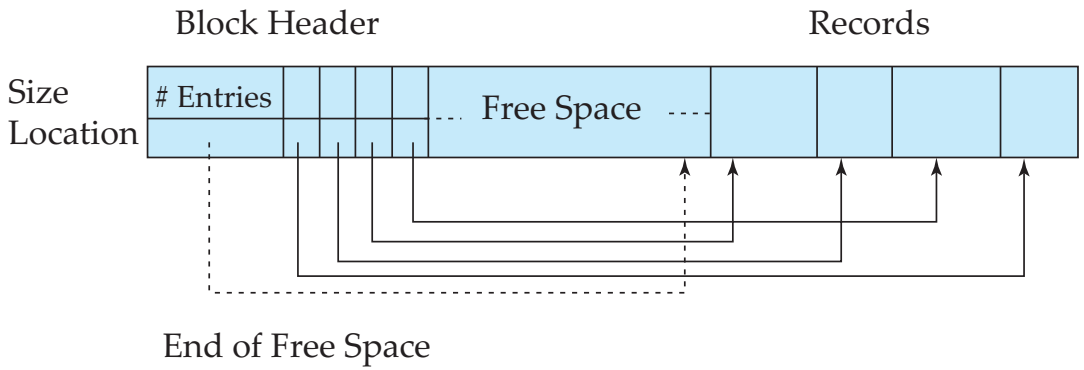
header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology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

Zapisi različite veličine

- Zapisi različite veličine nastaju u bazama podataka usled više razloga:
  - Čuvanje više vrsta zapisa u jednoj datoteci (tj. više relacija u jednoj datoteci)
  - Zapisi imaju polja koja dozvoljavaju varijabilnu dužinu poput stringova (**varchar**)
  - Zapisi koji dozvoljavaju ponavljanje polja (kod modela koji su prethodili relacionom)
- Polja (tj. atributi) se čuvaju u odgovarajućem redosledu
- Polja varijabilne dužine su predstavljena fiksnom veličinom
  tj. parom vrednosti (offset, lenght), a podatak se čuva nakon svih polja fiksne dužine
- NULL je evidentiran u vidu niza bitova (**null-value bitmap**)
- Npr. Ako bi zapis imao tri varchar atributa (dužina 5, 10 i 10 karaktera, respektivno)
  i jedan celobrojni atribut (veličine 8 bajtova)



Zapisi različite veličine – struktura stranice sa prorezima (slotted page structure)



- Svaki blok ima zaglavlje, a potom prostor za smeštanje zapisa
- Zaglavlje (slotted page header) sadrži:
  - Informaciju o broju zapisa u bloku
  - Informaciju o kraju slobodnog prostora u bloku
  - Informacije o lokaciji i veličini svakog od zapisa
- Zapisi mogu biti premeštani unutar stranice kako bi formirali kontinualni prostor bez slobodnog prostora među njima – informacije u zaglavlju moraju biti ažurirane
- Pokazivači na podatke (npr. Pokazivači indeksnih struktura) ne bi trebalo da pokazuju direktno na neki zapis, već na ulaz u zaglavlje koji odgovara tom zapisu

Čuvanje velikih objekata

- Za čuvanje velikih objekata, SQL predviđa tip podataka BLOB/CLOB
- Ograničenje da zapis mora biti manji od veličine disk bloka
- Moguća rešenja:
  - Čuvati velike objekte kao posebne datoteka u fajl sistemu
  - Čuvati velike objekte kao posebne datoteka kojima upravlja DBMS
  - Podeliti veliki objekat na delove i potom svaki od delova čuvati u posebnom zapisu u za to namenski formiranoj relaciji
    (npr. PostreSQL TOAST – dodatna tabela za svaku tabelu koju korisnik kreira)

