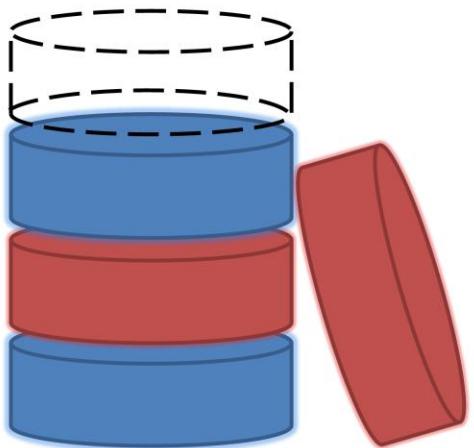


Indexarea bazelor de date



Regăsirea datelor

- Interogare folosind egalăți:
 - “*Find student name whose Age = 20*”
- Interogare folosind intervale:
 - “*Find all students with Grade > 8.50*”
- Parcurgerea secvențială a fișierului este costisitoare
- Dacă datele sunt sortate în fișier:
 - Căutare binară pentru găsirea primei înregistrări (*cost ridicat*)
 - Parcurgerea fișierului pentru găsirea celorlalte înregistrări.

Indecsi

- Indecsi sunt fișiere/structuri de date speciale utilizate pentru accelerarea execuției interogărilor.
- Un index se crează pe baza unei *chei de căutare*
 - *Cheia de căutare* e un atribut/set de attribute folosit(e) pentru a căuta înregistrările dintr-o tabelă/fișier.
 - *Cheia de căutare* este diferită de cheia primară / cheia candidat / supercheia
- Un index conține o colecție de înregistrări speciale și permite regăsirea eficientă a “*tuturor înregistrărilor tableei indexate pentru care cheia de căutare are valoarea k*”.

Characteristicile indecșilor

- *Propagarea modificărilor*
 - Adăugarea/ștergerea de înregistrări în tabela indexată implică actualizarea tuturor indecșilor definiți pentru acea tabelă.
 - Orice modificare a valorilor câmpurilor ce aparțin cheii de căutare presupune actualizarea indecșilor bazați pe acea cheie de căutare

Caracteristicile indecșilor

■ *Dimensiunea indecșilor*

- Deoarece utilizarea unui index presupune citirea acestuia în memoria internă → dimensiune indexului trebuie să fie redusă.
- Ce se întâmplă dacă indexul e (totuși) prea mare?
 - Utilizare structură de indexare parțială
 - Se indexează fișierul index
(stratificat sau pe o structură arborescentă)

- Teme importante:
 - Care este structura unei înregistrări de index?
(conținutul indexului)
 - Cum sunt organizate aceste înregistrări?
(tehnica de indexare)

Variante de structurare a conținutului unui index

- (1) înregistrările originale ce includ cheia de căutare
 - (2) $\langle k, rid \rangle$, rid – referință spre înreg. cu valoarea k a cheii de căutare
 - (3) $\langle k, \text{listă de } rid\text{-uri} \rangle$
- Alegerea variantei de stocare a intrărilor (înregistrărilor) din index este ortogonală cu tehnica de indexare.
- Exemple de tehnici de indexare: arbori B+, acces direct
 - Indexul mai conține și informații auxiliare de direcționare a căutărilor spre înregistrările căutate

Variante de structurare a conținutului unui index

■ Varianta (1):

- Indexul și înregistrările originale sunt stocate împreună
- Pentru o colecție de înregistrări se poate crea cel mult un index structurat conform variantei (1) (în caz contrar avem înregistrări duplicate → redundanță → potențiale inconistențe)
- Dacă înregistrările sunt voluminoase, numărul de blocuri în care se stochează înregistrările este mare → dimensiunea informației auxiliare din index este la rândul său mare.

