OBLIKOVANJE BAZA PODATAKA

Predavanje 07

Blic

- https://bit.ly/3fi0aU3

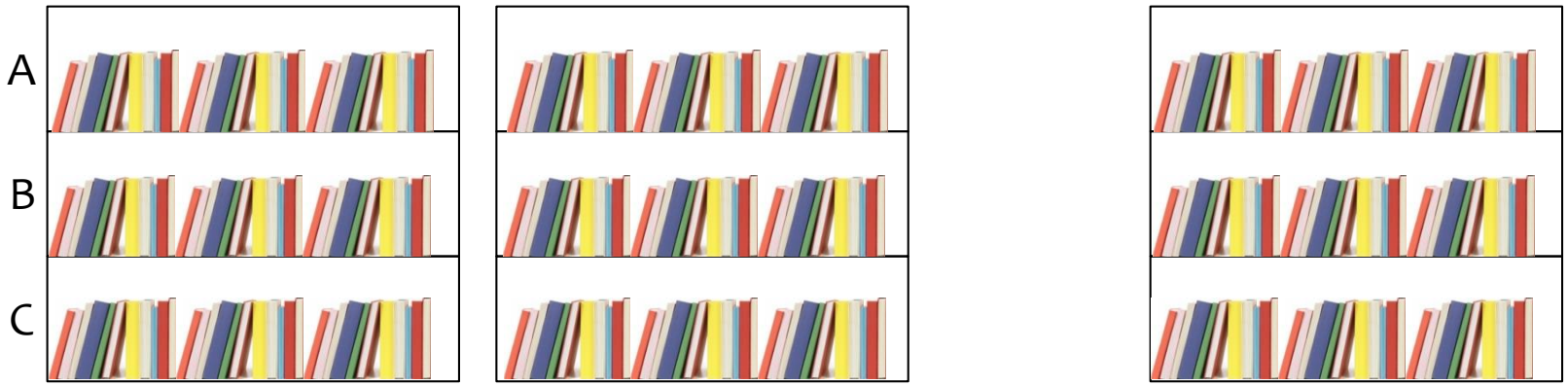
DEMO

- Primjer korisnosti indeksa

Indeksi u knjižnici

Analogija s knjižnicom (1/8)

- Imamo knjižnicu s ormarima i policama, ali nemamo računalo:



- Trenutno stanje knjižnice je da su knjige nabacane po policama ormara
  - Kažemo da imamo hrpu (engl. *heap*)

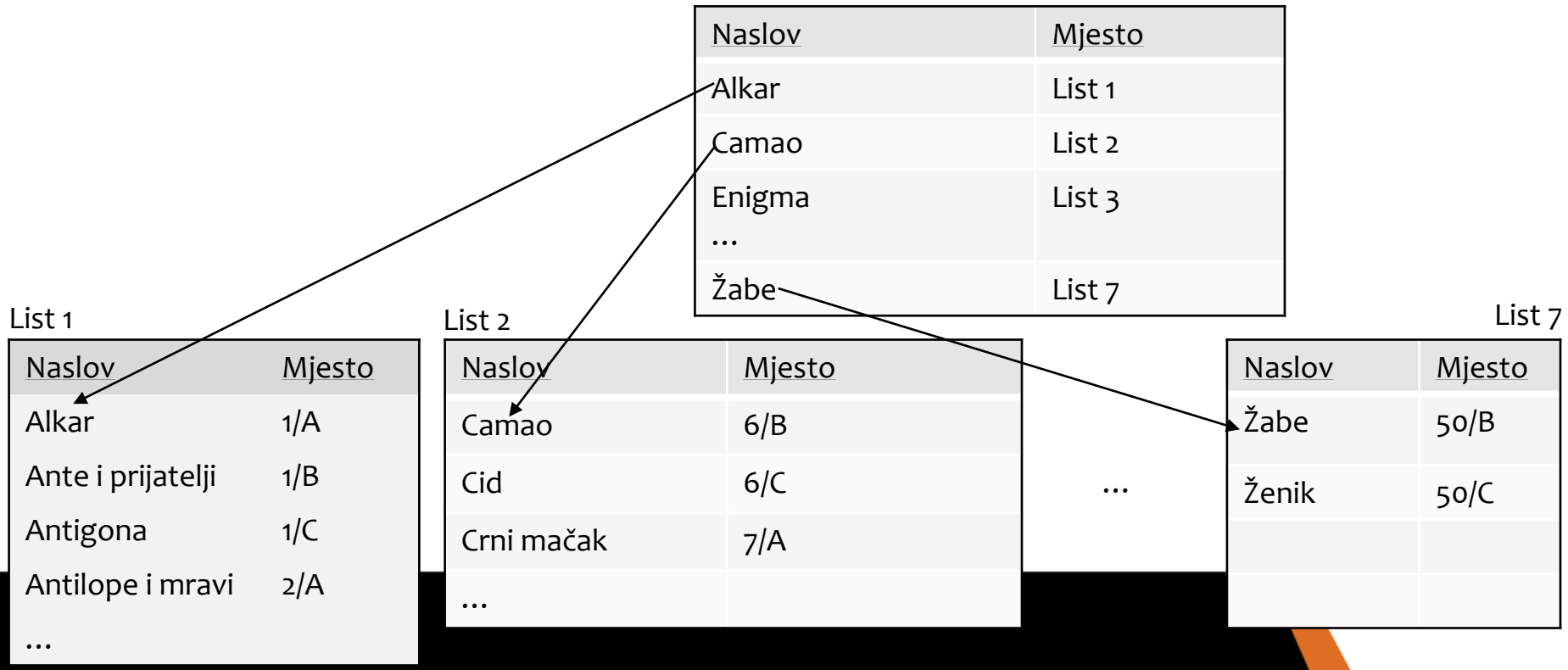
Analogija s knjižnicom (2/8)

