بسم الله الرحمن الرحیم

فصل اول

مقدمه:

در هر کامپیوتر دو محیط کلی وجود دارد:
۱ـ محیط درون ماشینی
۲ـ محیط برون ماشینی

۱ـ محیط درون ماشینی: از قطعات داخلی کامپیوتر تشکیل شده است و شامل: CPU – RAM - ROM - Register (ثبات) ... می‌باشد.

۲ـ محیط برون ماشینی: از دستگاه‌های جانبی متصل به کامپیوتر تشکیل شده است، که ***برخی برای ذخیره‌سازی*** اطلاعات استفاده می‌شوند. مانند: Disk drive – CD drive - Tape drive - ... و ***برخی دیگر برای مرتبط کردن*** محیط برون ماشینی و درون ماشینی استفاده می‌شوند مانند: keyboard (صفحه کلید)ـ network card (کارت شبکه) ـ sound card (کارت صدا) ـ ....

نکته: در بحث ذخیره و بازیابی اطلاعات، ابتدا با مفهوم عام حافظه آشنا می‌شویم سپس به بررسی سیستم ذخیره‌سازی و نحوه پیاده‌سازی فایل‌ها در محیط ذخیره‌سازی برون ماشینی (دیسک ـ نوار ـ CD ـ ...) می‌پردازیم.

یادآوری: حافظه در معنای عام هم شامل حافظه درون ماشینی (Register - ROM – RAM...) و هم شامل حافظه برون ماشینی (... - CD - Disk) می‌شود.

اهداف اصلی سیستم‌های ذخیره و بازیابی:

۱ـ صرفه‌جوئی در حافظه

۲ـ بالابردن سرعت عملیات

۳ـ بالا بردن سرعت دسترسی به اطلاعات

حافظه:

هر دستگاهی که قادر به ذخیره‌سازی و نگهداری اطلاعات باشد، به طوریکه استفاده کننده از آن بتواند در هر لحظه به آن اطلاعات ***دستیابی*** (Access) داشته باشد ***حافظه*** نامیده می‌شود.

نکته۱: در هر کامپیوتر با توجه به دو محیط برون ماشینی و درون ماشینی، انواع حافظه به دو قسمت تقسیم می‌شود:

۱ـ حافظه درون ماشینی (اصلی ـ اولیه): (Primary)

این نوع حافظه توسط CPU (پردازنده مرکزی) ***جهت اجرای برنامه‌ها*** مستقیماً مورد استفاده قرار می‌گیرد. مانند: Register – RAM - ...

۲ـ حافظه برون ماشینی (جانبی ـ ثانویه): (Secondary)

این نوع حافظه بیشتر برای ***حفظ و ذخیره‌سازی اطلاعات*** مورد استفاده قرار می‌گیرد، مانند: دیسک مغناطیسی ـ نوار مغناطیسی ـ دیسک نوری ـ ...

* در درس ذخیره و بازیابی موضوع اصلی حافظه برون ماشینی و بررسی ساختار پیاده‌سازی اطلاعات فایل‌ها روی آن حافظه است.

خصوصیات مشترک انواع حافظه:

هر حافظه دارای خصوصیات مشخصی می‌باشد، اما بین تمام حافظه‌ها خصوصیات مشترکی وجود دارد که می‌توان به موارد زیر اشاره کرد:

۱) نوشتن و خواندن

۲) نشانی‌پذیری (آدرس دهی) (Address ability)

۳) دستیابی‌پذیری (قابلیت دستیابی) (Access ability)

۴) ظرفیت (Capacity)

۵) زمان دستیابی (Access time)

۶) نرخ انتقال (Transfer rate)

خصوصیات دیگری نیز مانند: قابلیت جابجائی‌پذیر بودن ـ غیر فرار (مانا) یا فرار (نامانا) بودن ـ .... وجود دارد که ممکن است بعضی حافظه‌ها آن را داشته باشند و بعضی دیگر نداشته باشند.

۱ـ خواندن و نوشتن: هر حافظه‌ای این قابلیت را دارد که بتوان در آن نوشت (حداقل یک بار) و یا از آن اطلاعات را خواند (واکشی (Fetch)).

۲ـ نشانی‌پذیری (آدرس دهی): هر حافظه دارای یک سیستم آدرس دهی برای اطلاعاتی است که داخل خود نگهداری می‌کند و از طریق آن سیستم می‌توان به اطلاعات دسترسی پیدا کرد. مانند:

الف ـ حافظه RAM: آرایه‌ای ازبایت‌ها که هر کدام از بایت‌ها یک آدرس یکتا دارد که از 0 شروع می‌شود.

ب ـ دیسک نوری (CD): که درآن هر سکتور با سه عدد (سکتور: ثانیه: دقیقه) مشخص می‌شود.

ج ـ دیسک مغناطیسی: که در آن با سه عدد (شماره هد ـ شماره سیلندر ـ شماره سکتور) آدرس دهی انجام می‌شود.

۳ـ دستیابی‌پذیری: هر حافظه‌ای طریق مکانیسم آدرس دهی قابل دستیابی است.

۴ـ ظرفیت: هر حافظه برای نگهداری اطلاعات داخل خود اندازه یا ظرفیت مشخصی دارد که واحد آن معمولاً بیت (bit) یا بایت (byte) است.

۵ ـ زمان دستیابی (Access time) (مهم)

زمان لازم از لحظه‌ای که دستور خواندن / نوشتن صادر می‌شود تا لحظه‌ای که حافظه مورد نظر (داده مورد نظر) مورد دستیابی قرار گیرد.

به طور مثال: زمان دستیابی RAM حدود 120 نانوثانیه است در حالیکه زمان دستیابی Disk حدود 30 میلی‌ثانیه است. یعنی RAM حدود 30 برابر سریعتر از دیسک عمل می‌کند.

نکته مهم: هر چه زمان دستیابی(Access time) کمتر باشد، سرعت حافظه بیشتر خواهد بود و بالعکس.

Access time ↙	↔	↗ سرعت حافظه
(زمان دستیابی)		

۶ـ نرخ انتقال (Transfer rate): مقدار داده‌ای که در واحد زمان قابل انتقال است را نرخ انتقال می‌گوئیم و معمولاً واحد آن بایت در ثانیه (bps) است. هرچه سرعت حافظه بیشتر باشد، نرخ انتقال بیشتر خواهد بود.

خاصیت مانا بودن (غیر فرار ـ non volatile) و نامانا بودن (فرار ـ volatile)

۱ـ حافظه‌هائی که با قطع جریان برق پاک شوند (اطلاعات داخل آن از بین می‌رود)، حافظه‌های فرار نامیده می‌شود، (حافظه‌های اصلی (RAM ـ ثبات Register) معمولاً فرار هستند، به جز ROM) (درون ماشینی)

۲ـ حافظه‌هائی که با قطع جریان برق پاک نمی‌شوند (اطلاعات داخل آن از بین نمی‌رود)، حافظه‌های غیرفرار نامیده می‌شود، (حافظه‌های جانبی (دیسک ـ نوار مغناطیسی) معمولاً غیر فرار هستند.) (برون ماشینی)

* خاصیت ماناتائی یا نامانائی جزء خصوصیات مشترک انواع حافظه نیست.

□ دلایل استفاده از انواع مختلف رسانه‌های ذخیره‌سازی (حافظه‌های برون ماشینی)

۱ـ حافظه‌های درون ماشینی ظرفیت محدود دارند. (RAM ـ ...)

۲ـ حافظه‌های درون ماشینی سرعت بالائی دارند اما هزینه آن‌ها زیاد است. (گران بودن)

۳ـ برنامه‌ها معمولاً به حافظه‌ای بیش از حافظهٔ درون ماشینی نیاز دارند.

۴ـ حافظه‌های درون ماشینی نامانا هستند و اطلاعات ذخیره شده در آن‌ها با قطع جریان برق از بین می‌رود.

۵ ـ در بعضی مواقع لازم است چند Proccess (پردازش) (فرارونده) به صورت هم‌روند (موازی) (Concurrent) به داده دستیابی داشته باشند در این صورت داده باید حتماً روی دیسک ذخیره شود.

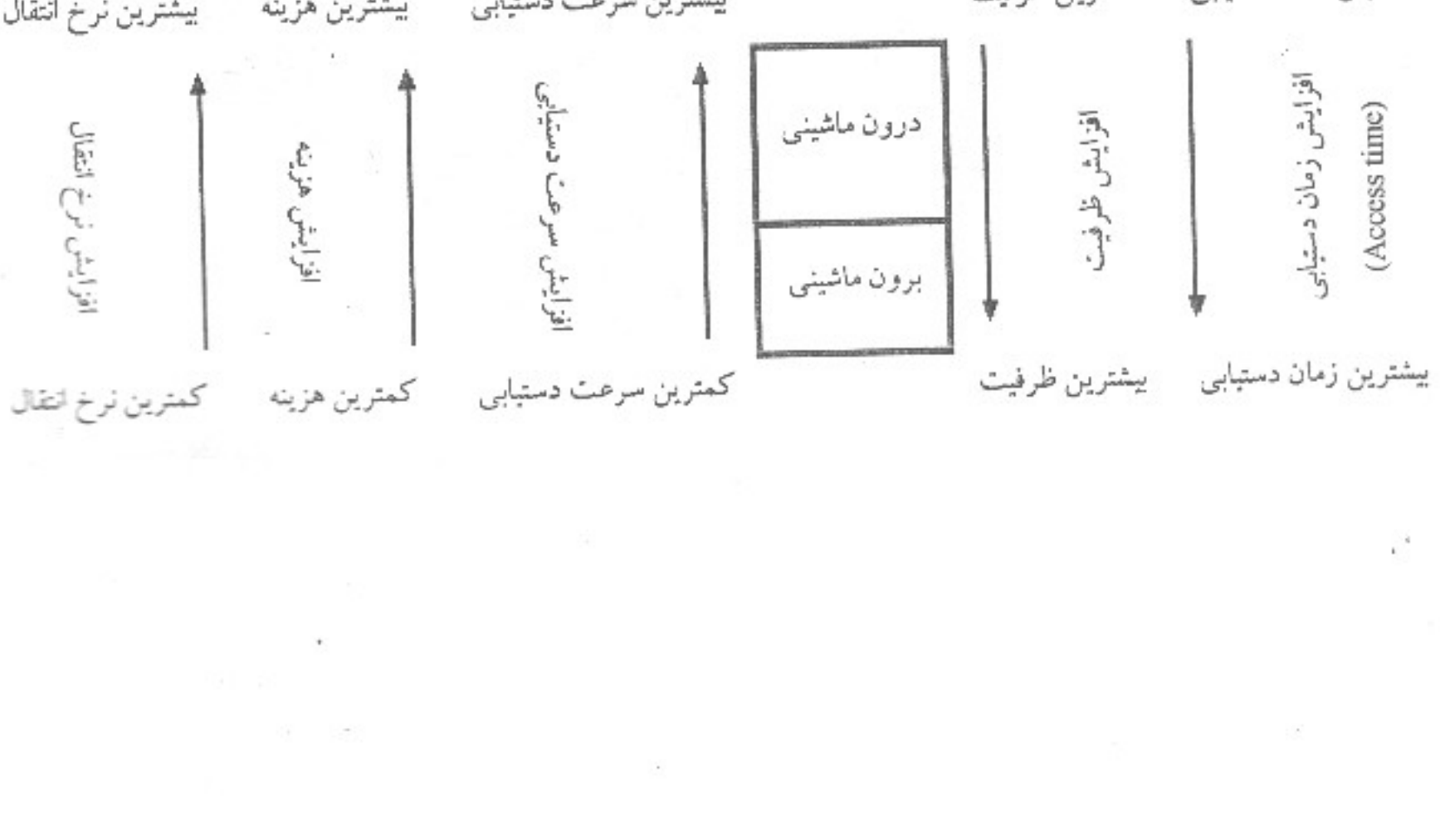
۶ـ بیشتر برنامه‌ها هنگام اجرا قسمت کمی از خود را برای اجرا نیاز دارند و بقیه قسمت‌های آن‌ها فقط باید نگهداری شود بنابراین بهتر است از حافظه جانبی برای آن‌ها استفاده کنیم.

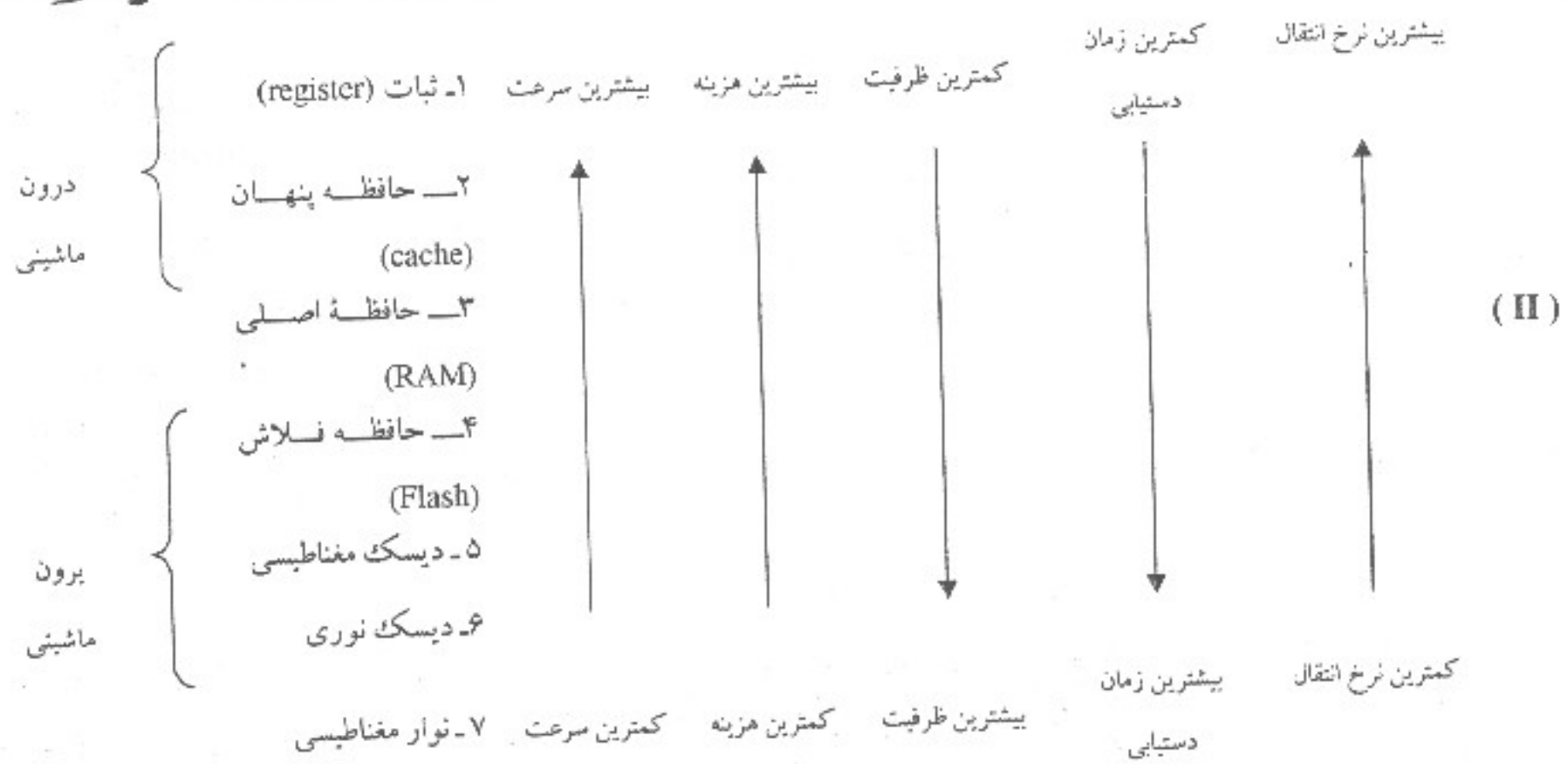
نکته: در هر کامپیوتر باتوجه به مزایای حافظه درون ماشینی (سرعت زیاد) و برون ماشینی (ظرفیت بالا ـ هزینه کم ـ مانا بودن) ترکیبی از هر دو نوع حافظه مورد استفاده قرار می‌گیرد.

سلسله مراتب حافظه (مهم)

با توجه به پارامترهای سرعت ـ ظرفیت ـ هزینه ـ زمان دستیابی سلسله مراتب حافظه از حافظه درون ماشینی تا حافظه برون ماشینی به شکل زیر نمایش داده می‌شود.

(I) کمترین زمان دستیابی کمترین ظرفیت بیشترین سرعت دستیابی بیشترین هزینه بیشترین نرخ انتقال





نتیجه‌گیری:

۱ـ حافظه‌های درون ماشینی:

بیشترین سرعت دستیابی ـ بیشترین نرخ انتقال ـ بیشترین هزینه ـ کمترین ظرفیت ـ کمترین زمان دستیابی (Access time)

۲ـ حافظه‌های برون ماشینی:

کمترین سرعت دستیابی ـ کمترین نرخ انتقال ـ کمترین هزینه ـ بیشترین ظرفیت ـ بیشترین زمان دستیابی (Access time)

انواع حافظه‌های جانبی (برون ماشینی) از نظر تکنولوژی ساخت

حافظه‌های جانبی از نظر تکنولوژی ساخت به ۴ قسمت تقسیم می‌شود:

۱) تکنولوژی الکترومکانیک (مانند: نوار منگنه ـ کارت منگنه)

۲) تکنولوژی الکترومغناطیس (مانند: نوار مغناطیسی ـ دیسک مغناطیسی ـ طبله (drum))

۳) تکنولوژی الکترواپتیک (مانند: دیسک نوری (CD))

۴) تکنولوژی الکترومغناپتیک (مغناطیسی ـ نوری) (مانند: دیسک‌های (MD) (Magentic optic) (ادغامی از ۲ نوع تکنولوژی امت 2 و 3 است)

در ادامه بحث به مطالعه و بررسی ویژگی‌های سه رسانه (حافظه) رایجتر یعنی نوار مغناطیسی ـ دیسک مغناطیسی ـ دیسک نوری (CD) می‌پردازیم.

نوار مغناطیسی

نوار مغناطیسی اساساً برای پردازش پی در پی یا ترتیبی (sequential) داده‌ها مورد استفاده قرار می گیرد.

نکته ۱: (مهم) از نظر تکنولوژی ساخت، نوارهای مغناطیسی را به ۴ دسته تقسیم‌بندی می‌کنند:

۱ـ ریل به ریل

۲ـ نوار کارتریج

۳ـ نوار کاست

۴ـ نوار صوتی (تطبیق داده شده با کامپیوتر)

نکته ۲: دستگاه نوارخوان(Tape Drive) دارای یک هد خواندن / نوشتن است که می‌تواند اطلاعات راروی نوار ضبط کرده و یا اطلاعات ضبط شده را، حس (sense) کرده و بخواند.

نحوه ذخیره‌سازی داده‌ها روی نوار

بر روی هر نوار مغناطیسی داده‌ها به صورت رشته‌های بیتی روی شیارهایی (track) که در سطح نوار وجود دارد، ذخیره می‌شود.

نکته ۱: (مهم)

بیت‌های یک کاراکتر روی شیارها و ***در عرض نوار*** نگهداری می‌شود.

نکته ۲: از نظر تعداد شیار دو نوع نوار وجود دارد:

۱) 7 شیاره

۲) 9 شیاره

نکته ۳: یکی از شیارهای نوار برای کنترل خطا به نام شیار کنترل پاریتی (parity) به کار می‌رود.

نکته ۴: در نوار دو نوع بیت پاریتی (parity bit) وجود دارد:

۱) **بیت پاریتی عرضی یا کاراکتری:** این بیت به ازاء هر کاراکتر ذخیره می‌شود.

۲) **بیت پاریتی طولی یا بلاکی:** این بیت به ازاء تعدادی کاراکتر یا یک بلاک ذخیره می‌شود.

✓ هر بلاک را مجموعه‌ای از کاراکترها یا رکوردها می‌گوییم.

کاراکتر 'A'

	0	
	1	
	0	
	0	
	0	
	0	
	0	
	1	
		← شیار پاریتی (کنترل خطا)

بیت‌های کاراکتر 'A'

مبنای ۲

0 1 0 0 0 0 0 1

= کد اسکی 65 = کاراکتر 'A'

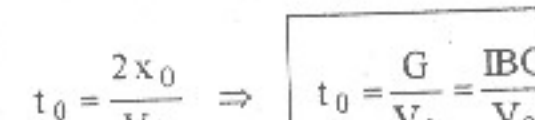
چگالی نوار

تعریف: تعداد بیت‌های ضبط شده در هر اینچ نوار را چگالی (Density) می‌گویند. (میزان داده‌ای که در واحد سطح داده می‌شود را چگالی می‌گویند)

واحد چگالی نوار، ***بیت در اینچ*** (bit perinch ***(bpi)***) بیان می‌شود، اما با توجه به ***نحوه نشست کاراکترها*** که ***روی عرض نوار*** قرار می‌گیرند به آن ***بایت در اینچ*** یا ***کاراکتر در اینچ*** نیز می‌گویند.

در مسائل چگالی نوار بایت در اینچ در نظر گرفته می‌شود.

گپ (GAP) : به فضای بلا استفاده (هرز = waste) بین دو گروه کاراکتر (بلاک) گپ گفته می‌شود.



نکته ۱: (مهم)

گپ در نوار مغناطیسی برای توقف یا حرکت دوبارهٔ هد خواندن / نوشتن مورد نیاز است.

توضیح: برای آن‌که هد خواندن / نوشتن بتواند داده‌های موجود روی نوار را بخواند باید به سرعت یکنواختی برسد که به آن سرعت حس می‌گویند و همچنین هنگام متوقف شدن عملیات خواندن / نوشتن مکانی مورد نیاز است تا هد از سرعت یکنواخت به سرعت 0 برسد، به این مکان که برای رسیدن به سرعت یکنواخت از سرعت 0 یا رسیدن به سرعت 0 از سرعت یکنواخت مورد نیاز است GAP (گپ) گفته می‌شود.

■ چون گپ فقط جهت تغییر سرعت مورد نیاز است، بنابراین هیچ داده‌ای روی آن نگهداری نمی‌شود و به آن فضای هرز (waste) نیز می‌گوئیم.

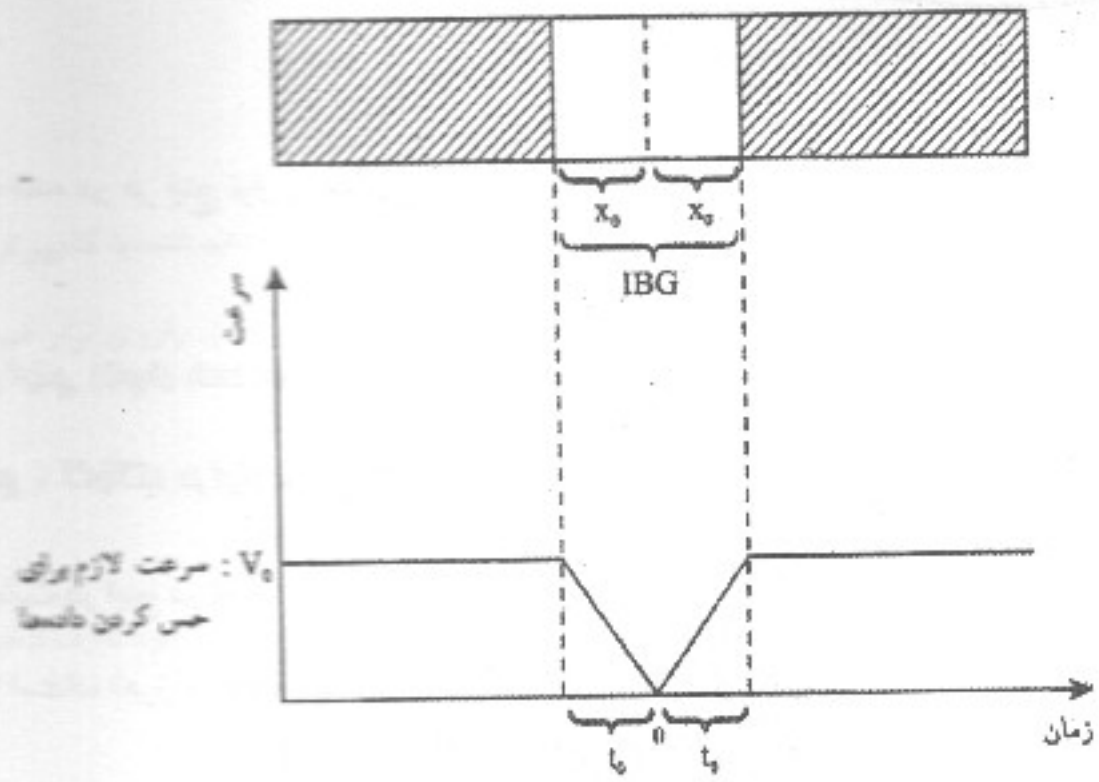
نکته ۲: در صورتی که گپ بین دو گروه کاراکتر (بلاک) باشد به آن IBG (Inter Block Gap) گفته می‌شود و در صورتی که بین دو رکورد باشد به آن IRG (Inter Record Gap) گفته می‌شود.

■ اغلب کلمهٔ GAP را به صورت IBG در نظر می‌گیرند. چون بهتر است به جای آن‌که بین رکوردها فاصله قرار دهیم بین بلاک‌ها فاصله بگذاریم تا فضای هرز (Waste) کمتری داشته باشیم.