Variante de structurare a conținutului unui index

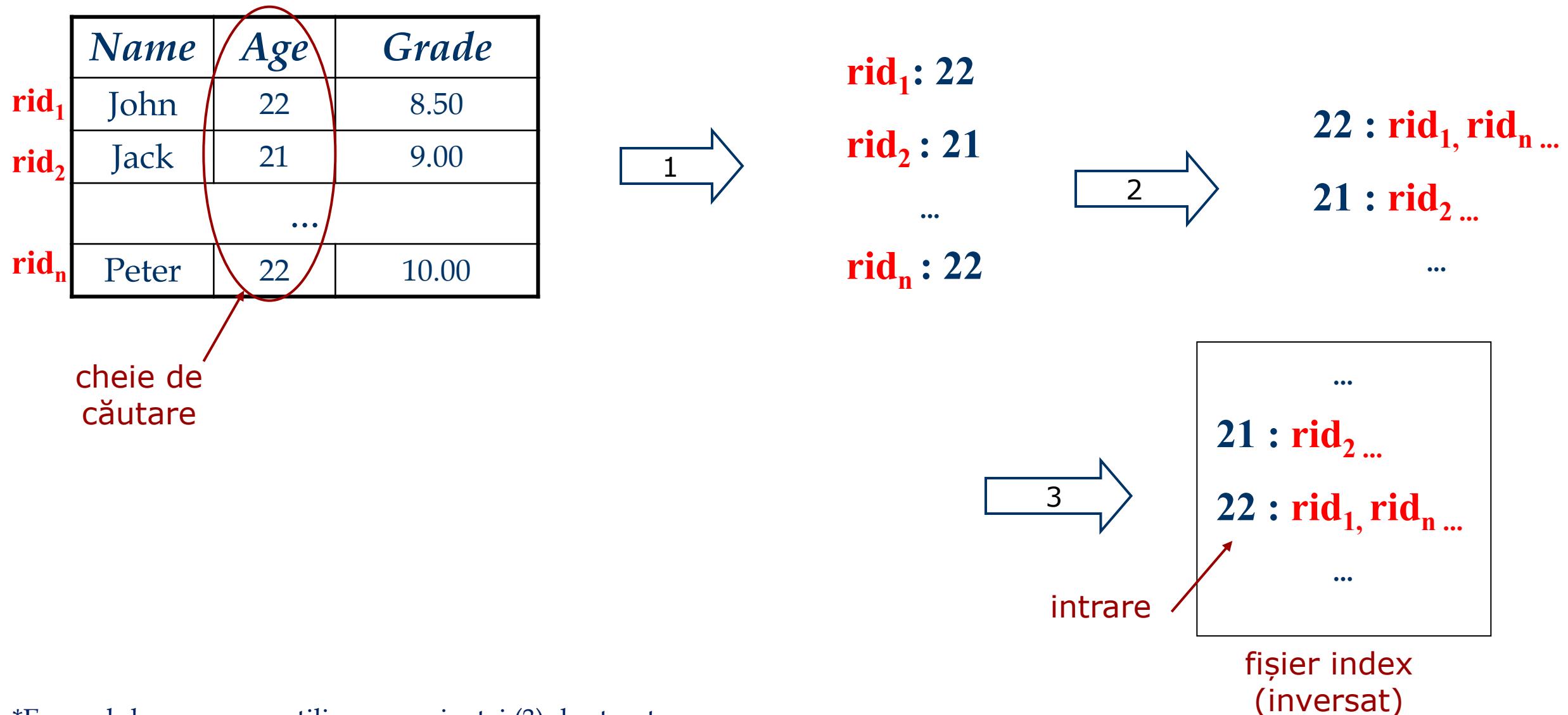
■ Variantele (2) și (3):

■ Intrările din index referă înregistrările originale

- Intrările din index sunt în general mai mici decât înregistrările (deci sunt variante mai eficiente decât varianta (1), mai ales dacă cheia de căutare este mică).
- Informația auxiliară din index folosită pentru a direcționa căutarea, este la rândul său mai redusă decât în cazul variantei (1).

■ Varianta (3) este mult mai compactă decât varianta (2), dar implică dimensiune variabilă a spațiului de stocare a intrărilor, chiar dacă cheia de căutare este de dimensiune fixă.