- Situacija 1: dolazi član i veli "Molim vas knjigu Sve o bazama"
  - Jedino rješenje: krećem od A police ormara 1 i idem prema C polici ormara 50 dok je ne nađem
  - Problem: postupak dugo traje
  - Pametno rješenje: sve knjige poslažem abecedno po nazivu

- Situacija 2: dolazi član i opet veli "Molim vas knjigu Sve o bazama"
  - Jedino rješenje: implementiram binarno pretraživanje hodanjem
  - Problem: postupak i dalje traje dulje nego je zaista potrebno

Analogija s knjižnicom (3/8)

- Pametno rješenje:
  - Na listove papira ću zapisati nazive prvih knjiga za svaku policu
  - Na dodatni (ishodišni) list papira ću zapisati nazive prvih knjiga na svim ostalim listovima



Analogija s knjižnicom (4/8)

- Ovakvu strukturu zovemo **klasterirani indeks** (engl. *clustered index*)

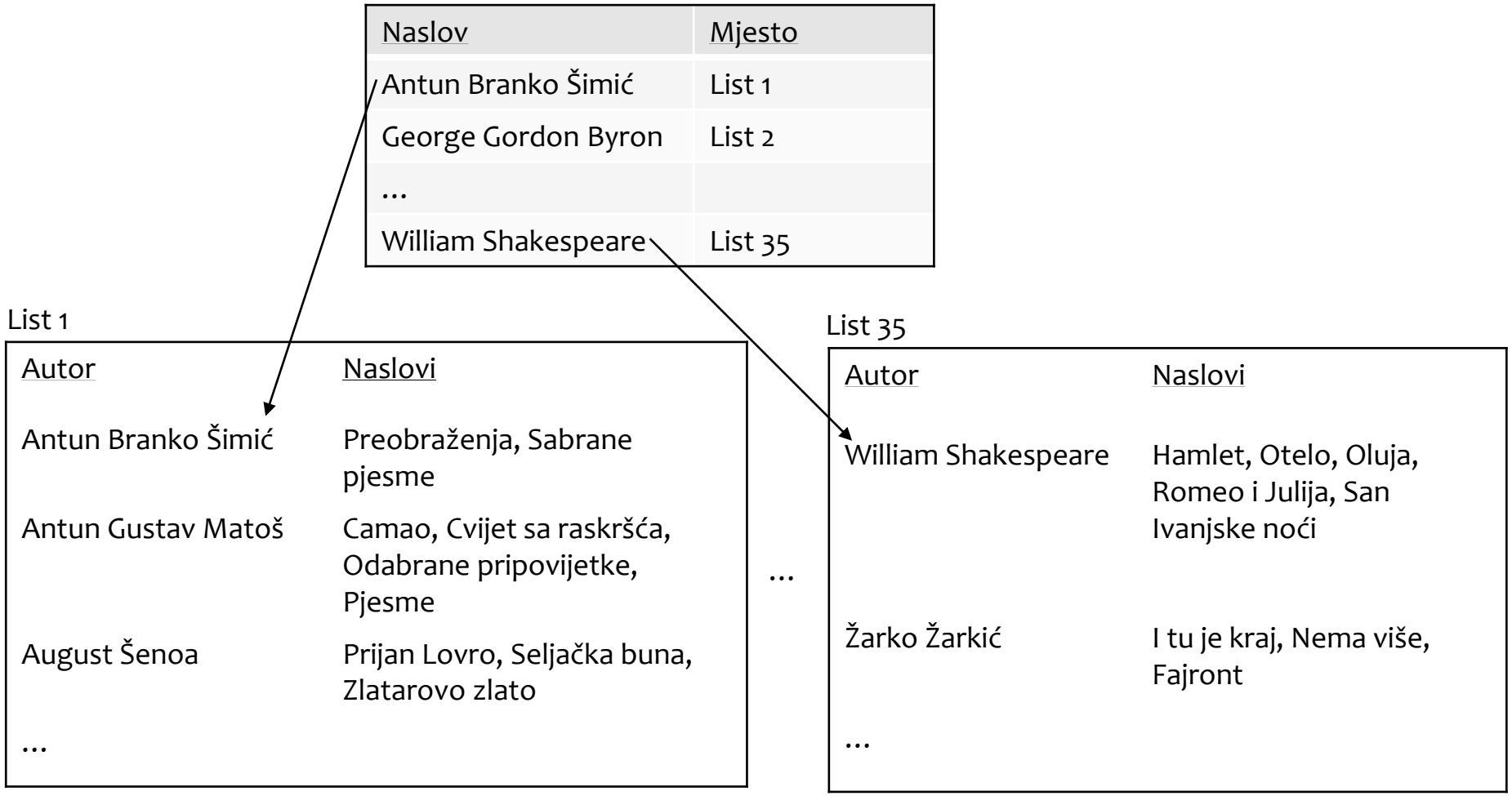
- Situacija 3: dolazi član i veli "Molim vas knjigu Cid"
  - Rješenje:
    - Uzimam ishodišni list i pronalazim da mi informacije o knjizi Cid pišu na listu 2
    - Uzimam list 2 i pronalazim da je knjiga smještena u ormaru 6 na polici C
    - Odlazim do ormara i uzimam knjigu (moram ipak pretražiti policu)
  - Brže ne može