Organizacija zapisa u datoteci

- Gomila (**heap**) – zapis može biti sačuvan bilo gde u datoteci gde ima dovoljno mesta
- Sekvencijlna (**sequential**) – zapisi se čuvaju u redosledu sortiranom na osnovu vrednosti jednog ili više atributa koji predstavljaju ključ pretrage (search key)
- Grupisanje više tabela (**multitable clustering**) – zapisi nekoliko različitih relacija se čuvaju u istoj datoteci (minimizacija I/O prilikom pristupa zapisima koji se spajaju u čestim upitima)
- B^+ stablo (**B^+ tree**) – zapisi se čuvaju u strukturi B^+ stabla (sortiranost čak i nakon ins/del)
  - Koristi se i za formiranje indeksa nad relacijama/tabelama
- Heš (**hash**) – heš funkcija se računa za atribut koji služi kao ključ pretrage, a rezultat definiše u kom bloku datoteke zapis treba da bude sačuvan
  - Koristi se i za formiranje indeksa nad relacijama/tabelama

Organizacija datoteke kao gomile zapisa (heap file)

- Zapis može biti sačuvan bilo gde u datoteci gde ima dovoljno mesta
- Zapisi se obično ne pomeraju nakon inicijalnog upisa
- Veoma je važno efikasno pronalaženje slobodnog prostora u datoteci
- Mapa slobodnog prostora (**free-space map**)
  - Niz sa po 1 ulazom po bloku
    Svaki ulaz je veličine od nekoliko bita do jednog bajta
    i beleži procenat bloka koji je slobodan
  - Na primer, ukoliko se koriste 3 bita po bloku,
    onda vrednost podeljena sa 8 daje informaciju o procentu bloka koji je slobodan

4	2	1	4	7	3	6	5	1	2	0	1	1	0	5	6

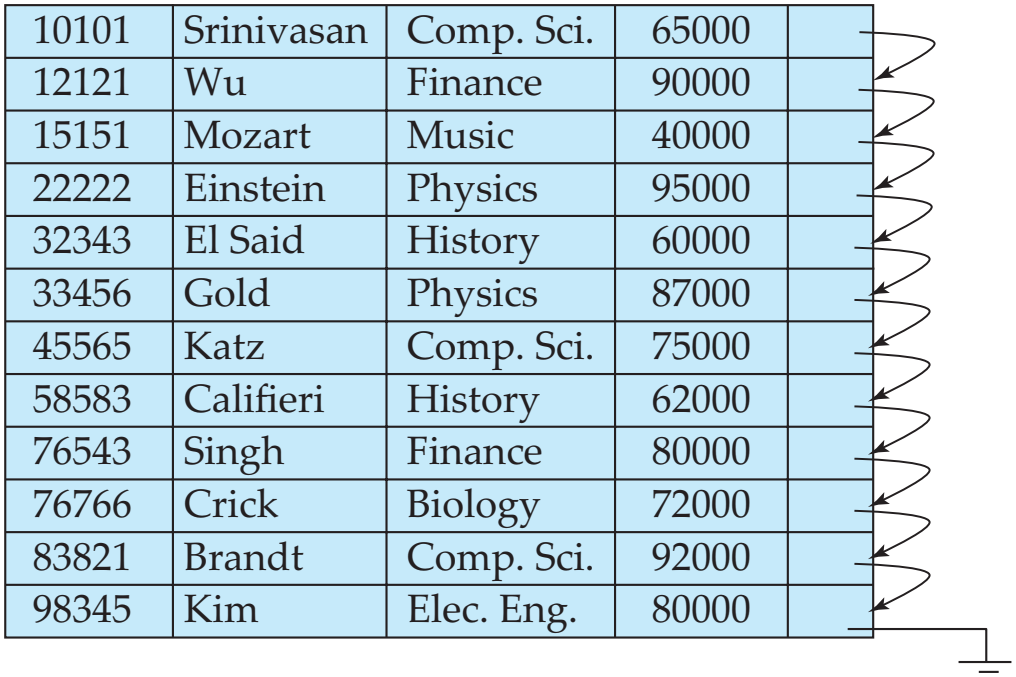
  - Radi efikasnije pretrage, može postojati mapa slobodnog prostora drugog nivoa
  - Na primer, svaki ulaz mape slobodnog prostora čuva informaciju o maksimumu slobodnog prostora u 4 ulaza u mapi slobodnog prostora prvog nivoa