Crearea unui index*



*Exemplul presupune utilizarea variantei (3) de structura

Clasificarea indecsilor

- Indecși *primari* vs *secundari*:
 - Un index *primar* se formează pe baza unei chei de căutare ce include cheia primară.
 - Un index e *unic* atunci când cheia de căutare conține o cheie candidat
 - Nu vor fi intrări duplicate
 - În general, indecsii *secundari* conțin duplicate

Clasificarea indecsilor

- Indecși *clustered* (*grupat*) vs. *un-clustered* (*negrupat*):
 - Un index este *clustered* (*grupat*) dacă ordinea înregistrărilor este aceeași (sau foarte apropiată) cu ordinea intrărilor din index.
 - Varianta (1) implică grupare; de asemenea în practică, gruparea implică utilizarea primei variante de structurare a indecsilor.
 - O tabelă poate fi indexată grupat (*clustered*) pentru cel mult o cheie de căutare.
 - Costul regăsirii datelor prin intermediul unui index e influențat *decisiv* de grupare!

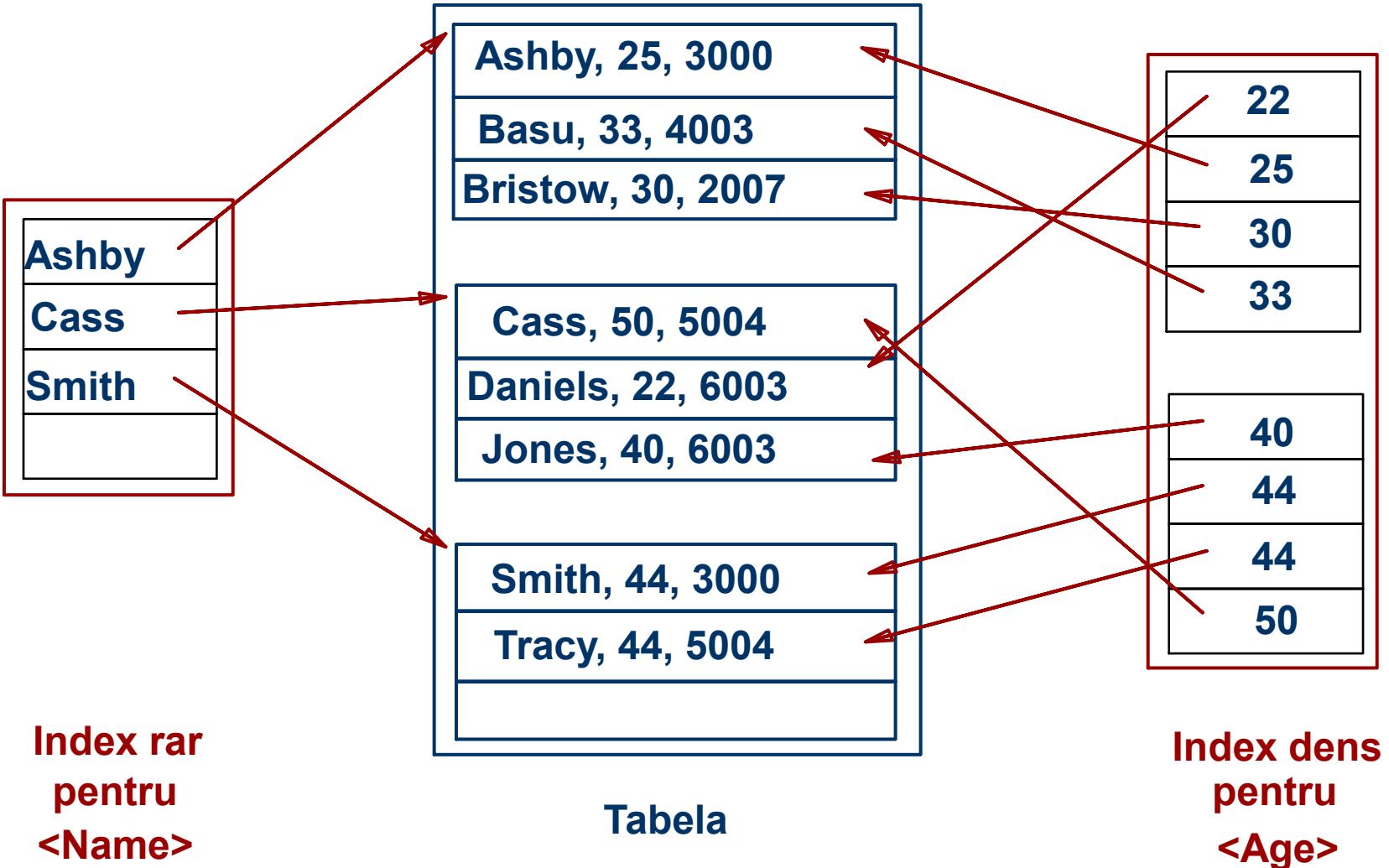
Index Clustered vs. Un-clustered

- Pentru a construi indecsi grupati:
 - Se ordoneaza inregistrarile in cadrul fisierului (*heap file*)
 - Se rezerva spatiu in fiecare pagină de memorie, pentru a se absorbi viitoarele inserari
 - dacă spatiul suplimentar e consumat, se vor forma liste inlantuite de pagini suplimentare
 - E necesara o reorganizare periodica pentru a se asigura o performanta ridicata
- Intretinerea indecsilor grupati e foarte costisitoare

Clasificarea indecsilor

- Indecși *denși* vs. *rari*:
- Un index este *dens* dacă are cel puțin o intrare pentru fiecare valoare a cheii de căutare (ce se regăsește în înregistrări)
 - Mai multe intrări pot avea aceeași valoare pentru cheia de căutare în cazul utilizării variantei (2) de stocare
 - Varianta (1) presupune implicit ca indexul e dens.
- Un index e *rar* dacă memorează o intrare pentru fiecare pagină de înregistrări
 - Toți indecșii rari sunt grupați (*clustered*)!
 - Indecșii rari sunt mai compacți.