Analogija s knjižnicom (5/8)

- Situacija 4: dolazi član i veli "Trebaju mi sve knjige koje je napisao Edgar Allan Poe"
  - Kako nam naš klasterirani indeks pomaže?
    - Baš nikako…
  - Jedino rješenje: obilazim sve ormare i police i pregledavam knjigu po knjigu

- Pametno rješenje: uzimam nove listove papira i na njih zapisujem autore i pokraj svakog knjige koje je napisao
  - Princip sličan onome za klasterirani indeks
  - Prvo ću pomoću tog novog indeksa pronaći koje sve knjige je napisao E. A. Poe, a onda ću iskoristiti klasterirani indeks kako bih pronašao jednu po jednu knjigu

Analogija s knjižnicom (6/8)



Analogija s knjižnicom (7/8)

- Ovakvu strukturu zovemo **neklasterirani indeks** (engl. *nonclustered index*)

- Situacija 5: dolazi član i veli "Trebaju mi sve knjige koje je napisao Henrik Ibsen"
  - Rješenje:
    - Uzimam ishodišni list neklasteriranog indeksa i pronalazim da mi informacije o Ibsenu pišu na listu 9
    - Uzimam list 9 i pronalazim da imam njegove knjige: Nora i Kuća lutaka
    - Sad svaku od knjiga tražim u klasteriranom indeksu po već opisanom principu
- Kako bismo pronašli knjigu po ISBN-u?

Analogija s knjižnicom (8/8)

- Situacija 6: dolazi član i veli "Želio bih saznati godine izdanja svih Kozarčevih knjiga"
  - Jedino rješenje: primijeniti opisani postupak, pronaći knjige i pročitati godine
  - Problem: dosta dugo traje
  - Bolje rješenje: u neklasterirani indeks po autorima uz svaku knjigu dopišemo i godinu
  - Sad upit mogu poslužiti direktno iz indeksa, bez odlaska do ormara
  - Ovakvi indeksi se nazivaju **neklasterirani pokrivajući indeksi** (engl. *nonclustered covering indexes*)
    - Sve potrebne informacije su dostupne u samom indeksu