زمان حرکت / توقف (t_0)

مدت زمانی است که هد خواندن / نوشتن طول گپ (GAP) را طی می‌کند تا به سرعت حس یا به سرعت 0 برسد، این زمان از رابطه زیر بدست می‌آید:

$$t_0 = \frac{2x_0}{V_0} \Rightarrow \boxed{t_0 = \frac{G}{V_0} = \frac{IBG}{V_0}}$$



در رابطه فوق:

t_0 : زمان حرکت / توقف (برحسب ثانیه یا میلی‌ثانیه)

V_0 : سرعت حس (برحسب اینچ در ثانیه یا اینچ در میلی ثانیه)

G : طول گپ (برحسب اینچ)

نکته بسیار مهم:

در صورتی که طول گپ (GAP) برحسب byte داده شود باید آن را با استفاده از چگالی D برحسب اینچ بدست آوریم، بصورت زیر:

$$\frac{G_{byte}}{D_{byte/inch}} = G_{inch}$$

مثال:

(I): اگر نوار مغناطیسی با سرعت 60 in / sec و طول گپ بین بلاک 0.15 in داشته باشیم زمان حرکت / توقف چند میلی‌ثانیه است؟

$$\left.\begin{aligned} V_0 &= 60 \text{ in/sec} \\ IBG &= 0.15 \text{ in} \end{aligned}\right\}$$

حل: چون طول گپ برحسب اینچ است نیازی به تبدیل واحد نداریم:

$$t_0 = \frac{IBG}{V_0} = \frac{0.15 \text{ in}}{60 \text{ in/sec}} = 0.0025 \text{ sec}$$

$$t_0 = 0.0025 \text{ sec} \times 1000 = \boxed{2.5 \text{ ms}}$$

(II): اگر سرعت حس نوک یک نوار برابر 150 اینچ بر ثانیه باشد و زمان رسیدن به این سرعت 2 میلی‌ثانیه باشد، IBG چند اینچ است؟

۱) 0.3 ۲) 0.75 ۳) 7.5 ۴) 300

حل:

$$\left.\begin{aligned} V_0 &= 150 \text{ in/sec} \\ t_0 &= 2 \text{ ms} = 0.002 \text{ sec} \end{aligned}\right\} \Rightarrow t_0 = \frac{IBG}{V_0} \Rightarrow IBG = 150 \text{ in/sec} \times 0.002 \text{ sec} = 0.3 \text{ in}$$

(III): اگر سرعت حس یک نوار 200 اینچ بر ثانیه باشد و طول گپ 25 بایت باشد، مطلوبست زمان حرکت / توقف نوار؟

(D = 5 bpi = چگالی)

حل: در این مسئله باید طول گپ را با استفاده از چگالی برحسب اینچ بدست آوریم.

$$\begin{cases} V_0 = 200\,{}^{in}/_{sec} \\ IBG = 25_{byte} \end{cases} \Rightarrow IBG_{in} = \frac{IBG_{byte}}{D_{byte/in}} = \frac{25}{5} = 5_{in} \Rightarrow t_b = \frac{IBG}{V_0} = \frac{5_{in}}{200_{in/sec}} = 0.025_{sec}$$

نحوه ذخیره‌سازی فایل روی نوار:

هر فایل معمولاً در قالب بلاک‌هائی (مجموعه‌ای از رکوردها) به‌صورت پی‌درپی روی نوار ذخیره می‌شود و اصطلاحاً می‌گوئیم فایل بلاک‌بندی شده است.

- هر بلاک را مجموعه‌ای از رکوردها فرض می‌کنیم.
- روی یک نوار ممکن است چند فایل نگهداری شود. دراین‌حالت هر فایل دارای نشانگر شروع (BOF) و نشانگر پایان (EOF) خواهد بود.

پارامترهای نوار: (مهم)

- پارامترهای اساسی نوار عبارتند از:

الف) سرعت نوار

ب) چگالی

ج) نرخ انتقال

- در نوارهای مغناطیسی دو دسته پارامتر وجود دارد:
  - ۱- پارامترهای ظرفیتی
  - ۲- پارامترهای زمانی

۱- پارامترهای ظرفیتی
- الف) چگالی (D)
- ب) طول نوار (L)
- ج) طول گپ (IBG)

۲- پارامترهای زمانی
- الف) سرعت لغزش نوار (اینچ بر ثانیه)
- ب) نرخ انتقال (بایت بر ثانیه)
- ج) زمان حرکت / توقف(میلی ثانیه)

در هر نوار مغناطیسی با داشتن پارامترهای ظرفیتی می‌توان ظرفیت اسمی و واقعی نوار را محاسبه کرد:

$$\text{ظرفیت اسمی} = L_{in} \times D_{byte/in}$$

B = اندازه یا طول هر بلاک (inch یا byte)

G= طول یا اندازه گپ (inch یا byte)

$$\text{درصد استفاده واقعی از نوار} = \frac{B}{B+G} \times 100$$

$$\text{ظرفیت واقعی} = (\text{ظرفیت اسمی}) \times (\text{درصد استفاده واقعی از نوار}) = (L \times D) \times \frac{B}{B+G}$$

نکته (مهم):

هرگاه رکوردهای یک فایل را بلاک‌بندی کنیم هر مجموعه از رکوردها در یک بلاک قرارمی‌گیرند، دراین‌حالت به تعداد رکوردهای موجود در هر بلاک، ضریب بلاک‌بندی یا فاکتور بلاک‌بندی گفته می‌شود که آن‌را با B_f نشان می‌دهند. تعداد رکوردهای هر بلاک به نوع بلاک‌بندی بستگی دارد که در فصل بعدی آن‌را مورد مطالعه قرار می‌دهیم.

(I) : $B =$ طول یا اندازه یک بلاک

$$\Rightarrow B = B_f \times R = \text{طول یا اندازه یک بلاک} \qquad (II):\begin{cases} B_f = & \text{تعداد رکوردهای هر بلاک} \\ R = & \text{طول یا اندازهٔ هر رکورد} \end{cases}$$

نکته۲ (مهم): در یک فایل بلاک‌بندی شده، بین هر دو بلاک متوالی یک Gap وجود دارد، بنابراین درصد استفاده واقعی از یک نوار به دو عامل بلاک و گپ بستگی دارد:

$$\text{درصد استفاده واقعی} = \frac{B}{B+G} \times 100$$

نکته ۳(مهم): باتوجه به رابطهٔ ظرفیت واقعی نوار دیده می‌شود که:

$$\text{ظرفیت واقعی} = L \times D \times \frac{B}{B+G}$$

افزایش ظرفیت نوار = افزایش طول نوار ـ افزایش چگالی ـ کاهش گپ ـ افزایش طول بلاک‌ها یا رکوردها

کاهش ظرفیت نوار = کاهش طول نوار ـ کاهش چگالی ـ افزایش گپ ـ کاهش طول بلاک‌ها یا رکوردها

نمونه سؤالات:

۱- فایلی را درنظر بگیرید با 10000 رکورد 80 بایتی که روی نواری با چگالی 1600_{bpi} ذخیره شده است، این فایل چه طولی از نوار را اشغال می‌کند؟

۱) 500 اینچ ۲) 700 اینچ ۳) 1000 اینچ ۴) 1200 اینچ

حل:

باتوجه به آن‌که ظرفیت فایل $10000 \times 80 = 800000$ بایت می‌باشد،

درنتیجه

$$\text{ظرفیت فایل} = \text{طول فایل} \times \text{چگالی} \Rightarrow 800000 = L \times 1600 \Rightarrow \boxed{L = 500}$$

۲- چگالی نواری 1800bpi و طول گپ آن 0.6_{in} می‌باشد، اگر اندازه بلاک‌ها 1400 بایت باشد، درصد استفاده واقعی از نوار چند درصد است؟

۱) 87 ۲) 45 ۳) 43.5 ۴) 56

حل: باید G را برحسب بایت یا B را برحسب اینچ بدست آوریم.

$$\text{درصد استفاده واقعی} = \frac{B}{B+G}\times 100$$

$$B = 1400_{byte}$$
$$G = 0.6_{in} = 0.6_{in} \times D_{bpi} = 0.6 \times 1800 = 1080_{byte} \qquad \Rightarrow \frac{B}{B+G}\times 100 = \frac{1400}{1400+1080}\times 100 = 56\%$$

۳- درصد استفاده واقعی نواری با مشخصات زیر کدام است؟

$$t_G = 4_{ms}\ ,\ v_0 = 125_{in/sec}\ ,\ B = 1200_{byte}\ ,\ D = \text{چگالی} = 1600_{bpi}$$

حل: ابتدا G را محاسبه می‌کنیم:

$$t_0 = \frac{G}{v_0} \Rightarrow G = t_0 \times v_0 = 0.004_{sec} \times 125_{in/sec} = 0.5_{in}$$

سپس برای محاسبه درصد استفاده واقعی از نوار یا G را برحسب بایت یا B را برحسب اینچ بدست می‌آوریم تا بتوانند باهم در رابطه استفاده شوند:

$$G = 0.5_{in} \Rightarrow G_{byte} = 0.5_{in} \times D_{byte/in} = 0.5 \times 1600 = 800\ byte$$

حال می‌توانیم درصد استفاده واقعی را بدست آوریم:

$$\text{درصد استفاده واقعی} = \frac{B}{B+G}\times 100 = \frac{1200}{1200+800}\times 100 = \%60$$

۴- فایلی را درنظر بگیرید با 1000 رکورد 80بایتی که روی نواری با چگالی 1600_{bpi} ذخیره شده است. اگر $B_f = 50$ و $IBG = 0.5_{in}$ باشد میزان واقعی استفاده از نوار کدام است؟

حل:

$$(I)\begin{cases} R = 80\,byte \\ B_f = \text{تعداد رکوردهای هر بلاک} = 50 \end{cases} \Rightarrow B = 80\times 50 = 4000byte \text{ اندازه بلاک}$$

$$(II)\begin{cases} IBG = G = 0.5_{in} \times 1600_{byte/in} = 800byte \end{cases}$$

$$\text{درصد استفاده واقعی نوار (II) و (I)} = \frac{B}{B+G}\times 100 = \frac{4000}{4000+800}\times 100 = \%83$$

۵- در یک نوار مغناطیسی درصورتی که ظرفیت اسمی نوار 2500 بایت و چگالی نوار $250^{byte}/_{ft}$ باشد درصورتی که فضای داده بلاک 200 بایت و تعداد IBG = 50 بایت باشد، میزان واقعی استفاده از نوار چقدر است؟

$$\text{ظرفیت واقعی} = (\text{ظرفیت اسمی}) \times (\text{درصد استفاده واقعی}) = (L\times D)\times\left(\frac{B}{B+G}\right) = 2500\times\frac{20}{200+50} = 2000byte$$

چند نکته:

۱- نرخ انتقال به دوصورت اسمی و واقعی بیان می‌شود، نرخ انتقال اسمی توسط کارخانه سازنده بیان می‌شود و نرخ انتقال واقعی قابل محاسبه است.

۲- دسترسی ترتیبی در نوارها سریع بوده و درحین حال حمل و نقل نوارها ساده است، و از نظر هزینه از دیسک‌ها ارزانتر هستند.

۳- درحال حاضر نوارها بیشتر برای بایگانی و آرشیو اطلاعات مورد استفاده قرار می گیرند.

دیسک مغناطیسی:

حافظه‌ای جانبی به‌صورت گردان با امکان دستیابی ***تصادفی و مستقیم*** به داده‌های ذخیره‌شده که اصطلاحاً به آن *(Direct Access Device) DASD* نیز می‌گویند.

■ دیسک‌گردان (disk drive) مجهز به هد خواندن / نوشتن است، این هد متصل به بازوئی است که می‌تواند رویهٔ دیسک را در مسیر شعاع آن پویش کند. (در بعضی دیسک‌ها مانند طبله بازو ثابت است)

دسته‌بندی دیسک‌ها:

دیسک‌ها از نقطه‌نظرهای متفاوتی تقسیم‌بندی می‌شوند:

۱- از نظر جابجائی پذیربودن:

الف) دیسک‌های ثابت

ب) دیسک‌های جابجاشدنی

۲- ازنظر ثابت یا متحرک‌بودن نوک (هد) خواندن / نوشتن:

الف) دیسک با هد ثابت

ب) دیسک با هد متحرک

* دیسک‌ها با هد ثابت، ***سرعت بالا*** و ***هزینه بالائی*** دارند. (حساس‌تر نیز هستند).

* دیسک‌ها با نوک (هد) ثابت هر شیار هد مخصوص خود را دارد. اما در دیسک‌ها با نوک (هد) متحرک، بازوی دیسک روی شیارها حرکت کرده و از یک شیار به شیار دیگر می‌رود.

۳- از نظر تعداد رویه در صفحه:

الف) یک رویه (single side)

ب) دورویه (Double side)

۴- از نظر تعداد لایه در رویه:

الف) تک‌لایه

ب) دولایه (جدیدتر)

۵- از نظر تعداد صفحات:

الف) تک‌صفحه‌ای (single platter)

ب) چند صفحه‌ای پک (Pack) (multiple platter)

نکته مهم: به دیسک‌های چند صفحه‌ای اصطلاحاً پَک (Pack) می‌گویند، در یک پک با n صفحه ***2n رویه*** وجود دارد که معمولاً از ***2n-2*** رویه آن برای ذخیره‌سازی استفاده می‌شود و دو رویه بالائی و پائینی را برای حفاظت استفاده می‌کنند (در حالت خاص در بعضی از انواع، از 2n رویه استفاده می‌شود).

۶ـاز نظر جنس صفحه:

الف- دیسک سخت (Hard disk)

ب-دیسک نرم (Floppy disk)

۷ـاز نظر تکنولوژی ساخت:

الف) دیسک مغناطیسی

ب) دیسک نوری

ج) دیسک نوری مغناطیسی

تقسیمات دیسک:

۱)**شیار** (track): به دوایر متحدالمرکز در هر رویه دیسک که محل ضبط بیت‌های اطلاعاتی می‌باشد شیار گفته می‌شود.

۲)**استوانه** (cylinder): تمام شیارهای هم شعاع در رویه‌های مختلف دیسک را استوانه می‌گویند، (در دیسک پک (pack) به تعداد شیارهای هر رویه استوانه داریم)

۳)**سکتور یا قطاع** (sector): هر شیار را می‌توان به قسمت‌های مساوی تقسیم کرد که به هر کدام یک سکتور گفته می‌شود، در عین حال هر شیار ازتعداد سکتور تشکیل شده است.

نکته مهم:

الف) شیارهای یک دیسک از بیرونی‌ترین شیار به سمت داخل از صفر شماره‌گذاری می‌شود.

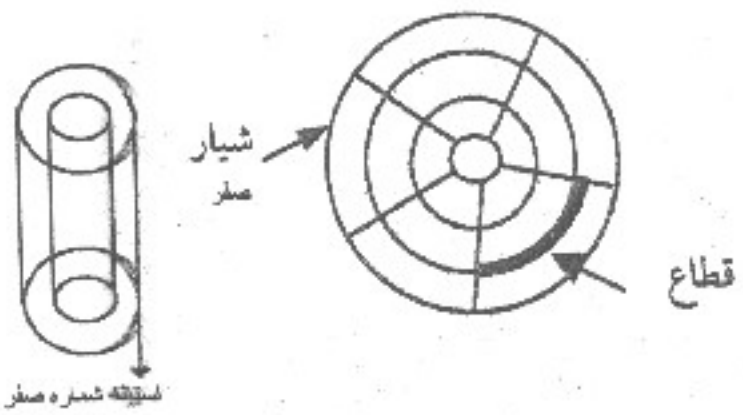
ب) اندازه هر سکتور معمولاً 512 بایت یا 0.5 کیلوبایت است.

ج) طول سکتورها در شیارهای بیرونی بیشتر از شیارهای داخلی است.

د) ظرفیت سکتورها و شیارهای بیرونی و داخلی با هم برابر است.

هـ) با توجه به مورد (د) چگالی شیارها و سکتورهای بیرونی کمتر از شیارها و سکتورهای داخلی است.

و) با توجه به مورد (هـ) فضای هدر رفته در شیارها و سکتورهای بیرونی بیشتر از شیارها و سکتورهای داخلی است.



ز) در دیسک‌ها همیشه دو نوع سرعت وجود دارد:

۱) سرعت خطی

۲) سرعت زاویه

در دیسک‌های مغناطیسی
۱ـ سرعت خطی در شیارهای بیرونی بیشتر از شیارهای درونی است
۲ـ سرعت زاویه‌ای ثابت است (CAV)

[CAV = Constant Angular Velocity]

نکته ۱: در دیسک‌های امروزی (IDE) با استفاده از تکنولوژی بیت Zone، ظرفیت دیسک‌ها بیشتر شده و از فضای دیسک بهتر استفاده می‌شود، در این دیسک‌ها تعداد سکتورها در شیارهای بیرونی بیشتر از شیارهای داخلی است. در نتیجه ظرفیت شیارهای بیرونی بیشتر از شیارهای داخلی است.

نکته ۲: همیشه دو نوع سکتور در دیسک وجود دارد:

۱) **سکتور سخت‌افزاری:** که توسط شرکت سازنده ایجاد می‌شود. (فرمت کردن سطح پائین)

۲) **سکتور نرم افزاری (بلاک):** از طریق سیستم‌عامل و با دستور format ایجاد شده، و امکان ضبط اطلاعات را میسر می‌کند.

■ تقسیمات بیان شده در اصل برای آدرس‌دهی به داده مورد نظر در فضای دیسک استفاده می‌شود، مؤلفه‌های آدرس دهی فیزیکی عبارتند از:

۱ـ شماره درایو

۲ـ شماره استوانه (شیار از رویه)

۳ـ شماره شیار در استوانه (رویه از استوانه)

۴ـ شماره سکتور (بلاک)

■ در دیسک‌های مغناطیسی با ساختار CAV اغلب برای آدرس‌دهی از یک آدرس 3 مؤلفه‌ای به صورت ***(شماره هد ـ شماره سیلندر (استوانه) ـ شماره سکتور)*** یا ***(شماره رویه از استوانه ـ شماره شیار در رویه ـ شماره سکتور)*** استفاده می‌شود.

■ در اکثر سیستم‌عامل‌های جدید، تعداد سکتور همجوار یا غیرهمجوار نوعی تقسیم‌بندی منطقی را تشکیل می‌دهند که به آن خوشه یا کلاستر (cluster) می‌گوئیم، جهت استفاده بهینه از فضای دیسک، یک فایل را به صورت گسسته روی کلاسترها نگهداری می‌کنند در این حالت زمان حرکت هد و همین‌طور زمان خواندن فایل افزایش پیدا می‌کند.

پارامترهای دیسک: (مهم)

۱) پارامترهای ظرفیتی
۲) پارامترهای زمانی

۱) پارامترهای ظرفیتی:

عبارتند از:

الف) اندازهٔ هر سکتور

ب) تعداد سکتور در هر شیار

ج) تعداد رویه در استوانه (تعداد هد خواندن / نوشتن)

د) تعداد شیار در رویه (تعداد استوانه)

هـ) چگالی

■ واحد چگالی دیسک مغناطیسی ***تعداد بیت‌ها در هر اینچ*** (Bit per Inch = BpI) یا ***تعداد شیارها در هر اینچ*** (Track per Inch = TpI) گفته می‌شود.

■ افزایش چگالی دیسک از قانونی به نام قانون هوگلند تبعیت می کند، به شرح زیر:

$$\text{چگالی (سال)} = 10^{\frac{\text{سال} - 1979}{10}} \quad Mb/inch^2$$

۲) پارامترهای زمانی:

عبارتند از:

الف) زمان پیگرد یا استوانهٔ جوئی (seek time)

ب) زمان درنگ دورانی یا تأخیر چرخشی (rotational lantency time)

ج) نرخ انتقال (transfer rate)

د) سرعت گردش دیسک (با واحد دور در دقیقه = RPM)

هـ) زمان استقرار (setting time)

نکته ۱: (مهم)

در بین پارامترهای زمانی 3 پارامتر اول، الف و ب و ج بیشترین اهمیت را داشته و تأثیر زیادی روی زمان دستیابی فایل دارند.

الف) زمان استوانه جوئی
ب) زمان درنگ دورانی
ج) نرخ انتقال،

الف) زمان استوانه‌جوئی (پیگرد= seek time):

مدت زمان لازم از زمانی که دستور خواندن / نوشتن صادر می‌شود تا زمانی که هد یا نوک خواندن / نوشتن به استوانه (سیلندر) مورد نظر برسد (یعنی استوانه‌ای که داده مورد نظر در آن وجود دارد.)

■ متوسط زمان استوانه‌جوئی را با S نشان می‌دهند (میلی ثانیه) که در دیسک‌ها با بازوی ثابت (مانند: طبله) این زمان 0 = S است.

■ زمان S در محیط‌های چند کاربره (multi user) بیشتر از محیط‌های تک کاربره است.

■ زمان S از زمان درنگ دورانی و نرخ انتقال مهمتر است.

ب) زمان درنگ دورانی (Rotational Latency time)

پس از آن‌که هد یا نوک خواندن / نوشتن به استوانه مورد نظر رسید، مدت زمانی که سپری می‌شود. تا ابتدای داده مورد نظر (بلاک یا سکتور مورد نظر) در اثر چرخش یا دوران دیسک به زیر نوک خواندن / نوشتن برسد.

■ (مهم) متوسط زمان درنگ دورانی را با r (میلی ثانیه) نمایش داده که نصف یک دوره کامل دیسک (2r) است و همیشه در رابطه زیر صدق می‌کند:

$$0 \le r = \text{متوسط زمان درنگ دورانی} \le 2r$$

■ (مهم) واحد چرخش دیسک «دور در دقیقه» یا «Rotation per minute» است که آن را با RPM نشان می‌دهند و متوسط زمان درنگ دورانی (r) را از روی آن به شکل زیر بدست می‌آورند:

$$r = \frac{1}{2} \times \frac{60 \times 1000}{RPM}$$

مثال: در صورتیکه در یک دیسک مغناطیسی سرعت گردش دیسک مغناطیسی 300 دور در دقیقه باشد. متوسط زمان درنگ دورانی (r) چقدر است؟

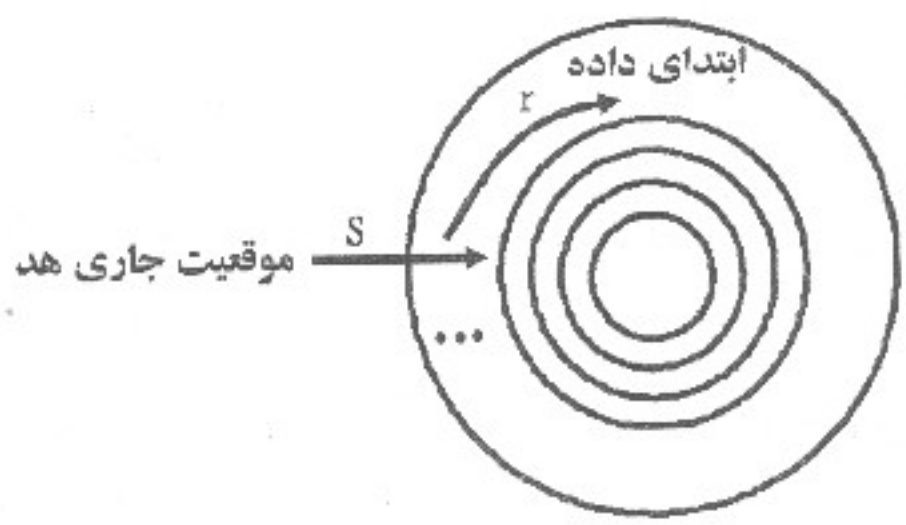
$$RPM = 300 \qquad \Rightarrow r = \frac{1}{2} \times \frac{60 \times 1000}{300} = 100 \text{ ms}$$

زمان دستیابی تصادفی (RAT = Random Access time)

■ مدت زمان لازم بین لحظه‌ای که دستور خواندن / نوشتن صادر می‌شود تا لحظه‌ای که ابتدای داده مورد نظر (سکتور مورد نظر) زیر هد خواندن / نوشتن قرار گیرد. را زمان دستیابی (تصادفی) گویند و مقدار آن از رابطه «$s + r$» بدست می‌آید.

$$RAT = \text{زمان دستیابی تصادفی} = s + r$$

- به عبارت بهتر متوسط زمان لازم برای رسیدن به ابتدای یک بلاک با مکان مشخص از یک مکان نامعین، «$s + r$» خواهد بود که همان زمان دستیابی تصادفی (RAT) می‌باشد.



ج) نرخ انتقال (transfer rate):

تعداد بایتی که در هر ثانیه از دیسک منتقل می‌شود را نرخ انتقال می‌گویند.

■ واحد نرخ انتقال بایت بر ثانیه (bps) است.

■ دو نوع نرخ انتقال وجود دارد:
- ۱- **نرخ انتقال اسمی:** کارخانه سازنده آن را بیان می‌کند.
- ۲- **نرخ انتقال واقعی:** می‌توان آن را محاسبه کرد.

د) زمان استقرار (setting time):

زمانی که نوک خواندن / نوشتن روی استوانه مورد نظر قرار گرفت (زمان استوانه‌جوئی)، مدت زمان کوتاهی طی می‌شود تا لرزش هد متوقف شده و روی استوانه (سیلندر) استقرار یابد.

■ زمان استقرار را معمولاً جزء زمان استوانه‌جوئی در نظر گرفته و به آن اضافه می‌کنند، و به عنوان پارامتر جداگانه‌ای مطرح نمی‌کنند.

متوسط زمان بی‌عیبی (Mean time to failure=MTTF):

عبارتست از متوسط زمانی که یک دیسک می‌تواند بدون عیب و ایراد کار کند. (معمولاً بین 200 تا 500 هزار ساعت است)

فرمت‌بندی دیسک:

شیارهای دیسک که محل قرار گرفتن بیت‌های اطلاعاتی است، به دو شکل سخت‌افزاری و نرم‌افزاری می‌تواند تقسیم‌بندی (فرمت بندی) شود.

الف) فرمت سخت افزاری: در این نوع فرمت‌بندی تقسیم‌بندی شیار به سکتورها از قبل توسط شرکت سازنده انجام شده و ثابت است.

ثابت ثابت

...	Gap	Data 1	Gap	Data 2	...

ب) فرمت نرم‌افزاری: در این نوع فرمت‌بندی که اغلب توسط سیستم‌عامل انجام می‌شود، در تقسیم‌بندی سکتورها، در ابتدای هر سکتور یک بخش کنترلی حاوی اطلاعات سکتور مانند: طول سکتور و ... قرار داده می‌شود.

اطلاعات درباره Data 1	Data 1	اطلاعات درباره Data 2	Data2

فلاپی دیسک (Floppy disk):

این دیسک‌ها که به دیسک‌های نرم شناخته می‌شوند، به عنوان یک حافظه جانبی قابلیت دستیابی مستقیم را دارد، به نکات زیر در مورد این دیسک‌ها توجه می‌کنیم:

۱ـ سرعت و ظرفیت به مراتب کمتر از دیسک‌های سخت (Hard disk) است.

۲ـ برخلاف دیسک‌های سخت که هد با سطح دیسک فاصله اندکی دارد، در فلاپی دیسک‌ها هد با سطح دیسک در تماس است.