Clasificarea indecsilor

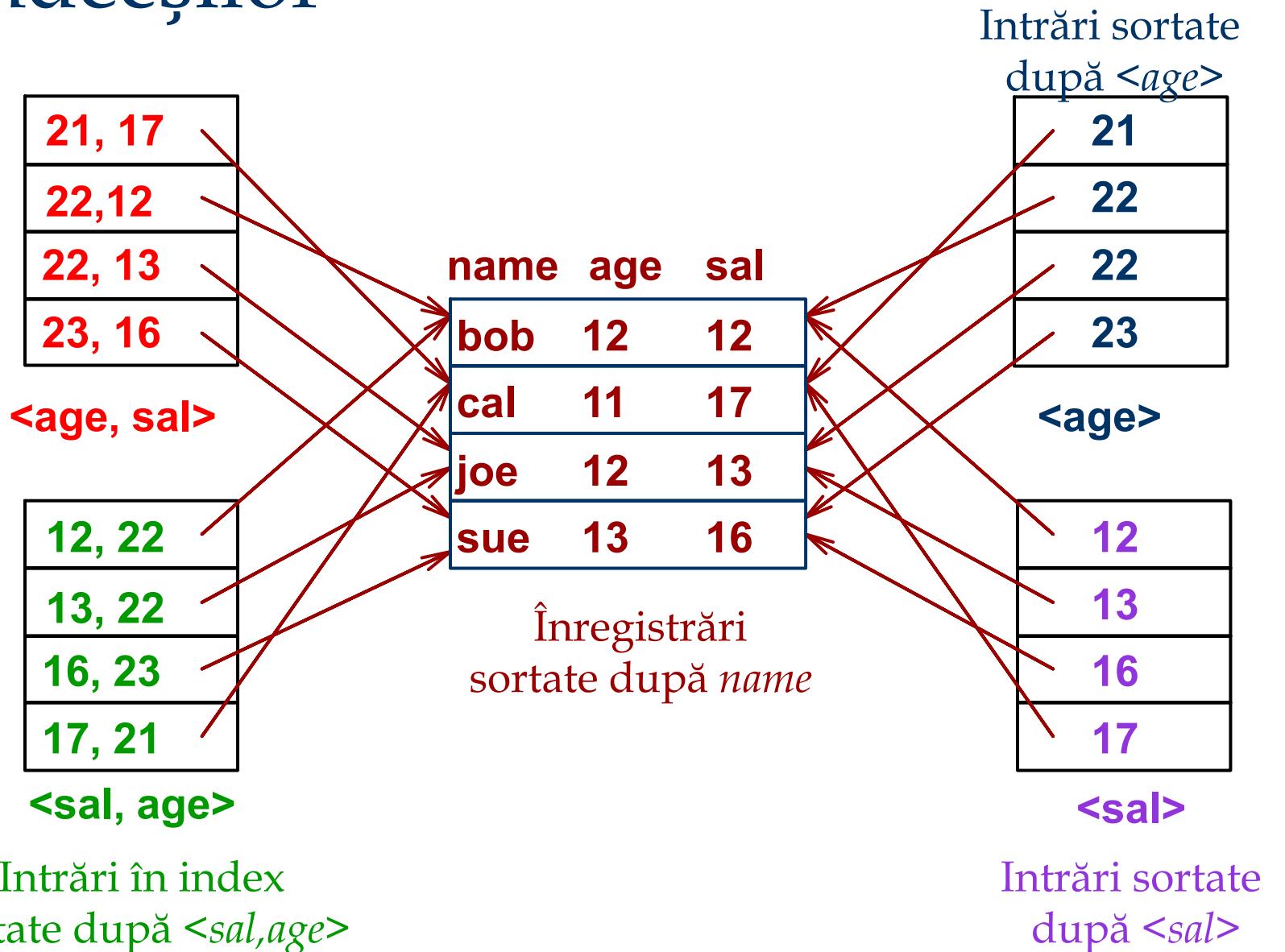