4	7	2	6

- Mapa slobodnog prostora se periodično čuva na disk, potpuno je prihvatljivo da mapa sadrži pogrešne (stare) vrednosti za neke ulaze (jer će to biti detektovano i korigovano)

Organizacija sekvencijalne datoteke (sequential file)

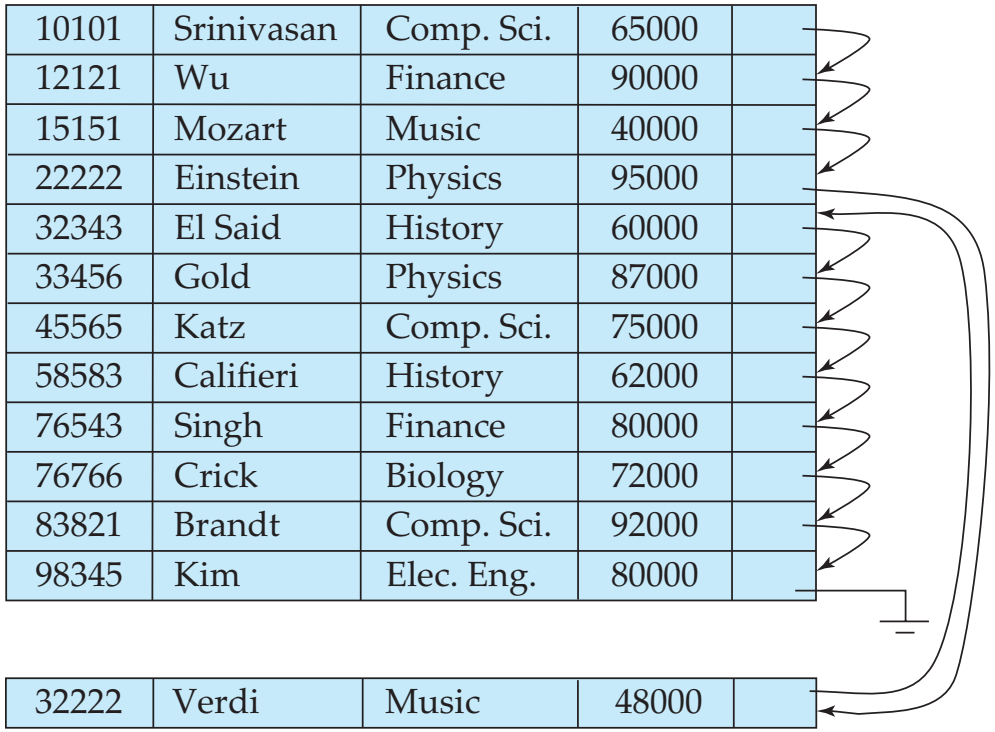
- Pogodna za aplikacija koje zahtevaju sekvencijalnu obradu čitave datoteka
- Zapisi su u datoteci uređeni (sortirani) na osnovu ključa pretrage (search key)



10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Organizacija sekvencijalne datoteke (sequential file)

- Brisanje zapisa – upotreba lanca pokazivača
- Dodavanje zapisa – lociranje pozicije gde zapisi treba da bude dodata
  - Ukoliko ima slobodnog prostora onda dodati tu
  - Ukoliko nema slobodnog prostora, onda dodati zapis u dodatni blok (**overflow block**)
  - U svakom slučaju, lanac pokazivača treba ažurirati
- Postoji potreba da se datoteka povremeno reorganizuje (kako bi se povratila sortiranost)



10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

32222	Verdi	Music	48000	

Organizacija datoteke sa grupisanje više tabela (multitable clustering)