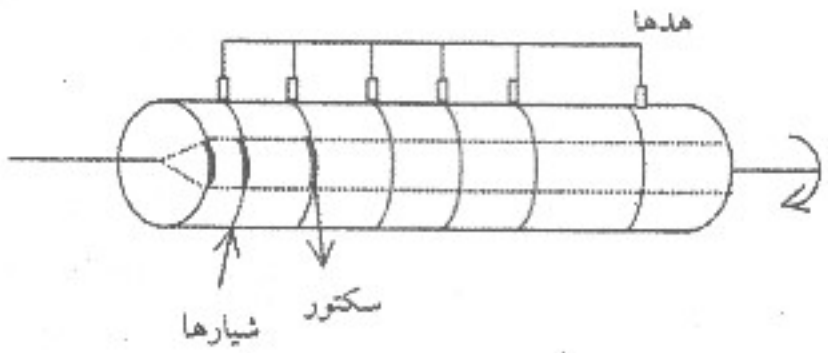
۳ـ فلاپی دیسک‌ها می‌توانند یک‌رویه (یک طرفه) یا دو رویه (دو طرفه) باشند.

۴ـ داده‌ها به صورت سریال ضبط می‌شود.

۵ـ فلاپی دیسک‌ها مجهز به حفره‌ای برای محافظت در مقابل نوشتن است (باز بودن حفره به مفهوم protect یا محافظت در مقابل نوشتن است)

۶ـ در نوعی از فلاپی دیسک حفره‌ای به نام index Hole وجود دارد که محل شروع سکتور صفر از هر شیار را مشخص می‌کند.

طبله (Drum):



این حافظه با یک استوانه که شیارهای آن روی سطح خارجی‌اش قرار گرفته‌اند، معادل با یک دیسک با نوک یا هد ثابت است.

نکات مهم:

- برای هر شیار یک نوک یا هد در نظر گرفته شده است، اگر تعداد نوک‌ها از شیارها کمتر باشد در این صورت نوک متحرک خواهد بود)
- زمان استوانه‌جوئی (S) در طبله $S = 0$ است.
- طبله از نوار و دیسک سریعتر بوده ولی ظرفیت آن کمتر است.
- هزینه بالا (چون برای هر شیار از یک هد استفاده می‌کند.)

موارد استفاده از طبله:

۱) قبل از ایجاد حافظه‌های چنبره‌ای از طبله به عنوان حافظه اصلی استفاده می‌شد.

۲) برای ضبط نرم‌افزارهائی که ثابت بوده و مرتباً مورد استفاده قرار می‌گیرند.

۳) برای ایجاد فایل‌های موقت بسیار فعال که مورد استفاده سیستم‌عامل و کامپایلرها هستند.

۴) به عنوان حافظه پشتیبان برای ماشین مجازی.

دیسک‌های با تغییر فاز

در این نوع دیسک‌ها، رویه دارای غشایی است که می‌تواند، در اثر تابش اشعه لیزر، دو حالت کریستال و یا نامشخص را به خود بگیرد. حالت اولیه غشا، نامشخص است و وقتی که اشعه به آن می‌تابد. حالت کریستالی به خود می‌گیرد و اگر اشعه به آن تابانیده نشود، به حالت نامشخص بازمی‌گردد.

برای خواندن اطلاعات، اشعه‌ای، با قدرت کمتر از حالت نوشتن، به آن تابانیده می‌شود تا نوری را منعکس کند. نوع نور، بسته به اینکه انعکاس از قسمت کریستالی باشد و یا قسمت نامشخص، فرق می‌کند. همین تفاوت در نوع نور، امکان می‌دهد تا دو حالت صفر و یک تشخیص داده شوند.

نکته: سرعت این دیسک‌ها حدوداً دو برابر دیسک‌های مغناطیسی ـ نوری است.

دیسک‌های دای ـ پولیمر

رویه در این نوع دیسک‌ها، دو لایه پولیمر دارد.

نکته ۱: ضبط اطلاعات به صورت زیر انجام می‌شود: لایه زیرین به وسیله لیزر گرم می‌شود و در نتیجه یک برآمدگی در لایه بالا ایجاد می‌گردد. سپس ناحیه برآمده را سرد می‌کنند. بدین ترتیب برآمدگی ثابت می‌ماند.

نکته ۲: پاک کردن اطلاعات به صورت زیر انجام می‌شود: لایه برآمده را به کمک یک اشعه لیزر با طول موج متفاوت با حالت اول، گرم می‌کنند و برآمدگی از بین می‌رود.

دیسک‌های نوری: (optical disk)

در دیسک‌های نوری از تکنولوژی اشعه لیزر برای ذخیره‌سازی داده‌ها استفاده می‌شود این برخلاف دیسک‌های مغناطیسی است که از مغناطیس کردن سطح برای ذخیره‌سازی استفاده می‌کنند.

نکته۱: (مهم)

استفاده از نور به جای مغناطیس کردن برای ذخیره‌سازی این مزیت را دارد که فضای ذخیره‌سازی برای یک بیت خیلی کمتر می‌شود.

نکته۲: استفاده از دیسک‌های نوری سبب کاهش فضای ذخیره‌سازی و افزایش ظرفیت حافظه می‌شود.

CD - ROM:

دیسکی است فقط خواندنی، بدین معنا که پس از ضبط اطلاعات روی آن، امکان دوباره نوشتن در آن وجود ندارد. از این رو به آن worm نیز می‌گویند (write once – Read many)

نکات مهم در CD – ROMها:

۱ـ CD – ROMها چگالی بالائی دارند، تقریباً 200 برابر دیسک‌های نرم (Floppy) و 20 برابر دیسک‌های سخت.

۲ـ CD – ROMها هزینه پائین و دوام بالائی دارند. (مزیت)

۳ـ زمان استوانه جوئی خیلی بیشتر از دیسک مغناطیسی بوده و نرخ انتقال نسبتاً پائین است، در نتیجه سرعت آن‌ها نسبت به دیسک مغناطیسی کمتر است. (نقطه ضعف)

۴ـ سرعت کم در CD – ROMها باعث شده، طراحی ساختار فایل‌ها در آن مشکل‌تر از دیسک‌های مغناطیسی است.

۵ ـ در ابتدا CDها جهت اجرای موسیقی طراحی شده بودند و دستیابی سریع و مستقیم به داده‌ها مدنظر نبود، و این به آن علت است که در CDها ظرفیت بالا و سرعت دستیابی متوسط است.

۶ ـ نوع دیگری از دیسک‌های نوری به نام DVD (Digital video Disk) نیز برای ذخیره‌سازی فیلم‌ها طراحی شده است که می‌توان آن‌ها را برای ذخیره‌سازی فایل‌ها مورد استفاده قرار داد. (ظرفیت DVDها معمولاً حدود 10 گیگابایت است)

۷ ـ CDهای اولیه یک طرفه بودند اما بعضی DVDها به صورت دو طرفه ساخته می‌شوند.

۸ـ بعضی CDها به صورت CD – RW هستند که امکان خواندن و نوشتن دوباره روی آن‌ها وجود دارد.

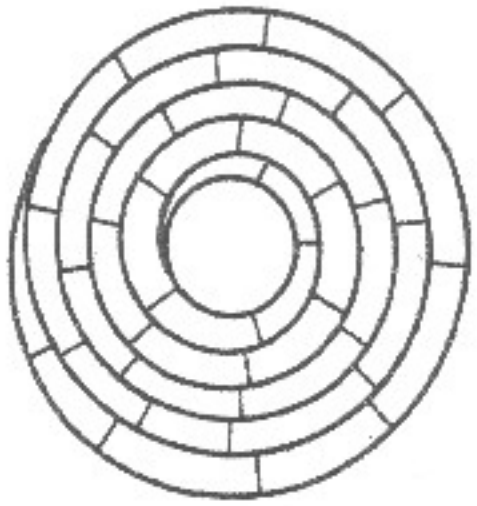
دیسک‌های نوری ـ مغناطیسی:

در این نوع دیسک‌ها سعی بر آن است تا با ادغام دو تکنولوژی نور و مغناطیس، دیسک‌هائی ایجاد کنیم که هم خاصیت ***پاک شدن و دوباره نویسی*** دیسک مغناطیسی و ***هم چگالی و ظرفیت بالای*** دیسک‌های نوری را داشته باشند.

نکته۱: دیسک‌های MO (Magnetic – optic) از دو تکنولوژی مغناطیسی ـ نوری استفاده می‌کنند و قابلیت خواندن و نوشتن را دارند.

نحوه ذخیره‌سازی داده‌ها در دیسک‌های نوری (CD): (مهم)

■ ساختار ذخیره‌سازی داده‌ها بر روی CD – ROM به شکل حلزونی است.



داده‌های CD – ROM بر روی یک شیار مارپیچ شده ذخیره می‌گردد که از مرکز تا لبه دیسک حدود 5 کیلومتر طول دارد.

■ CD – ROM از ابتدا برای ذخیره داده‌های صوتی مورد استفاده قرار می‌گرفت، این داده‌ها فضای ذخیره‌سازی زیادی می‌خواهند ولی جستجو و سرعت در آن‌ها زیاد مهم نیست، چون این داده‌های صوتی باید از ابتدا تا انتها بدون وقفه نواخته شوند.

■ برخلاف دیسک‌های مغناطیسی، اندازه سکتورهای داخلی و بیرونی باهم برابر است.

■ چگالی در کل شیار مارپیچ ثابت است.

نحوه خواندن داده‌ها ازروی CD – ROM:

برای خواندن داده‌ها از روی CD – ROM می‌بایست شیار مارپیچ با سرعت ثابتی از زیر پیکاپ نوری رد شود. به عبارت بهتر سرعت خطی باید ثابت باشد.(Constant Linear Velocity = CLV)

■ برای این‌که سرعت خطی ثابت باشد باید سرعت چرخش دیسک (زاویه‌ای) در لبه‌های بیرونی کندتر از لبه‌های داخلی باشد، به عبارت بهتر سرعت زاویه ثابت نیست.

نکته ۱: (مهم)

دیسک‌های مغناطیسی
- **۱ـ سرعت خطی متغیر:** در شیارهای بیرونی بیشتر و سریعتر از شیارهای درونی
- **۲ـ سرعت زاویه‌ای ثابت:** (CAV= Constant Angular velocity)

دیسک‌های نوری
- **۱ـ سرعت خطی ثابت:** (CLV = Constant Linear Velocity)
- **۲ـ سرعت زاویه‌ای متغیر:** در لبه‌های بیرونی کمتر و کندتر لبه‌های داخلی

نکته ۲: اگر در CD – ROMها به جای حالت CLV (سرعت خطی ثابت) از حالت CAV (سرعت زاویه‌ای ثابت) استفاده شود. آن‌گاه ظرفیت در حدود نصف ظرفیت موجود CD – ROMها می‌شد.

نکته۳: با آن‌که حالت CLV باعث بالا رفتن ظرفیت می‌شود اما از طرف دیگر زمان درنگ دورانی افزایش و سرعت آن کاهش پیدا می‌کند. چون در روش CLV روش سریعتری برای رسیدن به موقعیت مشخصی (مانند دیسک‌ها) وجود ندارد.

نحوه آدرس دهی در CD - ROM

در CD – ROMها جهت آدرس‌دهی از تکنیک دیگری استفاده می‌شود که در آن هر ثانیه از زمان اجرای موسیقی به 75 سکتور تقسیم می‌گردد که هر سکتور 2 کیلوبایت اطلاعات را ذخیره می‌سازد. طبق قرارداد اولیه هر CD حداقل باید بتواند یک ساعت موسیقی را ذخیره سازد. به عبارت دیگر هر CD حداقل باید بتواند 000 540 کیلو بایت داده را نگهداری کند:

$$3600 \times 75 \times 2K = 540\ 000KB$$

به این دلیل است که حداقل ظرفیت CDها 540 مگابایت می‌باشد.

■ هر سکتور در CD – ROM با سه عدد (سکتور : ثانیه : دقیقه) مشخص می‌گردد.

تکنیک‌های ضبط مغناطیسی اطلاعات

سطح نوار یا دیسک می‌تواند سه حالت مغناطیسی مثبت، منفی یا خنثی داشته باشد. از نظر نحوه مغناطیس شدن، تکنیک‌های ضبط اطلاعات متعددی ابداع شده است که در این‌جا چهار تای آن‌ها را شرح می‌دهیم:

۱ـ بازگشت به صفر (Return to Zero = Rz)

۲ـ بی‌بازگشت به صفر (Non Return to Zero = NRZ)

۳ـ بی‌بازگشت به صفر معکوس (Non Return to Zero Inverted = NRZI)

۴ـ کد کردن RRL (Run Length Limit)

عبور جریان در یک جهت سیم هد سبب می‌شود تا نوار در جهت خاصی مغناطیس شود و تغییر جهت جریان باعث تغییر جهت مغناطیس می‌گردد. حال این تکنیک‌ها را شرح می‌دهیم.

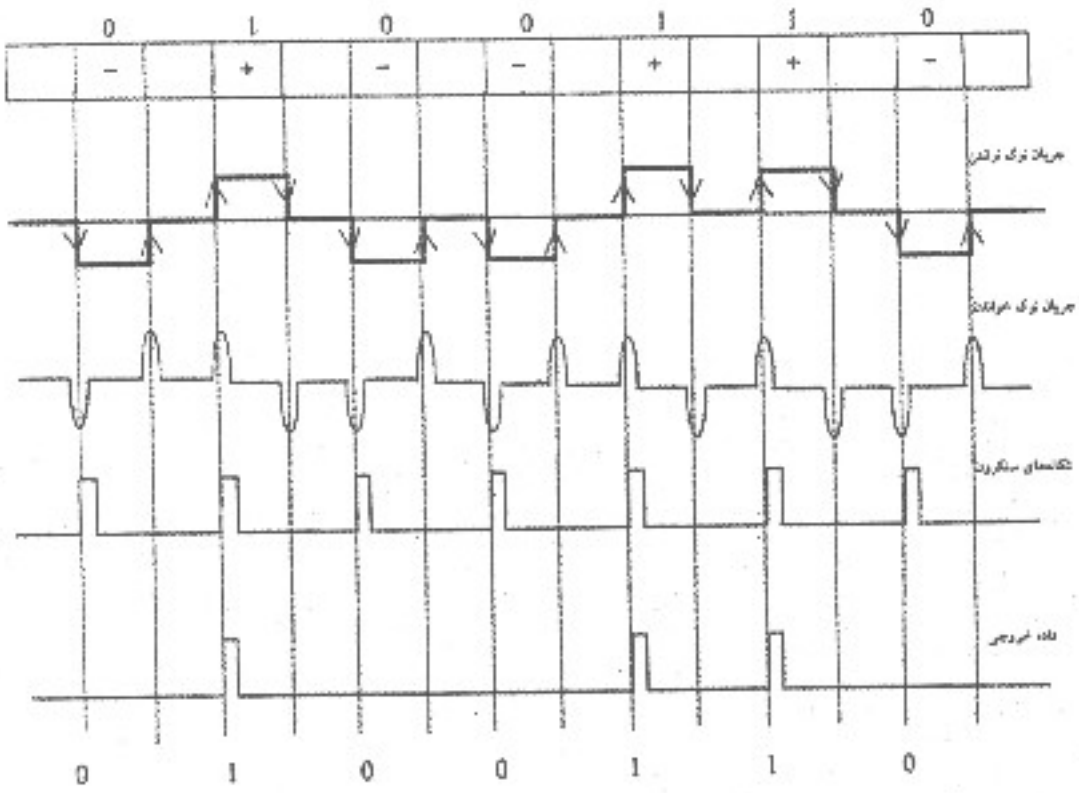
تذکر: تکنیک‌های مشهور دیگری که در این‌جا بیان نمی‌شوند عبارتند از PE، FM و MFM. روش‌های FM و MFM اغلب در فلاپی دیسک‌ها استفاده می‌شوند.

۱ـ تکنیک بازگشت به صفر (RZ)

عبور جریان نوشتن در یک سمت، سبب ایجاد نقطه مثبت مغناطیس روی سطح می‌شود عبور جریان. در سمت دیگر نقطه منفی و اگر جریانی وجود نداشته باشد نقطه حالت خنثی پدید می‌آید. ***در این روش در نقاطی که در آن‌ها جریان نوشتن صفر است عمل ضبط صورت نمی‌گیرد و به این دلیل به روش «بازگشت به صفر» معروف است.***

هنگام خواندن،وجود تغییر در حالت مغناطیس نقاط باعث پدید آمدن جریان در هد (نوک) می‌شود. مثلاً اگر سطح نوار تماماً به صورت مثبت مغناطیس شده باشد، جریانی از هد خواندن عبور نمی‌کند. هنگام خواندن اگر جهت جریان نوک خواندن مثبت باشد به "1" و اگر منفی باشد به صفر تعبیر می‌شود.

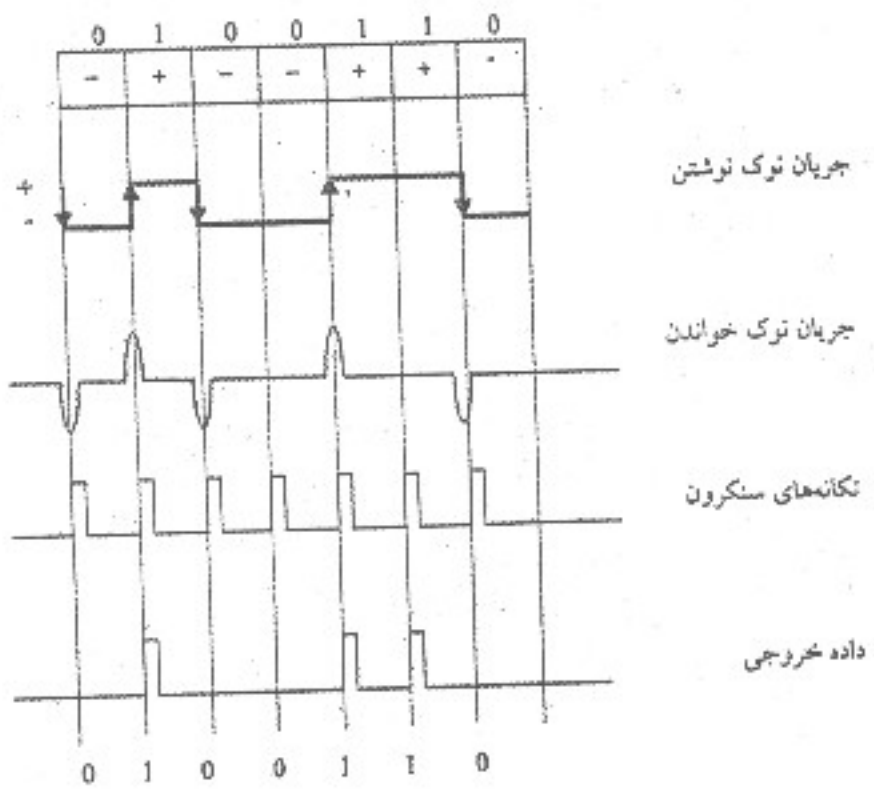
مثال ۱:



تکانه‌ها (پالس‌های) سنکرون شروع بیت‌ها را نشان می‌دهند. شکل جریان در هد خواندن را از روی شکل جریان در هد نوشتن بدست آورید. هرگاه جریان هد نوشتن بالا رونده بود،جریان هد خوانده مثبت است و هرگاه جریان هد نوشتن پائین رونده بود، جریان هد خواندن منفی است.

۲ـ تکنیک بی‌بازگشت به صفر (NRZ)

در این تکنیک جریان هد نوشتن هیچ‌گاه صفر نیست. به عبارتی دیگر بر روی نوار نواحی خنثی هیچ‌گاه وجود ندارد. ***هنگام خواندن جریان مثبت به 1 و جریان منفی به صفر تعبیر می‌شود. باید توجه داشت در این روش صفرهای پی‌درپی و یک‌های پی‌درپی تغییر در شار ایجاد نخواهد کرد. در واقع این روش را باید روش بی‌بازگشت به نواحی مغناطیس خنثی نامید.***

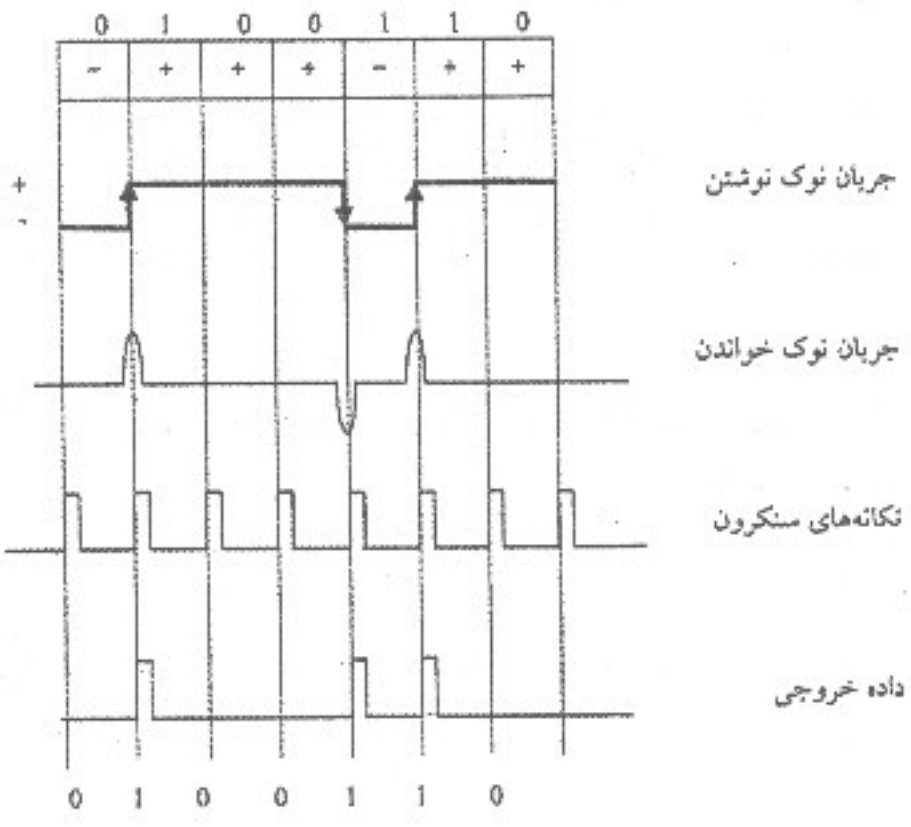


در هنگام نوشتن برای بیت "1" جریان مثبت و برای بیت "0" جریان منفی است و جریان صفر نداریم. در هنگام تعیین داده خروجی اگر جریان منفی باشد به صفر و اگر جریان مثبت باشد به "1" تعبیر می‌شود و اگر جریان وجود نداشته باشد، داده قبلی تکرار می‌شود.

۳ـ تکنیک بی‌بازگشت به صفر معکوس (NRZI)

در این روش هر تغییر شار (بدون توجه به جهت آن) نمایانگر "1" است و عدم تغییر شار نمایانگر "0" می‌باشد. در هنگام نوشتن، هنگام رسیدن به "1" جهت جریان در هد نوشتن را عوض می‌کنیم. در هنگام خواندن، اگر جریانی در هد خواندن باشد (در هر جهتی)، این جریان به "1" تعبیر می‌شود و اگر جریانی وجود نداشته باشد به صفر تعبیر می‌گردد.

مثال ۳:



تذکر مهم: چگالی ذخیره‌سازی اطلاعات یعنی تعداد بیت‌هائی که در واحد طول می‌تواند ذخیره شود بستگی به ،تعداد تغییر شار، در واحد طول دارد. هر چقدر تعداد تغییر شارها در واحد طول کمتر باشد، امکان ذخیره‌سازی بیتهای بیشتری در سطح وجود دارد. همانطور که در سه شکل قبلی مشاهده می‌کنید برای ذخیره و بازیابی رشته (0100110) در روش RZ تعداد 14 تغییر شار، در روش NRZ تعداد 5 تغییر شار و در روش NRZI تعداد 3 تغییر شار داریم. پس چگالی NRZI از دو روش دیگر بیشتر است.

۴ـ روش RLL (Run Length Limit)

تعداد عدم تغییر جهت شار، بین دو تغییر جهت متوالی را Run Length گویند. این پارامتر با دو مقدار Min و Max مشخص می‌شود. مثلا با نگاه کردن به جریان هد خواندن در روش NRZ متوجه خواهیم شد که حداقل هیچ عدم تغییر شار و حداکثر یک عدم تغییر شار، مابین دو تغییر شار متوالی وجود دارد. پس NRZ یک RLL 0,1 می‌باشد. به همین ترتیب روش NRZI یک RLL 0,2 می‌باشد (عدد اول Min و عدد دوم Max عدم تغییر شار است).

اگر بتوان اعداد RLL را بالا برد، به علت کم شدن تعداد تغییر شار، ظرفیت دیسک بالا می‌رود.

سوالات نمونه کنکور

۱ـ هر کدام روش ذخیره اطلاعات روی نوار، در نقاطی که جریان صفر است، اطلاعات ضبط نمی‌شود؟

۱) کدکردن فاز ۲) بازگشت به صفر ۳) بی‌بازگشت به صفر ۴) بی‌بازگشت به صفر معکوس

۲ـ در دیسک مغناطیسی:

۱) اطلاعات ذخیره شده در شیارهای بیرونی بیشتر از اطلاعات ذخیره شده در شیارهای درونی است.

۲) سرعت ذخیره اطلاعات روی شیارهای بیرونی از شیارهای درونی کمتر است.

۳) در شیارهای بیرونی دیسک چگالی ذخیره اطلاعات از شیارهای درونی دیسک کمتر است.

۴) چگالی ذخیره اطلاعات در شیارهای بیرونی و درونی برابر است.

۳ـ اگر نواری با سرعت حرکت 60 in/s و طول گپ بین بلاک 0.15 in داشته باشیم، زمان حرکت توقف نوار چند ms است؟

۱) 1 ۲) 2.5 ۳) 3 ۴) 5

۴ـ چگالی نواری 1800 bpi و طول گپ آن 0.6 in می‌باشد. اگر اندازه بلاک‌ها 1400 بایت باشد، درصد استفاده واقعی از نوار چند درصد است؟

۱) 87 ۲) 45 ۳) 56 ۴) 21

۵ـ access time (زمان دستیابی) عبارت است از:

۱) مدت زمانی بین لحظه‌ای که دستور خواندن/ نوشتن داده می‌شود تا لحظه‌ای که داده موردنظر دستیابی می‌شود.

۲) مدت زمانی که داده موردنظر از روی حافظه خوانده می‌شود.

۳) مدت زمانی که داده موردنظر بر روی حافظه نوشته می‌شود.

۴) مدت زمانی که مکان داده در روی حافظه معین می‌شود.

۶ـ کدام عبارت درمورد حافظه‌های اصلی نسبت به حافظه‌های ثانویه صحیح است؟

۱) ظرفیت کمتر ـ سرعت دستیابی بیشتر

۲) ظرفیت بیشتر ـ سرعت دستیابی بیشتر

۳) ظرفیت کمتر ـ سرعت دستیابی کمتر

۴) ظرفیت بیشتر ـ سرعت دستیابی کمتر

۷ـ کدام عبارت در مورد Gap نادرست است؟

۱) از نظر ذخیره‌سازی گپ فضای هرز (Waste) می‌باشد.