Clasificarea indecsilor

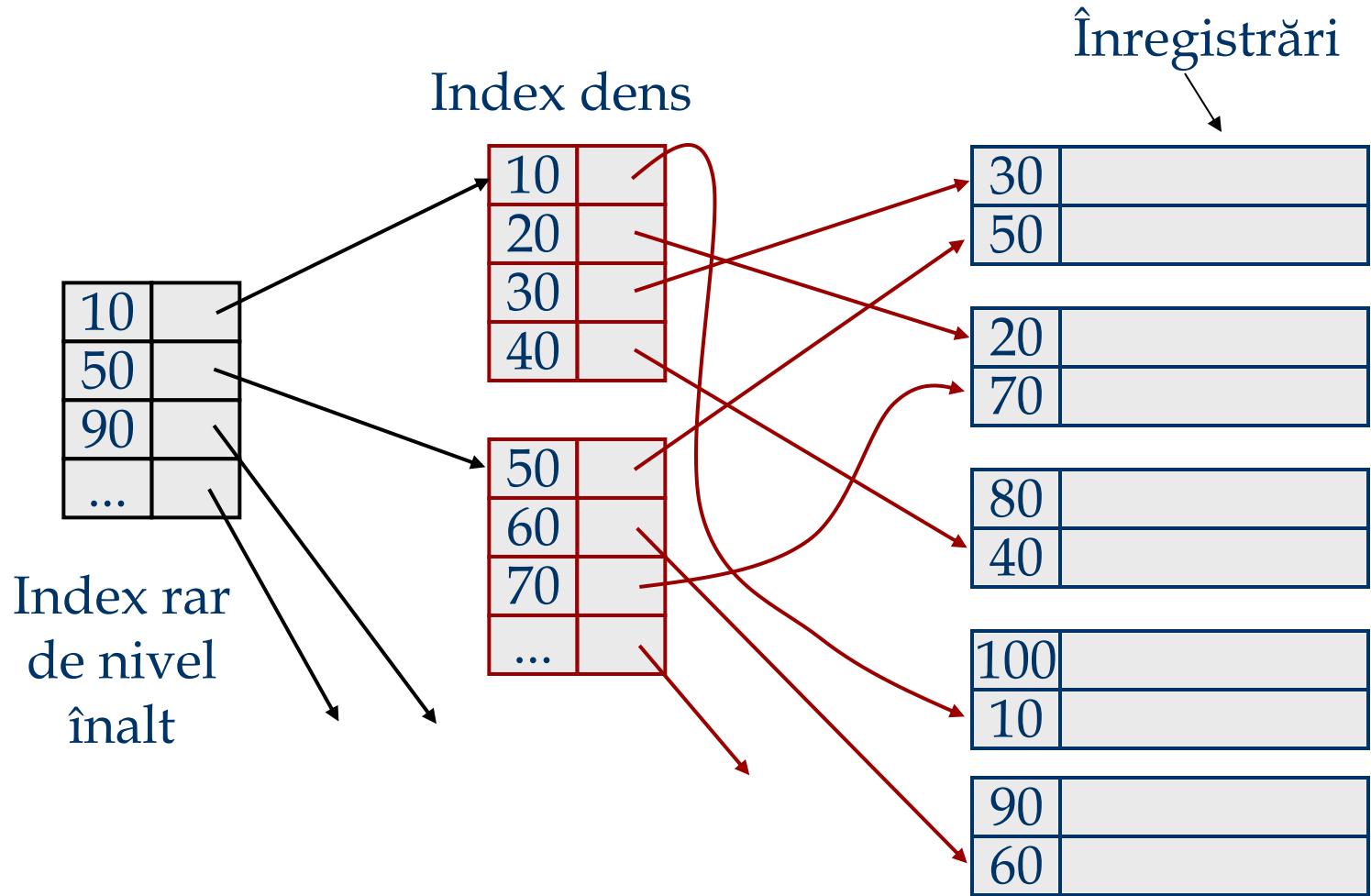
Chei de căutare compuse:

- Utile atunci când căutarea se realizează după o combinație de câmpuri
 - Interrogări cu egalitate (valoarea fiecărui câmp e egală cu o valoare constantă):
 $age=20$ și $sal = 25$
 - Interrogări cu interval (valoare unui câmp nu e o constantă):
 $age=20$ și $sal > 30$
- Intrările din index sunt sortate după cheia de căutare pentru a putea fi utile interrogărilor cu interval.
 - Ordine lexicografică, sau
 - Ordine spațială.

Clasificarea indecsilor



Indecși multinivel (indexare multiplă)



Exemplu

- Înregistrările tablei Students sunt stocate într-un fișier sortate după câmpul *Age*. Fiecare pagină poate stoca un număr maxim de 3 înregistrări

<i>ID</i>	<i>Name</i>	<i>Age</i>	<i>Grade</i>	<i>Course</i>
53831	Melanie	11	7.8	Music
53832	George	12	8.0	Music
53666	John	18	9.4	ComputerS
53688	Sam	19	9.2	Music
53650	Sam	19	9.8	ComputerS

- Determinați intrările în fiecare dintre următorii indecsi (folosiți *<page_id, slot no>* pentru identificarea unui tuplu).

Exemplu

ID	Name	Age	Grade	Course
53831	Melanie	11	7.8	Music
53832	George	12	8.0	Music
53666	John	18	9.4	ComputerS
53688	Sam	19	9.2	Music
53650	Sam	19	9.8	ComputerS

1. *Age* - Dens, varianta (1)

- întreaga tabelă

2. *Age* - Dens, varianta (2)

(11,<1,1>) (12,<1,2>)(18,<1,3>)(19,<2,1>)(19,<2,2>)

3. *Age* - Dens, varianta (3)

(11,<1,1>) (12,<1,2>)(18,<1,3>)(19,<2,1>,<2,2>)

4. *Age* - Rar, varianta (1)

- un astfel de index nu poate fi construit (definiție)

Exemplu

5. *Age* - rar, varianta (2)

(11,<1,1>) (19,<2,1>) - ordinea intrărilor e semnificativă

6. *Age* - rar, varianta (3)

(11,<1,1>) (19,<2,1>,<2,2>) - ordinea intrărilor e semnificativă

7. *Grade* - Dens, varianta (1)

7.8, 8.0, 9.2, 9.4, 9.8

8. *Grade* - Dense, varianta (2)