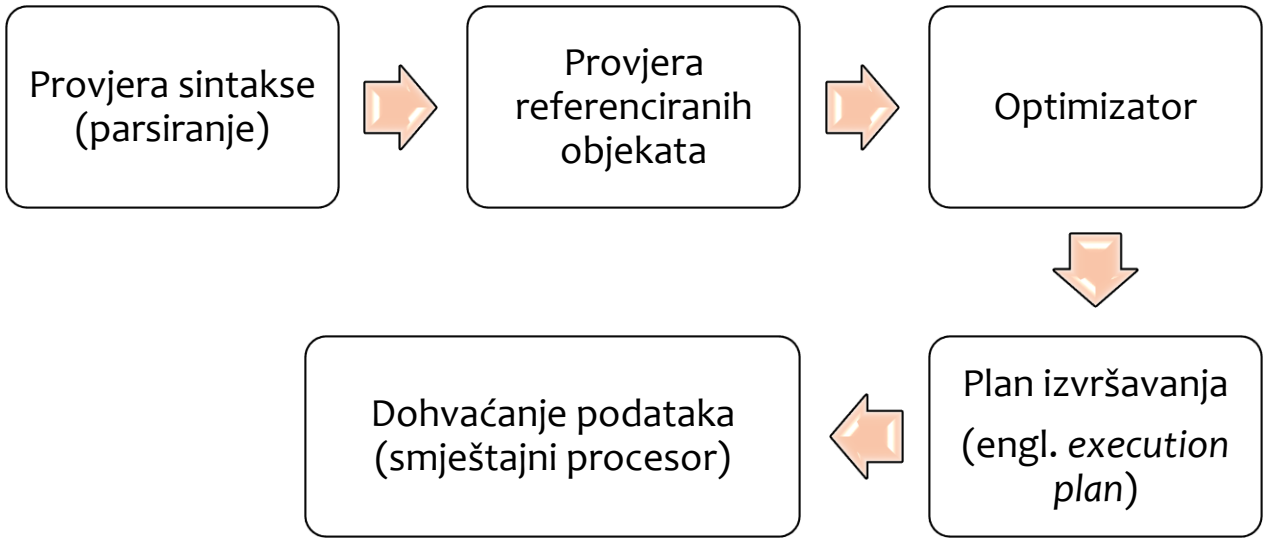
Zaključak

- Možemo zaključiti sljedeće:
  - Dobro je imati klasterirani indeks
  - U knjižnici može postojati najviše jedan klasterirani indeks
    - Knjige mogu biti poslagane samo po jednom svojstvu (npr. po nazivu)
  - Možemo imati više neklasteriranih indeksa (npr. po autoru, po ISBN-u, po godini izdanja, …)
    - Ako neklasterirani indeks u potpunosti zadovoljava upit, zovemo ga pokrivajućim
    - Kreiramo ga dodavanjem svojstava (npr. godina izdanja uz autora)
  - Upiti prema svojstvu indeksa će imati koristi od tog indeksa; ostali upiti neće

Indeksi u bazama podataka

Optimizator

- Svaki DML upit prolazi sljedeće korake:



- **Optimizator** analizira koje tablice i stupci sudjeluju u upitu
  - Za svaku tablicu odabire **indeks** koji će se koristiti za pronalazak i dohvaćanje podataka
  - Rezultat rada optimizatora je **plan izvršavanja**

Opće karakteristike indeksa

- Svaki indeks se definira na **jednom ili više stupaca tablice**
  - Na svakom stupcu može se definirati i smjer (uzlazni ili silazni)
  - Općenito, indeks je najefikasniji na stupcu koji sadržava puno različitih vrijednosti (visoko selektivan stupac)
    - Uz svaki indeks se vežu i **statistike**, objekti koji sadržavaju razdiobu vrijednosti u tablici

- Svaki indeks troši prostor na disku
  - Ovisi o vrsti indeksa, veličini tablice te o tipu podataka stupaca od kojih se sastoji

- Indeksi ubrzavaju `SELECT`, a usporavaju `INSERT`, `UPDATE` i `DELETE` operacije

Implementacija indeksa

Tipovi indeksa

- U SQL Serveru postoji više tipova indeksa:
  - **Klasterirani indeksi** (engl. *clustered indexes*)
  - **Neklasterirani indeksi** na klasteriranoj tablici (engl. *nonclustered index on clustered table*)
  - Neklasterirani indeksi na hrpi (engl. *nonclustered index on heap*)
  - XML primarni i sekundarni, *full-text*, prostorni, …
- U ovom kolegiju zanimat će nas samo prva dva tipa
- Prema organizaciji, svaka tablica je:
  - Hrpa: retci su poredani onako kako su umetani u tablicu
  - Klasterirana: retci su poredani prema klasteriranom indeksu

Implementacija indeksa

- Indeksi se najčešće implementiraju kao B-stabla

- Kod klasteriranih indeksa:
  - Svi nelisni čvorovi su indeksne stranice i sadržavaju **putokaze**
  - Lisni čvorovi su podatkovne stranice i sadržavaju **retke**

- Kod neklasteriranih indeksa:
  - Svi čvorovi su indeksne stranice i sadržavaju **putokaze**
- **Putokazi** su vrijednosti ključa indeksa

Klasterirani indeksi

- Svaka tablica može imati **samo jedan** klasterirani indeks
  - Preporuka je da ga svaka tablica ima
  - Retci na stranicama su fizički poslagani po tom indeksu
- Ključeve indeksa čine vrijednosti stupaca indeksa
- Ako drukčije nije specificirano, kreiranjem primarnog ključa se implicitno kreira i klasterirani indeks
  - Dobro za dohvat podataka po primarnom ključu
- Svi su retci složeni po klasteriranom indeksu
  - Umetanje ili izmjena retka uzrokuje njegovo smještanje ili premještanje kako bi se očuvao redoslijed
  - Za performanse dobro koristiti surogatne primarne ključeve