۲) گپ متمایزکننده بین دو بلاک می‌باشد.

۳) وجود گپ بر روی دیسک بدون استفاده می‌باشد.

۴) گپ جهت حرکت هد مورد نیاز می‌باشد.

۸ـ در کدام گزینه حافظه‌ها بر اساس سرعت مرتب شده‌اند؟

۱) ثبات ـ حافظه اصلی ـ حافظه پنهان ـ دیسک نوری

۲) ثبات ـ حافظه پنهان ـ حافظه اصلی ـ دیسک نوری

۳) حافظه پنهان ـ حافظه اصلی ـ ثبات ـ دیسک نوری

۴) حافظه پنهان ـ ثبات ـ حافظه اصلی ـ دیسک نوری

۹ـ فایلی را در نظر بگیرید با 10000 رکورد 80 بایتی که روی نواری با چگالی 1600bpi ذخیره شده است این فایل چه طولی از نوار را اشغال می‌کند؟

۱) 500 اینچ ۲) 700 اینچ ۳) 1000 اینچ ۴) 1200 اینچ

۱۰ ـ تعداد سکتورهای نرم‌افزاری تراک بیرونی یک دیسک مغناطیسی نسبت به تراک‌های داخلی آن.....

۱) برابر است. ۲) بستگی به چگالی تراک بیرونی دارد.

۳) بیشتر است. ۴) بستگی به نحوه فرمت‌بندی دارد.

۱۱ ـ کدام یک از جملات زیر نادرست است؟

۱) سرعت خطی در دیسک نوری یکسان است.

۲) سرعت زاویه‌ای در دیسک نوری یکسان است.

۳) سرعت خطی در هر یک از تراک‌های دیسک مغناطیسی متفاوت است.

۴) سرعت زاویه‌ای در دیسک مغناطیسی یکسان است.

۱۲ ـ زمان دستیابی (access time) کدام مورد زیر است؟

۱) مدت زمانی که داده موردنظر از روی حافظه خوانده می‌شود.

۲) مدت زمانی بین لحظه‌ای که دستور خواندن یا نوشتن داده می‌شود تا لحظه‌ای که داده موردنظر دستیابی می‌شود.

۳) مدت زمانی که داده موردنظر روی حافظه نوشته می‌شود.

۴) مدت زمانی که مکان داده روی حافظه معین می‌شود.

۱۳ ـ در کدامیک از موارد زیر حافظه‌ها بر اساس سرعت مرتب شده‌اند؟

۱) حافظه پنهان ـ ثبات ـ حافظه اصلی ـ دیسک نوری ۲) ثبات ـ حافظه اصلی ـ حافظه پنهان ـ دیسک نوری

۳) حافظه پنهان ـ حافظه اصلی ـ ثبات ـ دیسک نوری ۴) ثبات ـ حافظه پنهان ـ حافظه اصلی ـ دیسک نوری

۱۴ ـ کدام عبارت زیر گپ (GAP) را مشخص می‌کند؟

۱) فضای بین دو کلاستر ۲) فضای بین دو فایل

۳) فضای بین دو گروه (بلاک) رکورد ۴) فضای بین دو پارتیشن

۱۵ ـ کدام عبارت در مورد گپ (GAP) نادرست است؟

۱) از نظر ذخیره‌سازی گپ فضای هرز (Waste) می‌باشد.
۲) گپ متمایزکننده دو بلاک می‌باشد.
✓ ۳) وجود گپ بر روی دیسک بدون استفاده می‌باشد.
۴) گپ جهت حرکت هد موردنیاز می‌باشد.

۱۶ ـ عوامل مؤثر بر افزایش ظرفیت نوار مغناطیسی TAPE کدامند؟

۱) افزایش چگالی
۲) افزایش طول رکورد
۳) کاهش طول گپ
✓ ۴) هر سه مورد

۱۷ ـ اهداف اصلی سیستمهای ذخیره و بازیابی اطلاعات چیست؟

۱) پیاده‌سازی سیستمهای اطلاعاتی
۲) صرفه‌جویی در حافظه
۳) بالابردن سرعت عملیات
✓ ۴) موارد ۲ و ۳

۱۸ ـ تعریف چگالی نوار مغناطیسی (TAPE) چیست؟

۱) تعدادبایتهای ذخیره شده در هر اینچ
۲) تعداد بیتهای ذخیره شده در هر سانتیمتر
۳) تعدادبایتهای ذخیره شده در هر سانتیمتر
✓ ۴) تعداد بیتهای ذخیره شده در هر اینچ

۱۹ ـ فایلی را در نظر بگیرید با 10000 رکورد 80 بایتی که روی نواری با چگالی 1600 بیت در اینچ ذخیره شده است این فایل چه طولی از نوار را اشغال می کند؟

۱) 500 اینچ
۲) 700 اینچ
۳) 1000 اینچ
۴) 1200 اینچ

۲۰ ـ زمان وقفه دورانی (Rotation latency) کدام است؟

۱) زمان‌خواندن اطلاعات یک قطاع از دیسک
۲) زمان یک دور کامل دیسک
۳) مدت‌زمانی است که طول می کشد تا نوک هد به شیار مربوطه برسد.
✓ ۴) مدت‌زمانی است که بعد از قرارگرفتن نوک هد بر روی شیار مربوطه طول می کشد تا اولین قسمت اطلاعات به هد برسد.

۲۱ ـ کدام یک از گزینه‌های زیر عامل اصلی کاهنده نرخ واقعی انتقال در نوار مغناطیسی می‌باشد؟

۱) بلاک
✓ ۲) زمان حرکت ـ توقف
۳) چگالی ـ توقف
۴) گپ

۲۲ ـ یک دیسک چند صفحه‌ای (Pack) با n صفحه دارای چند رویه است و از چند رویه آن برای ذخیره‌سازی استفاده می‌شود؟

۱) n^2 رویه دارد که از همه آن برای ذخیره‌سازی استفاده می‌شود.
۲) $2n$ رویه دارد که از همه‌آن برای ذخیره‌سازی استفاده می‌شود.
✓ ۳) $2n$ رویه دارد که از $2n-2$ رویه آن برای ذخیره‌سازی استفاده می‌شود.
۴) n^2 رویه دارد که n^2-2 برای آن ذخیره‌سازی استفاده می‌شود.

۲۳ ـ گزینه صحیح کدام است؟

۱) زمان یک دور دیسک ≤ زمان درنگ دورانی ≤ 0

۲) زمان درنگ دورانی ≤ زمان یک دور دیسک ≤ 0

۳) زمان استوانه‌جویی ≤ زمان درنگ دورانی ≤ زمان یک دور دیسک

۴) زمان درنگ دورانی < زمان استوانه‌جوئی ≤ زمان درنگ دورانی

۲۴ ـ فایلی را در نظر بگیرید با 10000 رکورد 80 بایتی روی نواری به چگالی 1600bpi ، اگر $B_f = 50$ و طول IBG = 0.5 باشد، میزان واقعی استفاده از نوار کدام است؟

۱) 0.83 ۲) 0.86 ۳) 0.87 ۴) 0.90

۲۵ ـ در کدام نوع از حافظه‌ها تعداد استوانه‌ها برابر یک است؟

(آموزشکده‌های فنی ـ ۸۳)

۱) دیسک پک (DISK PACK) ۲) دیسک مغناطیسی

۳) طبله مغناطیسی (DRUM) ۴) دیسک نوری

فصل دوم

مفاهیم مقدماتی : (سیستم فایل ـ طرح‌ها ـ تکنیک‌ها)

☑ **فیلد (field):** مکان ذخیره‌سازی یک واحد معنائی داده و نامدار را یک فقره اطلاع یا فیلد می گوئیم.

واحد معنائی داده حالت تجزیه‌ناپذیری (Atomicity) دارد. یعنی با تجزیه آن به داده‌هائی می‌رسیم که معنا ندارد.

☑ **رکورد (record):** مجموعه‌ای از فیلدها تشکیل یک رکورد را می‌دهند، البته فیلدهای مربوط به هم در مورد یک فرد، شی یا پدیده که به آن موجودیت (Entity) می گوئیم.

☑ **موجودیت (Entity):** پدیده، فرد یا شی که در مورد آن می‌خواهیم اطلاع کسب کنیم، مانند: کارمند، درس، دانشجو، کارگاه، .....

☑ **محیط عملیاتی:** به محیطی که عملیات پردازش، ذخیره و بازیابی در رابطه باآن انجام می‌شود، مانند: حسابداری سازمان ـ بایگانی یک اداره ـ انباری یک کارخانه

☑ **صفت خاصه:** هر بخش از یک محیط عملیاتی به زیر مجموعه‌ای از صفات خاصه نیاز دارد.

به‌طور مثال: نوع ***موجودیت دانشجو*** در محیط ***عملیاتی دانشگاه*** احتیاج به صفات خاصه نام، نام‌خانوادگی ـ ..... دارد.

اطلاع: هر صفت‌خاصه دارای دو مؤلفه است: اسم صفت خاصه + مقدار صفت خاصه

وقتی که این جفت مؤلفه اسم صفت خاصه و مقدار صفت خاصه در مورد یک موجودیت وجود داشته باشد می گوئیم ***اطلاع*** حاصل شده است.

☑ رکورد از سه دیدگاه مورد بررسی واقع می‌شود:

۱) سطح انتزاعی

۲) سطح برنامه کاربر

۳) سطح ذخیره‌سازی

☑ **کلید (key):** صفت خاصه ساده یا مرکبی است که دو خاصیت زیر را دارا می‌باشد:

۱) در نمونه‌های‌مختلف رکورد در دوره حیات فایل یکتائی (unique) مقدار داشته باشد.

۲) طول آن حتی‌الامکان کوتاه باشد . (minimality)

کلید رکورد درواقع شناسه یک نوع موجودیت است. و به کمک هر مقدار آن یک نمونه از موجودیت از هر نمونه دیگر متمایز می گردد.

☑ **فایل file:** مجموعه‌ای است نامدار از نمونه‌های مختلف یک نوع یا چند نوع رکورد، البته ممکن است فایل مجموعه‌ای از رکوردها نباشد و صرفاً دنباله‌ای ازکاراکترهای بی‌ساختار باشد.

✓ ساختار فایل به دو شکل منطقی و فیزیکی است. در ***ساختار منطقی*** فایل سازمانی است که بر اساس آن رکوردهای منطقی کنار هم گرد آمده‌اند. و چگونگی ارتباطات و پیوندهای بین رکوردهای منطقی را نشان می‌دهد. در ***ساختار فیزیکی*** چگونگی ذخیره‌سازی بلاک‌های فایل در رسانه (دیسک) بررسی می‌شود.

✓ مفهوم فایل در معنای عام دارای سه ویژگی است:

۱) اندازه بزرگی به حدی که به یک‌باره در حافظه درون ماشینی (حافظه اصلی) نمی‌گنجد.
۲) پایایی ـ یعنی داده‌های آن از بین نمی‌روند (مانا هستند) مگر به درخواست کاربر از روی دیسک
۳) اشتراکی بودن بین تعدادی کاربر مجاز

☑ **مکانیسم عینی ذخیره سازی:**

هر چه فایل یک مکانیسم انتزاعی است، اما در نهایت باید عینیت پیدا کند یعنی به صورت یک مکانیسم عینی در محیط فیزیکی ذخیره‌سازی شود. از نظر سیستم فایل فیزیکی، فایل از ***تعدادی بلاک*** تشکیل شده است و ممکن است تقسیمات دیگری مانند باکت ـ خوشه ـ ..... داشته باشد.

نکته‌امهم: در سطح انتزاعی : مفهوم رکورد را مستقل از جنبه‌های مربوط به نمایش آن در محیط منطقی سطح برنامه کاربر و پیاده‌سازی آن در محیط ذخیره‌سازی مطرح می‌کنیم.

به‌طور مثال یک شی در سطح انتزاعی در سطح منطقی دیدکاربر مفهوم رکورد منطقی و در محیط ذخیره‌سازی مفهوم رکورد فیزیکی را دارد.

دانشجو ← نام ـ نام‌خانوادگی ـ ..... ← محیط ذخیره‌سازی فیزیکی

(سطح انتزاعی) (سطح منطقی دیدکاربر) (سطح فیزیکی محیط ذخیره‌سازی)

☑ انواع مختلف موجودیت‌ها توسط صفات خاصه آن‌ها از یکدیگر متمایز می‌شوند. هر نوع موجودیت دارای مجموعه‌ای از صفات خاصه است که باید با توجه به نیازهای اطلاعاتی محیط عملیاتی از بین آن‌ها انتخاب کرد.

☑ **نکته۲: (مهم)**

رکورد از دید برنامه‌ساز:

رکورد از دیدکاربر برنامه‌ساز را رکورد منطقی می‌گوئیم، مجموعه‌ای است دارای ساختار مشخص مبتنی بر طرح خاص و دارای نام با تعدادی فیلد مشخص.

☑ **طرح رکورد نشان‌دهندهٔ *نوع رکورد*:** دو طرح برای ساختار رکورد وجود دارد:

۱) طرح با قالب ثابت مکان

۲) طرح با قالب غیرثابت مکان

۱)طرح با قالب ثابت مکان:

* در این طرح در هر فیلد فقط مقدار صفت خاصه ذخیره می‌شود. این که محتوای هر فیلد مربوط به کدام صفت خاصه است. در تعریف ساختار رکورد و به کمک مکان فیلد در قالب از پیش ثابت مشخص می‌شود بنابراین مکان یک فیلد در نمونه‌های مختلف رکورد ثابت است.

در این طرح معمولاً تعداد فیلدها و طول هر فیلد در تمام نمونه‌های متعلق رکورد ثابت است اما گاهاً ممکن است متغیر باشد.

* احتیاجی به ذخیره‌سازی اسم صفت خاصه نیست.

۲) طرح با قالب غیر ثابت مکان:

در این طرح مکان یک فیلد در نمونه‌های مختلف رکورد از پیش مشخص و ثابت نیست. بنابراین در هر فیلد از هر نمونه رکورد هم اسم صفت خاصه و هم مقدار آن ذخیره می‌شود.

محل فیلدها در این طرح در نمونه‌های مختلف رکورد ثابت نیست. $A_1 = V_1,\ A_2 = V_2,\ A_3 = V_3$ / $A_1 = V_2,\ A_3 = V_3,\ A_4 = V_4$

* در عین حال در این طرح ***تعداد فیلدها*** و ***طول آن‌ها*** و ***مکان آن‌ها*** در نمونه‌های مختلف رکورد یکسان نیست. البته ممکن است در مواردی طول ثابت نیز داشته باشد.

* **نتیجه‌گیری** رکورد منطقی از دید برنامه‌ساز از نظر طول بر دو نوع است:

۱) رکورد با طول ثابت

۲) رکورد با طول متغیر

در هر دو قالب ***ثابت مکانی*** و ***غیرثابت مکانی*** ممکن است رکورد با طول ثابت و متغیر وجود داشته باشد.

☑ **دلایل متغیرشدن طول رکورد : (مهم)**

الف) طول مقدیر یک یا بیش از یک صفت خاصه (فیلد) متغیر باشد

مانند: نام و نام‌خانوادگی ـ آدرس

ب) تعداد صفت خاصه مورد نیاز برای نمونه‌های مختلف یک نوع موجودیت متفاوت باشد.

ـ به ویژه زمانی که نمونه‌ها قابل رده‌بندی باشند.

مانند: کارمند قراردادی ـ رسمی در یک شرکت.

وام ـ مالیات ـ بیمه ـ شماره کارمندی ـ نام خانوادگی ـ نام : کارمند رسمی

ـ مالیات ـ شماره کارمندی ـ نام خانوادگی ـ نام: کارمند قرار دادی

ج) پدیده فقرهاطلاع تکرار شونده یا گروه اطلاع تکرار شونده

- به عبارت بهتر صفت خاصه ***چند مقداری*** داشته باشیم.

✓ **فقره اطلاع تکرار شونده:** یعنی به ازاء یک اسم صفت خاصه بیش از یک مقدار در نمونه‌های مختلف از یک نوع رکورد وجود داشته باشد. گاهاً به آن صفت چند مقداری می گوییم.

نام	نام‌خانوادگی	درس
علی	زارع	فیزیک - ریاضی - شیمی
پویا	احمدی	ریاضی - فیزیک

صفت خاصه ساده چند مقداری

✓ همانطور که دیده می‌شود وجود فقره اطلاع تکرار شونده فایل را ***به طور منطقی نامسطح*** می کند. ساده‌ترین راه برای مسطح کردن در نظر گرفتن قالب خطی برای رکورد است. که در آن بر اساس مقادیر صفت خاصه، فیلد در نظر گرفته شود.

نام	نام‌خانوادگی	درس۱	درس۲	درس۳
علی	زارع	شیمی	ریاضی	فیزیک
پویا	احمدی	فیزیک	ریاضی	

فایل مسطح شده

- گروه اطلاع تکرار شونده از چند فقره اطلاع تکرار شونده تشکیل می‌شود. که در فیلدهای مرکب چند مقداری در نظر گرفته می‌شود.(البته صفت خاصه مرکب لزوماً تکرار شونده نیست)

اگر تعداد دفعات تکرار در نمونه رکوردها یکسان باشد آن‌گاه طول رکورد ثابت می‌ماند.

☑ **۳ـ رکورد ذخیره شده (در محیط ذخیره‌سازی) (سطح فیزیکی) : (مهم)**

رکورد در سطح فیزیکی علاوه بر داده‌هائی که با استفاده از رکورد در سطح منطقی ذخیره می کند، بخش دیگری را نیز ذخیره می کند که به آن بخش کنترلی - پیشوندی - غیرداده‌ای یا metasection می گوئیم. بخش کنترلی یا پیشوندی در محیط ذخیره‌سازی در نظر گرفته می‌شود. و از دید کاربر کاملاً پوشیده است.

بخش داده‌ای	بخش غیرداده‌ای

رکورد در سطح فیزیکی

نکته مهم: بخش غیرداده‌ای حاوی اطلاعاتی است که سیستم فایل برای پردازش رکورد به آن‌ها نیاز دارد. این بخش معمولاً از تعدادی فیلد تشکیل شده است شامل:

۱ـ طول رکورد
۲ـ نوع رکورد
۳ـ نشانه‌روها (اشاره گرها - نشان‌نماها)
۴ـ فلاگ‌های عملیاتی
۵ ـ فلاگ‌های حفاظتی
۶ـ اطلاعاتی خاص در بعضی از ساختار

۱ـ طول رکورد: وقتی که رکوردها با طول متغیر است. در نظر گرفتن فیلدی برای درج طول، یکی از تکنیک‌های مشخص کردن محدوده رکورد است. در مورد رکوردها با طول ثابت احتیاجی به این فیلد نیست.

۲ـ نوع رکورد: در فایل‌هائی که بیش از یک نوع رکورد ذخیره می‌شود، فیلدی برای درج کد نوع در نظر گرفته می‌شود.

فایل تک نوعی: فایلی که یک نمونه رکورد ذخیره می‌کند. مثال: (فقط دانشجو)

فایل چند نوعی: فایلی که چند نمونه رکورد ذخیره می‌کند. مثال: (دانشجو ـ استاد ـ درس)

۳ـ نشانه‌روها (اشاره‌گرها ـ نشان‌نماها): نشانه‌رو اساساً آدرسی است که در فیلد جای داده می‌شود و از نقطه‌ای از فایل مکان "داده‌ای" را در نقطه دیگر مشخص می‌کند. برای پیاده‌سازی ارتباط ساختاری میان رکوردها (ساختار منطقی) و ایجاد ساختار فیزیکی از نشانه‌روها استفاده می‌شود.

✓ نشانه‌رو یک مبدأ و یک مقصد دارد، از این نقطه نظر نشانه‌روها حالت‌های زیر را دارند:

۱) رکورد به رکورد

۲) رکورد به بلاک

۳) بلاک به بلاک

۴) بلاک به رکورد

۵) گروهی بلاک به گروهی دیگر

۶) فایل به فایل

از نشانه‌رو برای نشان دادن رکورد بعدی، قبلی و یا سرزنجیره رکوردها نیز استفاده می‌شود.

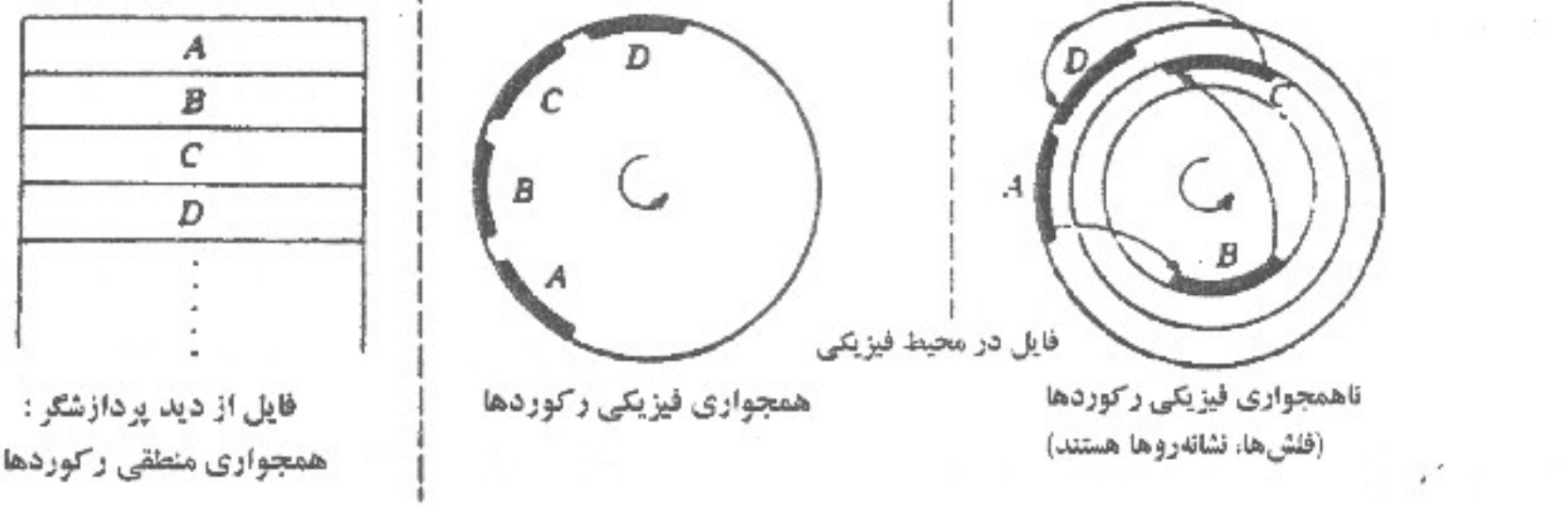
✓ **نشانه‌روها از نظر نوع نشان‌دهی بر سه گونه است:**

۱) **نشان‌دهی در سطح فیزیکی:** که سیستم تولید می‌کند.

۲) **نشان‌دهی نسبی:** این نشان‌دهی در نهایت به نشان‌دهی فیزیکی می‌رسد.

۳) **شناسه رکورد:** که نوعی نشانه‌رو ضمنی به رکوردها است.

نکته مهم: ساختار منطقی فایل ساختاری است که ارتباط بین رکوردها را از دید برنامه فایل پردازشگر نشان می‌دهد. به‌طور مثال: نظم بر اساس شماره کارمندی در فایل کارمندان ترتیب قرارگرفتن رکوردهای کارمندان را مشخص می‌کند. اما رکوردهای منطقی همجوار در این ساختار ممکن است به‌طور فیزیکی همجوار نباشند. که در این حالت از نشانه‌روها برای برقراری همجواری منطقی استفاده می‌شود.



۴ـ **فلاگ‌های عملیاتی:** فلاگ‌های عملیاتی به دو منظور ایجاد می‌شود:

۱) عملی که قرار است روی رکورد انجام شود. مانند حذف

۲) عملی که روی رکورد انجام شده است. مانند بهنگام‌سازی

* این فلاگ‌ها بیشتر در فایل‌های اشتراکی استفاده می‌شود.

* این فلاگ‌ها در بعضی مواقع، خود به عنوان یک رکورد است و در فایل جداگانه‌ای ذخیره می‌شود. (مانند: log فایل‌ها در پایگاه داده‌ها)

* بهنگام‌سازی می‌تواند شامل تغییرات ـ ایجاد ـ یا درج باشد.

۵ ـ **فلاگ‌های حفاظتی:** فلاگ حفاظتی برای این در نظر گرفته می‌شود تا رکورد از دستیابی غیرمجاز مصون بماند. برای همین به آن فلاگ قفل در سطح رکورد نیز می‌گویند. به‌طور کلی حق دستیابی برای کاربران در دو سطح ***خواندن*** و ***نوشتن*** است.

* در حالت نوشتن باید مشخص شود که به چه منظوری انجام می‌شود.

۱) برای درج رکورد جدید

۲) برای حذف رکورد

۳) برای بهنگام‌سازی رکورد

که هر کدام به طور جداگانه می‌توانند انتخاب شوند.

* این فلاگ هم در فایل‌های اشتراکی نیز استفاده می‌شود.

* سطوح حفاظت از نقطه نظر فلاگ حفاظت در سطح‌های مختلفی انجام می‌شود:

۱ـ سطح رکورد

۲ـ سطح بلاک

۳ـ سطح گروهی از بلاک‌ها

۴ـ سطح فایل

۵ـ سطح گروهی از فایل‌ها

۶ـ در بعضی موافع حتی در سطح فیلد

۶ـ **اطلاعات خاص:** در بعضی ساختارهای فایل گاهی احتیاج به نگهداری اطلاعاتی در مورد ساختار هستیم که از فیلد اطلاعات خاص استفاده می‌شود.

بلاک بندی: (مهم)

بلاک قالبی است با ساختارمشخص و شامل تعدادی رکورد، جای دادن چندین رکورد در این قالب بزرگتر را ***بلاک‌بندی*** می گوئیم.

بلاک را ***صفحه*** (page) ـ ***چارچوب*** (Frame) و***حفره*** (slot) نیز ***می‌گویند.***

* B_f (**ضریب بلاک‌بندی**):تعداد رکوردهای درون بلاک را ضریب بلاک‌بندی می گویند.

* مقدار B_f (**ضریب بلاک‌بندی**): روی تعداد بلاک‌های فایل‌ها ـ زمان کل عملیات ورودی ـ خروجی برای پردازش تمام رکوردهای منطقی تأثیر می گذارد. و همچنین تاثیرمستقیمی در استفاده بهینه از فضای دیسک دارد.