- Čuvati nekoliko relacija u jednoj datoteci
  - Na primer: *department* (dept_name, building, budget)
    instructor (ID, name, dept_name, salary)

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Physics	Watson	70000

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

Comp. Sci.	Taylor	100000	
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
Physics	Watson	70000	
33456	Gold	Physics	87000

- Dobar izbor kada su česti upiti spajanja (npr. *department* ∞ *instructor*)
  ili upiti koji uključuju jedan red iz *department* i sve odgovarajuće redove iz *instructor*
- Loš izbor za upit koji zahteva samo redove iz *department*
  (rešenje – lanac pokazivača da bi se povezali zapisi jedne od relacija)
- Dolazi do formiranja zapisa različite veličine

Particionisanje (table partitioning)

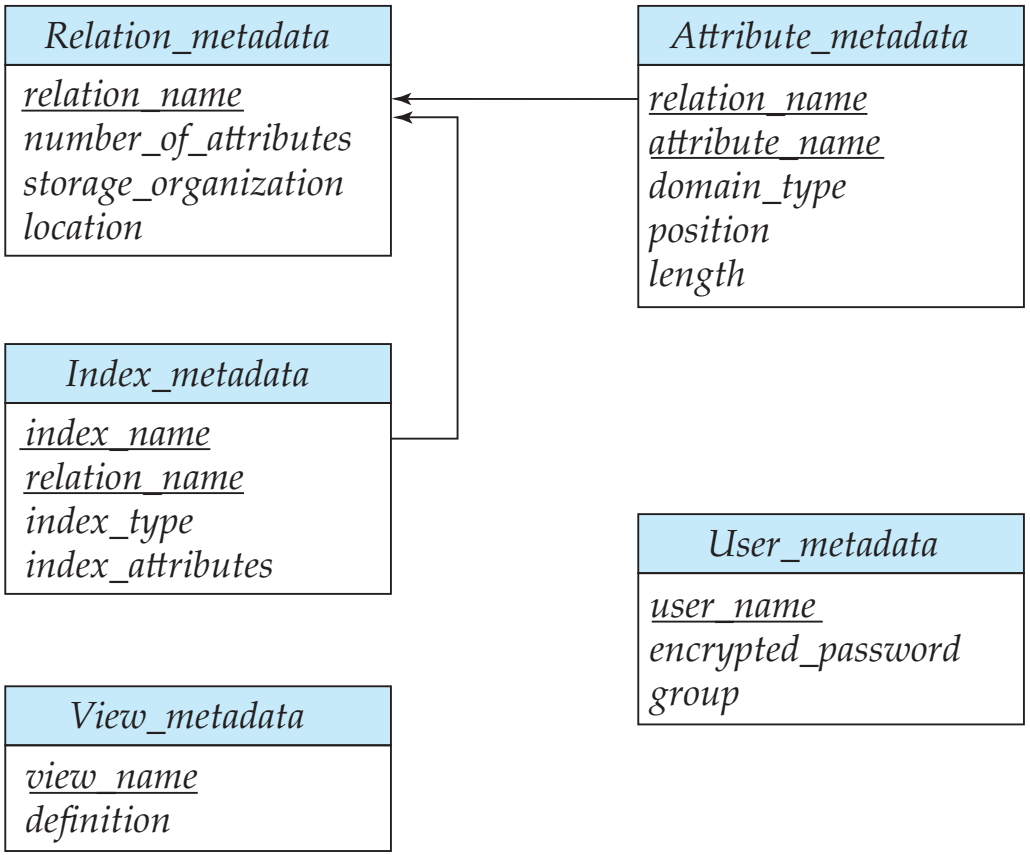
- Particionisanje tabele – zapisi jedne relacije se mogu particionisati na manje relacije koje se čuvaju odvojeno
- Na primer, tabela *transactions* se može particionisati na *transactions_2018*, *transactions_2019* itd.
- Upiti nad *transactions* moraju pristupiti zapisima u svim participima
  - Izuzev u slučaju kada upit ima uslov *year=2019*, kada je dovoljna samo jedna particija
- Prednosti
  - Smanjuje cenu nekih operacija (npr. upravljanje slobodnim prostorom)
  - Omogućava da se različite particije čuvaju na različitim uređajima (npr. tekuća godina na SSD, a prethodne godine na magnetnim HDD)

