

Indeksi

Indeksi

- Indeksi predstavljaju dodatne strukture kojima je omogućen neskevencijalni (direktni) pristup podacima.
- Jednom fajlu može biti pridruženo više indeksa. Indeks se definira nad atributom ili skupom atributa, tzv. indeksnim poljem, koji predstavlja ključ po kojem se vrši pretraga indeksne strukture umesto celih fajlova sa podacima.
- Indeksni fajl se sastoji iz **indeksnih zapisa** (*index entry*) u formi



kojima su povezane vrednosti ključa/indeksiranog polja sa informacijom o lokaciji slogova podataka (*data entry*) sa navedenom vrednošću ključa.

Informacija o lokaciji sloga – pointer ka slogu u fajlu sa podacima ili adresa bloka.

- Prednost upotrebe
veća količina potrebne informacije (vrednosti ključa po kom se vrši pretraga) se učitava u manjem broju blokova -> brža pretraga po indeksiranom polju
-

Podela

- Primarni vs. Sekundarni

Da li uređenje slogova u fajlu sa podacima prati uređenje indeksiranih polja u indeksu?

- Gusti vs. Retki

Da li se u indeksu nalaze sve vrednosti indeksiranog polja koje se pojavljuju i u fajlu sa podacima?

- Klasterovani vs. Neklasterovani (Grupišući/Negrupišući)

Da li se su podaci u fajlu uređeni prema vrednosti ključa u indeksu ili ne?
(klasterovani indeks nije isto što i klasterovana fajl oranizacija)

- Jednonivoiski vs. Višenivoiski

Da li se u indeksnoj strukturi definiše 'indeks nad indeksom' ili ne?

Vrste i merenje kvaliteta indeksa

- Dve osnovne vrste indeksa:
 - **Uređeni**. Implementirani kao strukture uređene po vrednosti ključa i prilagođene broj pretrazi.
 - **Hash indeksi**. Implementirani kao heš tabele. Zasnovani na uniformnoj distribuciji vrednosti u kolekciju korpi/bucket-a.
 - Dve vrste pomoćnih stuktura:
 - Uređena stabla
 - Heš tabele
 - Metrike za procenu indeksa
 - Vreme pristupa traženom podatku
 - Vreme dodavanja
 - Vreme brisanja
 - Prostorno prekoračenje
-

Hash indeksi / Direktna organizacija

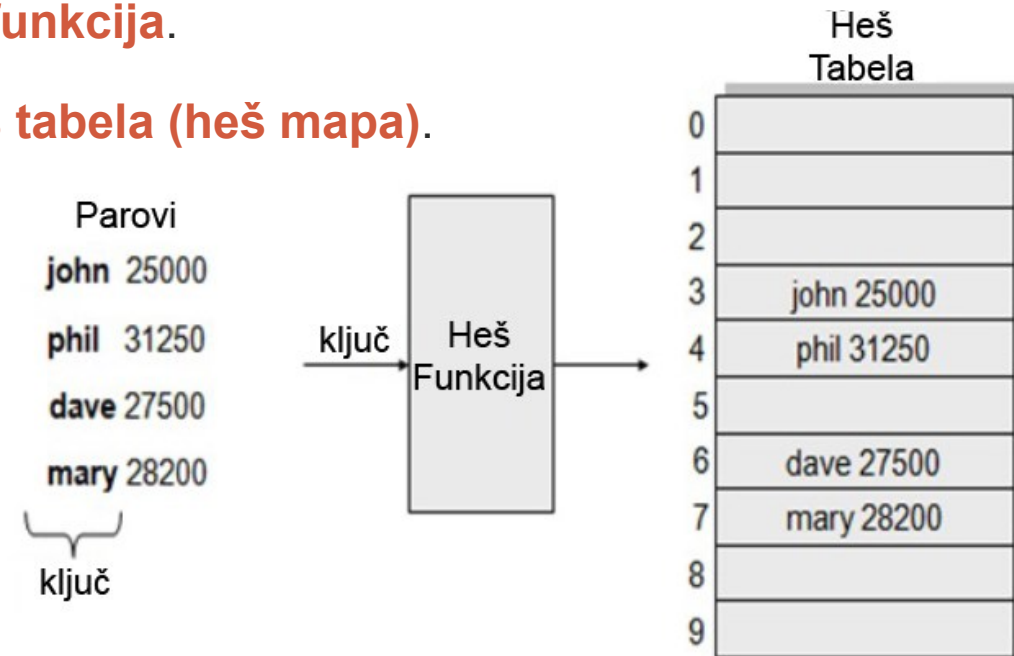
Organizacija fajlova

Heširanje

- Heširanje – tehnika pretrage podataka. Na osnovu vrednosti ključa naći podatak pridružen ključu. Ključ i podatak su smešteni u strukturu - mapa/tabela. U postupku heširanja se pozicija traženog para ključ-vrednost određuje direktno.
- Za određivanje pozicije para ključ-vrednost se koristi **heš funkcija**.
- Parovi ključ-vrednost su smešteni u strukturi nazvanoj **heš tabela (heš mapa)**.
- Heš funkcijom se vrši transformisanje ključa u ceo broj (**heš vrednost**) koji pripada opsegu indeksa u tabeli i **predstavlja poziciju u tabeli** na kojoj se nalazi traženi par.

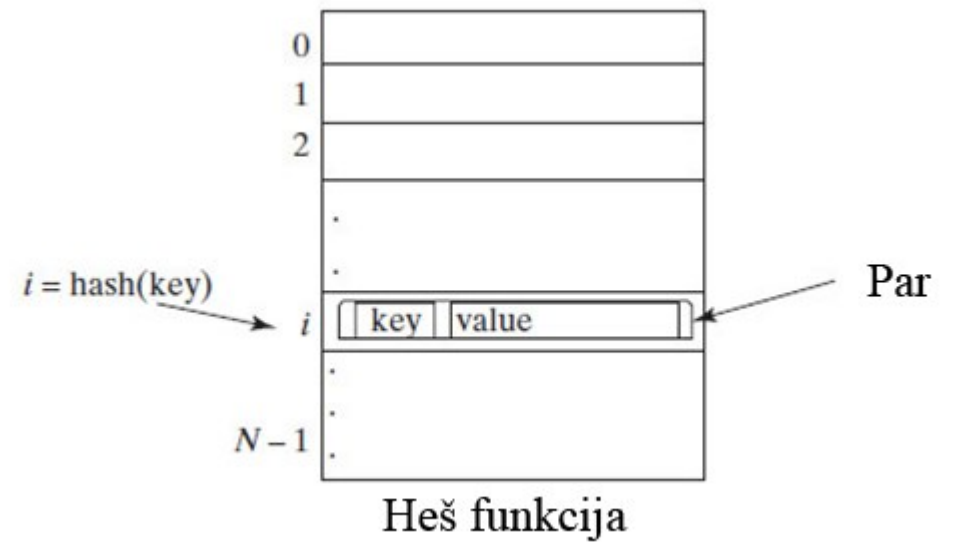
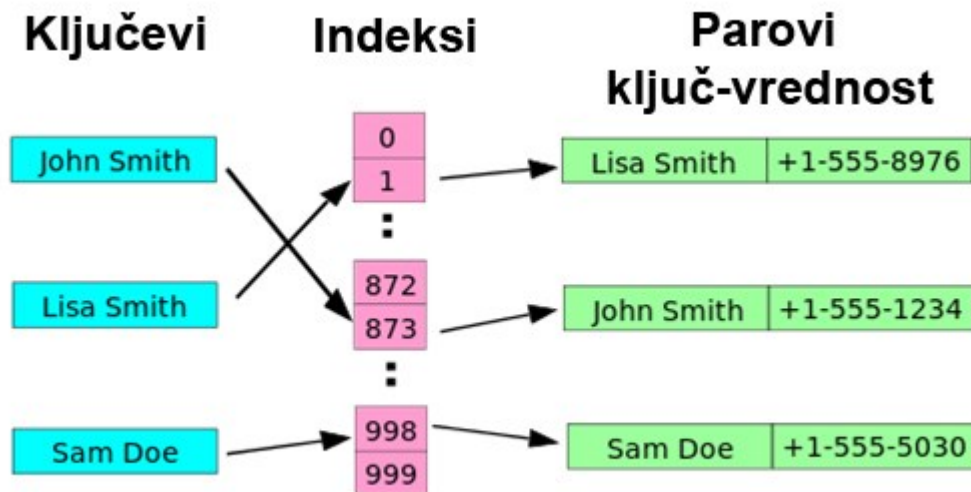
ključ -> pozicija u tabeli -> <ključ, vrednost>

↑
heš vrednost ili heš kod



Hash tabela

- Heš tabela – struktura podataka koja koristi heš funkciju za efikasno preslikavanje ključeva u njima pridružene vrednosti.



Heš tabela - kompleksnost

- Kompleksnost (n – broj vrednosti ključa)
 - Memorijska kompleksnost $O(n)$
 - Kompleksnost operacije: - prosečna $O(1)$, najgora $O(n)$
 - Pri odabiru tehnike heširanja, tj. implementaciji heš tabela, potrebno je:
 - Odbrati efikasnu heš funkciju.
 - Odrediti hešing šemu – izabrati način razrešavanja kolizije ključeva.
 - Manja kolizija -> Veća heš tabela
 - Manja tabela -> veća kolizija -> sporije pronalaženje traženih slogova.
-

Heš funkcija – osobine

- Mora vratiti broj od 0 do tablesize (veličina heš tabele).
- Idealno, heš funkcija bi trebalo da preslikava svaki mogući ključ u zaseban indeks, ali ovaj zadatak je u praksi najčešće neostvariv.
- Heš kolizije / sudari – različiti ključevi sa istim heš vrednostima.
- Ako je veličina heš tabele manja od broja različitih ključeva, kolizija je nužna.
- Svakoj heš vrednosti je pridružen jedan skup parova ključ-vrednost.

Najgora moguća hash funkcija

Sve vrednosti ključa se slikaju u istu heš vrednost.