* از ***دید برنامه پردازشگر (منطقی)*** فایل مجموعه‌ای از رکوردهاست ولی از نظر ***سیستم فایل (فیزیکی)*** فایل از تعدادی بلاک تشکیل یافته است.

نکته مهم: بلاک کمترین مقدار داده‌ای است که در یک عمل ورودی/ خروجی توسط سیستم فایل بین بیرون و درون ماشین مبادله می‌شود. و به عبارت بهتر واحد عملیات خواندن و نوشتن است.

البته ممکن است در حین انجام عمل ورودی/خروجی چندین بلاک رد و بدل شود.

* واحد رد و بدل اطلاعاتبین ***حافظه اصلی*** و ***حافظه جانبی بلاک*** است.

* در اکثر اوقات محاسبه B_f به‌صورت زیر انجام می گیرد:

$$B_f = \left\lfloor \frac{B}{R} \right\rfloor$$

☑ انتخاب مناسب B_f باعث‌صرفه‌جویی در استفاده از حافظه و بالابردن سرعت عملیات می‌شود.

* بین بلاک‌ها، IBG یا همان GaP وجود دارد. (برای سرعت حس و توقف)

تعیین محدودهٔ رکوردهای بلاک‌بندی شده: (مهم)

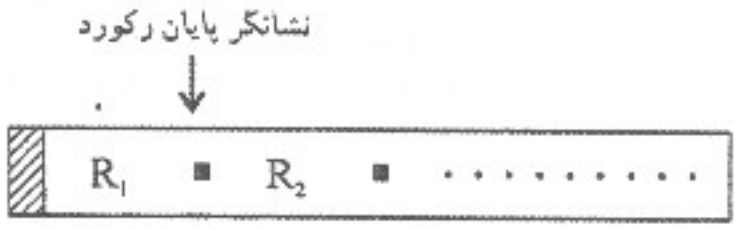
بعد از بلاک‌بندی رکوردهابایدبا تکنیکی محدوده رکوردها را در بلاک مشخص کرد. که دو وضعیت پیش می‌آید.

*۱) **رکوردها با طول ثابت**:نیاز به تکنیک خاصی نیست کافی است یک‌بار طول رکورد در "راهنمای فایل" درج شود.

* ۲) **رکوردها با طول متغیر:** سه روش کلی وجود دارد:

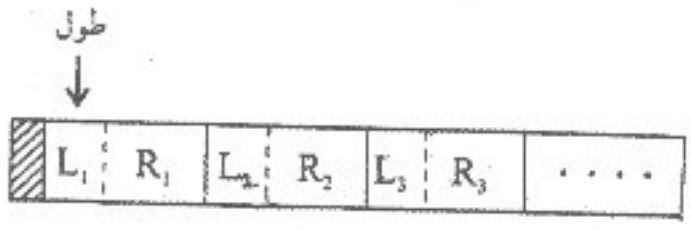
الف) درج نشانگر پایان رکورد.

نشانگر پایان رکورد در انتهای رکورد درج می‌شود ←



ب) درج طول در بخش غیرداده‌ای رکورد.

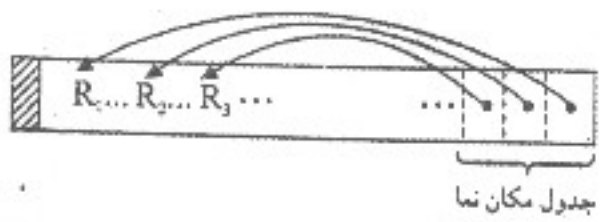
برای هر رکورد یک فیلد طول در نظر می‌گیریم ←



ج)ایجاد جدول مکان‌نما.

آدرس نسبی رکورد در بلاک نسبت به ←

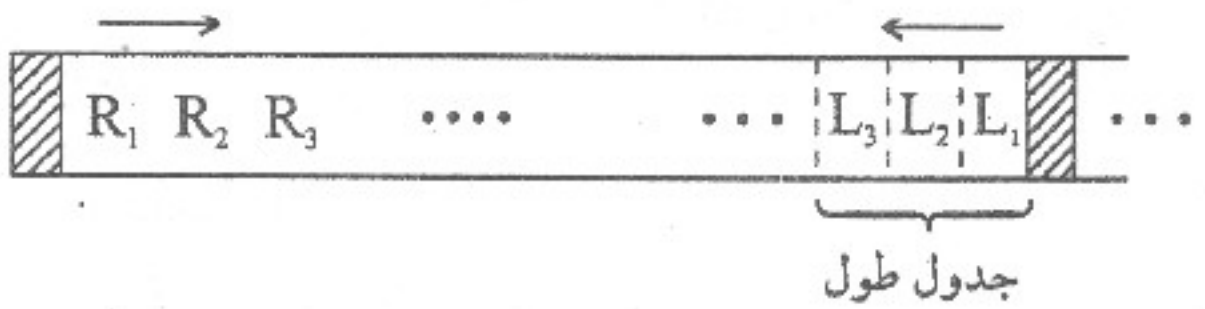
آغاز بلاک در جدول درج می‌شود.

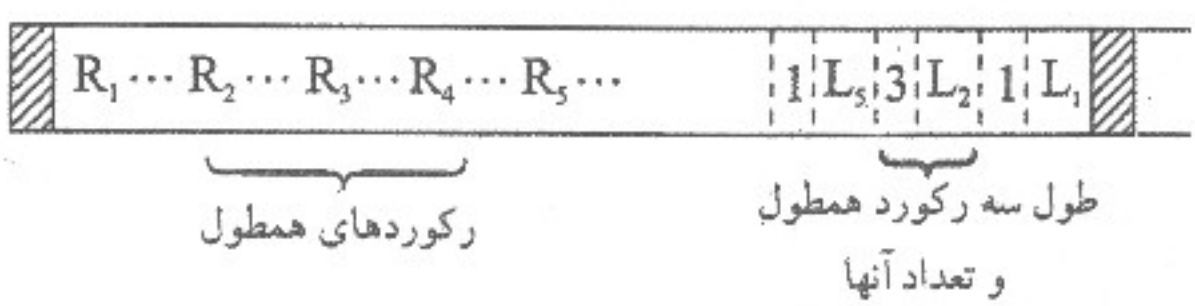


تکنیک‌های تعیین محدوده رکورد

* در تکنیک جدول مکان‌نما با استفاده از جدول مکان‌نما برای هر رکورد تخصیص یا اشاره‌گر به‌صورت پویا انجام شده و از فضای بلاک بهتر استفاده می‌شود.

* در بعضی سیستم‌ها ترکیبی از تکنیک‌های فوق نیز ممکن است دیده شود؛ به‌طور مثال یک روش استفاده از جدول طول با استفاده از روش ب و ج است. با این تفاوت که در مدخل هر رکورد به جای اشاره‌گر، طول رکورد قرار داده می‌شود. بنابراین برای رکوردهای هم طول نیز می‌توان صرفه‌جوئی کرد.





تکنیک‌های بلاک‌بندی : (مهم)

به طور کلی بلاک‌بندی رکوردها در ۳ تکنیک خلاصه می‌شود.

۱) بلاک‌بندی رکوردها با طول ثابت و یکپاره و گاهاً دوپاره

۲) بلاک‌بندی رکوردها با طول متغیر و دوپاره

۳) بلاک‌بندی رکوردها با طول متغیر و یکپاره

* در هر یک از ۳ تکنیک میزان حافظهٔ هرزی وجود دارد که در هر کدام با دیگری متفاوت است.

* در محاسبات حافظه هرز علائم زیر به صورت عمومی وجود دارد:

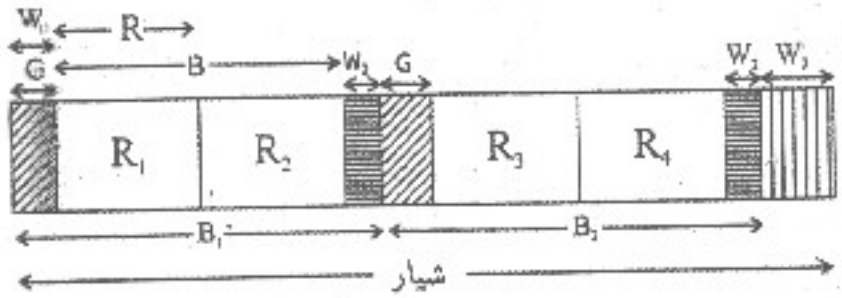
حافظه هرز به ازاء یک رکورد = W_2

تعداد بلاک‌های داخل هر شیار = T_f = (فاکتور تراکینگ)

تعداد رکوردهای هر بلاک = B_f = (فاکتور بلاک‌بندی)

بین بلاک‌ها G = Gap

۱) تکنیک بلاک‌بندی رکوردها با طول ثابت به‌صورت یکپاره:



طول رکوردها در این روش ثابت = R

در این تکنیک ۳ نوع حافظه هرز داریم:

۱) $W_1 = G$: گپ به طول G بایت بین بلاک‌ها (IBG)

۲) W_2 : حافظه هرز ناشی از جانشدن رکورد دیگر در بلاک (***که ممکن است با انتخاب مناسب طول بلاک و طول رکورد 0 شود.***) $0 \le W_2 \le R-1$

۳) W_3 : حافظه هرز ناشی از جانشدن بلاک دیگر در شیار

$$B_f = \left\lfloor \frac{B}{R} \right\rfloor$$

W_2 = به‌طور متوسط نصف طول رکورد ثابت

$$W_B = W_1 + W_2 + \frac{W_3}{T_F'} = G + \frac{R}{2} + \frac{W_3}{T_f}$$

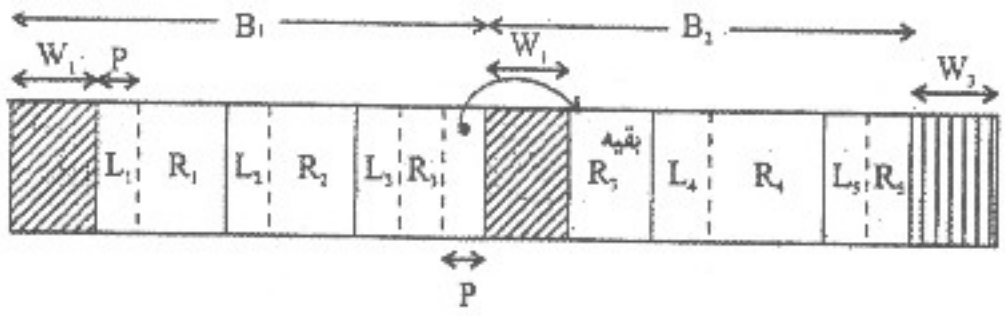
$$W_R = \frac{W_B}{B_f} = \frac{1}{B_f}\left(G + \frac{R}{2} + \frac{W_3}{T_f}\right)$$

۲) تکنیک بلاک‌بندی رکوردهای با طول متغیر و دوپاره:

در این تکنیک در صورتیکه R_3 نتواند در بلاک اول جای گیرد آن‌را به‌صورت دوپاره بوسیله یک اشاره‌گر در دو بلاک همجوار ذخیره می‌کنیم.

P= طول فیلد طول و یا طول فیلد اشاره‌گر

R = متوسط اندازه هر رکورد (طول)



بلاک‌بندی رکوردهای با طول متغیر و دوپاره

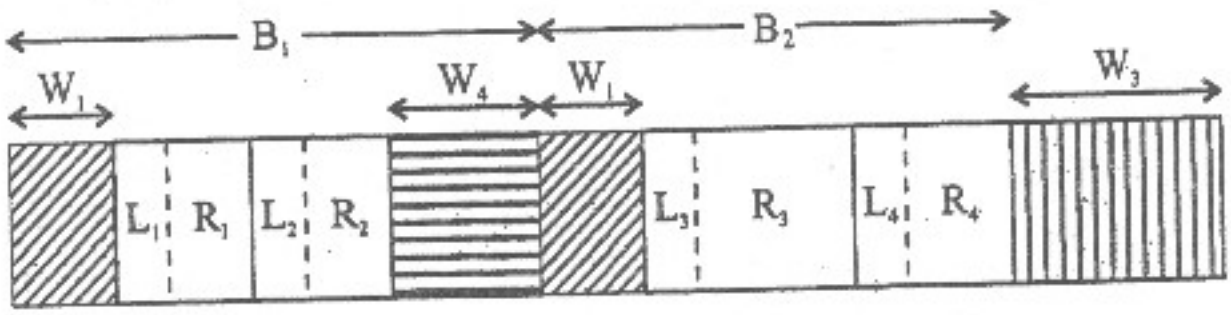
$$B_f = \frac{B-P}{R+P}$$

$$W_B = G + \underbrace{P}_{\text{فضای هرز ناشی از فیلد طول رکوردها}} + \underbrace{B_f \times P}_{\text{فضای هرز اشاره‌گر}} + \frac{W_3}{T_f}$$

$$W_R = \frac{W_3}{B_f}$$

۳) تکنیک بلاک‌بندی رکوردهای با طول متغیر و یکپاره:

در این تکنیک یک حافظه هرز ناشی از جانشدن R_3 در بلاک اول ایجاد می‌شود که به آن W_4 می‌گوئیم.



بلاک‌بندی رکوردهای با طول متغیر و یکپاره

R = متوسط اندازه رکورد P = طول فیلد طول

$$B_f = \frac{B-\frac{R}{2}}{R+P}$$

$$W_B = G + \underbrace{B_f * P}_{\text{فضای هرز ناشی از فیلد طول رکوردها}} + \underbrace{\frac{R}{2}}_{\text{متوسط } w_4} + \frac{W_3}{T_f}$$

$$W_R = \frac{W_B}{B_f}$$

☑ **مقایسه تکنیک‌های بلاک‌بندی:** (مهم)

۱ـ تکنیک اول به‌سادگی پیاده‌سازی می‌شود. اما در عوض انعطاف‌پذیری ندارد. در صورت تغییر طول رکورد چاره‌ای جز تعریف و ایجاد مجددفایل نیست. ***تغییر طول رکورد*** در مدت حیات فایل یکی از ***جنبه‌های رشد فایل*** است. و همچنین افزایش ***تعداد نمونه‌های رکورد*** باعث***افزایش اندازه فایل*** می‌شود.

۲ـ تکنیک‌های دوم و سوم نسبت به اولی انعطاف‌پذیرترند. ولی نرم‌افزار پیچیده‌ای را در سیستم درخواست می کنند.

۳ـ درمقایسه تکنیک‌های دوم و سوم ***تکنیک دوم*** از نظر حافظه مقرون به صرفه‌تر است. (منظور میزان حافظه هرز کمتر)

۴ـ تکنیک دوم نسبت به تکنیک سوم دارای نرم‌افزار پیچیده‌تری است. چون نیاز به مدیریت اشاره‌گرها برای وضعیت دوپاره رکوردها دارد.

۵ـ درتکنیک سوم این مشکل مطرح است که حداکثر اندازه یک رکورد به علت یک پاره بودن تکنیک فقط به اندازهٔ طول بلاک است اما در تکنیک دوم چنین محدودیتی نداریم و انعطاف‌پذیری داریم.

۶ـ ***درتکنیک سوم*** به علت وجود حافظه‌های هرز ناشی از جانشدن آخرین رکورد بلاک ***طول فایل افزایش پیدا کرده*** بنابراین ***زمان خواندن کل فایل بیشتر*** می‌شود که در ***تکنیک دوم*** این زمان به علت دوپاره بودن رکوردها و نبودن این حافظه ***هرز کمتر*** است.

۷ـ درتکنیک دوم خواندن رکورد دوپاره زمان بیشتری می‌خواهد. زیرا سیستم باید دوبلاک را بخواند. در واقع اگر اندازهٔ بافر همان اندازهٔ بلاک‌باشد. واکشی رکورد دوپاره به دوبار عمل خواندن / نوشتن احتیاج دارد. که در تکنیک سوم این زمان کمتر است. (چون رکورد دوپاره ندارد) البته می‌توانیم با استفاده از باکت‌بندی (باکت = مجموعه‌ای از چند بلاک) این مشکل تکنیک دوم را حل کنیم. با خواندن دو بلاک در یک بافر این مشکل حل می‌شود.

✱ درتکنیک دوم بهینه این است که یک رکورد بر دو بلاک همجوار حتماً دوپاره ذخیره شود زیرا ممکن است در طول حیات، سیستم فایل رشدکند که در این حالت با شیفت درون بلاکی برای قسمت اضافه شده مکان اختیار می کنیم.

☑ **مزایا و معایب بلاک‌بندی:** (مهم)

✓ **مزایا:**

۱) **کاهش دفعات ورودی/خروجی (صرفه‌جوئی در زمان):** و کاهش زمان اجرای برنامه فایل پرداز (البته این زمان به بافرینگ و زمان پردازش محتوی بلاک نیز بستگی دارد.)

۲) **صرفه‌جوئی در مصرف رسانه ذخیره‌سازی از طریق کاهش گپ‌ها:**

(به جای گپ بین رکوردها (IRG) از گپ بین بلاک‌ها (IBG) استفاده می‌شود.)

بنابراین اندازه فایل کمتر و زمان پردازش فایل کمتر می‌شود.

✓ **معایب:**

۱) ***کار نرم افزاری بیشتر*** برای عملیات بلاک‌بندی و بلاک گشائی (عکس عمل بلاک‌بندی) که در بلاک گشائی رکوردهای منطقی در اختیار کاربر قرار می گیرد.

۲) ***مصرف حافظه بیشتر (حافظه اصلی)*** به خاطر لزوم بافرینگ

۳) ***بالارفتن احتمال اشتباه در مبادله اطلاعات*** به خاطر افزایش مقدار داده‌ای که منتقل می‌شود.

نکته مهم:

✓ **باکت:** مجموعه‌ای از تعدادی بلاک که می‌تواند طی یک دستور واحد خواندن به بافر منتقل شود. باکت در حالت خاص می‌تواند تک بلاکی باشد.

● باکت‌های فایل معمولاً در محیط فیزیکی ذخیره‌سازی همجوارند. اما گاه ممکن است چنین نباشد بلاک‌ها درون باکت هم معمولاً همجوارند اما گاه ممکن است ناهمجوار باشند.

● در بعضی از سیستم‌ها به باکت Partition-segment-train نیز گفته می‌شود.

✓ **خوشه** (Cluster) : گاه به تعدادی بلاک همجوار یا سکتور همجوار خوشه گفته می‌شود. تعداد بلاک در خوشه را اندازه خوشه می گویند.

✓ به مجموعه‌ای ***از شیارهای درون یک استوانه*** و یا ***تعدادی استوانه همجوار گسترش*** می گویند. بنابراین ***گسترش*** در نهایت تعدادی بلاک است و می‌توان آن‌را مترادف ***مفهوم باکت*** دانست.

☑ **موضعی بودن رکوردها (لوکالیتی): (مهم)**

تعریف: میزان همسایگی (نزدیکی) فیزیکی رکوردهای منطقاً همجوار را لوکالیتی رکوردها می گوئیم.

* اگر دو رکورد R_1 و R_2 منطقاً همجوار باشند و R_1 واکشی شده باشد. در این صورت R_1 را رکورد فعلی و R_2 را رکورد منطقاً بعدی R_1 می گوئیم.

* باید توجه کنیم که همجواری منطقی رکوردها براساس نظمی است که به‌صورت منطقی توسط برنامه و توسط برنامه‌نویس تعیین شده است. (مثلاً نظم صعودی افراد یک شرکت بر اساس شماره دانشجوئی) ***همجواری منطقی لزوماً در محیط فیزیکی پیاده‌سازی نمی‌شود***. بلکه بستگی به ساختار فایل و نحوه تخصیص فضای ذخیره‌سازی به فایل دارد.

* تنها در فایل ***ترتیبی کلیدی فیزیکی*** می‌توانیم هر دو ***همجواری منطقی و فیزیکی*** را آن هم در ***لود اولیه*** تأمین کنیم.

* در مفهوم لوکالیتی پارامتر زمان پنهان است و اساساً هنگامی تأثیر زمانی دارد که پردازش سریال رکوردها (بر اساس نظم منطقی خاصی) مدنظر باشد.

* ***در صورتیکه همجواری فیزیکی رکوردهائی که منطقاً همجوارند برقرار نباشد لوکالیتی ضعیفتر شده*** و زمان خواندن سریال فایل نوک خواندن/نوشتن دائماً در حالت حرکت بین استوانه‌ها است.

* مهم:

خاصیت ***موضعی بودن (لوکالیتی)*** گاه با اصطلاح ***خوشه‌ای بودن*** بیان می‌شود. که بر دو نوع است.

۱) **خوشه‌ای بودن درون فایل:** موضعی بودن رکوردهای درون فایل

۲) **خوشه‌ای بودن بین فایل‌ها:** موضعی بودن رکوردهای دو فایل یا بیشتر که منطقاً به یکدیگر مرتبط هستند. یعنی رکورد منطقاً بعدی ممکن است در سفایل دیگری باشد.

☑ **درجات لوکالیتی:** (مهم)

● میزان نزدیکی فیزیکی رکوردهای منطقاً همجوار (لوکالیتی) دارای درجاتی است. البته مطلوب این است که همجواری منطقی با همجواری فیزیکی پیاده‌سازی شود. (هنگامی که پردازش سریال فایل موردنظر باشد) ***هر چه لوکالیتی رکوردها قوی‌تر باشد، زمان پردازش سریال آن‌ها کمتر می‌شود.***

● لوکالیتی در طیفی از قوی به ضعیف را به صورت زیر ارزیابی می کنیم:

۱ـ «رکورد بعدی» درهمان بلاکی باشد که رکورد فعلی است و بلاک در بافر باشد پس در این وضعیت : I/O نداریم ـ $r=0$ و $s=0$ ***کمترین زمان را برای بدست آوردن رکورد بعدی داریم.***

↓ I/O داریم

۲ـ «رکورد بعدی» در بلاک بلافاصله بعدی، بلاک حاوی رکورد فعلی باشد از همان استوانه در این وضعیت : I/O داریم ـ $r=0$ و $s=0$.

↓ $r>0$ داریم

۳ـ «رکورد بعدی» در همان استوانه باشد که رکورد فعلی است در این وضعیت : I/O داریم ـ $r>0$ و $s=0$

۴ـ «رکورد بعدی» روی استوانه هم‌شماره باشد از دیسکی دیگر یعنی فایل توزیع شده روی چند دیسک اما بر استوانه‌های هم شمارهٔ در این وضعیت: **I/O داریم** ـ $r>0$ ـ $s=0$.

↓ $s>0$ داریم

۵ـ «رکورد بعدی» در استوانه همجوار است. در این وضعیت : I/O داریم ـ $r>0$ ـ $s>0$

۶ـ رکورد بعدی در یک استوانه شناخته شده است. یعنی در پردازش رکورد فعلی مشخص می‌شود که رکورد بعدی در کدام استوانه است. به عبارت بهتر آدرس رکورد بعدی از رکورد فعلی بدست می‌آید. **در این وضعیت:** I/O داریم ـ $r>0$ ـ $s>0$

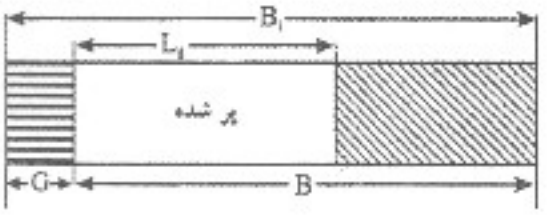
۷ـ رکورد بعدی روی استوانه ناشناخته‌است ولی آدرس آن با انجام محاسبات بدست می‌آید. در این وضعیت :I/O داریم ـ $r>0$ ـ $s>0$

۸ـ **رکورد بعدی:** روی استوانه ناشناخته است و آدرس آن با مراجعه به فایل دیگر بدست می‌آید. **در این وضعیت:** I/O داریم ـ $r>0$ ـ $s>0$

۹ـ رکورد بعدی: روی رسانه‌ای است که در حال حاضر در drive نیست. **در این وضعیت:** I/O داریم ـ $r>0$ ـ $s>0$

☑ **چگالی لود اولیه**(loading Density) : (مهم)

در صورتی که بتوانیم حجم عملیات ذخیره‌سازی (مانند عمل درج) در فایل را بعد از لود اولیه در ابتدا تخمین بزنیم می‌توانیم در هر بلاک مقداری فضای رزرو پیش‌بینی کنیم. ***یعنی در لود اولیه تمام فضای بلاک را پر نکنیم تا از این فضای رزرو برای انجام عملیات ذخیره‌سازی بعداً استفاده کنیم.***



چگالی لود اولیه

نکته مهم: چگالی لود اولیه به صورت درصد از اندازه بلاک بیان می‌شود واین درصد با داشتن نسبت $\frac{L_d}{B} < 1$ بدست می‌آید. در حالت چگالی کمتر از صددرصد حافظه هرز پدید آمده نیز باید دربرآورد میزان استفاده واقعی حافظه منظور گردد.

- **مزایا:**

۱) ایجاد ناحیه رزرو سبب می‌شود لوکالیتی رکوردهای فایل بهتر حفظ شود. زیرا از پراکندگی نشست رکوردها روی حافظه جانبی تا حدی جلوگیری می‌کند. چون میزان پراکندگی فایل در زمان دستیابی تصادفی و همچنین پردازش سریال فایل نقش مهمی دارد.

۲) ایجاد ناحیه رزرو انجام بعضی عملیات روی فایل را تسهیل و تسریع می‌کند. مثلاً هنگام عمل درج با استفاده از شیفت درون بلاکی رکورد دلخواه درج می‌شود.

- **معایب:**

۱) وجود ناحیه رزرو نوعی حافظه هرز است و سبب افزایش اندازه فایل شده و ***زمان خواندن کل فایل*** زیاد می‌شود.

۲) اگر توزیع درج رکوردها در بلاک‌ها یکنواخت نباشد در بعضی بلاک‌ها در انتهای بلاک حافظه هرز پیدا می‌شود.

نکته: همیشه ممکن است، لوکالیتی رکوردهای فایل، در دوره حیات فایل به علت عملیات محیط فیزیکی (درج ـ فیزیکی ـ بهنگام‌سازی) به تدریج از قوی به ضعیف میل می‌کند، که در این حالت فایل با سازماندهی مجدد درست می‌شود.

فایل در محیط فیزیکی:

چگونگی نشست فایل در محیط فیزیکی به چگونگی تخصیص فضا به فایل بستگی دارد، در هر صورت بلاک‌های هر فایل باید در بلاک‌هائی از دیسک قرار داده شود.

انواع نشست فایل در محیط فیزیکی:

۱) نشست پیوسته

۲) نشست گسسته

۱ـ نشست پیوسته:

در این نشست هر فایل در بلاک‌های فیزیکی همجوار به‌صورت پیوسته روی دیسک ذخیره می‌شود.