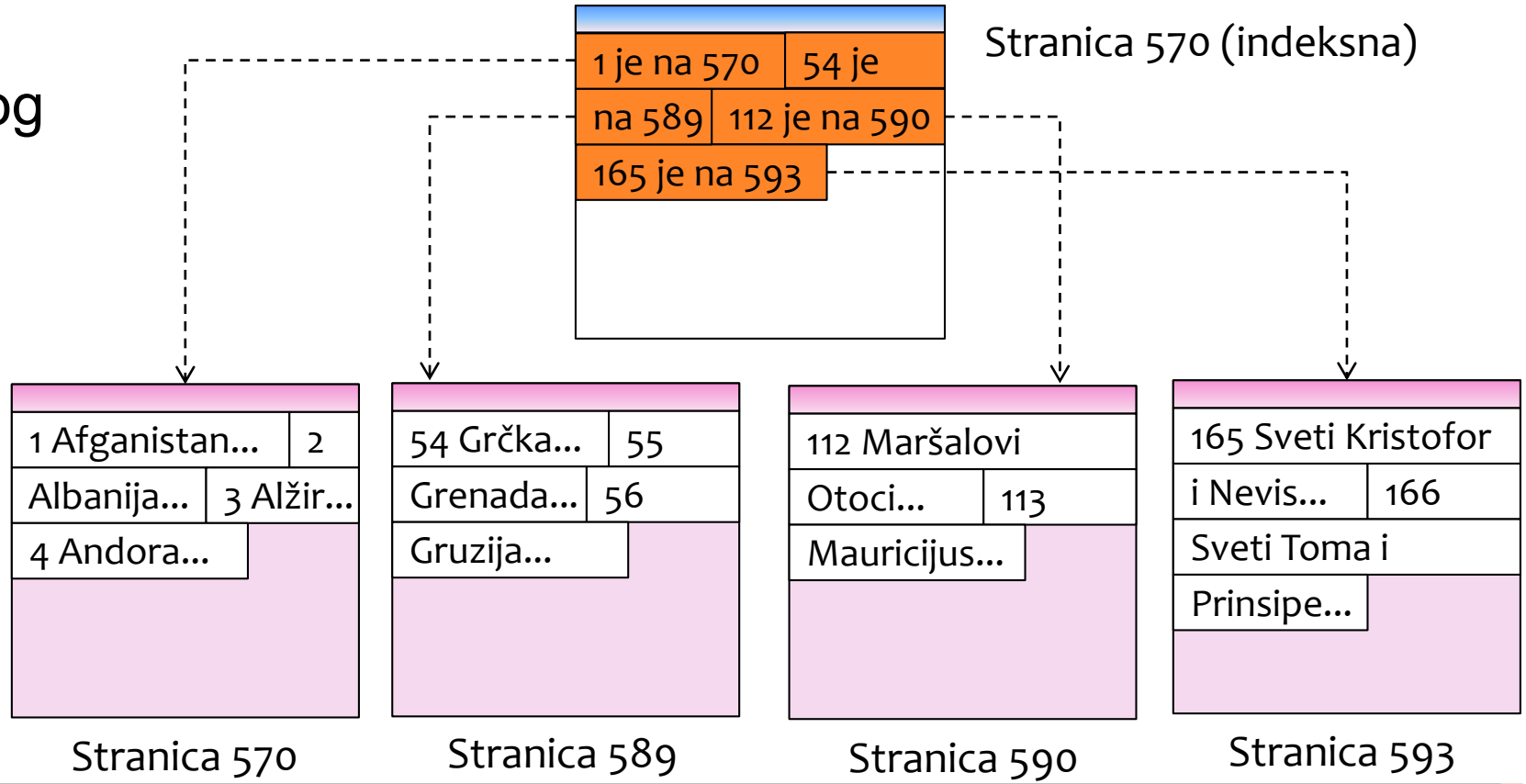
Primjer klasteriranog indeksa (1/2)

- Pokrenuti skriptu VUA23-OBP-Pred-07-Kreiranje tblDrzava.sql

- Skripta izrađuje tablicu tblDrzava bez primarnog ključa i bez klasteriranog indeksa
  - Pogledati koliko stranica ima tablica i zapisati njihove brojeve
  - Staviti primarni ključ na tablicu
  - Pogledati koliko stranica ima tablica i zapisati njihove brojeve

Primjer klasteriranog indeksa (2/2)

- Tablica tblDrzava se sastoji od 196 redaka na 4 stranice
  - Primarni ključ i ključ klasteriranog indeksa je Drzava_ID



Korištenje klasteriranog indeksa

- Primjer upita:

```
SELECT NazivHrKratki FROM tblDrzava WHERE Drzava_ID = 136
```

- Optimizator odlučuje koristiti klasterirani indeks:
  1. RDBMS pretraživanje indeksa započinje od korijena koji sadrži ključeve indeksa (vrijednosti za Drzava_ID)
  2. Pretražuje ishodišni čvor dok ne nađe stranicu na kojoj se nalazi traženi ključ (Drzava_ID = 136)
  3. Odlazi na stranicu 590
  4. Pretražuje ju dok ne dođe do retka s ključem 136
  5. Iz tog retka uzima vrijednost stupca NazivHrKratki

Neklasterirani indeksi

- Kako nam klasterirani indeks pomaže pronaći Maltu?