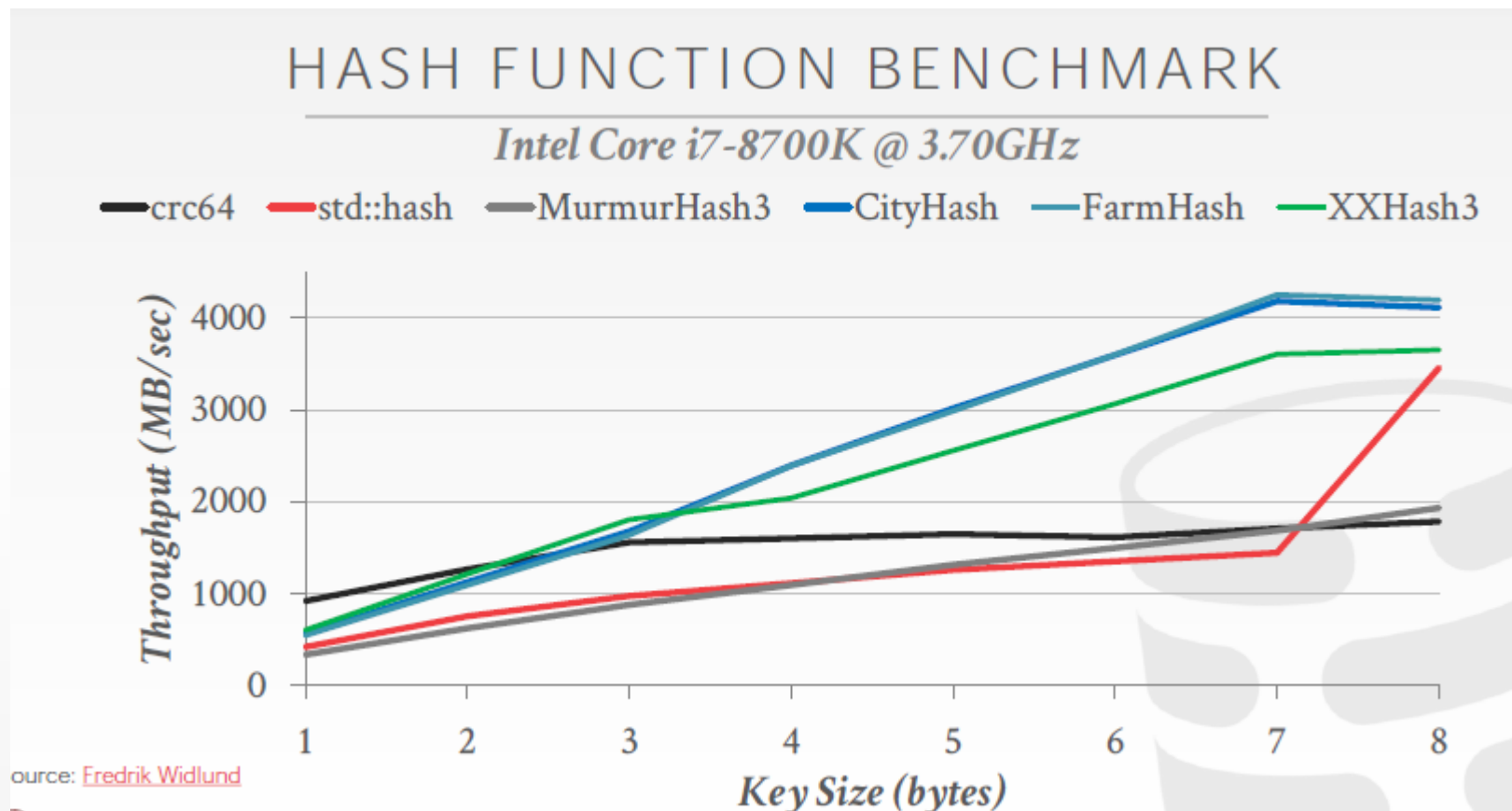
Idealna hash funkcija

Ravnomerno raspodeljuje ključeve tako da je svakoj heš vrednosti pridružen isti broj slogova.

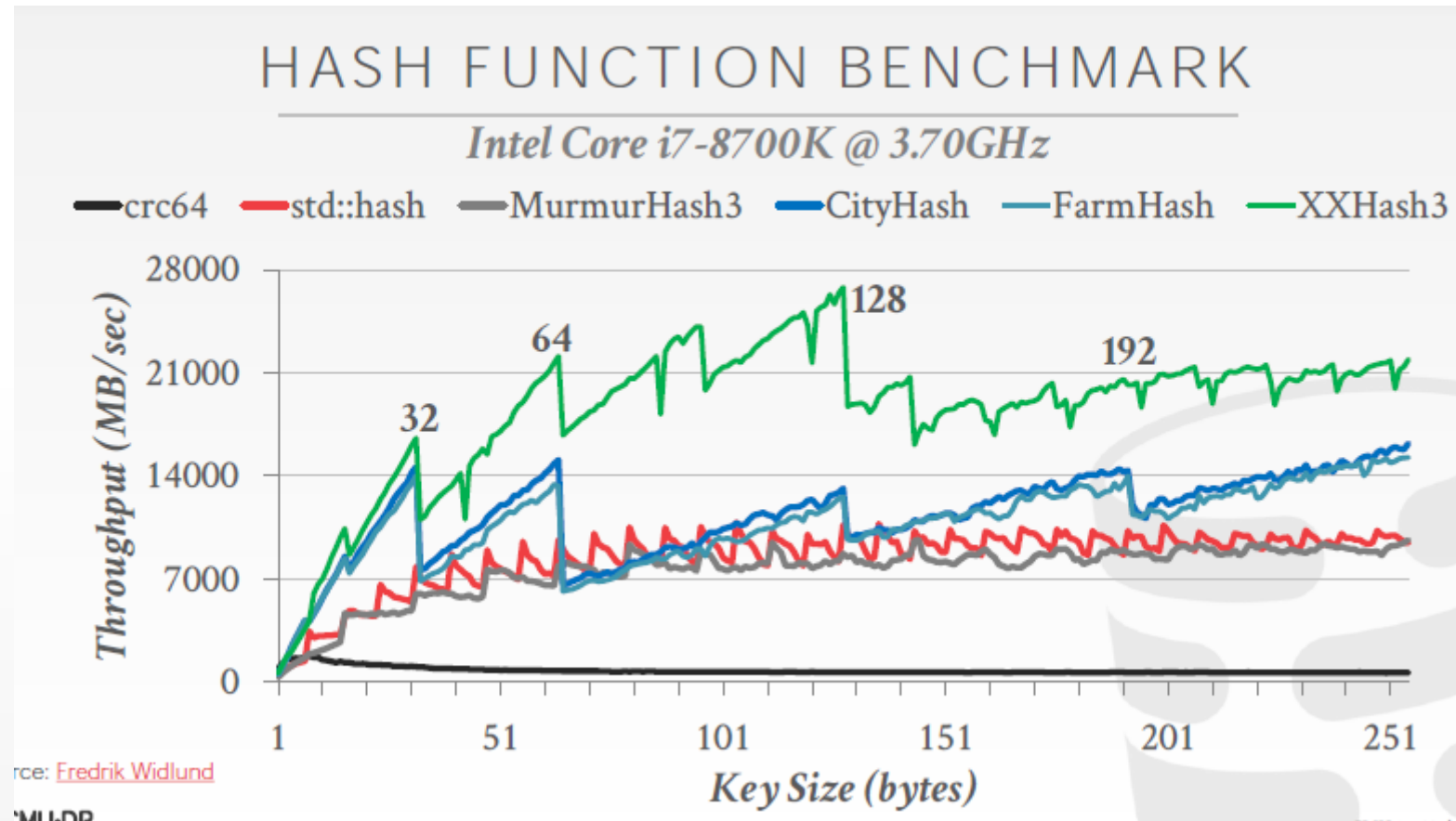
Hash funkcija

- Za svaki ključ vraća njegovu integer reprezentaciju.
 - U DBMSovima se izbegavaju kriptografske heš funkcije.
 - Faktori na osnovu kojih se vrši odabir su brzina i mala stopa kolizije ključeva.
 - **CRC-64(1975)** – detekcija grešaka na mrežama.
 - **MurmurHash(2008)** - fast, general purpose hash function.
 - **Google CityHash(2011)** - modifikovan Murmur, prilagođen kraćim ključevima (<64 bytes).
 - **Facebook XXHash(2012)** - From the creator of zstd compression.
 - **Google FarmHash(2014)** – novija verzija CityHash sa boljim stopama kolizije.
-

Hash funkcija



Hash funkcija



Heš funkcija sa kolizijom – osobine

Poželjne osobine:

- Uniformna distribucija.
 - Svakoj heš vrednosti se pridružuje isti broj vrednosti ključa iz skupa svih mogućih vrednosti.
 - Distribucija je slučajna.
 - U proseku, za svaku heš vrednost postoji približno isti broj parova ključ-vrednost koji su mu dodeljeni, bez obzira na trenutnu raspodelu vrednosti ključa.
Hash vrednost ne sme da bude u korelaciji sa bilo kojim kriterijumom uređivanja vrednosti ključa, npr. Alfabetским ili uređenjem prema dužini zapisa ključa.
-

Heš funkcija – osobine primer

Plate (ogranak, prosecna_plata)

Heš funkcija definisana nad poljem ogranak koja mapira ogranke sa i-tim početnim slovom u i-ti bucket.

- Neuniformna distribucija vrednosti ključa, jer ima više mesta sa početnim slovom B nego X.

Heš funkcija definisana nad poljem račun koja podrazumeva 10 opsega, 1–10,000, 10,001–20,000, ...