Rečnik podataka (data dictionary)

- Rečnik podataka – (tzv. system catalog) čuva metapodatke (podatke o podacima)
- Informacija o relacijama
  - Nazive relacija
  - Nazive, tipove i dužine atributa svake od relacija
  - Nazive i definicije pogleda
  - Integritetska ograničenja
- Informacija o korisnicima i njihovim lozinkama
- Statističke i opisne podatke
  - Broj redova u svakoj od relacija
  - Broj različitih vrednosti za svaki od atributa
- Informacije o fizičkoj organizaciji datoteka
  - Kako se čuva relacija (sequential, multitable cluster, …)
  - Fizička lokacija relacije
- Informacije o indeksima

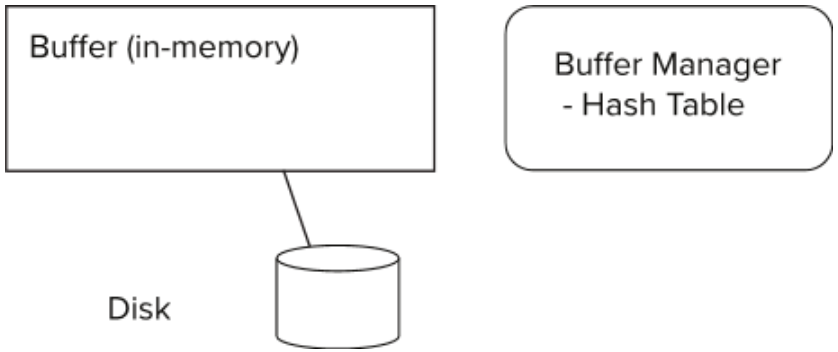
Rečnik podataka (data dictionary)

- Relaciona reprezentacija metapodataka na disku
- Specijalizovane strukture podataka dizajnirane za efikasan pristup u memoriji



Pristup sistemu za čuvanje podataka

- Blokovi su jedinice alociranja prostora na sistemu za čuvanje i jedinice prenosa podataka
- DBMS pokušava da minimizira broj transfera blokova između diska i memorije
  Broj pristupa disku se može smanjiti time što se što veći broj blokova čuva u memoriji
- Bafer (**buffer**) – deo memorije u kome se čuvaju kopije disk blokova
- Menadžer bafera (**buffer manager**) – podsistem odgovoran za upravljanje prostorom bafera u memoriji



Menadžer bafera

- Kada je potrebno dohvatiti blok sa diska, menadžer bafera prima poziv
  - Ako je blok već u baferu, menadžer bafera vraća adresu tog bloka
  - Ako blok nije u baferu, onda menadžer bafera
    * Alocira prostor u baferu (za potrebe smeštanja bloka)
      Zamenjuje (tj. izbacuje) neki drugi blok, ukoliko je to potrebno,
      kako bi obezbedio prostor za novi blok
      Zamenjeni blok upisuje na disk samo ako je on bio modifikovan
      od prethodnog trenutka kada je bio upisan na, odnosno, dohvaćen sa diska
    * Učitava blok sa diska u bafer i vraća adresu tog bloka pozivaocu koji je tražio blok