- Svaka tablica može imati proizvoljan broj neklasteriranih indeksa
  - Za pronalazak Malte bi dobro došao neklasterirani indeks po stupcu NazivHrKratki

- Preporuke za neklasterirane indekse:
  - Staviti po jedan na svaki strani ključ (zbog podrške spajanjima)
  - Ostale postaviti nakon analize najčešćih upita
  - Ne bi ih trebalo biti puno više od 5 po jednoj tablici

Karakteristike neklasteriranih indeksa

- Svi čvorovi sadrže putokaze (ključeve indeksa)
  - Ključ neklasteriranog indeksa čini:
    - Vrijednost stupca na kojeg je indeks postavljen
    - Ključ klasteriranog indeksa (referencira klasterirani indeks)
  - Primjerice, tblDrzava ima klasterirani indeks na Drzava_ID i neklasterirani indeks na NazivHrKratki
    - Ključevi klasteriranog indeksa su: 1, 2, 3, ...
    - Ključevi neklasteriranog indeksa su: Afganistan (Drzava_ID = 1), Albanija (Drzava_ID = 2), Alžir (Drzava_ID = 3), …

Princip rada neklasteriranih indeksa

- Princip rada neklasteriranih indeksa:
  1. Optimizator odluči koristiti indeks
  2. Kreće od korijena i prati putokaze dok se ne pronađe lisni čvor koji sadržava traženi ključ
  3. Iz traženog ključa se izdvoji ključ klasteriranog indeksa
  4. Odlazi se na klasterirani indeks po podatke

- Dobri za dohvaćanje **manjeg broja redaka** zbog posjećivanja i klasteriranog indeksa (engl. *lookup*)
  - Za dohvaćanje većeg broja redaka su **jako štetni po performanse** i optimizator ih neće izabrati

Pokrivajući neklasterirani indeksi

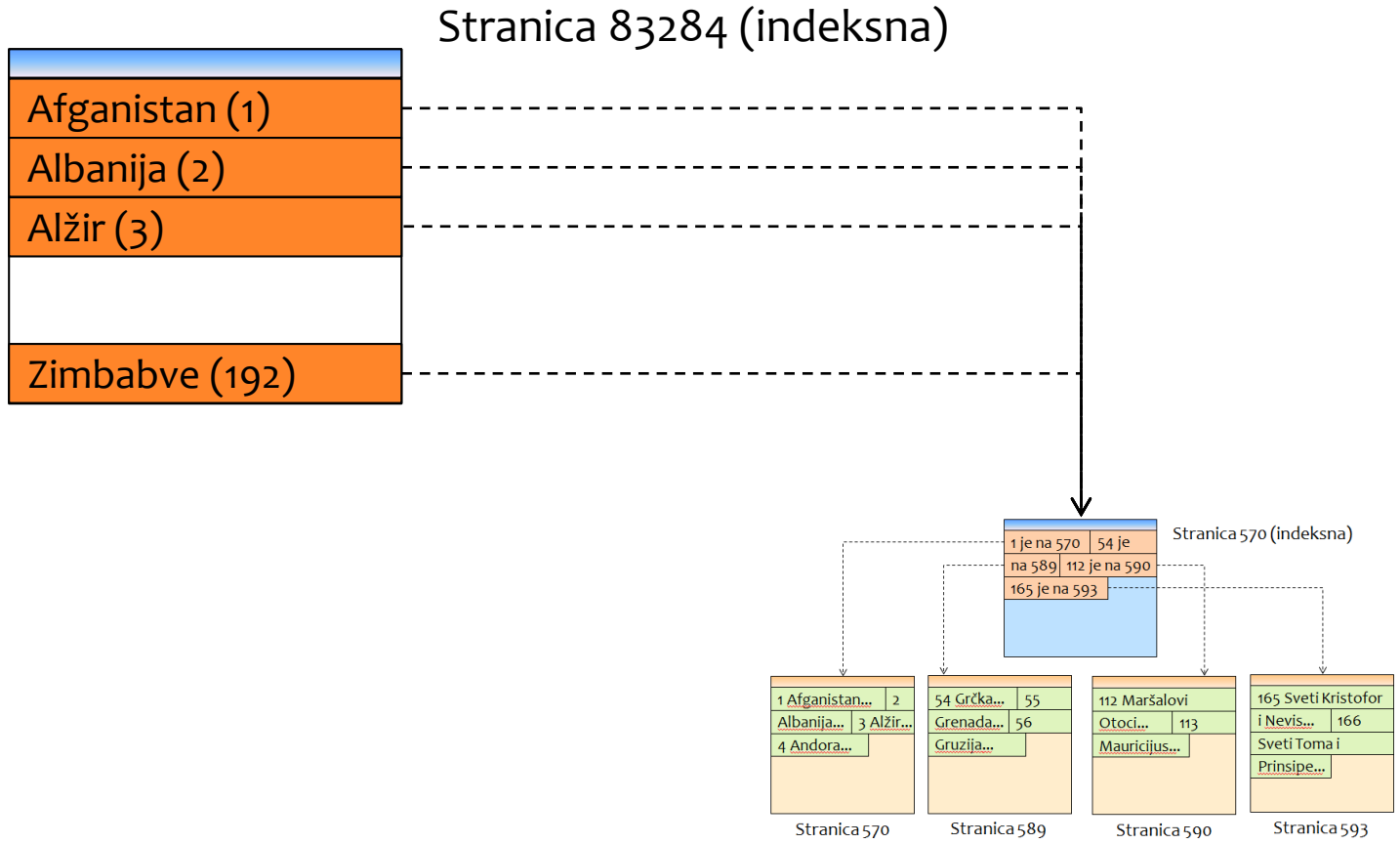
- Posebna situacija je kad ključ neklasteriranog indeksa sadrži sve tražene podatke
  - Može se preskočiti odlazak na klasterirani indeks
  - Takve indekse zovemo **pokrivajući** indeksi
  - Pokrivaju cijeli upit i dobri su za dohvat i većeg broja redaka