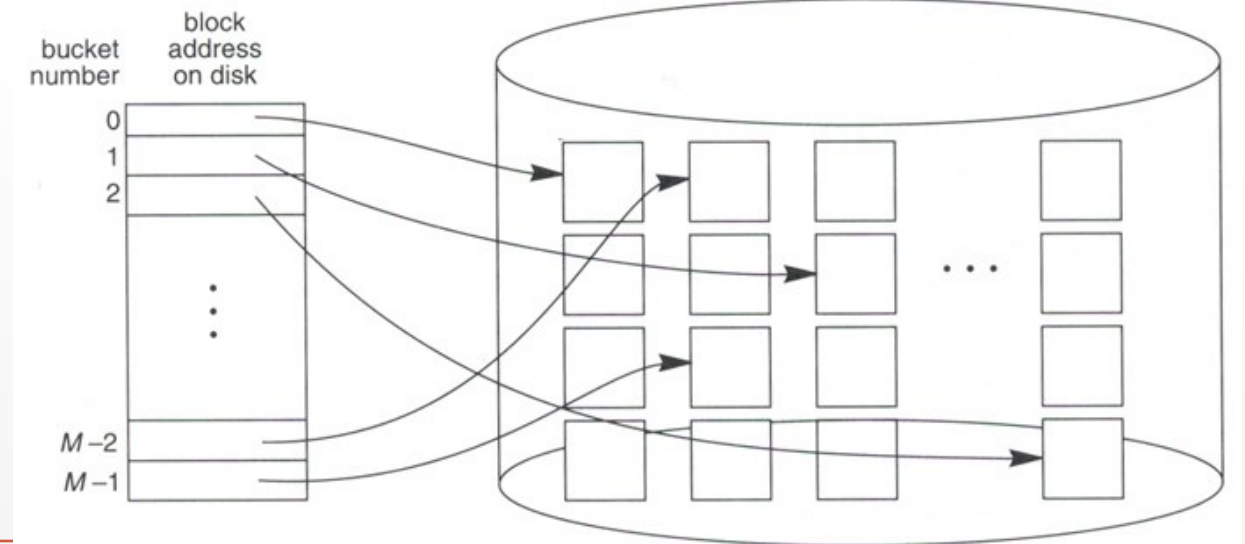
- Raspodela jeste uniformna, ali nije slučajna, jer je realno očekivati da će najveći broj plata biti u rasponu od 40001-50000.

Dobra heš funkcija ima prosečno vreme pronalaženja traženog sloga blisko maloj konstantnoj vrednosti, koja je nezavisna od broja trenutno upisanih vrednosti.

Unutrašnje i spoljašnje heširanje

- Unutrašnje heširanje – heš tabela sadrži cele slogove, u svakom slotu se nalazi jedan slog.
- Slogovi se nalaze na disku. Heš tabela sadrži samo ključ i pokazivač na lokaciju gde se slog može pronaći. Adresni prostor se sastoji od bucket-a. Svi slogovi sa istom heš vrednošću se nalaze u istom bucket (jedan bucket – minimalno jedan blok)

	Name	Ssn	Job	Salary
0				
1				
2				
3				
		⋮		
$M - 2$				
$M - 1$				



Hešing šema

- Fiksni broj heš vrednosti – **statičko heširanje**
 - Definirati heš funkciju na osnovu trenutne veličine skupa slogova.
 - Definirati heš funkciju na osnovu pretpostavljene veličine fajla u budućnosti. Trošenje memorijskog prostora preko potrebnog.
 - Periodična reorganizacija heš strukture u skladu sa rastom fajla. Vremenski zahtevno. Potrebna zabrana pristupa tokom reorganizacije.
 - **Dinamičko heširanje** – skup heš vrednosti koje se menja tokom vremena.
-

Statičko heširanje

Na osnovu načina na koji se rešava problem kolizije (situacije u kojoj je lokacija na koju treba smestiti novi slog već zauzeta), razlikujemo (glavni metodi):

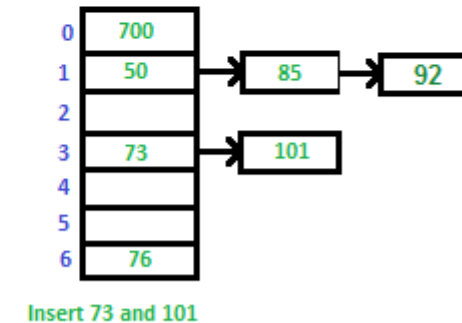
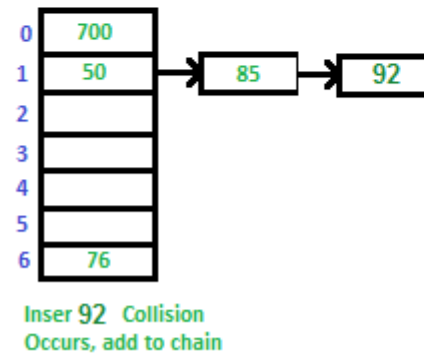
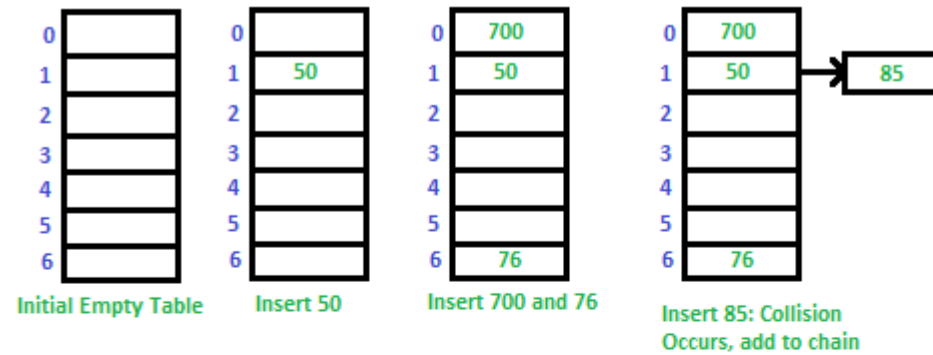
- **Otvoreno heširanje – otvoreno adresiranje**
kolizije se razrešavaju u okviru tabele
 - **Zatvoreno heširanje – posebno (odvojeno) ulančavanje**
kolizije se razrešavaju van tabele
-

Otvoreno adresiranje

- Svi parovi ključ-vrednost su smešteni u heš tabeli. Dakle, tabela mora biti dovoljno velika.
 - Svodi se na pronalaženje neke druge lokacije u heš tabeli različite od one dobijene heš funkcijom.
 - Formulirati pravilo koje za svaki ključ određuje ispitni niz kao niz adresa u tabeli koje se proveravaju pri umetanju novog ključa.
 - Neke od tehnika otvorenog adresiranja:
 - linearno pretraživanje ([linear probing](#))
 - slučajno pretraživanje
 - kvadratno pretraživanje
 - dvostruko heširanje
-

Odvojeno ulančavanje

Za svaku vrednost heš ključa postoji posebna lista slogova.



Upis/Brisanje – spoljašnje zatvoreno heširanje

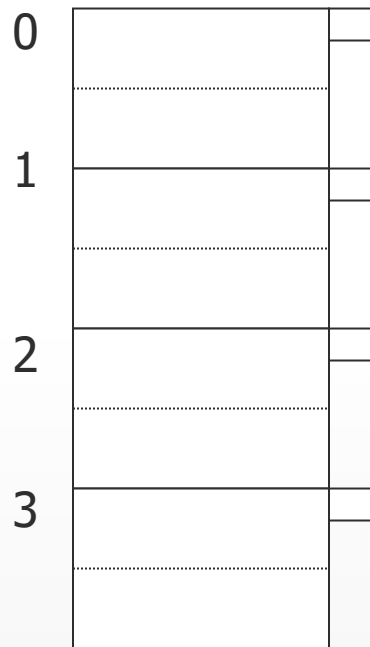
INSERT:

$h(a) = 1$

$h(b) = 2$

$h(c) = 1$

$h(d) = 0$



Upis/Brisanje

INSERT:

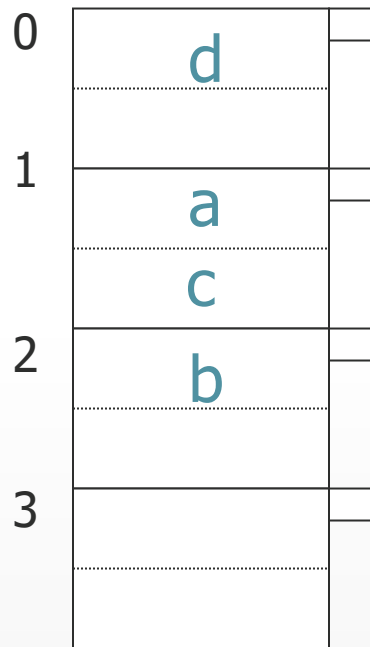
$h(a) = 1$

$h(b) = 2$

$h(c) = 1$

$h(d) = 0$

$h(e) = 1$



Upis/Brisanje

INSERT:

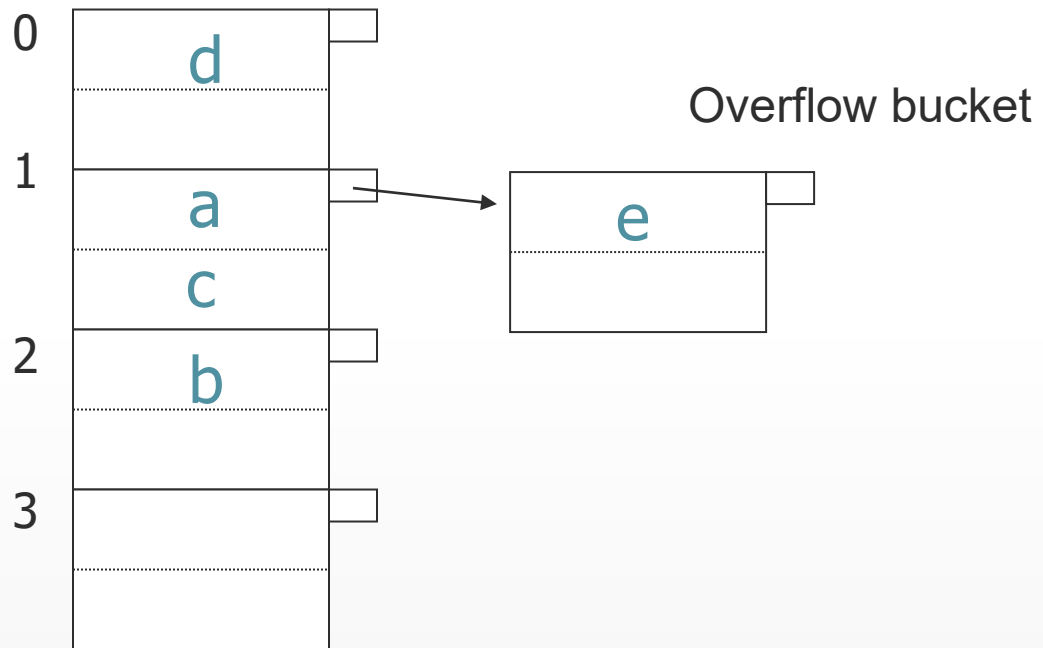
$h(a) = 1$

$h(b) = 2$

$h(c) = 1$

$h(d) = 0$

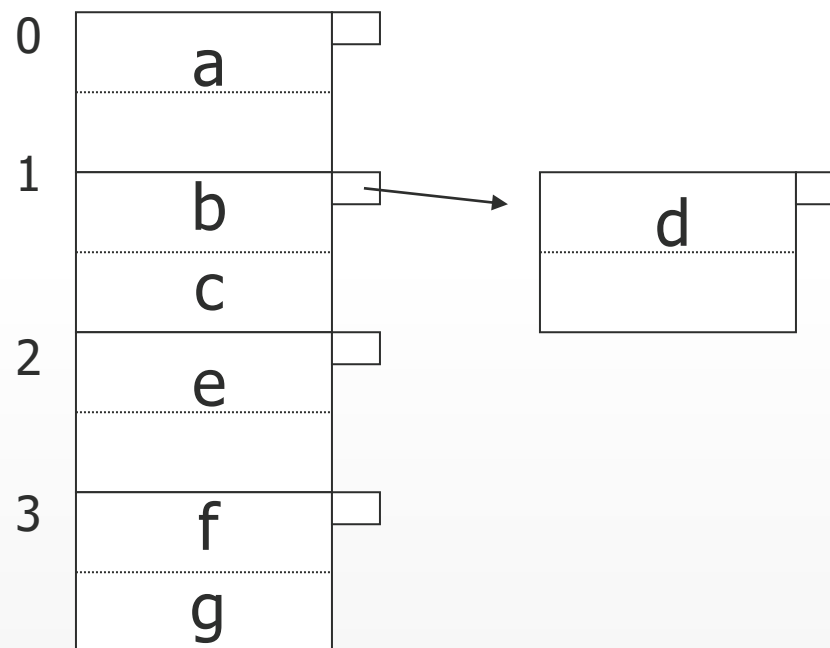
$h(e) = 1$



Upis/Brisanje

Delete:

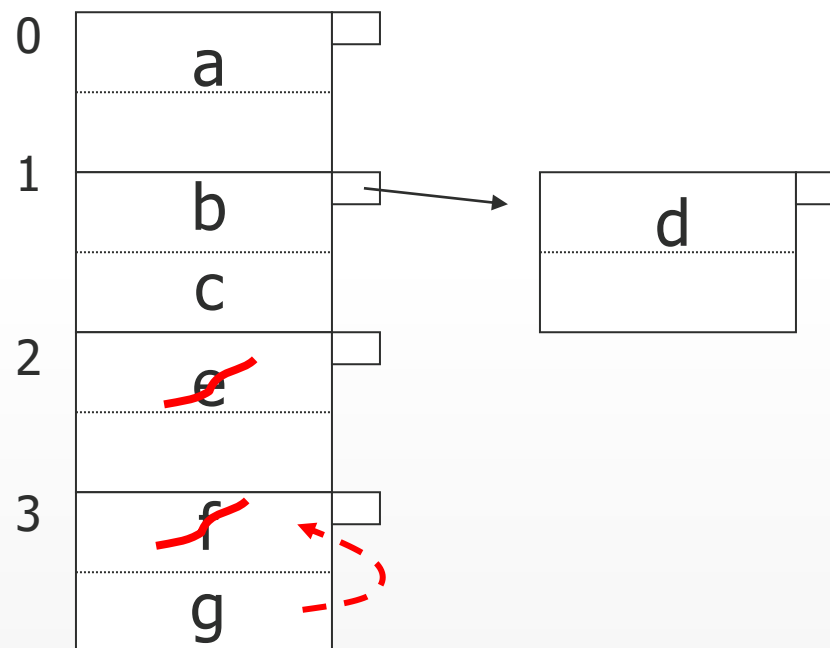
e
f



Upis/Brisanje

Delete:

e
f



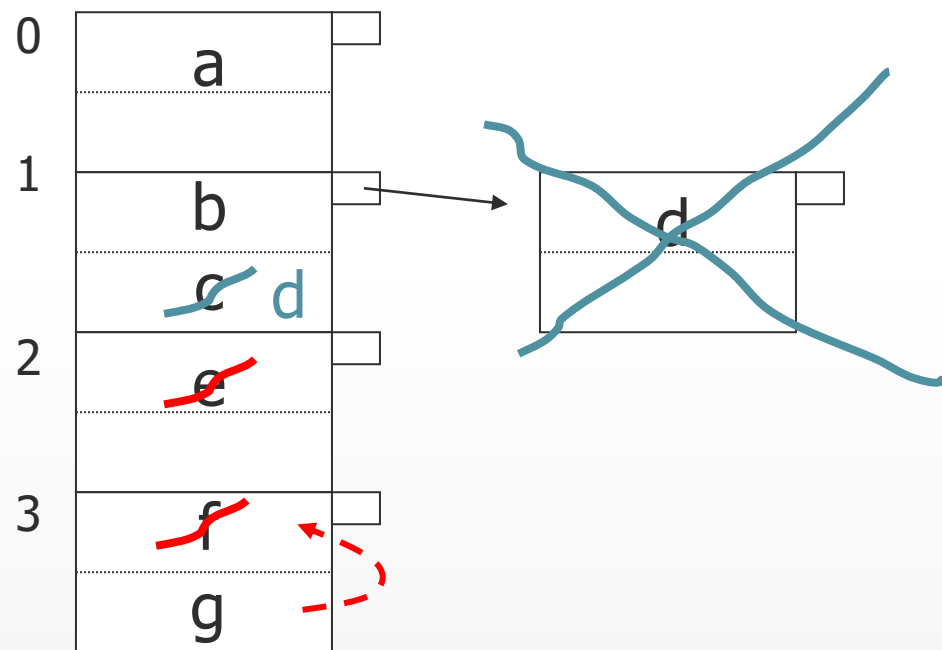
Upis/Brisanje

Delete:

e

f

c



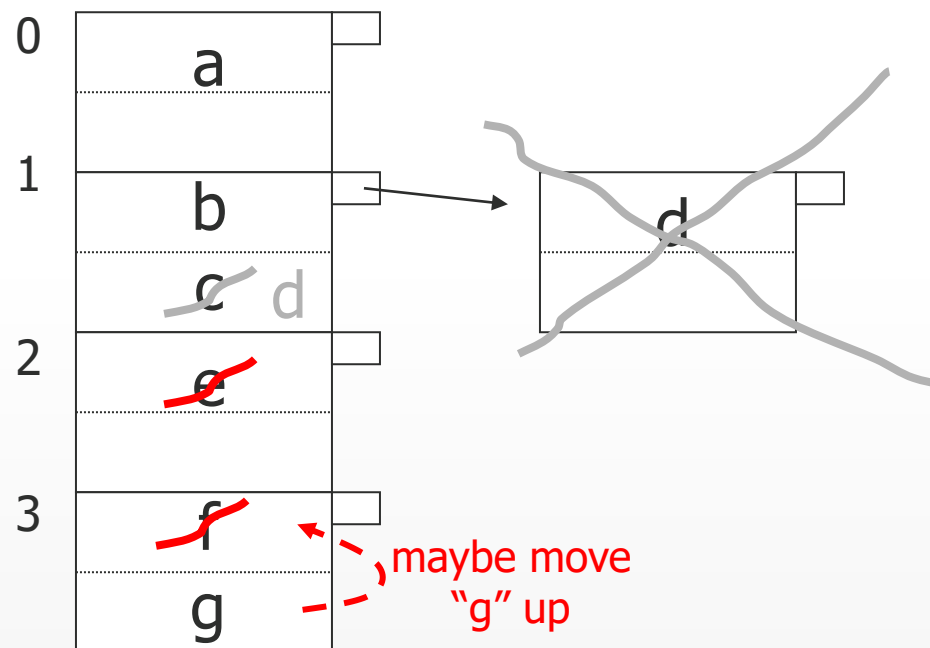
Upis/Brisanje

Delete:

e

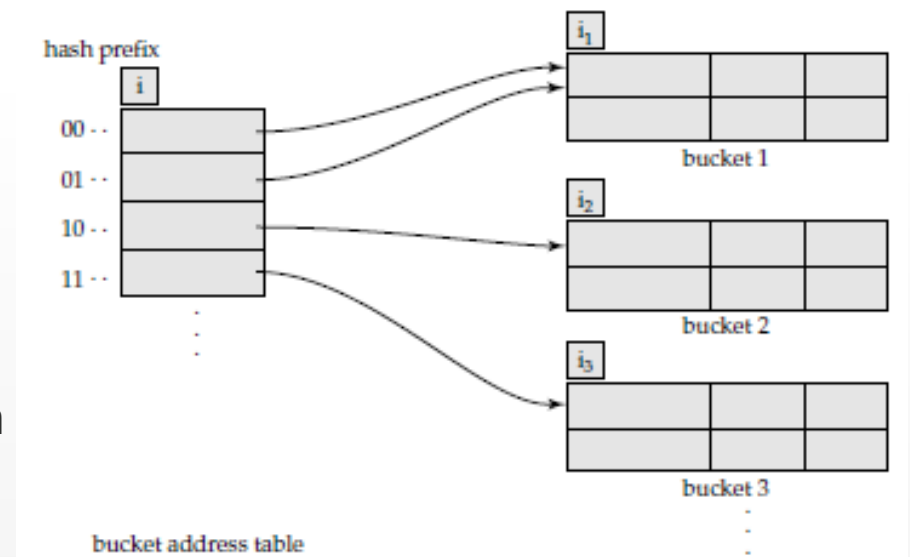
f

c



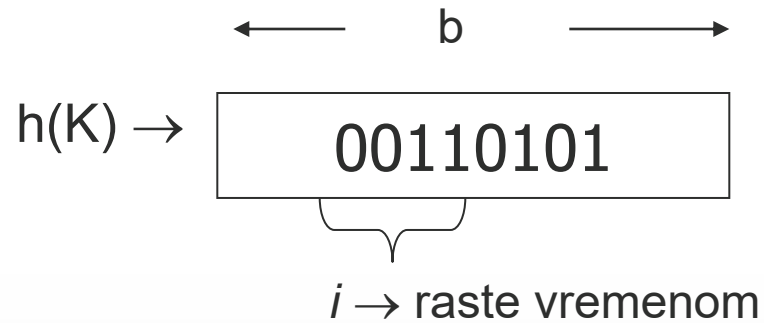
Dinamičko heširanje

- Statičko spoljašnje heširanje - kada datoteka raste, raste dužina lanaca prekoračilaca, što degradira performanse rasute organizacije fajla.
- Rešenje: dinamičko heširanje.
- Jedna od tehnika – proširivo heširanje (*Extendable hashing*)
- Adresna tabela sadrži zaglavlje u kojoj se nalazi dužina hash prefiksa.
- Funkcija heširanja prevodi ključ u binarni kod i uzima prvih i bitova kao takozvani pseudoključ.
- Svaki bucket ima svoju lokalnu dužinu prefiksa (*lokalna dubina bloka*) koja označava koliko prvih bitova pseudo ključa svih zapisa u bucket-u se poklapa.