(7.8,<1,1>)(8.0,<1,2>)(9.2,<2,1>)(9.4,<1,3>)(9.8,<2,2>)

ID	Name	Age	Grade	Course
53831	Melanie	11	7.8	Music
53832	George	12	8.0	Music
53666	John	18	9.4	ComputerS
53688	Sam	19	9.2	Music
53650	Sam	19	9.8	ComputerS

Exemplu

ID	Name	Age	Grade	Course
53831	Melanie	11	7.8	Music
53832	George	12	8.0	Music
53666	John	18	9.4	ComputerS
53688	Sam	19	9.2	Music
53650	Sam	19	9.8	ComputerS

9. *Grade* – Dense, varianta (3)

(7.8,<1,1>)(8.0,<1,2>)(9.2,<2,1>)(9.4,<1,3>)(9.8,<2,2>)

10. *Grade* – Rar, varianta (1)

- un astfel de index nu poate fi construit (definiție)

11. *Grade* – Rar, varianta (2)

- nu e posibil (valorile cheii de căutare nu sunt ordonate)

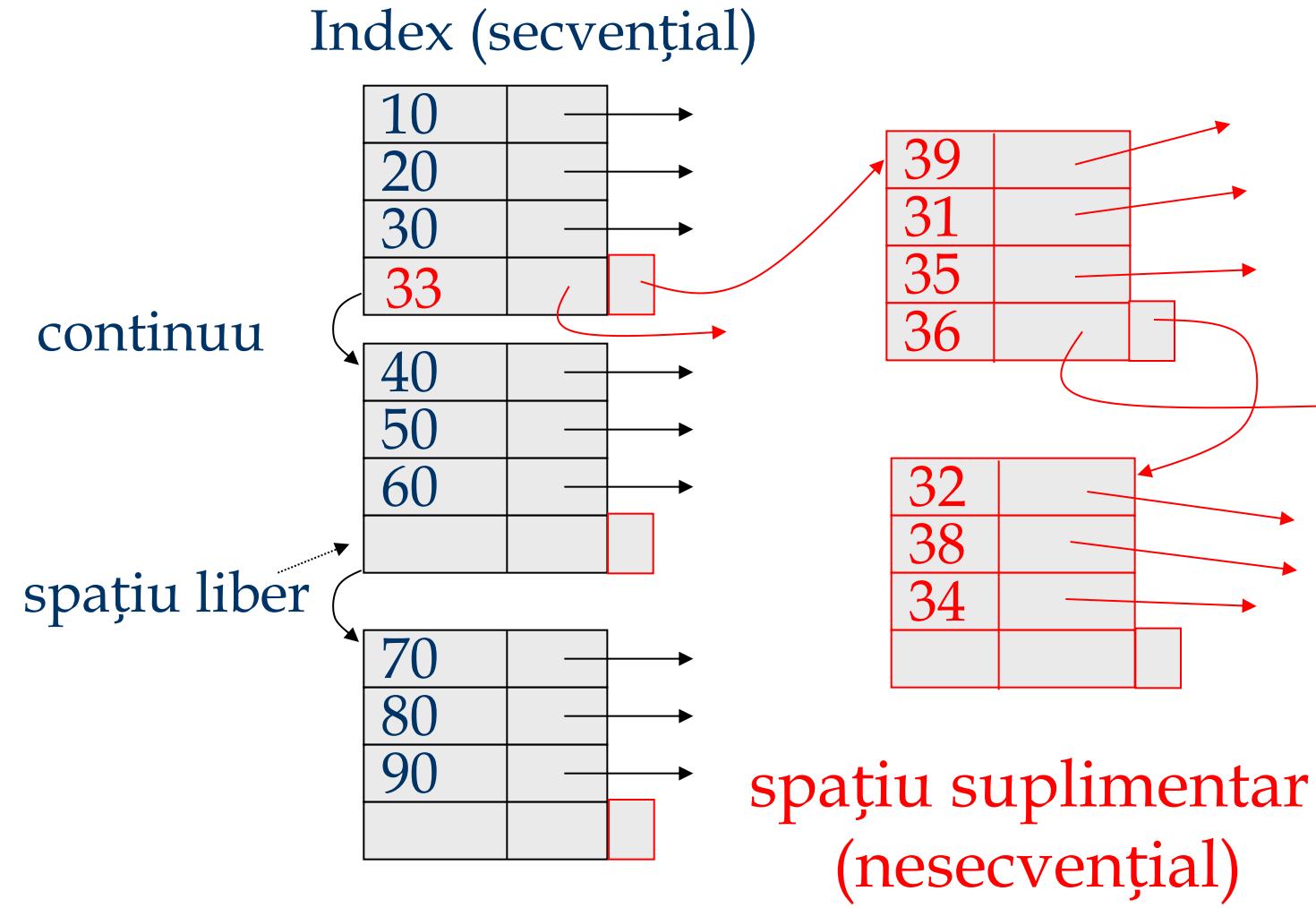
12. *Grade* – Rar, varianta (3)