- RDBMS nudi mehanizam uključivanja dodatnih stupaca u ključ neklasteriranih indeksa, tako da se on sastoji od:
  - Vrijednost stupca na kojeg je indeks postavljen
  - Ključ klasteriranog indeksa
  - **Vrijednosti dodatnih stupaca koji služe pokrivanju upita**

- Nedostatak: ključ indeksa postaje veći pa manje zapisa stane na jednu stranicu

Primjer neklasteriranog indeksa

- Na tablici tblDrzava radimo neklasterirani indeks na NazivHrKratki



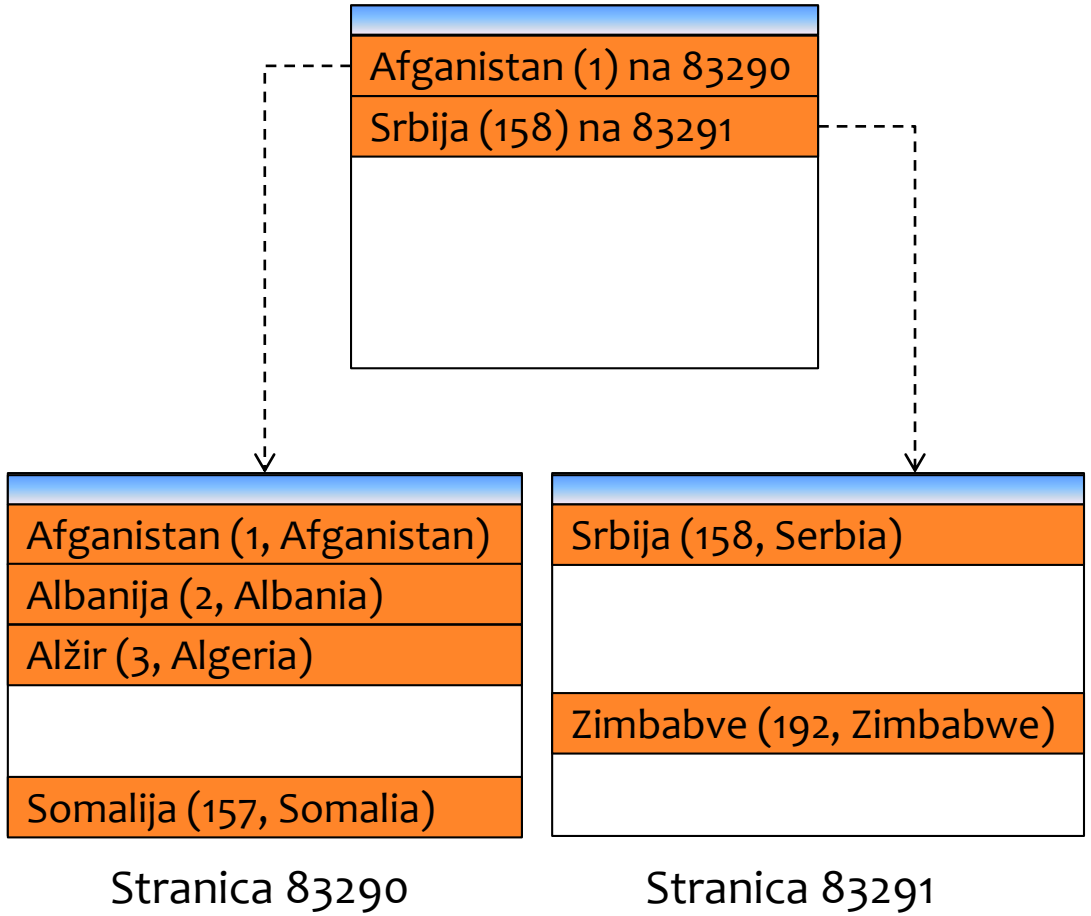
Korištenje neklasteriranog indeksa

- Primjer upita:

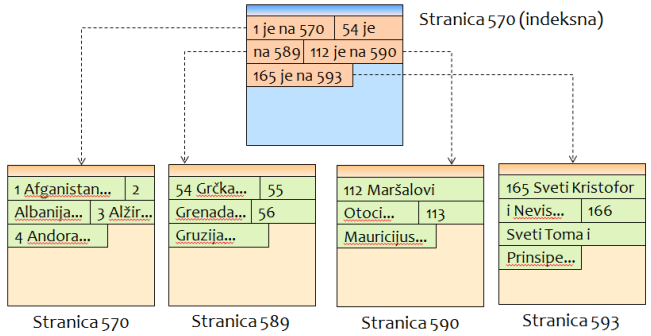
```
SELECT NazivHrKRatki, NazivEnKratki
FROM tblDrzava
WHERE NazivHrKratki = 'Kina'
```

- Optimizator odlučuje koristiti neklasterirani indeks:
  1. RDBMS pretraživanje indeksa započinje od ishodišta koje sadržava ključeve neklasteriranog indeksa
  2. Pretražuje ishodišni čvor dok ne nađe ključ "Kina (85)", ali mu nedostaje NazivEnKratki
  3. Uzima ključ klasteriranog indeksa (Drzava_ID = 85)
  4. Odlazi na ishodište klasteriranog indeksa i obavlja pretragu po ključu 85 te uzima NazivEnKratki

Primjer neklasteriranog pokrivajućeg indeksa



- Napravili smo indeks koji pokriva prethodni upit



Korištenje neklasteriranog pokrivajućeg indeksa

- Ako na novom indeksu zadamo isti upit:

```
SELECT NazivHrKRatki, NazivEnKratki
FROM tblDrzava
WHERE NazivHrKratki = 'Kina'
```

- Optimizator odlučuje koristiti novi neklasterirani indeks:
  1. RDBMS pretraživanje indeksa započinje od ishodišta
  2. Pronalazi da se na čvoru 83290 nalazi traženi ključ
  3. Odlazi na čvor 83290 i pronalazi ključ "Kina (85, China)"
  4. Pošto su svi traženi podaci pronađeni, nema potrebe za odlaskom na klasterirani indeks

T-SQL za kreiranje indeksa

- Izrada klasteriranog indeksa:
  - Implicitno, definiranjem primarnog ključa ili eksplicitno:

    ```
    CREATE CLUSTERED INDEX naziv ON tablica(stupac1, …)
    ```
- Izrada neklasteriranog indeksa:

  ```
  CREATE NONCLUSTERED INDEX naziv ON tablica(stupac1, …)
  ```
- Izrada neklasteriranog pokrivajućeg indeksa:

  ```
  CREATE NONCLUSTERED INDEX naziv ON tablica(stupac1, …) INCLUDE (stupac1, …)
  ```
- `ALTER INDEX` ne može promijeniti definiciju indeksa (iznimka)!
  - Služi za održavanje indeksa
- Uklanjanje oba tipa indeksa:

  ```
  DROP INDEX indeks ON tablica
  ```

Optimizacija upita

Optimizacija upita pomoću indeksa

- Vrlo složen postupak, mi ćemo promatrati njegov maleni dio
- Naša procedura:
  1. Uključiti `SET STATISTICS IO ON`
  2. Izvršiti upit i zabilježiti broj pretraženih stranica (engl. *logical reads*)
  3. Algoritam optimizacije:
     a. Osigurati da svaka tablica ima klasterirani indeks
     b. Identificirati sve stupce koji sudjeluju u `WHERE` dijelu upitu i jednog od njih izabrati za ključ neklasteriranog indeksa
     c. Sve ostale stupce koji su dio upita uključiti u indeks s `INCLUDE`
        i. Ne zaboraviti da neklasterirani indeks u sebi automatski sadrži vrijednost klasteriranog indeksa – ne dodavati ga ponovno
  4. Izvršiti upit i vidjeti je li došlo do poboljšanja

Primjeri

1. Optimizirajte upit: `SELECT DatumIzdavanja FROM Racun WHERE DatumIzdavanja BETWEEN '20010702' AND '20010702 23:59:59'`
   a. Koliko stranica je pregledao RDBMS?
   b. Napravite indeks koji optimizira upit
   c. Koliko sad stranica pregled RDBMS?
   d. Uklonite indeks

2. Optimizirajte upit: `SELECT IDRacun, DatumIzdavanja FROM Racun WHERE DatumIzdavanja BETWEEN '20010702' AND '20010702 23:59:59'`

Primjeri

3. Optimizirajte upit: `SELECT IDRacun, BrojRacuna, DatumIzdavanja FROM Racun WHERE DatumIzdavanja BETWEEN '20010702' AND '20010702 23:59:59'`
   a. Pomaže li nam postojeći indeks? Zašto? Kako ćemo riješiti problem?

4. Optimizirajte upit:
```
SELECT r.IDRacun, SUM(s.Kolicina) AS Kolicina,
SUM(s.UkupnaCijena) AS UkupnaCijena
FROM Racun AS r
INNER JOIN Stavka AS s ON r.IDRacun = s.RacunID
WHERE r.DatumIzdavanja BETWEEN '20040126' AND '20040126 23:59:59'
AND BrojRacuna LIKE 'SO6275%'
GROUP BY r.IDRacun
```