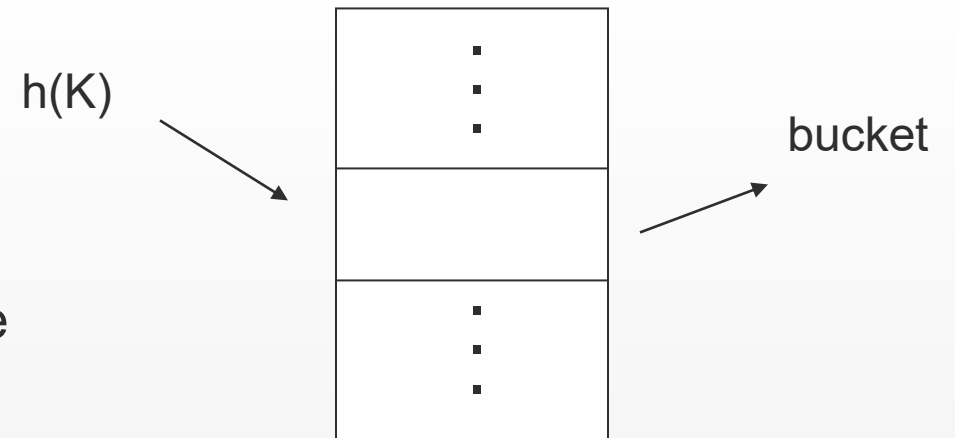
Proširivo heširanje

(a) Upotreba i od b bitova vrednosti heš funkcije za datu vrednost ključa

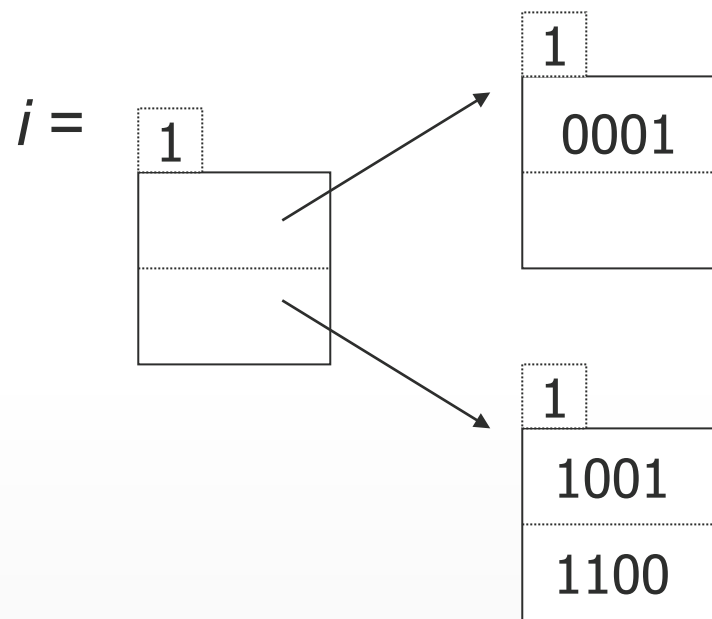


(b) Upotreba posebne strukture – **direktorijum**.

Strukturu čini zaglavlje u kome se daje dubina direktorijuma (d), a zatim 2^d pokazivača na adrese blokova.