- nu e posibil (valorile cheii de căutare nu sunt ordonate)

Indecși conventionali

- Avantaje:
 - Simpli
 - Indexul e un fișier secvențial → potrivit pentru parcurgeri
- Dezavantaje:
 - Inserările și/sau ștergerile sunt costisitoare
 - Se pierd secvențialitatea & echilibrul

Exemplu



Înțelegere *workload*

- Pentru fiecare interogare utilizată:
 - Care sunt relațiile/tabelele accesate?
 - Care sunt atributele/câmpurile returnate?
 - Care dintre atrbute sunt implicate în condiții de selecție/join? Cât de selective sunt aceste condiții?

- Pentru fiecare actualizare:
 - Care dintre atrbute sunt implicate în condiții de selecție/join? Cât de selective sunt aceste condiții?
 - Ce tip de actualizare este (INSERT/DELETE/UPDATE), și care sunt atrbutele afectate?

Alegerea indecșilor

- Ce index trebuie creat?
- Pentru fiecare index, care e varianta de structurare utilizată?
- **Abordare:** Se analizează cele mai importante interogări. Pentru fiecare se consideră cel mai optim plan de execuție folosind indecșii existenți. Vom crea un nou index doar dacă acesta va genera un plan mai bun.
 - Evident, acest lucru implică să înțelegem cum evaluatează un SGBD interogările și cum sunt create **planurile de execuție!**
- Înainte de a crea un nou index, trebuie să evaluăm impactul actualizării acestuia!
 - Indecșii accelerează execuția interogărilor, dar încetinesc actualizările. De asemenea, necesită spațiu suplimentar de stocare.

Alegerea indecșilor

- Atributele din clauza WHERE sunt chei de căutare “candidat”
 - Egalitățile sugerează o structură cu acces direct
 - Intervalele sugerează utilizarea unei structuri arborescente
 - Gruparea e foarte utilă în aceste situații; dar și în cazul egalităților atunci când numărul de duplicate este mare.
- Atunci când clauza WHERE are mai multe condiții se pot lua în considerare chei de căutare compuse
 - Ordinea atributelor e importantă pentru a evalua condițiile cu intervale
 - Pentru interogările importante acești indecși pot determina strategii de execuție **index-only**
- Vor fi aleși indecșii de care beneficiază cele mai multe interogări posibile. Deoarece se poate defini un singur index grupat pe o tabelă, acesta va fi ales astfel încât să beneficieze de acest index o interogare foarte importantă

Chei de căutare compuse

- Pentru returnarea înregistrărilor din *Employees* cu $age=30$ AND $sal=4000$, un index pe $\langle age, sal \rangle$ e mai potrivit decât un index pe age sau unul pe sal .
- Dacă avem: $20 < age < 30$ AND $3000 < sal < 5000$:
 - Un index arborescent grupat pe $\langle age, sal \rangle$ sau $\langle sal, age \rangle$ e foarte bun.
- În cazul: $age=30$ AND $3000 < sal < 5000$:
 - Index grupat $\langle age, sal \rangle$ e mai bun decât index pe $\langle sal, age \rangle$
- Indeksii compusi sunt mai voluminoși și sunt actualizați mai des