مثال: در یک دیسک با بلاک‌های 1 کیلوبایتی، یک فایل 50 کیلو بایتی، 50 بلاک به‌هم پیوسته را اشغال می‌کند.

■ **مزایا:**

۱ـ پیاده‌سازی آسان: با داشتن آدرس اولین بلاک روی دیسک می‌توان به بقیه بلاک‌ها دسترسی پیدا کرد.

۲ـ کارائی بالا: کل فایل را می‌توان با یک عمل خواندن از روی دیسک خواند.

■ **معایب:**

۱ـ حداکثر اندازه فایل باید در مرحله ایجاد فایل، معلوم باشد.

۲ـ پدیده بندبندشدگی در فضای دیسک به وجود می‌آید، که این به علت آن است که فضای پیوسته بزرگ اشغال شده و فضای خالی کوچک به صورت گسسته به وجود می‌آید.

* پدیده بند بند شدگی را می‌توان با تکنیک یکپارچه‌سازی یا فشرده‌سازی از بین برد.

۲ـ نشست ناپیوسته:

در این نشست، هر فایل روی تعدادی بلاک غیر همجوار در سطح دیسک ذخیره می‌شود.

■ انوع روش‌ها در نشست ناپیوسته:

۱) ایجاد لیست پیوندی

۲) ایجاد لیست پیوندی با جدول راهنما

۳) تکنیک گرهٔ I $\left(\underbrace{\text{Index node}}_{\text{I-node}}\right)$

۱ـ ایجاد لیست پیوندی:

در این روش بلاک‌های فایل براساس ترتیب منطقی آن‌ها به یکدیگر پیوند می‌شوند (به عبارت بهتر بلاک‌های غیر همجوار با اشاره‌گرهائی به هم پیوند می‌شوند)

مزایا:

الف) پدیده بند بند شدگی بوجود نمی‌آید.

ب) بلاک‌های فایل به آسانی پیدا می‌شوند. (سیستم آدرس اولین بلاک را داشته و بقیه بلاک‌ها با پیمایش زنجیرهٔ پیوند بلاک‌ها پیدا می‌شوند)

ج) خواندن پی‌درپی فایل آسان است.

معایب:

الف) دستیابی تصادفی به رکوردها کُند است. (چون باید از ابتدای لیست پیوندی پیمایش را انجام داد و در بدترین حالت رکورد مورد نظر انتهای لیست است)

۲ـ ایجاد لیست پیوندی با جدول راهنما:

در این روش جدولی در حافظه اصلی (RAM) ایجاد می‌شود و برای هر بلاک فیزیکی دیسک یک مدخل در این جدول نگهداری می‌شود.

نحوه کار: در این روش با مشخص شدن اولین بلاک (مدخل) فایل در مدخل هر بلاک فایل شماره بلاک بعدی قرار داده می‌شود.

به طور مثال: اگر فایل F1 از بلاک شماره 4 شروع شود و بلاک‌های بعدی آن 7 , 9 , 12 , 15 باشد در مدخل بلاک 4، شماره 7 و در مدخل بلاک 7 شماره 9 و ... قرار داده می‌شود.

0		
1		
2		
3		
4	7	← بلاک اول فایل F1
5		
6		
7	9	
8		
9	12	
10		
11		
12	15	
13		
14		
15	EOF	
⋮		
	جدول راهنما	

مزایا:

الف) دستیابی تصادفی سریعتر خواهد بود. (به علت وجود جدول راهنما در حافظه اصلی)

ب) با داشتن آدرس اولین بلاک می‌توان به بلاک‌های دیگر هم دستیابی پیدا کرد.

ج) خواندن پی‌درپی فایل آسان است.

معایب:

الف) کمبود حافظه اصلی (تمام جدول راالزما باید در حافظه اصلی همیشه ماندگار (Resident) باشد.

نکته: کاربرد این روش در محیط سیستم‌عامل MS – DOS است.

۳ـ تکنیک گره I (I – node):

در این روش جدول کوچکی به نام I ایجاد می‌شود و توسط آن تعیین می‌شود که هر بلاک فیزیکی روی دیسک مربوط به کدام بلاک فایل است.

نحوه کار: در جدول I، صفات خاصه فایل و آدرس بلاک‌های فیزیکی فایل قرار دارد، در صورتی که فایل کوچک باشد، همین جدول کافی است در غیر این‌صورت برای فایل‌های بزرگ در یکی از مدخل‌های این جدول آدرس بلاک‌های دیگر دیسک قرار داده می‌شود، و این عمل برای فایل‌های بزرگتر می‌تواند ادامه پیداکند.

نکته مهم: این روش در سیستم‌عامل یونیکس (unix) مورد استفاده قرار می‌گیرد.

نتیجه‌گیری: تخصیص فضا به صورت ناپیوسته (گسسته) مطلوب تر به نظر می‌رسد زیرا پیدا کردن فضاهای خالی کوچک گسسته بسیار آسان‌تر از پیدا کردن فضاهای بزرگ پیوسته یا یکپارچه است.

■ نشست فایل به صورت گسسته ممکن‌است روی چند دیسک باشد. (آرایه‌ای از دیسک‌ها) به فایلی که بدین صورت توزیع شود، فایل توزیع شده (Distributed) یا فایل چند پاره یا اوراق شده (stripped) نیز می‌گویند.

تکنیک اوراق شدن در سطوح مختلفی مثل: کاراکتر ـ رکورد ـ بلاک ... می‌تواند انجام شود.

مدیریت بلاک‌های آزاد:

برای مدیریت فضاهای خالی دیسک ۲ روش وجود دارد:

۱ـ **ایجاد لیستی از چند بلاک دیسک:** هر بلاک در این روش حاوی شماره بلاک‌های آزاد است.

۲ـ **استفاده از بیت نقش (bitmap):** در این روش برای یک دیسک با n بلاک، n بیت در نظر گرفته می‌شود، بیت بلاک‌های آزاد را 1 و بیت بلاک‌های تخصیص یافته (اشغالی) را 0 قرار می‌دهیم.

تکنیک‌های تولید نسخه پشتیبان:

۱) استفاده از نیمه دو دیسک

۲) تولید دامپ‌های تدریجی

۳) آینه‌سازی (Mirroring)

در هر محیط ذخیره و بازیابی، همیشه لازم‌است که نسخه‌های پشتیبان به صورت دوره‌ای ایجاد شود.

■ برای تولید نسخه پشتیبان (Backup) برای یک فلاپی دیسک از فلاپی دیسک استفاده می‌شود.

■ برای تولید نسخه پشتیبان (Backup) برای یک دیسک سخت (Hard disk) معمولاً از نوراهای مغناطیسی استفاده می‌شود، البته اگر ظرفیت دیسک سخت زیاد باشد، تولید نسخه پشتیبان در نوارها بسیار زمان‌گیر می‌شود.

تکنیک‌های تولید نسخه پشتیبان به شرح زیر می‌باشد:

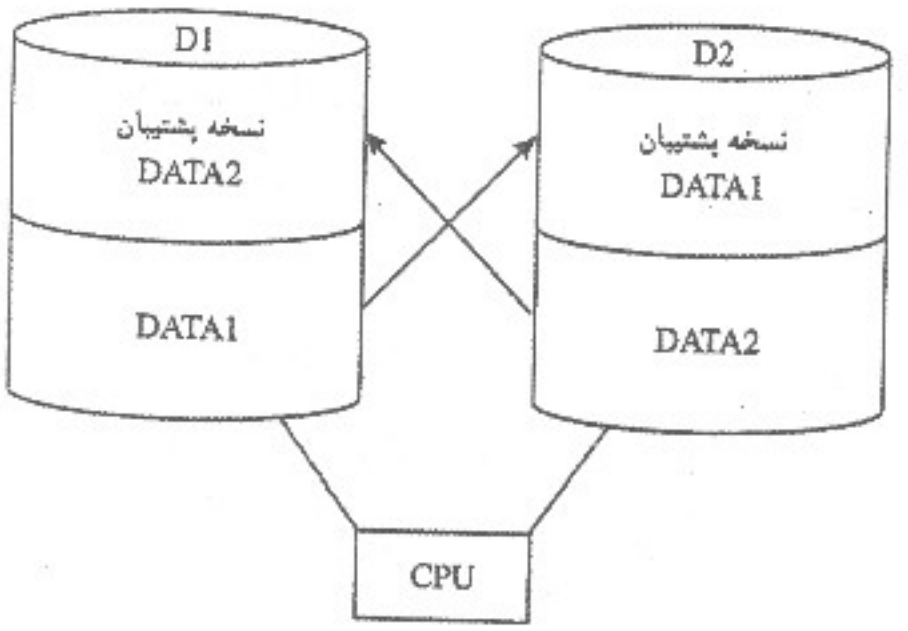
۱ـ استفاده از نیمه دو دیسک:

در این روش دو دیسک لازم است، و فضای هر دیسک را به دو نیمه تقسیم می‌کنیم:

۱ـ نیمه داده‌ها

۲ـ نیمه پشتیبان

نیمه داده‌ای هر دیسک روی نیمه پشتیبان دیسک دیگر نیز ذخیره می‌شود.



تکنیک نیمه دو دیسک

■ در این حالت هرگاه یکی از دو دیسک مشکلی پیدا کند، دیسک دیگر علاوه بر داده خود، نسخه پشتیبان آن دیسک را نگهداری کرده است.

۲ـ تولید دامپ‌های تدریجی: (incremental dump)

در این روش نسخه پشتیبان داده‌ها به صورت دامپ فایل‌هائی در دوره‌های هفتگی یا ماهانه تولید می‌شود، در عین حال دامپ فایل‌هائی که از آخرین تولید نسخه پشتیبان تغییر کرده‌اند به صورت روزانه دوباره تولید می‌شوند.

نکته: این روش تولید نسخه پشتیبان ***حافظه زیادی*** را اشغال می‌کند، و معمولاً برای تولید آن از ***نوار مغناطیسی*** استفاده می‌شود.

۳ـ آینه‌سازی (mirroring)ـ (مهم)

در این روش معمولاً از دو یا بیش از دو دیسک استفاده می‌شود.

زمانی که از دو دیسک استفاده می‌کنیم، در این روش ***نوشتن در هر دو دیسک*** انجام می‌شود، اما ***عمل خواندن فقط از یک دیسک،*** صورت می‌گیرد.

* نوشتن در دیسک دوم (دیسک آینه) کمی با تأخیر انجام می‌شود (این تأخیر به علت مطمئن شدن از صحت نوشتن در دیسک اول است) و اگر دیسکی دچار نقص شود، دیسک دوم (دیسک آینه) را می‌توان مورد استفاده قرار داد.

☑ **سطوح مختلف آدرس‌دهی: (بسیار مهم)**

با توجه به آن فایل در سه سطح ۱ـ سطح کاربر برنامه (انتزاعی) ۲ـ سطح منطقی (برنامه پردازشگر) ۳ـ سطح فیزیکی در نظر گرفته می‌شود.
سطوح آدرس‌دهی نیز در این سطوح بررسی می‌شوند.

۱ـ سطح کاربر برنامه (انتزاعی)

الف) محتوائی (مقداری): در این آدرس‌دهی کاربر مقدار یک یا چند صفت خاصه را جهت جستجو مشخص می‌کند. مانند صفت خاصه کلید.

ب) نسبی: در این آدرس‌دهی کاربر فایل را به‌صورت یک ساختار خطی می‌بیند که هر رکورد شماره یکتا دارد که از 0 شروع می‌شود.

ج) نمادین (symbolic): در این حالت رکورد دلخواه توسط یک نام، مشخص و آدرس‌دهی می‌شود.

۲) در سطح منطقی فایل:

در این سطح سیستم فایل کل فضای ذخیره‌سازی را بصورت یک آرایه از بلاک‌ها درنظر می‌گیرد. (یا سکتورها) و از تعداد بلاک‌ها نیز مطلع است.

«هر بلاک شماره‌ای دارد که از 0 شروع شده و به آن ***آدرس نسبی RBA*** گفته می‌شود» قسمت منطقی سیستم فایل آدرس تولید شده در سطح برنامه را دریافت کرده و آن‌را به آدرس RBA تبدیل می‌کند و سپس این RBA را به سمت بخش فیزیکی می‌دهد.

* سیستم فایل منطقی با داشتن تعداد، نوع و ظرفیت هر یک از رسانه‌های محیط فیزیکی طیف مقادیر RBA را برای هر رسانه و نیز در کل فضای ذخیره‌سازی مشخص می‌کند.

فرض کنیم در دیسک D_1, D_2 با مشخصات زیر داریم:

نام دیسک	تعداد استوانه	تعداد شیار در استوانه	تعداد بلاک در شیار
D_1	c_1	t_1	b_1
D_2	c_2	t_2	b_2

ظرفیت D_1 به بلاک $= c_1 \times t_1 \times b_1$

ظرفیت D_2 به بلاک $= c_2 \times t_2 \times b_2$

$$0 \le RBA_{D_1} \le S_1 - 1$$

$$S_1 \le RBA_{D_2} \le S_1 + S_2 - 1$$

طیف مقادیر RBA در کل فضای ذخیره‌سازی: $0 \le RBA \le S_1 + S_2 - 1$

تبدیل آدرس داده شده در یک برنامه به آدرس نسبی بلاک RBA : (مهم)

اگر آدرس داده‌شده در برنامه، آدرس نسبی رکورد باشد بخش منطقی باید آن‌را به RBA تبدیل کند.

آدرس در برنامه ⟵ سیستم فایل منطقی ⟵ RBA

برای تبدیل آدرس نسبی‌رکورد به RBA به‌صورت زیر عمل می‌کنیم:

i : شماره رکورد موردنظر

R : طول هر رکورد

$*\text{ByTe off set}_{rec}$ = بایت شروع رکورد $= (i-1)\times R$

$*rba_{rec}$ = آدرس نسبی بلاک حاوی رکورد مورد نظر نسبت به آغاز فایل $= \left\lfloor \frac{(i-1)\times R}{B} \right\rfloor$

$*RBA_{rec}$ = آدرس نسبی بلاک حاوی مورد نظر نسبت به اولین بلاک $= RBA_{BOF} + rba_{rec}$

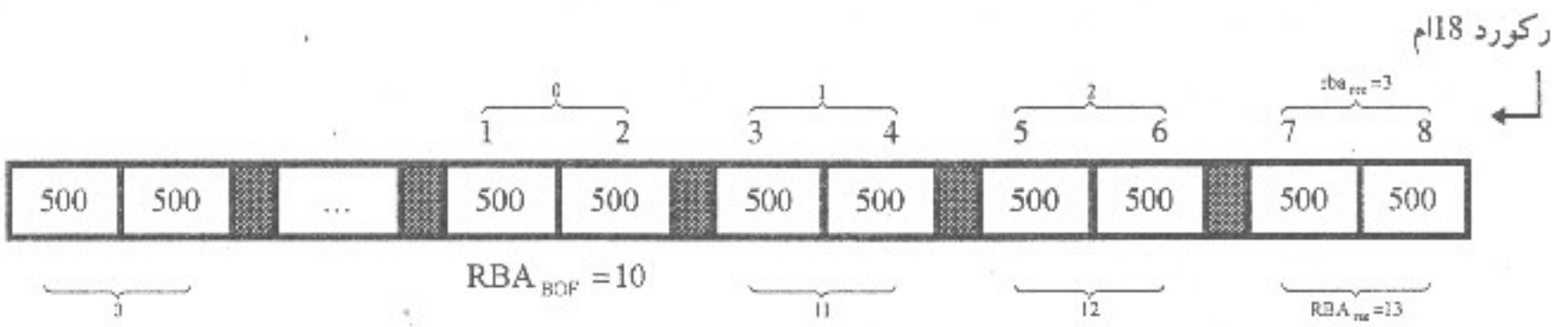
RBA_{BOF} : آدرس نسبی بلاک اول فایل که در صورت نبوده 0 فرض می‌شود.

مثال: در یک فایل $B = 1000, R = 500, RBA_{BOF} = 10$ مطلوبست آدرس نسبی رکورد 8ام فایل؟

بایت شروع = $\text{Byte off set} = (8-1)\times 500 = 3500$

$$rba_{rec} = \left\lfloor \frac{(8-1)\times 500}{1000} \right\rfloor = 3$$

$$RBA_{rec} = RBA_{BOF} + rba_{rec} = 10 + 3 = 13$$



۳- آدرس‌دهی در سطح فیزیکی: در این سطح آدرس RBA تبدیل به آدرس فیزیکی قابل ذخیره‌سازی می‌شود.

آدرس فیزیکی ⟵ سیستم فیزیکی فایل ⟵ RBA آدرس نسبی بلاک ⟵ سیستم منطقی فایل ⟵ آدرس در برنامه

در این وضعیت با داشتن اطلاعات زیر این امکان وجود خواهد داشت؛ و می‌توانیم آدرس RBA را به شماره استوانه ـ شیار ـ بلاک تبدیل کنیم.

- #CyL: شماره سیلندر (استوانه)
- #trk: شماره شیار نسبت به اول سیلندر
- #bLk: شماره بلاک یا سکتور نسبت به اول شیار

$$*\text{CyL\#} = \left\lfloor \frac{(\text{RBA} - \text{RBA}_{\text{Device}})}{(t \times b)} \right\rfloor$$

$$*\text{trk\#} = \left\lfloor \frac{(\text{RBA} - \text{RBA}_{\text{Device}}) \bmod (t \times b)}{b} \right\rfloor$$

$$*\text{BLK\#} = (\text{RBA} - \text{RBA}_{\text{Device}}) \bmod b$$

* درصورتیکه $\text{RBA}_{\text{Device}}$ بیان نشود آن را 0 فرض می کنیم.

* $t \times b$: ظرفیت هر استوانه به بلاک است که:

b = تعداد بلاک در شیار

t = تعداد شیار یا تعداد هد یا تعداد رویه در هر سیلندر

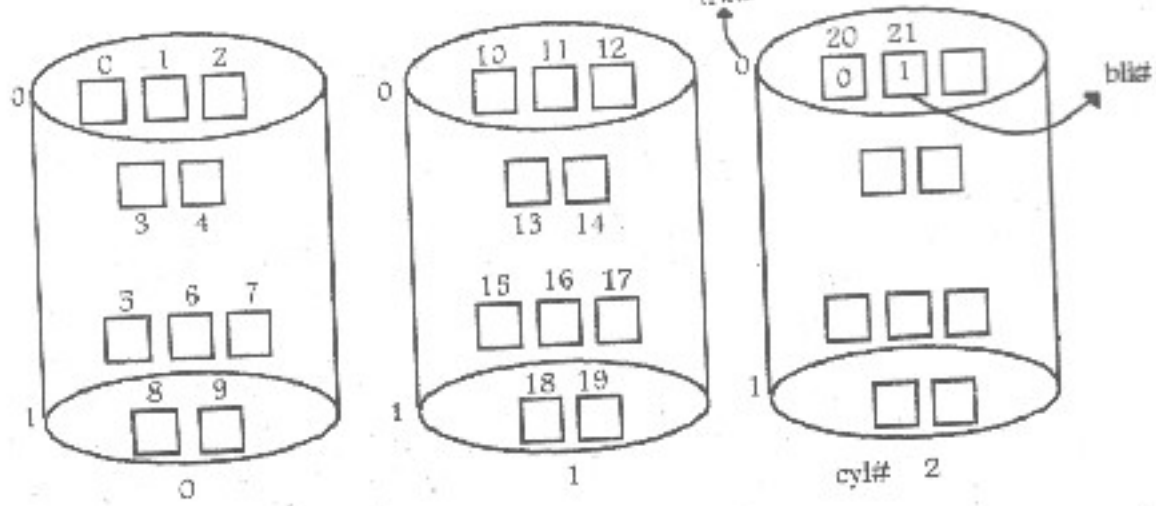
مثال: در یک دیسک که هر شیار آن 5 بلاک دارد و هر سیلندر آن 2 شیار دارد. آدرس فیزیکی بلوکی با RBA = 21 به فرم (CyL#, trk#, blk#) کدام است؟

☜ $\text{RBA}_{\text{Device}}$ مطرح نشده آن را 0 فرض می کنیم.

حل تشریحی:

تعداد بلاک در شیار b = 5

تعداد شیار در سیلندر t = 2



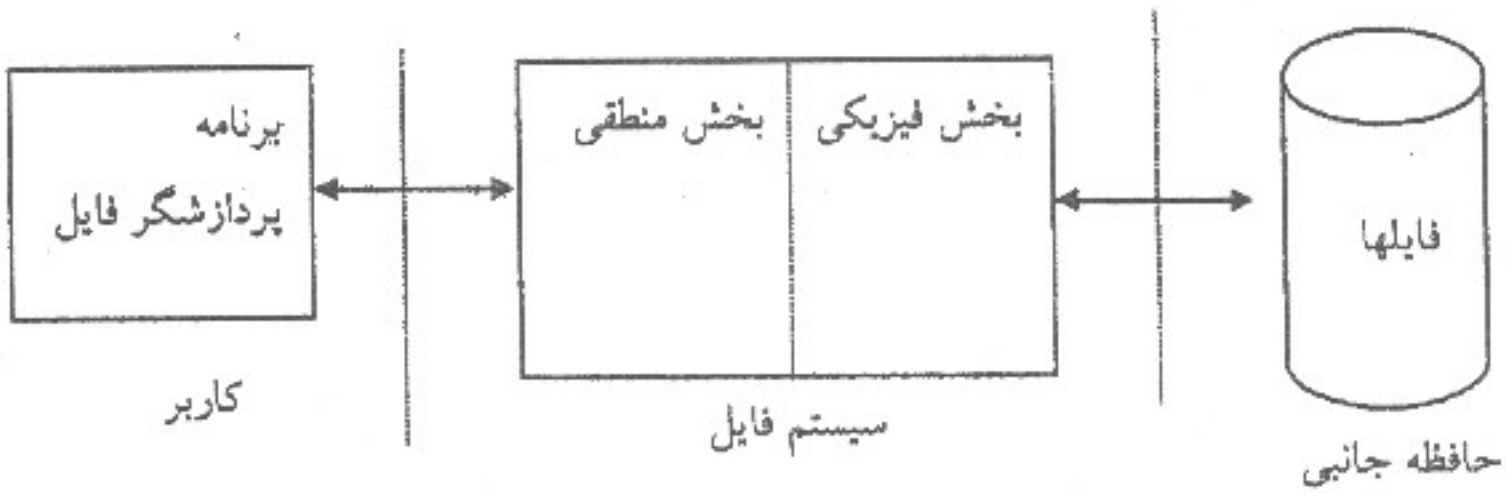
$$\text{CyL\#} = \left\lfloor \frac{\text{RBA}}{t \times B} \right\rfloor = \left\lfloor \frac{20}{2 \times 5} \right\rfloor = 2$$

$$\text{blk\#} = \text{RBA} \bmod b = 21 \bmod 5 = 1$$

$$\text{trk\#} = \left\lfloor \frac{\text{RBA} \bmod (t \times b)}{b} \right\rfloor = \left\lfloor \frac{21 \bmod (2 \times 5)}{5} \right\rfloor = 0$$

بافر و بافرینگ

سیستم فایل: قسمتی از سیستم‌عامل که با فایل در ارتباط است، و به دو ناحیه منطقی و فیزیکی تقسیم می‌شود:



- **بخش منطقی** سیستم فایل دستورات کاربر مانند: باز و بسته کردن فایل و خواندن و نوشتن فایل را انجام می‌دهد.

- **بخش فیزیکی** سیستم فایل به طور مستقیم به فایل‌های موجود برروی حافظه جانبی دستیابی دارد. (این قسمت، دستورات گرفته شده از قسمت منطقی سیستم فایل را به دستوراتی جهت صدور به حافظه جانبی تبدیل می‌کند).

نکته ۱: (مهم)

سیستم فایل فیزیکی سه عمل اصلی را انجام می‌دهد:

۱ـ مکان‌یابی

۲ـ خواندن از روی حافظه فیزیکی

۳ـ نوشتن برروی حافظه فیزیکی

نکته ۲: به بخش فیزیکی سیستم فایل، (Access Method) بخش شیوه دستیابی نیز گفته می‌شود.

■ بافر (Buffer):

ناحیه‌ای است از حافظه اصلی (RAM)، که برای ایجاد هماهنگی بین سرعت و زمان عملیات پردازنده (CPU) عملیات ورودی/خروجی (I/O) استفاده می‌شود.

نکته ۱: در هر بافر در صورتیکه فایل بلاک‌بندی نشده باشد، [illegible] بخش شیوه دستیابی نیز گفته می‌شود. [illegible] دارد که در هر بار دستور I/O از رسانه ذخیره‌سازی خوانده شده و در بافر قرار می‌گیرد.

منطقه بافر (Buffer Pool)

در سیستم فایل، بافر معمولاً از منطقه‌ای از حافظه اصلی به برنامه فایل پرداز تخصیص داده می‌شود که به آن منطقه بافرها (Buffer Pool) می‌گویند. (و گاه از حافظه نهان استفاده می‌شود).

نحوه ایجاد بافرها

بافرها به سه روش ساخته می‌شود:

۱) با ایجاد ناحیه‌ای از حافظه در برنامه و با اجرای یک ماکرو که محتوای بافر را با فایل‌های تحت پردازش مرتبط می‌کند (در این حالت برنامه‌ساز خود بافر را ایجاد می‌کند).

۲) با اجرای یک ماکرو، که از سیستم‌عامل درخواست ایجاد بافر می‌کند.

۳) خود سیستم‌عامل وقتی که فایل باز می‌شود، اقدام به ایجاد بافر(ها) می‌کند و پس از بسته شدن فایل، بافر(ها) را بازپس می‌گیرد.