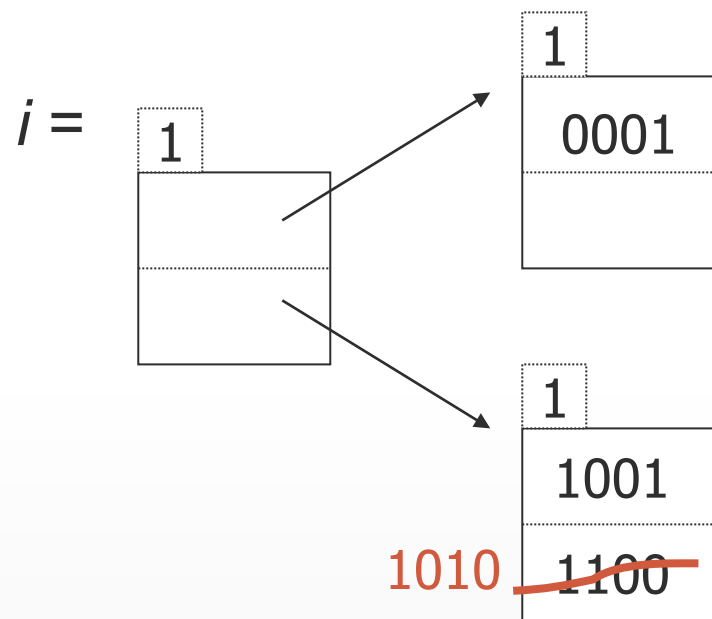


Proširivo heširanje

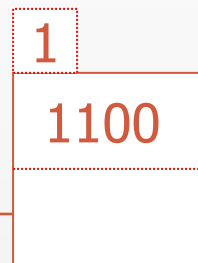


Insert 1010

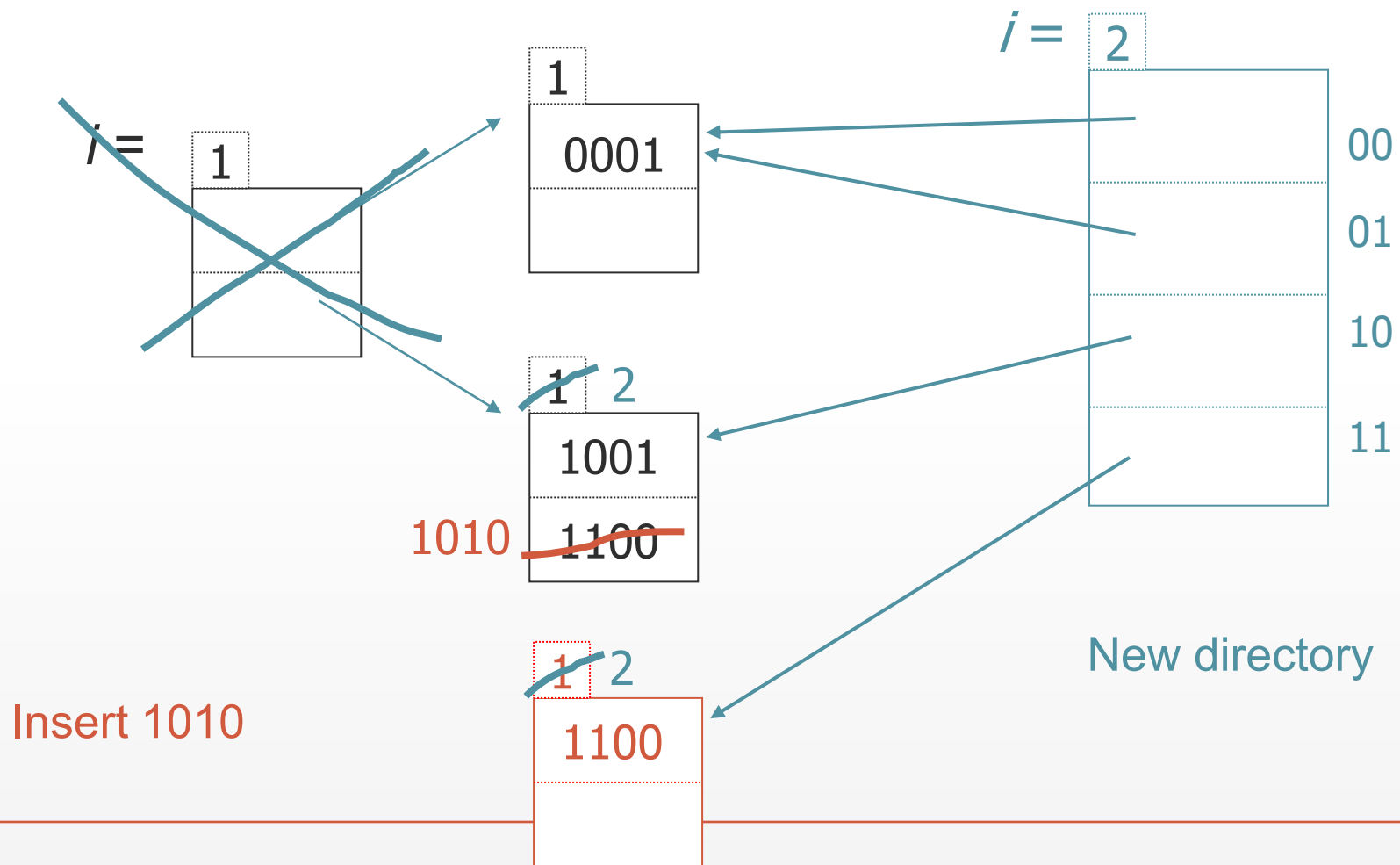
Proširivo heširanje



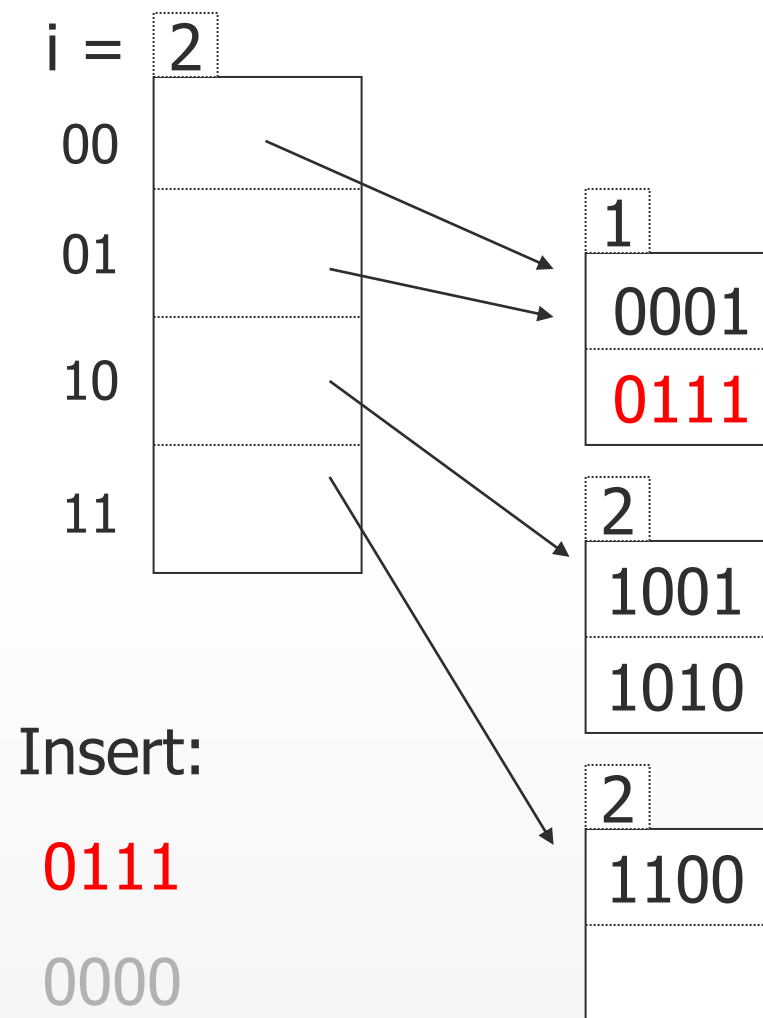
Insert 1010



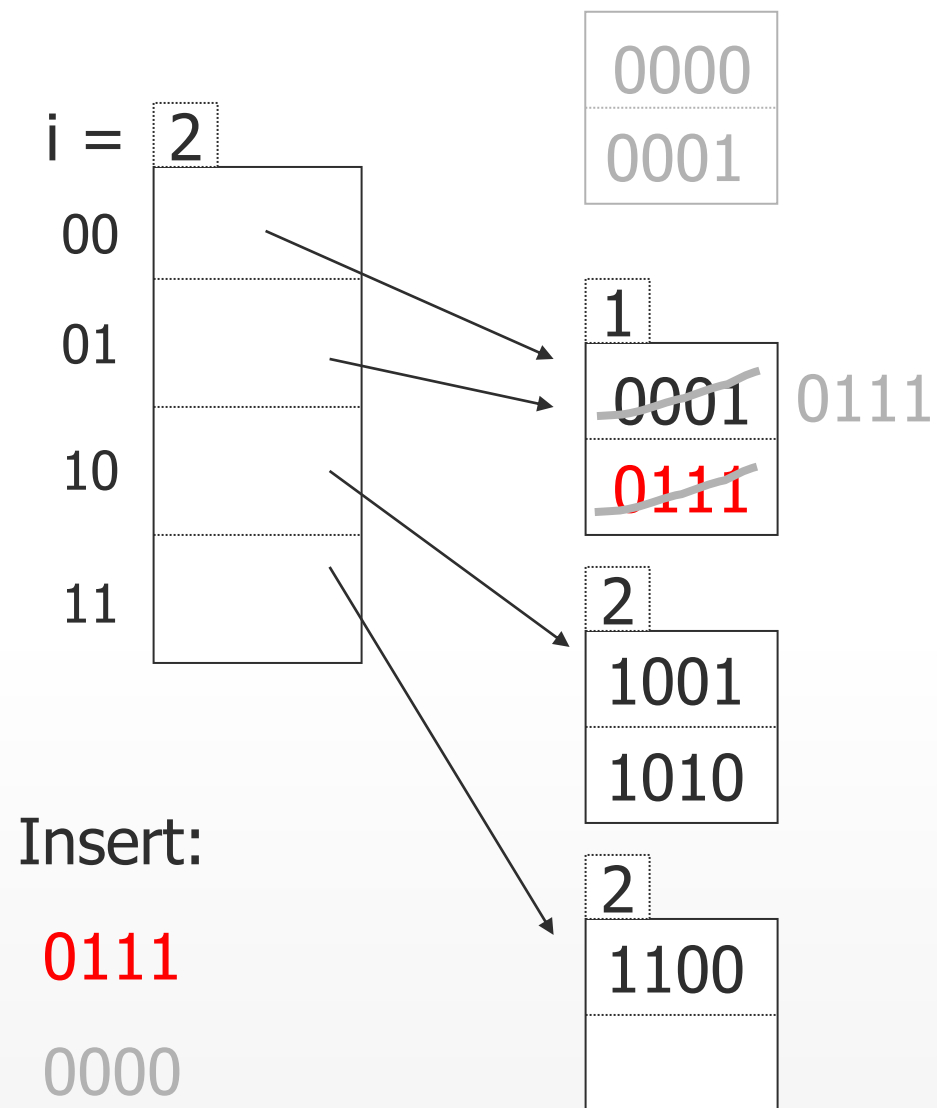
Proširivo heširanje



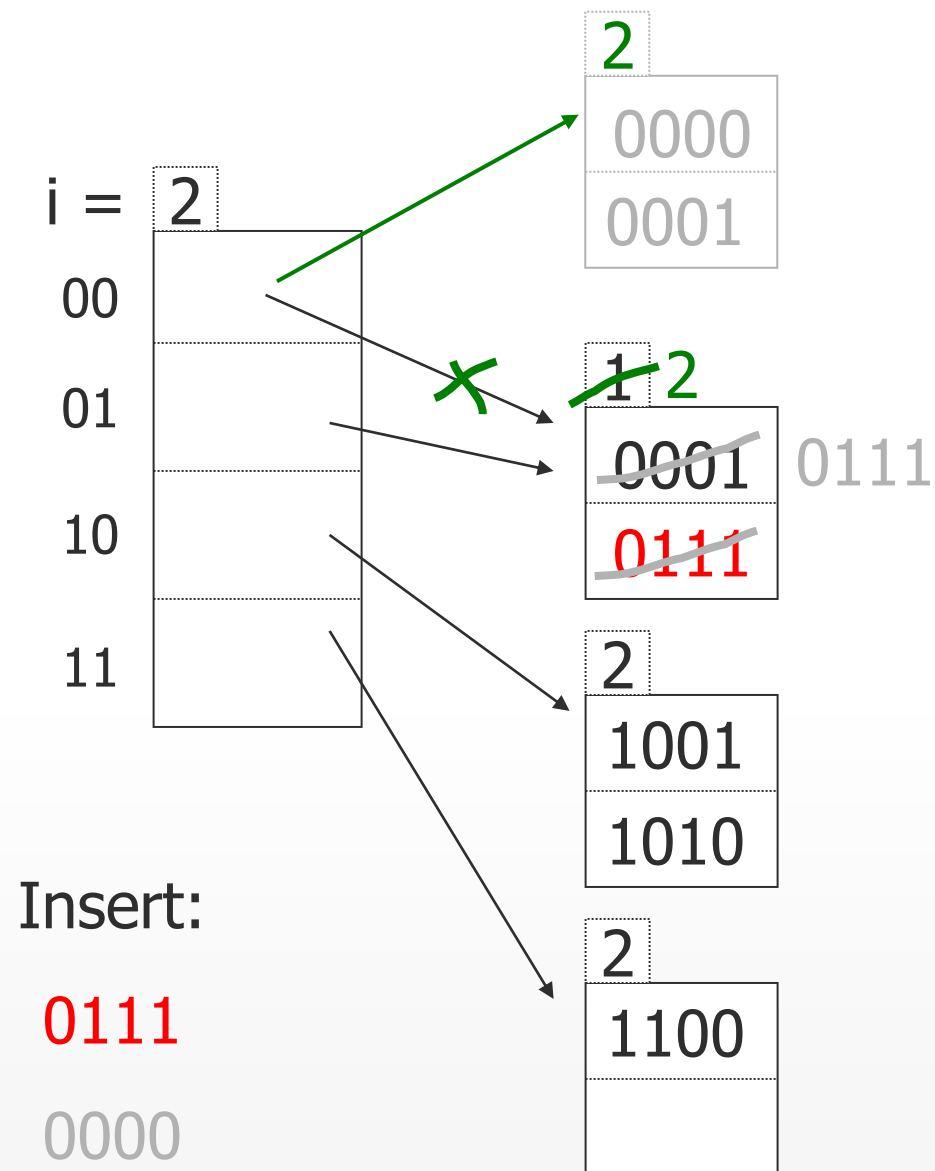
Proširivo heširanje



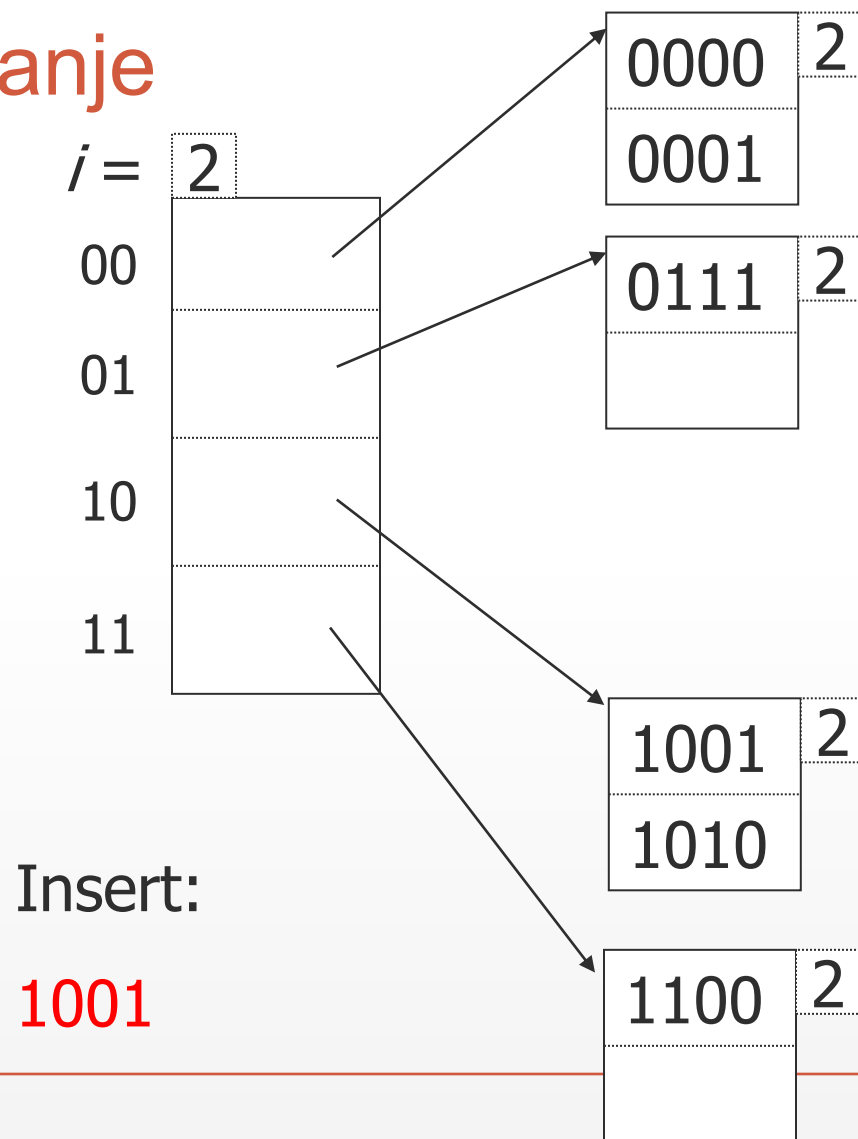
Proširivo heširanje



Proširivo heširanje



Proširivo heširanje



$i = 2$

00

01

10

11

Insert:

1001

0000

0001

2

0111

2

1001

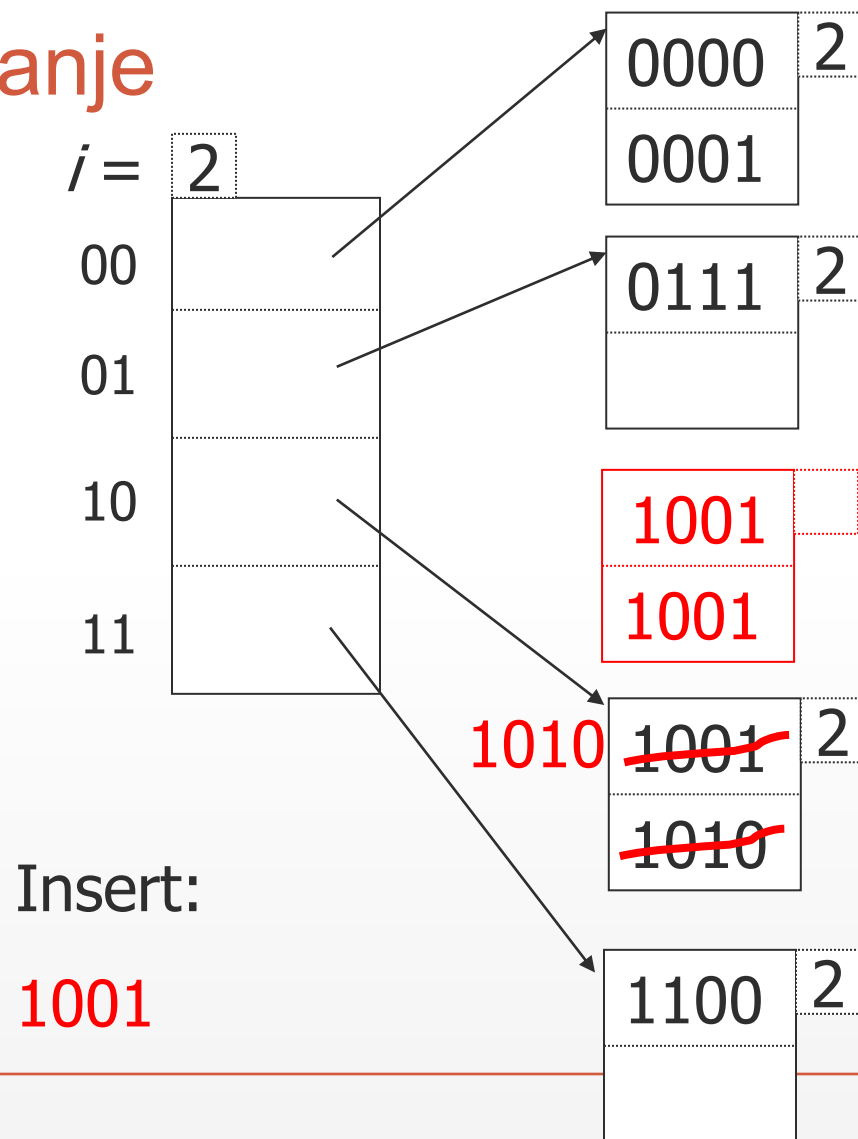
1010

2

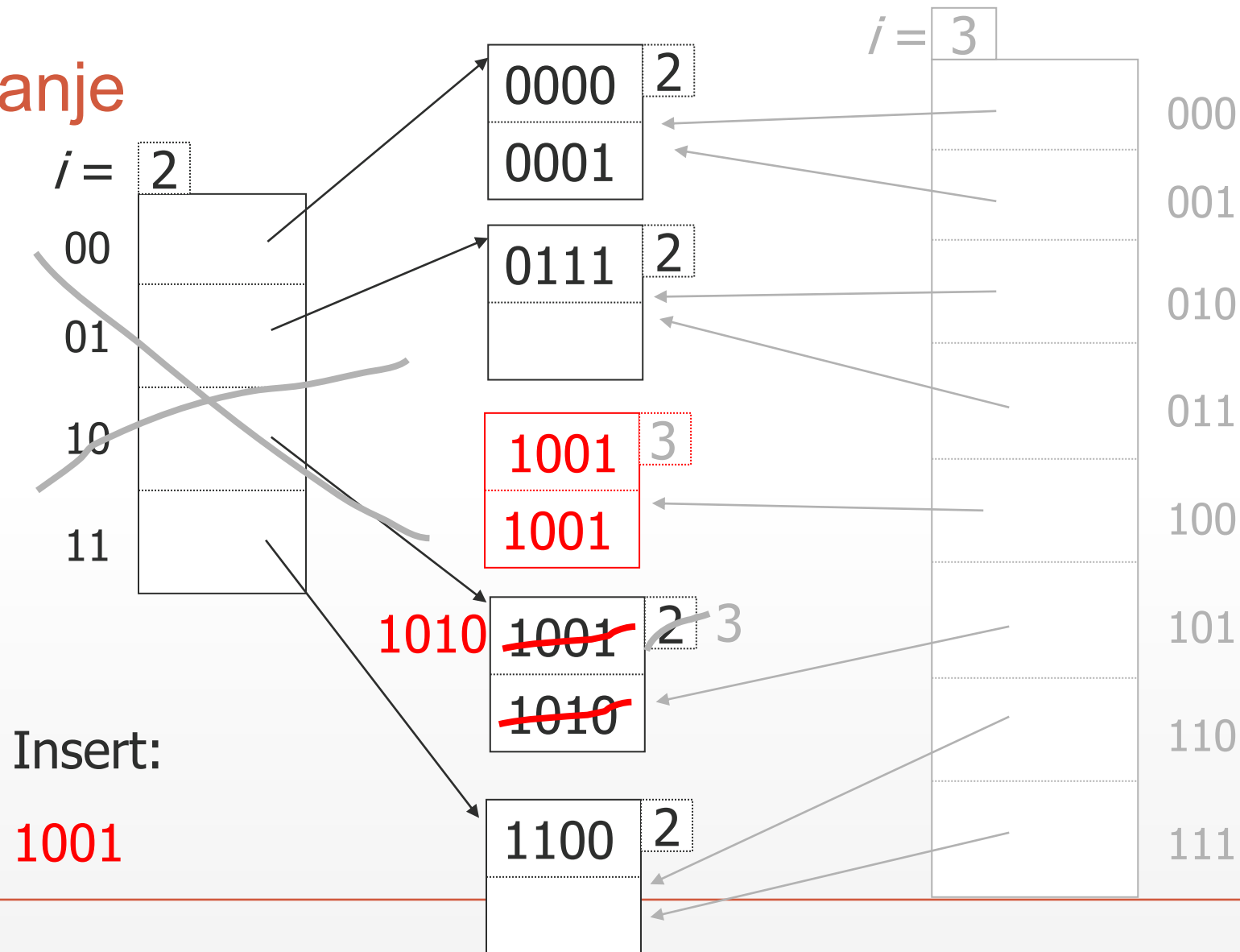
1100

2

Proširivo heširanje

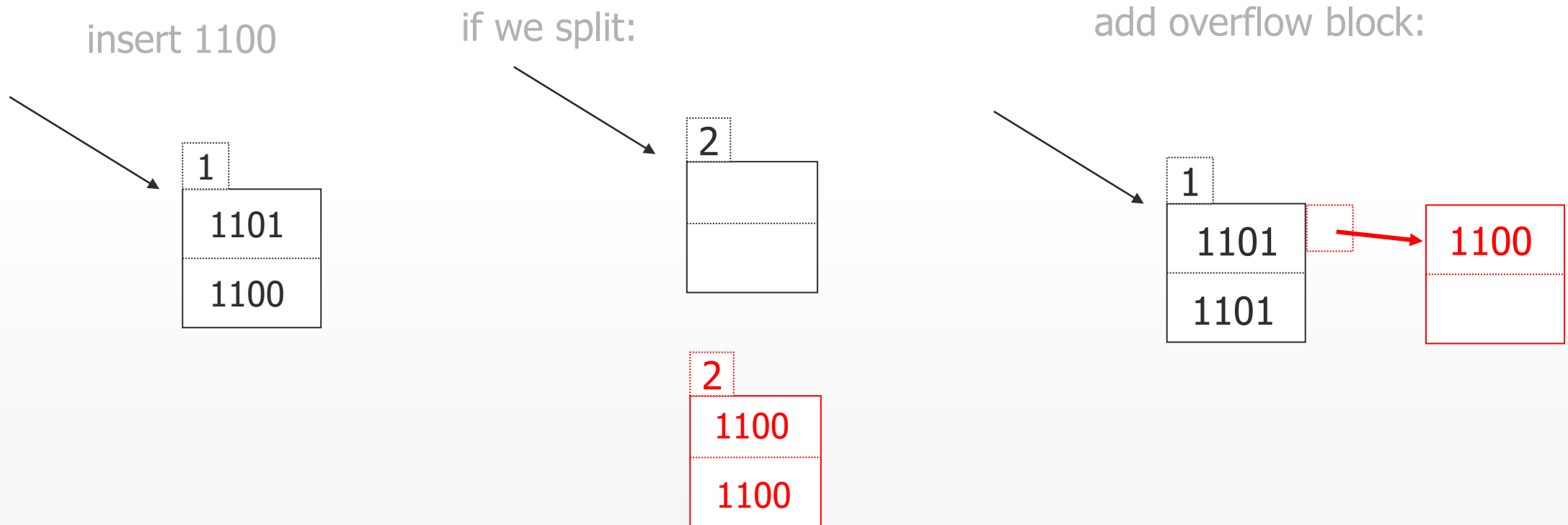


Proširivo heširanje



Proširivo heširanje

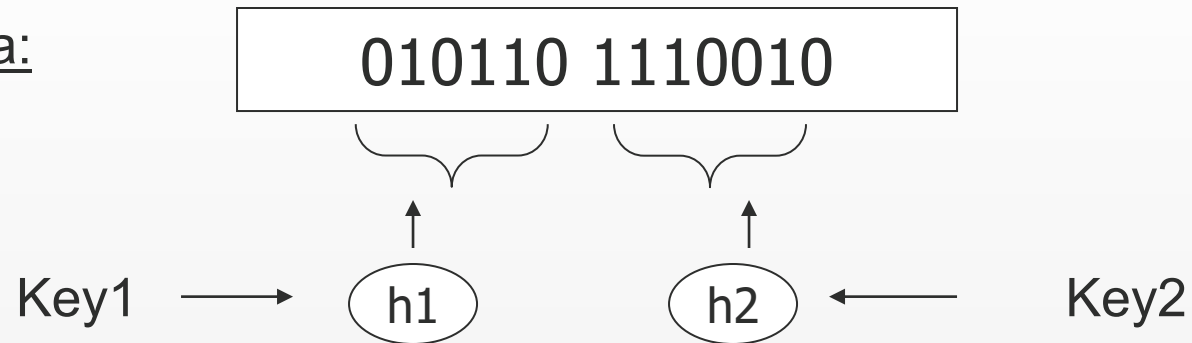
- U slučaju duplikata vrednosti ključa koristi se ulančavanje.



Podeljeno heširanje

- Definisanje heš funkcije nad više ključeva.
- Heš funkcija se primenjuje na svaki ključ posebno, pa se zatim dobijene vrednosti spajaju.
- Ovakvo heširanje ne podržava upite po opsegu vrednosti.

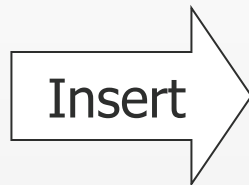
Idea:



Podeljeno heširanje

h1(toy)	=0	000
h1(sales)	=1	001
h1(art)	=1	010
.		011
h2(10k)	=01	100
h2(20k)	=11	101
h2(30k)	=01	110
h2(40k)	=00	111

<Fred>
<Joe> <Sally>



<Fred,toy,10k>,<Joe,sales,10k>,<Sally,art,30k>

Podeljeno heširanje

h1(toy)	=0	000	<Fred>
h1(sales)	=1	001	<Joe> <Jan>
h1(art)	=1	010	<Mary>
.		011	
h2(10k)	=01	100	<Sally>
h2(20k)	=11	101	
h2(30k)	=01	110	<Tom> <Bill>
h2(40k)	=00	111	<Andy>
.			

- Find Emp. with Dept. = Sales $\wedge$ Sal=40k
-

Podeljeno heširanje

h1(toy)	=0	000	<Fred>
h1(sales)	=1	001	<Joe> <Jan>
h1(art)	=1	010	<Mary>
.		011	
h2(10k)	=01	100	<Sally>
h2(20k)	=11	101	
h2(30k)	=01	110	<Tom> <Bill>
h2(40k)	=00	111	<Andy>
.			

- Find Emp. with Dept. = Sales $\wedge$ Sal=40k

Podeljeno heširanje

h1(toy)	=0	000
h1(sales)	=1	001
h1(art)	=1	010
.		011
.		
h2(10k)	=01	100
h2(20k)	=11	101
h2(30k)	=01	110
h2(40k)	=00	111
.		
.		

<Fred>
<Joe> <Jan>
<Mary>
<Sally>
<Tom> <Bill>
<Andy>

- Find Emp. with Sal=30k

look here

Podeljeno heširanje

h1(toy)	=0	000	<Fred>
h1(sales)	=1	001	<Joe> <Jan>
h1(art)	=1	010	<Mary>
.		011	
h2(10k)	=01	100	<Sally>
h2(20k)	=11	101	