Menadžer bafera

- Kada je potrebno dohvatiti blok sa diska, menadžer bafera prima poziv
  - Ako je blok već u baferu, menadžer bafera vraća adresu tog bloka
  - Ako blok nije u baferu, onda menadžer bafera
    * Alocira prostor u baferu (za potrebe smeštanja bloka)
      Zamenjuje (tj. izbacuje) neki drugi blok, ukoliko je to potrebno, kako bi obezbedio prostor za novi blok
      Zamenjeni blok upisuje na disk samo ako je on bio modifikovan od prethodnog trenutka kada je bio upisan na, odnosno, dohvaćen sa diska
    * Učitava blok sa diska u bafer i vraća adresu tog bloka pozivaocu koji je tražio blok
- Zakačen blok (**pinned block**) – blok u memoriji za koji je zabranjeno upisati ga na disk
  - Pin se radi pre čitanja podataka sa, odnosno, upisivanja podataka u blok
  - Unpin se radi kada se čitanje odnosno upis završi
  - Više konkurentnih pin/unpin operacija je moguće (brojač **pin count**, uklanjanje iz memorije je moguće samo ako je brojač jednak 0)
- Deljeni i ekskluzivni katanac na baferu (**shared, exclusive lock**)
  - Potrebni radi sprečavanja čitanja sadržaja bloka dok se on pomera ili menja
  - Čitanje zahteva deljeni katanac, a upis ekskluzivni katanac
  - Samo jedan ekskluzivni katanac u nekom trenutku (bez deljenih katanaca)

Menadžer bafera – politike zamene (buffer-replacement policies)

- Većina operativnih sistema zamenjuje blok koji je najdavnije korišćen
  LRU (**least recently used**) strategija
  - Ideja LRU – koristiti prethodni obrazac pristupa blokovima
    za potrebe predikcije budućih pristupa
  - LRU može biti loš izbor za neke upite
- Upiti imaju dobro definisan obrazac pristupa (npr. Sekvencijalno skeniranje)
  pa stoga DBMS može koristiti tu informaciju za predikciju potrebnih blokova
- Mešovita strategija je poželjna – strategija dobija nagoveštaje od optimizatora upita
- Ukoliko bi se koristio samo LRU, primer lošeg obrasca pristupa bi mogao biti
  upit koji zahteva spajanje dve relacije *r* i *s* algoritmom ugnežđenih petlji:

for each tuple t_r in r do
    for each tuple t_s in s do
        if the tuples t_r *and* t_s *match …*

Menadžer bafera – politike zamene (buffer-replacement policies)

- Strategija izbacivanja odmah (**toss-immediate**) – oslobođa prostor zauzet blokom čim se završi obrada poslednjeg zapisa u tom bloku
- Strategija izbacivanja najskorije korišćenog bloka, tj. MRU (**most recently used**) strategija kod koje DBMS postavlja pin na blok koji se trenutno obrađuje. Nakon što se završi obrada poslednjeg zapisa u tom bloku, skida se pin sa bloka (unpin), i blok postaje najskorije korišćen blok
- Menadžer bafera može koristiti statističke informacije koje se odnose na verovatnoću da će zahtev koristiti određenu relaciju
  - Na primer, rečniku podataka (system catalog) se često pristupa i zbog toga ga treba stalno držati u memoriji
- Operativni sistem ili menadžer bafera mogu menjati redosled upisa
  - Može dovesti do oštećenja struktura podataka na disku
    * Ulančana lista blokova sa nedostajućim blokovima na disku
    * Fajl sistem obavlja proveru konzistencije kako bi detektovao takve situacije
  - Pažjivo raspoređivanje upisa može eliminisati pojavu takvih problema

Menadžer bafera – politike zamene (buffer-replacement policies)

- Menadžer bafera podržava nasilno izbacivanje (**forced output**) blokova iz memorije na disk za potrebe oporavka
- Stalni bafer upisa (**nonvolatile write buffer**) – ubrzava upise na disk time što se upis blokova obavlja u stalni bafer trenutno, a upis na disk optimizuje odnosno, radi se promena redosleda upisa radi minimizacije pomeranje ruke diska
- Disk dnevnik (**log disk**) – disk posvećen čuvanju sekvence ažuriranje blokova
  - Koristi se kao i stalni bafer upisa
  - Upis na disk dnevnik je veoma brz jer ne zahteva pretragu i pozicioniranje
- Fajl sistem sa vođenjem dnevnika (**journaling file system**)
  - Upis podataka odman u NV-RAM ili disk dnevnik
  - Promena redosleda upisa bez vođenja dnevnika: rizik od gubitka podataka