چگونگی دستیابی برنامه به محتوای بافر

برنامه به دو صورت می‌تواند به محتوای بافر دستیابی داشته باشد:

۱) روش موسوم به اسلوب انتقالی (Move mode):

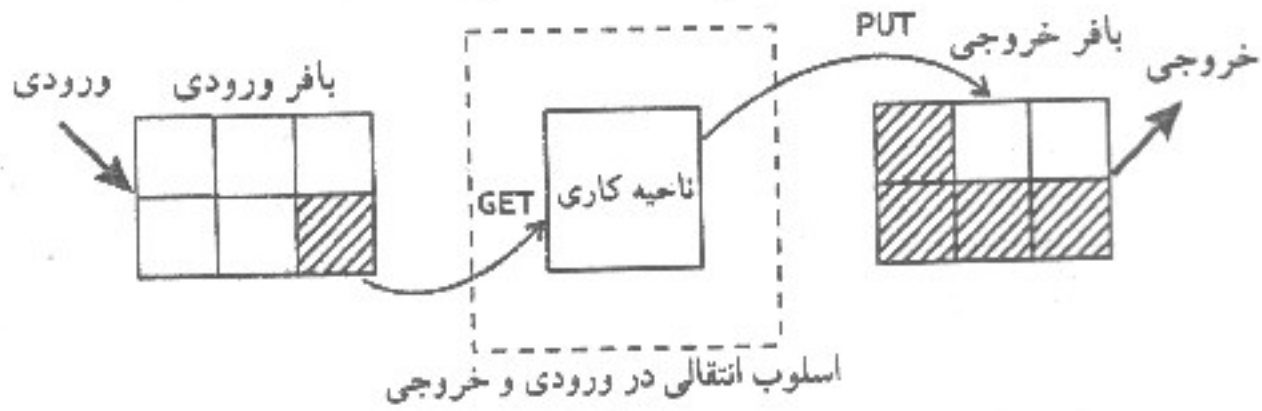
۲) روش موسوم به اسلوب مکان نمائی (Locate mode) یا مکان‌گیری (یا اسلوب تعویض (Exchange Mode)

۱) روش موسوم به اسلوب انتقالی (Move mode):

در روش انتقالی، برنامه بافر خاص خود را دارد (همان ناحیه‌کاری کاربر) و به بافر دسترسی ندارد. رکورد از بافر ورودی به ناحیه کاری کاربر در برنامه منتقل شده و یا از ناحیه کاری کاربر در برنامه به بافر خروجی منتقل می‌شود.

عمل بلاک‌بندی و بلاک گشائی در این روش توسط سیستم‌عامل انجام می‌شود.

نکته: در این روش
۱ـ یک فرمان برای هر یک از بافرهای ورودی ـ خروجی ورود دارد، دو عمل نمی‌توانند از یک بافر انجام شوند.
۲ـ کاربر نیاز به ناحیه‌کاری خاص خود را دارد.

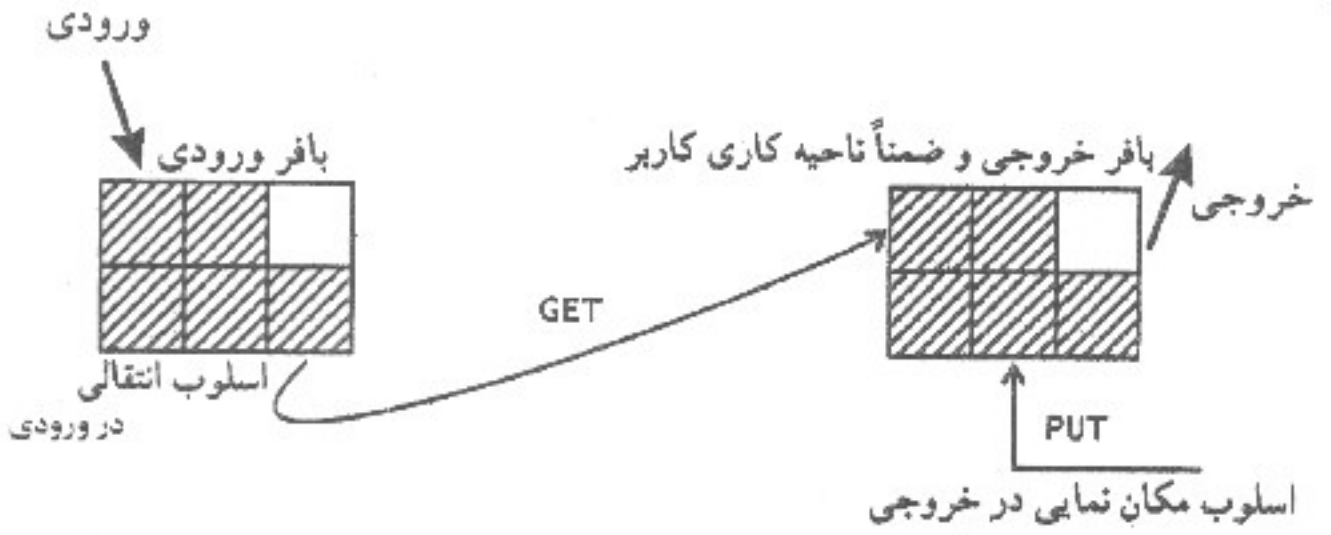


اسلوب انتقالی

۲) روش موسوم به اسلوب مکان نمائی (Locate mode) یا مکان‌گیری (یا اسلوب تعویض (Exchange Mode)

در روش مکان‌نمائی، ناحیه‌کاری خاص کاربر وجود ندارد. و سیستم‌عامل مکان بافر حاوی رکورد مورد نظر کاربر را به شکلی مانند: قرار دادن آدرس آن در یک ثبات یا در ناحیه‌ای از برنامه در اختیار برنامه قرار می‌دهد؛ در واقع کاربر از همان بافر به عنوان ناحیه‌کاری استفاده می‌کند.

عمل بلاک‌بندی و بلاک گشائی توسط برنامه انجام می‌شود.



اسلوب مکان‌نمایی

نکته: استفاده از دو روش انتقالی و مکان‌نمائی در هر یک از دو عمل ورودی ـ خروجی امکان‌پذیر است.

بافر از نظر محل ایجاد

دو نوع بافر وجود دارد:

۱) بافر سخت‌افزاری

۲) بافر نرم‌افزاری

۱) **بافر سخت‌افزاری:** بافری است که در بعضی از دستگاه‌ها مثلاً در کنترلر رسانه‌های ذخیره‌سازی، به ویژه در رسانه‌های کندتر مثل کارت‌خوان و نوار کاغذی‌خوان و چاپگر به گستردگی مورد استفاده قرار می‌گیرد.

۲) **بافر نرم‌افزاری:** ناحیه‌ایست در حافظه اصلی یا در حافظه نهان و توسط سیستم‌عامل، طبق الگوریتم‌های تخصیص بافر در مدیریت حافظه، در اختیار برنامه‌های فایل‌پرداز قرار داده می‌شود.

نکته مهم: هر دو نوع بافر به CPU و پردازنده ورودی / خروجی امکان می‌دهند تا با همپوشانی زمانی (Time Overlapping) عمل کنند، به این ترتیب که بافر سخت‌افزاری با سرعت رسانه ذخیره‌سازی پر می‌شود و پس از پر شدن، محتوایش با سرعت انتقال کانال به کامپیوتر وارد شده، طی چرخه‌های زمانی خاصی به بافر اصلی، که نرم‌افزاری است، منتقل می‌گردد.

انواع بافرینگ

انواع بافرینگ، از نظر تعداد بافرهایی که به عملیات ورودی/خروجی برنامه فایل‌پرداز تخصیص می‌یابد، به شرح زیر می‌باشد:

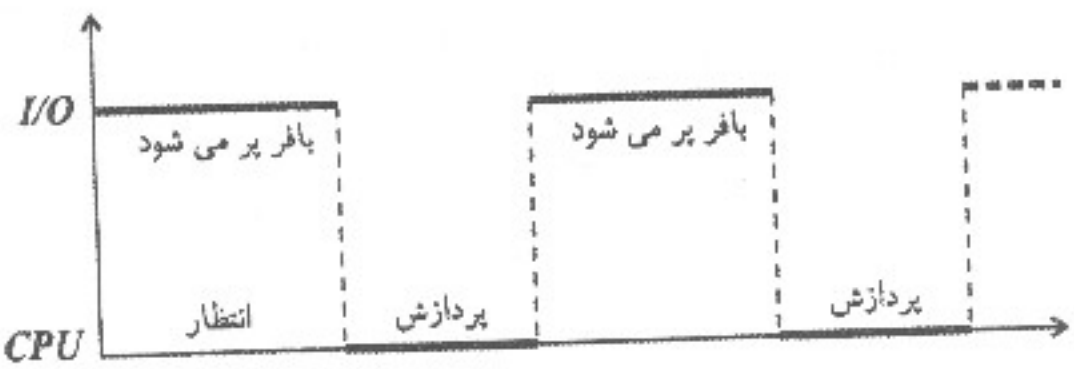
۱) بافرینگ ساده (Single buffering)

۲) بافرینگ مضاعف (Double buffering (Buffer swapping)

۳) بافرینگ چندگانه (Multiple buffering) (در حالت خاص **چرخشی**)

۱)بافرینگ ساده

در این نوع بافرینگ، یک بافر در اختیار برنامه فایل‌پرداز قرار داده می‌شود، در بافرینگ ساده، طبعاً زمان انتظار واحد پردازش مرکزی و اجرای برنامه افزایش می‌یابد براساس نمودار زیر:



نمودار کلی بافرینگ ساده در پردازش انبوه فایل

نکته: در اثنایی که بافر پر می‌شود، پردازش مرکزی حالت عاطل (Idle) دارد. در محیط چند برنامه‌ای می‌توان از این زمان برای برنامه‌های دیگر استفاده کرد. می‌بینیم که در این حالت امکان همروندی عملیات CPU و عملیات پردازشگر ورودی/خروجی وجود ندارد، البته در اسلوب مکان‌نمایی. در اسلوب انتقالی چون برنامه بافر خاص خود را دارد و در صورتی‌که فایل بلاک‌بندی نشده باشد، این همروندی تا حدی امکان‌پذیری است.

۲)بافرینگ مضاعف

یا دو بافر، می‌توان در اثناء خواندن یک بلاک و انتقال آن به یک بافر، محتوای بافر دیگر را که پر است، پردازش کرد.

نکته: در پردازش فایل‌ها به طور پی‌درپی و انبوه (یعنی تعداد زیادی بلاک خوانده می‌شوند)، حتماً لازم است دو بافر در اختیار داشته باشیم، وگرنه عملیات نه سریع خواهد بود و نه کارا.

(مهم) شرط کارائی بافرینگ مضاعف: زمانی را که واحد پردازش مرکزی برای پردازش محتوای یک بافر، مصرف می‌کند. باید کمتر از زمانی باشد که پردازنده ورودی خروجی و کنترل کننده دیسک برای انتقال بلاک به یک بافر لازم دارند، یعنی داشته باشیم:

$$\text{شرط کارائی بافرینگ مضاعف} \Leftrightarrow C_B < b_tt \text{ یا } C_B \le \frac{B+G}{t} \text{ یا } C_R \le \frac{R+W_R}{t}$$

پارامترها عبارتند از:

B:طول بلاک

G:طول گپ

t: نرخ انتقال رسانه

b_tt: زمان انتقال یک بلاک $= \frac{B}{t}$

C_B: زمان لازم برای پردازش محتوای یک بافر (یک بلاک)

C_R: زمان لازم برای پردازش یک رکورد

اگر شرط زمانی فوق‌الذکر برقرار نباشد، روند نمای عملیات دیگر صادق نخواهد بود، و بافرینگ مضاعف کارایی خود را از دست می‌دهد و نرخ انتقال واقعی کاهش می‌یابد.

ارتباط عملیات خواندن/نوشتن با نحوه بافرینگ

با توجه به تعداد بافر و این‌که سیستم با چه روشی عمل کند، حالات مختلفی ممکن است وجود داشته باشد، به شرح زیر:

۱) سیستم از بافر استفاده نمی‌کند و فقط ناحیه کاری داریم: نمی‌توان بلاک‌بندی انجام داد. رکوردها به طور مجزا به ناحیه‌کاری کاربر در حافظه منتقل می‌شوند. عمل خواندن، رکورد به رکورد انجام می‌شود. این شیوه اصطلاحاً به «روش مبنایی (Basic)» موسوم است.

۲) یک بافر داریم و ناحیه کاری نداریم: می‌توان بلاک به بلاک خواند (یا نوشت) عملیات خواندن یا نوشتن «پیشرس» توسط کاربر، دیگر امکان‌پذیر نیست (سیستم از اسلوب مکان‌نمایی استفاده می‌کند).

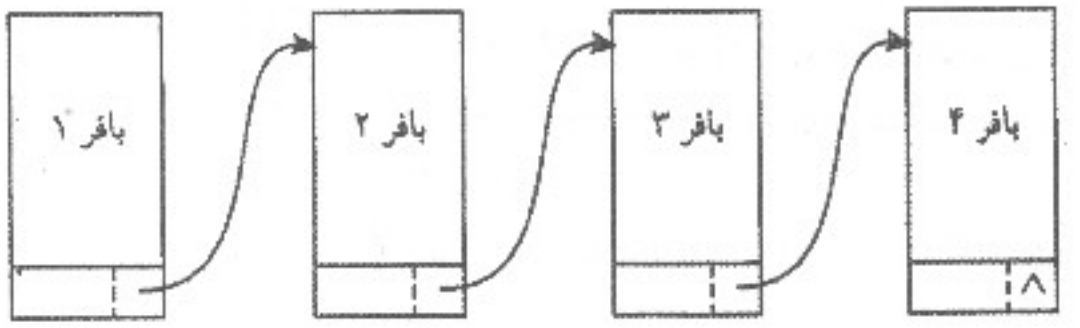
۳) یک بافر و یک ناحیه‌کاری داریم: می‌توان بلاک به بلاک خواند، بلاک بعدی را نیز می‌توان خواند و نوشت اما فقط در اثناء پردازش آخرین رکورد بلاک (با اسلوب انتقالی).

۴) دو بافر داریم (بافرینگ مضاعف) و ناحیه‌کاری نداریم: پردازش محتوای یک بافر، در اثناء پر یا خالی شدن بافر دیگر انجام می‌گیرد (با اسلوب مکان‌نمایی).

۵) دو بافر داریم و ناحیه‌کاری هم داریم (اسلوب انتقالی): همروندی عملیات ورودی/خروجی و پردازش رکوردها کامل است، سیستم با اسلوب انتقالی عمل می‌کند.

بافرینگ چندگانه پیشرس و مفهوم صف

در پردازش انبوه فایل‌ها، سیستم فایل‌ها، با استفاده از بافرینگ چندگانه، می‌تواند رکوردهای فایل را از پیش خوانده، در بافر بگذارد، یعنی در هر لحظه، رکورد بعدی در بافر است. چنین تکنیکی به صف‌بندی موسوم است که با استفاده از پیشدستی (Anticipation) در انجام عملیات ورودی/ خروجی پیاده‌سازی می‌شود و لذا به نوعی بافرینگ پیشرس نیاز دارد. برای این کار سیستم، چند بافر را به کار می‌گیرد.



بافرینگ چندگانه

نکته مهم: بافرینگ چندتایی معمولاً به صورت چرخشی پیاده‌سازی می‌شود، از این رو به آن، بافرینگ چرخشی (Circular buffering) هم می‌گویند.

جمع بندی:

دیدیم که بافرینگ برای ایجاد هماهنگی بین پردازشگر مرکزی و پردازشگرورود/خروجی است. به ویژه در بافرینگ مضاعف، همروندی بین عملیات این دو بخش امکان‌پذیر می‌شود. اما گاهی ممکن است میانگین درخواست‌های پردازش بیش از حدی باشد که پردازشگر ورودی/خروجی بتواند پاسخ دهد. در این وضع، حتی با بافرینگ چندتایی هم، تمام بافرها ممکن است پر شوند و پردازشگر مرکزی پس از پردازش آن‌ها باز هم مجبور شود انتظار بکشد و همروندی به تمامی حاصل نشود. اما در محیط چند برنامه‌ای، وقتی که فعالیت‌های گوناگون ورودی و خروجی و پردازش‌های متعدد وجود دارد، بافرینگ می‌تواند کارایی سیستم و نیز برنامه‌ها را افزایش دهد.

ملاحظاتی در طراحی فایل

طراحی فایل عبارتست از فرآیند تعیین یک ساختار (یا سازمان) فایل به نحوی که نیازهای مشخص کاربر پایانی را برآورده کند و زمان پاسخ‌دهی (Response time) به درخواست‌هایش را به حداقل برساند.

نکته مهم: فرآیند طراحی فایل در اساس دو مرحله دارد:

۱) طراحی فایل منطقی، که عبارتست از انتخاب یک ساختار فایل (از بین ساختارهایی که سیستم فایل ارائه می‌کند) با طراحی یک ساختار جدید.

۲) طراحی ساختار فایل فیزیکی، که خود گام‌هایی دارد. و گام‌های آن به قرار زیر است:

۱ـ انتخاب ضریب بلاک‌بندی

۲ـ تخصیص بافرها برای عملیات ورودی/خروجی

۳ـ اندازه فایل فیزیکی

۴ـ مکان بلاک در حافظه خارجی

۵ـ طراحی یا انتخاب شیوه دستیابی مناسب

۶ـ انتخاب کلید اصلی از بین صفات خاصه رکورد و کلید ثانوی

۷ـ **در نظر داشتن رشد فایل:** فایل‌ها به دو دسته کلی پویا یا ایستا تقسیم می‌شوند. در فایل‌هایی پویا (Dynamic File)، اندازه فایل در اثر عملیات تغییر دهنده (درج، حذف، بهنگام‌سازی) مرتب تغییر می‌کند. وقتی که تغییرات در فایل زیاد باشد، می‌گوییم فایل بسیار ناماناً (Highly volatile) است. در نظر گرفتن وضعیت رشد فایل برای تخمین حجم عملیات لازم در دستیابی به رکوردها، لازم است.

۸ـ تعیین زمان و پریود سازماندهی مجدد فایل.

تکنیک‌های بهبود کارائی سیستم فایل

داشتن مدیریت بافرینگ کارا و امکانات تخصیص بافر، نقش مهمی در بهبود کارائی پردازش تمام فایل به ویژه در حالت ***پی دا پی*** دارد.

علاوه بر این روش‌های:

۱) کاهش زمان درنگ دورانی (r)

۲) کاهش زمان استوانه‌جوئی (s)

۳) افزایش سرعت عملیات پردازش فایل باعث بالا رفتن کارائی سیستم فایل می‌شود.

■ **تکنیک‌های کاهش زمان درنگ دورانی**

۱) تداخل بلاک‌ها (درهم چینی بلاک‌ها) (interleaving)

۲) تغییرمکان نقطه شروع شیارها (track staggering)

۳) پراکنده‌خوانی

۱- تداخل بلاک‌ها (درهم‌چینی بلاک‌ها):

در این روش بلاک‌ها را بصورت یک‌درمیان و یا دو درمیان و ... روی شیارها قرار می‌دهند. تا شرط $C_B \leq b_t$ برقرار باشد.(یادآوری $C_B \leq b_t$ شرط کارائی در بافرینگ مضاعف)

نکته ۱: (مهم)

زمانی که می‌خواهیم بلاک‌ها را مرتب بخوانیم یعنی اول بلاک B1 و بعد بلاک B2 و ... به دلایل زیر مجبور به استفاده از این روش خواهیم بود:

۱) به علت کمبود حافظه اصلی فقط 1 بافر داریم و امکان استفاده از 2 بافر و پیاده‌سازی بافرینگ مضاعف وجود ندارد.

۲) $C_B \geq b_t$ که این رابطه همان شرط عدم کارائی در بافرینگ مضاعف است.

نکته ۲:

اگر بلاک‌ها را پشت‌سرهم روی شیارها قرار دهیم باتوجه به دلایل بیان شده در (نکته۱) امکان خواندن بلاک B_2 بعد از بلاک B_1 بلافاصله امکان‌پذیر نیست چون زمان پردازش بلاک B_1 $\left(C_{B_1}\right)$ بیشتر از زمان خواندن آن بلاک است $\left(b_t\right)$ بنابراین زمانی که به ابتدای B_2 می‌رسیم، پردازشگر آماده پردازش نیست.

بنابراین باید یک دور کامل دیسک زده شود (2r) تا دوباره به ابتدای B_2 برسیم.

نکته ۳:

با استفاده از این تکنیک دیگر نیازی به زمان 2r برای خواندن بلاک بعدی نداریم. و زمان به نصف کاهش پیدا می‌کند.

این کاهش زمان باعث:

۱- کاهش زمان انتقال فایل

۲- افزایش نرخ انتقال انبوه (t') می‌شود.

نکته۴: این روش را اصطلاحاً چیدن بلاک‌ها به‌صورت n درمیان نیز می گویند، روابط مهم زیر وجود خواهند داشت:

(I) (ضریب تداخل) $i_f = n+1$

* $\text{interleaving factor} = i_f$

مثال: درصورتی‌که بلاک‌ها را 2درمیان قرار دهیم ضریب تداخل کدام است؟

$n = 2 \Rightarrow i_f = n+1 = 3$ ضریب تداخل

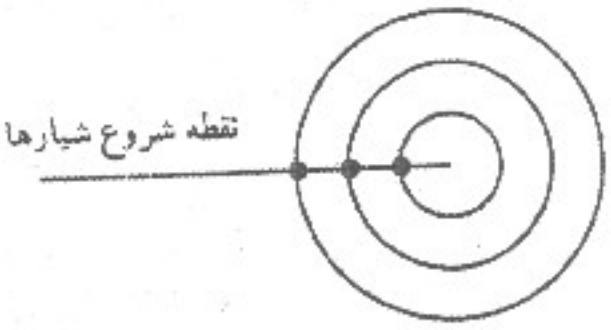
$$\text{(II)}\quad t'_{\text{interleaving}} = \frac{1}{i_f} \times t'$$

* $t'_{\text{interleaving}}$ = نرخ انتقال جدید بعد از چیدن بلاک‌ها بصورت n درمیان است.

یادآوری: به‌دلیل عدم کارائی بافرینگ مضاعف $C_B \geq b_{II} = \frac{B+G}{t}$ از این روش استفاده کرده‌ایم که در این حالت $t_{\text{interleaving}}$ جدید بانوجه به رابطه (II) کاهش پیداکرده و درنتیجه $b_{II} = \frac{B+G}{t}$ بزرگتر شده درنتیجه $C_B \leq b_{II}$ و شرط کارائی فراهم شده است.

۲- تغییرمکان نقطه شروع شیارها: (track staggering)

درحالت کلی نقطه شروع تمام شیارها در هر صفحه یک دیسک روی یک شعاع قرار دارد.



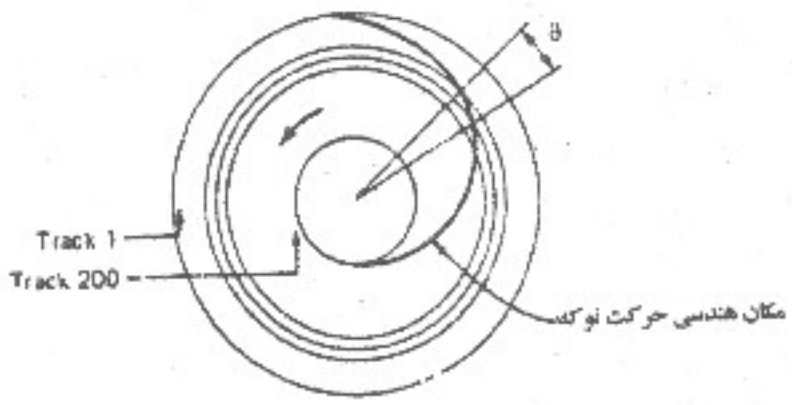
نقطه شروع شیار در تکنولوژی معمولی

نکته۱:

در این روش نقطه شروع هر شیار را نسبت به شیار قبلی به اندازه زاویه θ تغییر مکان می‌دهیم. به این‌ترتیب زمان درنگ دورانی کاهش پیداکرده و نرخ انتقال بیشتر می‌شود.

نکته۲:

این روش اساساً سخت‌افزاری است و بیشتر زمانی استفاده می‌شود که زمان استوانه جویی برای رفتن به استوانه بعدی (S_1) خیلی کمتر از زمان درنگ دورانی باشد. $(r < S_1)$



تغییر مکان نقطه آغاز شیار

نکته۳:

مقدار θ باتوجه به زمان رفتن به استوانه همجوار بعدی از رابطهٔ زیر بدست می‌آید.

$$\theta = \frac{360}{60 \times 1000} \times \text{rpm} \times \text{Max}(s_1)$$

مثال: درصورتی‌که چرخش دیسک معادل 1200 دور در دقیقه باشد و زمان رفتن به استوانه بعدی حداکثر 10باشد اندازه θ برای تغییر مکان نقطه شروع هر شیار نسبت به شیار قبلی کدام است؟

$$\left.\begin{matrix}\text{rpm} = 1200 \\ \text{Max}(s_1) = 10\end{matrix}\right\} \Rightarrow \theta = \frac{360}{60 \times 1000} \times 1200 \times 10 = 72$$

۳) پراکنده خوانی

اگر ماهیت پردازش فایل چنان باشد که نظم خاصی مورد نظر نباشد، به عبارت دیگر بتوان بلاک‌ها را، به طور پراکنده خواند، در این صورت می توان، به شرط داشتن تعداد کافی بافر، بلاک‌های یک استوانه را، به ترتیبی که زیر نوک خواندن/ نوشتن قرار می‌گیرند، خواند و به بافرها منتقل کرد، تا بعداً مورد پردازش قرار گیرند. در چنین حالتی، متوسط زمان درنگ دوران، برای رسیدن به آغاز یک بلاک، نصف زمان انتقال بلاک خواهد بود، یعنی:

$$\boxed{r = \frac{1}{2} \times \frac{B+G}{t}}$$

کاربرد: در خواندن فایل‌هایی که نظم مطرح نیست (مانند:پی‌درپی) اما درحالت سریال که براساس نظم خاصی باید فایل خوانده شود کاربرد ندارد.

■ تکنیک‌های کاهش زمان استوانه‌جویی (مهم)

این تکنیک‌ها عبارتند از:

۱) استفاده از دیسک‌های با بازوی ثابت
۲) توزیع فایل روی چند دیسک
۳) استفاده از الگوریتم‌های مناسب برای حرکت دادن بازوی دیسک
۴) اعمال ملاحظات خاص در جای‌دهی رکوردها در فایل

۱) دیسک‌های با بازوی ثابت

در این دیسک‌ها، به ازاء هر شیار از رویه، یک نوک خواندن/ نوشتن به بازو متصل است و بازو حرکت ندارد.

نکته۱: در این تکنیک زمان s استوانه‌جوئی صفر است.