h2(30k)	=01	110	<Tom> <Bill>
h2(40k)	=00	111	<Andy>
.			

- Find Emp. with Dept. = Sales
-

Podeljeno heširanje

h1(toy)	=0	000
h1(sales)	=1	001
h1(art)	=1	010
.		011
h2(10k)	=01	100
h2(20k)	=11	101
h2(30k)	=01	110
h2(40k)	=00	111
.		

<Fred>
<Joe> <Jan>
<Mary>
<Sally>
<Tom> <Bill>
<Andy>

- Find Emp. with Dept. = Sales

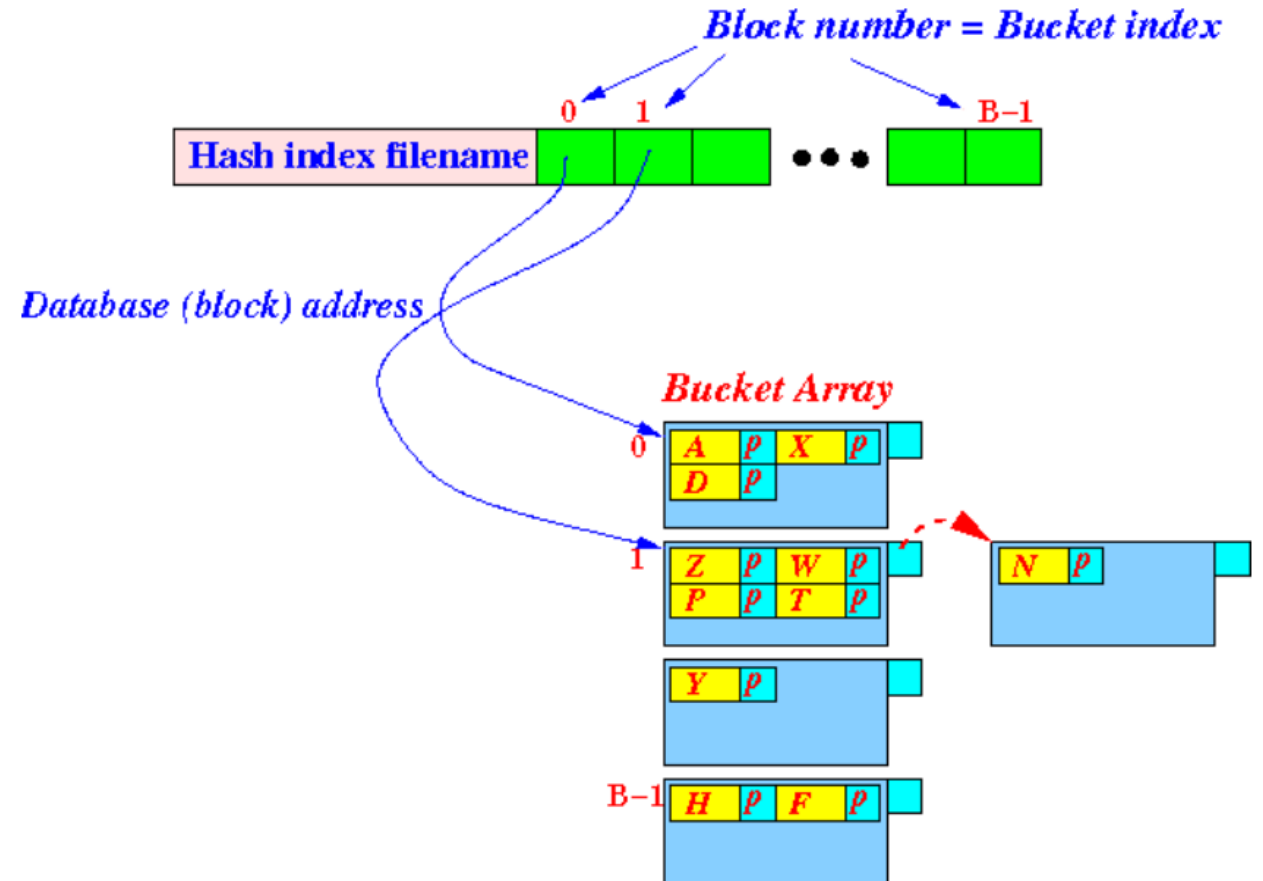
look here

Heširanje u DBMS-ovima

- U algoritmima koji implementiraju operacije relacione algebre (npr., join, selekcija, projekcija)
 - Za direktnu organizaciju fajlova – ne koristi se često
 - Dva načina alociranja:
 - Cela heš tabela u radnoj memoriji (in-memory hash indices).
 - Disk-based hashing – bucket-i (u kojima se nalaze slogovi) na sekundarnom skladištu.
-

Direktna (heš, rasejana, rasuta) organizacija fajlova

- Jedna relacija može imati samo jedan atribut direktnog pristupa, a više indeksiranih atributa.



Statičko heširanje – broj bucket-a

Prekoračenje veličine bucket-a

- Nedovoljan broj Bucket-a

$$nB > nr/fr,$$

nB – broj bucket-a

nr - ukupan broj slogova

fr – broj mesta u bucket-u

- Bucket skew

- Više slogova ima istu vrednost ključa pretrage.
- Odabrana heš funkica nema uniformnu distribuciju vrednosti ključa.

Da bi se smanjilo nepotrebno trošenje stranica, a izbeglo prekoračenje bucket-a, obično se za smeštaj slogova ostavlja oko 20% više prostora u datoteci podataka i zoni prekoračenja, nego što je to potrebno s obzirom na broj i veličinu slogova.

$$(nr/fr) * (1+d)$$

gde je d obično 0.2

Heš-bazirani indeksi

- Heš fajl organizacija se može primeniti na fajlove sa podacima, ali i na indekse.

Heš fajl organizacija

data stored in file hashed on age, no index used

Ovakva organizacija se može nazvati i klasterisanim hash indeksom