Čuvanje podataka po kolonama (column-oriented storage)

- Termin koji se takođe koristi je kolonsko predstavljanje (**columnar representation**) Čuvanje svakog atributa neke relacije zasebno
- Na primer: *instructor* (ID, name, dept_name, salary)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

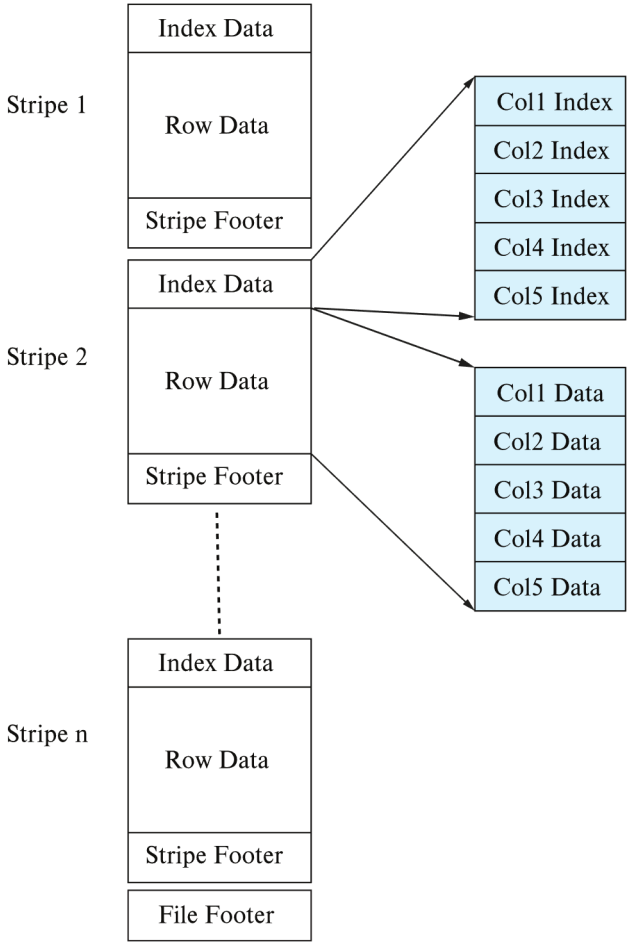
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Čuvanje podataka po kolonama (column-oriented storage)

- Prednosti:
  - Smanjen broj I/O operacija ako se pristupa samo nekim atributima
  - Poboljšanje CPU cache performansi
  - Poboljšanje kompresije podataka
  - Omogućavanje vektorkse obrade (vector processing) na modernim CPU
- Nedostaci:
  - Cena rekonstruisanja celog reda
  - Cena brisanja ili ažuriranja reda
  - Cena dekompresije podataka
- Jako efikasna kod analitičkih obrada u skladištima podataka (OLAP)
  npr. agregiranje svih vrednosti kolone je efikasnije nego kod čuvanja podataka po redovima
- Čuvanje podataka po redovima je poželjnije kod transakcionih sistema (OLTP)
- Neki DBMS podržavaju oba načina čuvanja (**hybrid row/column stores**)

Čuvanje podataka po kolonama (column-oriented storage)

- ORC i Parquet: fajl formati sa kolonskim predstavljanjem podataka u datoteci
- Veoma popularni za obradu velikih količina podataka (**big-data**)
- Na primer, ORC fajl format:



Baze podataka u memoriji (main-memory database)

- Mogu čuvati zapise direktno u memoriji, bez potrebe za menadžerom bafera
- Čuvanje podataka po kolonama se može koristiti
  kod baza podataka u operativnoj memoriji
  za potrebe aplikacija u oblasti podrške odlučivanjima (**decision support**)
  - Kompresija podataka smanjuje potrebnu količinu operativne memorije