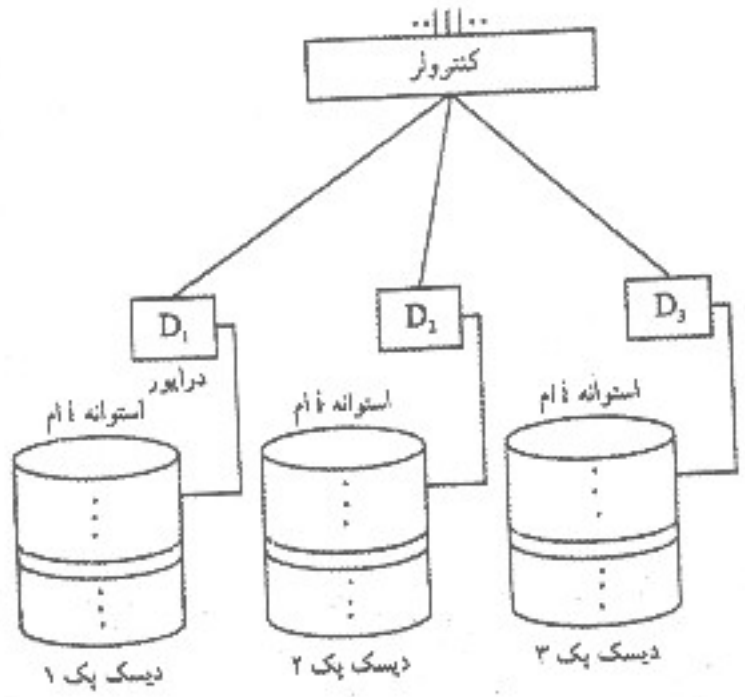
نکته۲: این روش اصولاً سخت‌افزاری است، و ***هزینهٔ آن بالا*** است.

۲) توزیع فایل روی چند دیسک

در این تکنیک فایل را روی استوانه‌های هم شماره از چند دیسک پک جای می‌دهند.

نکته۱: در این تکنیک زمان استوانه‌جوئی (s) صفر است.

نکته۲: این تکنیک نرم‌افزاری بوده و در جهت شبیه‌سازی محدود تکنیک اول، یعنی استفاده از دیسک‌های با بازوی ثابت است.



چند دیسک پک

تکنولوژی RAID

ایده اصلی این تکنولوژی، کوتاهانه این است که به جای استفاده از یک واحد دیسک با ظرفیت بالا، از چند واحد دیسک کوچکتر به صورت یک آرایه استفاده شود.

نکات مهم:

۱ـ RAID مجموعه‌ای‌است از دیسک‌ها که از نظر سیستم‌عامل به صورت یک واحد دیسک منطقی دیده می‌شود.

۲ـ داده‌ها روی دیسک‌های مختلف توزیع می‌شوند.

۳ـ داده‌ها با میزانی از افزونگی (Redundancy) (تکرار) ذخیره می‌شوند تا کار ترمیم (Recovery) داده‌ها در صورت بروز خرابی (Failure) (نقص) تسهیل و تسریع شود (با ذخیره‌سازی اطلاعات پاریتی یک دیسک در دیسک دیگر)

مزیت:

با توزیع داده‌ها، همزمانی هتیابی به آن‌ها امکان‌پذیر می‌شود و از این رهگذر کارایی سیستم فایل در عملیات I/O افزایش می‌یابد.

عیب:

استفاده از چند واحد دیسک، احتمال بروز نقص را افزایش می‌دهد و به همین دلیل با پذیرش افزونگی در ذخیره‌سازی اطلاعات، داده‌های خراب شده را ترمیم می‌کند.

۳) الگوریتم‌های کنترل حرکت بازو در محیط چند برنامه‌ای

در محیط چند برنامه‌ای که در آن، سیستم باید به درخواست‌های ورودی/خروجی چند برنامه پاسخ دهد حرکت بازوی دیسک باید با توجه به چند و چون درخواست‌های ورودی/خروجی صورت گیرد (که از نوعی بی‌نظمی برخوردار است، هرچند ممکن است توزیع آن‌ها قابل ارزیابی باشد). زیرا احتمال دارد سیستم برای یافتن داده مورد نظر یک برنامه، بازوی دیسک را از یک کرانه کران دیگر آن ببرد، یعنی از یکی از

شیارهای بیرونی به شیار کاملاً درونی و برعکس. این حرکت‌های بازوی دیسک طبعاً زمان استوانه‌جویی را افزایش می‌دهد. در چنین سیستمی باید حرکت بازوی دیسک براساس الگوریتمی برنامه‌ریزی و کنترل شود تا متوسط زمان مزبور به حداقل برسد.

در این قسمت بعضی از الگوریتم‌های زمانبندی جهت کاهش زمان s را شرح می‌دهیم. این روش‌ها عبارتند از FCFS، SSFT، SCAN و LOOK پیاده‌سازی و استفاده از این الگوریتم‌ها برعهده سیستم‌عامل است.

الف) زمانبندی FCFS

ساده‌ترین الگوریتم (First Come – First Serviced) یا FIFO می‌باشد. این روش عادلانه‌ترین روش است ولی غالباً سریع‌ترین روش نیست. در این روش درخواست‌ها در صفی قرار داده شده و به ترتیب ورود سرویس‌دهی می‌شوند.

مثال: شماره سیلندرهای درخواستی موجود در صف یک دیسک به ترتیب از چپ به راست عبارتند از:

98 , 183 , 37 , 122 , 14 , 124 , 65 , 67

اگر در ابتدای کار هد در سیلندر 53 باشد، کل مجموع حرکت هد در الگوریتم FIFO چند سیلندر خواهد بود؟

حل: هد ابتدا از سیلندر 53 به 98 می‌رود با $45 = 53 - 98$ حرکت. به همین ترتیب از سیلندر 98 به سراغ سیلندر 183 می‌رود با $183 - 98 = 85$ حرکت. پس:

$$(98-53)+(183-98)+(183-37)+(122-37)+(122-14)+(124-14)+(124-65)+(67-65)=$$

$$45+85+146+85+108+110+59+2=640$$

مشکل این روش آن است که هد ممکن است حرکت‌های شدیدی داشته باشد مثلاً از 122 به 14 برود و دوباره به 124 برگردد.

ب) زمانبندی SSTF

SSTF مخفف عبارات Shortest Seek Time First می‌باشد. در این روش درخواستی با حداقل زمان جستجو نسبت به موقعیت فعلی دهد، انتخاب می‌شود، یعنی سیلندری که شماره آن با شماره مکان جاری هد کمترین تفاضل را داشته باشد. (به عبارت بهتر حرکت بازوی دیسک همیشه در جهت رکورد s است که کمترین زمان برای حرکت در بازو را طلب کند)

مثال: اگر شماره سیلندرهای درخواستی به صورت 67 , 65 , 124 , 14 , 122 , 37 , 183 , 98 , 53 بوده و هد در ابتدا در سیلندر 53 باشد، ترتیب سرویس‌دهی و کل مجموع حرکت در هد در الگوریتم SSTF چه اندازه خواهد بود؟

حل:

$$53 \xrightarrow{12} 65 \xrightarrow{2} 67 \xrightarrow{30} 37 \xrightarrow{23} 14 \xrightarrow{84} 98 \xrightarrow{24} 122 \xrightarrow{2} 124 \xrightarrow{59} 183$$

$$12-2+30+23+84+24+2-59=236$$

همان‌طور که مشاهده می‌کنید نسبت به 640 حرکت در FCFS بهبود زیادی یافته است.

زمانبندی SSTF مشکل قحطی‌زدگی یا تعویق نامحدود (Starvation) را دارد. مثلاً فرض کنید در صف درخواست 124 و 14 را داشته باشیم و دیسک در حال سرویس‌دهی به سیلندر 14 است در این حال درخواست‌های 17 و 23 و غیره که نزدیک 14 هستند وارد شود. بدین ترتیب درخواست 124 مرتباً به عقب می‌افتد.

تذکر: الگوریتم SSTF بهینه نیست. اگر مثال فوق را به صورت زیر از چپ به راست سرویس دهیم جمع کل حرکات هد 208 سیلندر می‌شود:

53 , 37 , 14 , 65 , 67 , 98 , 122 , 124 , 183

ج) زمانبندی SCAN

در روش SCAN (پویش) هد دیسک مرتباً از یک انتهای دیسک به سمت انتهای دیگر حرکت می‌کند و هر بار که به سیلندری برسد که نیاز به سرویس‌دهی دارد، به آن سرویس می‌دهد. در شروع این روش علاوه بر دانستن مکان جاری هد باید جهت شروع حرکت آن را نیز بدانیم. مثلاً می‌توان هد را در ابتدای کار در جهت سیلندری حرکت داد که کوتاه‌ترین فاصله را با هد دارد و سپس در همان جهت حرکت هد را ادامه داد.

مثال: اگر شماره سیلندرهای درخواستی به صورت 67 , 65 , 124 , 14 , 122 , 37 , 183 , 98 بوده و هد در ابتدا در سیلندر 53 باشد و به سمت سیلندرهای بیرونی (با شماره کمتر) در حرکت باشد، ترتیب سرویس‌دهی در روش SCAN چه می‌شود؟

53 → 37 → 14 → تغییر جهت → 65 → 67 → 98 → 122 → 124 → 183

الگوریتم SCAN به الگوریتم آسانسور (elevator algorithm) نیز معروف است چرا که مانند آسانسور یک ساختمان عمل می‌کند. در این روش اگر درخواستی به صف برسد که جلوی هد باشد، این درخواست سریعاً سرویس داده می‌شود. ولی اگر درخواست درست پشت سر هد باشد، بایستی صبر کند تا هد به انتهای دسیک رفته، تغییر جهت داده و برگردد. بنابراین مشکل این روش آن است که تقاضایی که بلافاصله پس از عبور هد از یک سیلندر برای آن سیلندر دریافت می‌شود، به تعویق می‌افتد. روش بعدی تا حدی این مشکل را برطرف می‌سازد.

د) زمانبندی C – SCAN

C – SCAN مخفف Circular Scan یا پویش چرخشی است. این روش نسبت به روش قبلی زمان انتظار یکنواخت‌تری را پدید می‌آورد. در روش C – SCAN مانند SCAN هد در یک جهت (مثلاً از داخل به خارج) حرکت کرده و در مسیر خود به تمام درخواست‌ها سرویس می‌دهد. ولی هنگامی که به انتهای دیسک رسید سریعاً به اول دیسک برمی‌گردد و در این حرکت برگشتی سریع، هیچ سرویس‌دهی انجام نمی‌دهد.

مثال: اگر شماره سیلندرهای درخواستی به صورت 67 , 65 , 124 , 14 , 122 , 37 , 183 , 98 بوده و هد در ابتدا در سیلندر 53 باشد و به سمت سیلندرهای داخلی (با شماره بیشتر) سرویس‌دهی را انجام می‌دهد، ترتیب سرویس‌دهی در روش C – SCAN چه خواهد بود؟

حل:

53 → 65 → 67 → 98 → 122 → 124 → 183 → 14 → 37

هـ) زمانبندی LOOK و C – LOOK

این دو روش اصلاح شده دو روش SCAN و C – SCAN می‌باشند که در آن‌ها الزاماً حرکت از ابتدای دیسک شروع نمی‌شود و تا آخرین سیلندر نیز ادامه نمی‌یابد بلکه از اولین درخواست شروع شده و به آخرین درخواست ختم می‌گردد. یعنی در مثال‌های قبلی اگر تعداد کل

سیلندرها 300 باشد، در روش SCAN یا C – SCAN پس از سرویس‌دهی به سیلندر 183 هد به سراغ سیلندر 300 رفته و دوباره به سمت دیگر دیسک برمی‌گردد. ولی در روش LOOK پس از سرویس‌دهی به سیلندر 183، چون جلوتر از خود سیلندر منتظر سرویسی را نمی‌بیند از همان جا برمی‌گردد.

مقایسه الگوریتم‌ها: با آن‌که SSTF رایج است ولی مشکل قحطی‌زدگی را دارد. روش‌های SCAN و C – SCAN در سیستم‌هایی که بار سنگینی روی دیسک وجود دارد، بهتر کار می‌کنند. برای هر صف خاص از درخواست‌ها، ممکن است یک ترکیب بهینه قابل یافتن باشد، ولی محاسبات لازم برای یافتن آن هزینه‌بر است. احتمالاً در آینده الگوریتم‌های زمانبندی دیسک، در کنترلر سخت‌افزاری درون دیسک انجام می‌پذیرند. عادلانه‌ترین الگوریتم FCFS می‌باشد ولی این روش از نظر سرعت بهینه نیست. زمانبندی SSTF به سود سیلندرهای میانی نسبت به سیلندرهای داخلی‌تر و خارجی‌تر متمایل است.

۴) اعمال ملاحظاتی خاص در جای‌دهی رکوردها

اعمال ملاحظات خاص بستگی به ساختار فایل و وضعیت داده‌ها دارد.

به طور مثال: جای‌دهی رکوردها براساس بسامد (تکرار) دستیابی به آن‌ها است که موجب می‌شود تا متوسط زمان استوانه‌جویی کاهش یابد. رکوردهای با بسامد بالای دستیابی را، در استوانه‌های میانی دیسک و رکوردهای با بسامد کمتر را، در استوانه‌های بیرونی‌تر، جای داده می‌شوند.

مثال ۱: مثلاً، در طرح تخصیص ناپیوسته فضا برای بلاک‌های دیسک، با استفاده از روش I – node، می‌توان I-nodeها را در استوانه‌های میانی ذخیره کرد.

مثال ۲: به عنوان مثال، می‌توان از محیط فیزیکی ذخیره‌سازی سیستم رزرواسیون خطوط هوایی نام برد. رکوردهایی که حاوی اطلاعاتی در مورد رزرواسیون هستند، مرتباً مورد مراجعه قرار می‌گیرند. در حالی که رکوردهای حاوی اطلاعات آماری در مورد مسافران حتی ممکن است فقط ماهی یک بار خوانده شوند، بسامد دستیابی (Access Frequency) به رکوردهای اخیر طبعاً خیلی کمتر خواهد بود.

نکته: یکی دیگر از ملاحظاتی که می‌توان اعمال کرد و جنبه سخت‌افزاری دارد، عبارتست از استفاده از دیسک‌های سریعتر برای ذخیره‌سازی رکوردهایی که بسامد دستیابی به آن‌ها بالا است.

■ تدابیری برای تسریع پردازش فایل

۱) برای ایجاد حافظه نهان برای دیسک (Disk Cache) (گاه موسوم به بافر نهان) (Buffer Cache)

۲) فایل‌های نگاشته در حافظه اصلی (Memory Mapped files)

۳) طراحی سیستم فایل کاراتر

۱) ایجاد حافظه نهان برای دیسک

این حافظه نهان در اساس بافری است بسیار بزرگ که می‌تواند چندین بلاک را در خود جای دهد. وقتی که سیستم می‌خواهد بلاک حاوی رکورد مورد نظر برنامه کاربر را بخواند، علاوه بر این بلاک، تعدادی بلاک دیگر را نیز طی یک عمل واحد انتقال فیزیکی به این بافر انتقال

می‌دهد، سپس بلاک حاوی رکورد مورد نظر به بافر کوچکتری که در اختیار سیستم فایل است، منتقل می‌شود و از این‌جا رکورد، در اختیار برنامه کاربر قرار می‌گیرد.

نکته۱: برای پیاده‌سازی این بافر، یک راه این است که بافری بزرگ در حافظه اصلی، (در صورتی که سیستم تنگناهای حافظه‌ای نداشته باشد) در منطقه بافرها و یا در ناحیه‌ای خاص در بخشی از حافظه اصلی در نظر گرفته شود. راه دیگر این‌که، بافر بزرگ در کنترلر دستگاه ذخیره‌سازی باشد.

کاربرد: اگر فایل به طور تصادفی پردازش شود، این تکنیک باعث کاهش کارایی می‌شود. زیرا اساساً برای هنگامی که پردازش فایل به صورت انبوه انجام گیرد، طراحی شده است.

۲) فایل‌های نگاشته در حافظه اصلی

این تکنیک در سیستم‌های عاملی که برای مدیریت حافظه، از حافظه مجازی Virtual Memory استفاده می‌کنند، در مبادله داده‌ها بین حافظه ثانوی و حافظه اصلی به کار برده می‌شود.

نکته۱: حافظه به قسمت‌هایی با اندازه مساوی موسوم به صفحه تقسیم شده است. اندازه صفحه در سیستم‌های مختلف، معمولاً بین 256 تا چند هزار بایت است، ولی در یک سیستم خاص، صفحات هم اندازه‌اند.

نکته۲: در این سیستم‌ها فضای آدرسی (Address Space) یک پردازش (فرارونده) حاوی برنامه و داده‌ها، نیز به صفحاتی تقسیم می‌شود. آدرس‌های فضای آدرسی فرارونده، توسط یک مکانیسم نگاشت در حافظه اصلی به آدرس‌های حافظه اصلی تبدیل می‌گردند. صفحات فرارونده روی رسانه ذخیره‌سازی سریع، معمولاً دیسک، جای دارند (گسترش حافظه اصلی). برای اجرای فرارونده، در هر لحظه صفحاتی از آن، به حافظه اصلی آورده می‌شوند.

نکته۳: ممکن است صفحه‌ای که در یک لحظه از اجرای فرارونده، مورد نیاز باشد، هنوز به حافظه اصلی آورده نشده باشد، در این صورت ***حالت نقص صفحه (Page fault) بروز می‌کند.*** اجرای فرارونده متوقف می‌شود تا صفحه مورد نیاز در حافظه اصلی نگاشته شود. ***در این سیستم‌ها، مرتباً، صفحات بین حافظه ثانوی و حافظه اصلی برده و آورده می‌شوند.***

۳) طراحی سیستم فایل کاراتر

سیستم فایل هم، مثل هر تولید دیگری در مهندسی نرم‌افزار، می تواند دستخوش دگرگونی شود. سیستم فایل با ساختار لاگ (Log Structure File System (LFS))، که در زیر به آن اشاره می‌کنیم، یکی از این این تغییرات است.

در این سیستم فایل، ایده اصلی این است که بخش قابل توجهی از درخواست‌های خواندن را مستقیماً با استفاده از حافظه نهان خاص سیستم فایل پاسخ داد بی‌آن‌که نیازی به دستیابی مکرر به دیسک باشد. بنابراین اکثر دستیابی‌های به دیسک به منظور انجام عمل نوشتن خواهد بود و نیز مکانیسم خواندن پیشرس (Read ahead) دیگر چندان کارا نخواهد بود.

با توجه به این نکات، طراحان سیستم LFS چنین اندیشیدند که در هر مرحله ایجاد فایل، تمام داده‌های نوشتنی را در بافر جای دهند و سپس محتوای بافر(ها) به طور متناوب در یک سگمنت از دیسک نوشته شود (در انتهای فایلی به نام Log). در واقع در این روش، کل دیسک به صورت یک Log دیده می‌شود که چندین سگمنت دارد و در ابتدای هر سگمنت، اطلاعاتی در مورد محتوای سگمنت وجود دارد.

برای کسب اطلاع بیشتر از ساختار LFS، می‌توان به منابع سیستم‌عامل از جمله سیستم‌عامل یونیکس مراجعه کرد.

مجموعه سوالات کنکور نمونه

۲۶ - کدام لوکالیتی درجه‌اش از بقیه موارد بالاتر است؟

۱) رکورد بعدی در استوانه فعلی است ولی در بافر نیست.

۲) رکورد بعدی در استوانه همجوار است.

۳) رکورد بعدی روی استوانه‌ای است که شماره‌اش با انجام محاسباتی بدست می‌آید.

۴) رکورد بعدی روی استوانه هم شماره با استوانه فعلی و در یک volume دیگر است.

۲۷ - در بافرینگ مضاعف در چه صورتی کارآیی و سرعت انتقال کاهش می‌یابد؟

۱) فایل خواندنی/ نوشتنی از نوع ترتیبی باشد.

۲) سرعت انتقال یک رکورد از سرعت پردازش محتوای بافر فعلی بیشتر باشد.

۳) سرعت پردازش پردازنده از سرعت انتقال یک بلاک بیشتر باشد.

۴) سرعت پردازش محتوای بافر از سرعت انتقال یک بلاک کمتر باشد.

۲۸ - در کدام مورد زمان استوانه‌جویی تغییری نمی‌کند؟

۱) افزایش اندازه شکاف (GAP) بین سکتورهای شیار

۲) افزایش بازوهای دیسک به تعداد شیارهای موجود

۳) توزیع فایل روی چند دیسک

۴) کنترل نحوه حرکت بازوی هد برای جوابگویی به درخواستها در یک محیط چندبرنامه‌ای

۲۹ - از نظر تعداد بافرهایی که به عملیات برنامه پردازشگر فایل تخصیص می‌یابد چند نوع بافرینگ وجود دارد؟

۱) دو نوع (بافرینگ ساده - چندگانه)

۲) سه نوع (بافرینگ ساده - مضاعف - چندگانه)

۳) چهارنوع (بافرینگ ساده - چندگانه)

۴) پنج نوع (بافرینگ ساده - مضاعف - سه‌گانه - چندگانه - جدول تخصیص فایلها FAT)

۳۰ - برای تعیین محدوده رکورد در بلاک برای رکوردهای با طول متغیر چند نوع تکنیک وجود دارد؟

۱) متغیر (محدوده رکورد - طول متغیر - اندازه بلاک در رکورد متغیر)

۲) 1 تا (ایجاد جدول مکان‌نما)

۳) 2 تا (درج طول - ایجاد جدول مکان‌نما)

۴) 3 تا (درج نشانگر پایان رکورد - درج طول - ایجاد مکان‌نما)

۳۱- مجموعه‌ای که از تعدادی بلاک تشکیل شده و از نظر سیستم فایل یک واحد مبادله‌ای است و طی یک دستور خواندن به بافر منتقل می‌شود:

۱) باکت Bucket ✓ ۲) بلوک Block ۳) شیار Track ۴) سکتور Sector

۳۲- کدام گزینه نادرست است؟

۱) سیستم فایل نرم‌افزاری است که میزان پیچیدگی و حجم آن به نوع ساختار فایلی که باید ایجاد کند بستگی دارد.

۲) وظیفه بخش منطقی سیستم فایل انجام درخواستهای کاربر می‌باشد.

۳) وظیفه بخش فیزیکی سیستم فایل دستیابی فیزیکی به فایلها می‌باشد.

✓۴) هرچه لوکالیتی (هم محلی بودن) رکوردها قویتر باشد زمان پردازش سریال بیشتر خواهد شد.

۳۳- کمترین میزان حافظه هرز مربوط به تکنیک بلاک‌بندی ..... می‌باشد.

۱) رکوردهای با طول ثابت و یکپاره ۲) رکوردهای با طول متغیر و یکپاره

۳) رکوردهای با طول متغیر و دوپاره ✓ ۴) هر دو گزینه ۱ و۲

۳۴- در کدام تکنیک بلاک‌بندی می‌توان رکوردی بزرگ‌تر از اندازه بلاک ذخیره کرد؟

۱) بلاک‌بندی رکورد باطول متغیر و یکپاره ۲) بلاک‌بندی رکورد باطول ثابت و یکپاره

✓۳) بلاک‌بندی رکورد باطول متغیر و دوپاره ۴) هر دو گزینه۱ و۲

۳۵- کدام اطلاعات، در بخش غیرداده‌ای رکورد در نشست فیزیکی قرار ندارد؟

۱) فلاگ حذف ۲) فلاگ قفل رکورد ✓ ۳) کلید رکورد ۴) طول رکورد متغیر

۳۶- از کدام تکنیک برای کاهش زمان درنگ دورانی استفاده می‌شود؟

۱) الگوریتم‌های مناسب جهت حرکت نوک خواندن و نوشتن

۲) چندین نوک خواندن و نوشتن

۳) جای دادن بلاک‌ها به طور چند در میان

۴) قرار دادن نقطه شروع شیارها به صورت منحنی

۳۷- فایلی با 10^4 رکورد ثابت و $B_f = 10$ و چگالی لود اولیه 60% مفروض است. تعداد بلاک‌های اشغال شده توسط فایل کدام است؟

۱) 1600 ۲) 1601 ۳) 1666 ۴) 1667

۳۸- اگر تقاضای خواندن شیارها به ترتیب از چپ به راست به صورت زیر باشد و نوک خواندن و نوشتن در شیار 17 قرار داشته باشد، چنانچه حرکت نوک طبق الگوریتم SSTF کنترل شود، ترتیب خواندن شیارها چگونه می‌باشد؟ 3,15,7,29,16

۱) 16 ,15,29,7,3 ۲) 16,15,7,3,29

۳) 29,16,15,7,3 ۴) 3,7,15,16,29

۳۹- کمترین میزان حافظه هرز مربوط به کدام تکنیک بلاک‌بندی زیر می‌باشد؟

۱) رکوردهای با طول متغیر و دوپاره
۲) رکوردهای با طول ثابت و یکپاره
۳) رکوردهای با طول متغیر و یکپاره
۴) موارد۲ و۳

۴۰- از نظر تعداد بافرهایی که به عملیات برنامه پردازشگر فایل تخصیص می‌یابد چند نوع بافرینگ وجود دارد؟

۱) دو نوع (بافرینگ ساده ـ چندگانه)
۲) سه نوع (بافرینگ ساده ـ مضاعف ـ چندگانه)
۳) چهارنوع (بافرینگ ساده ـ مضاعف ـ سه‌گانه ـ چندگانه)
۴) پنج نوع (بافرینگ ساده ـ مضاعف ـ سه‌گانه ـ چندگانه جدول تخصیص فایلها FAT)

۴۱- در یک برنامه، برای دسترسی به یک رکورد خاص اعلام شده، «رکورد مربوط به رئیس» این نشانی‌دهی به کدام روش است؟

۱)آدرس
۲) محتوایی
۳) نسبی
۴) نمادی

۴۲- کاربرد اشاره‌گرها در فایل باعث ایجاد نظم ..... می‌شود.

۱) فیزیکی فایل در محیط فیزیکی
۲) فیزیکی فایل در محیط منطقی
۳) منطقی فایل در محیط فیزیکی
۴) منطقی فایل در محیط منطقی

۴۳- مشکل اصلی بلاک‌بندی رکوردها با طول متغیر و به صورت یکپارچه کدام است؟

۱) متغیر شدن طول رکورد باعث می‌شود نتوان تعداد رکوردها در بلاک را به طور تقریبی تخمین زد.
۲) متغیر شدن طول رکورد باعث می‌شود نتوان رکوردها را دقیق آدرس‌دهی کرد.
۳) متغیر شدن طول رکورد می‌تواند باعث شود رکورد از بلاک بزرگتر شده و نتوان آنرا درج کرد.
۴) یک پارچه بودن طول رکورد باعث می‌شود نرم‌افزار پیچیده‌ای را برای مدیریت آن بکار بریم.

۴۴- کمترین مقدار داده‌ای که در یک عمل ورودی / خروجی توسط سیستم فایل بیرون و درون ماشین مبادله می‌شود کدام گزینه است؟

۱) بلاک
۲) رکورد
۳) فایل
۴) فیلد

۴۵- بخش غیر داده‌ای در یک رکورد فیزیکی چیست؟

۱) فیلد طول رکورد، فیلد نوع رکورد، فیلد اشاره‌گر
۲) فیلد طول رکورد، فیلد نوع رکورد، فلگها ـ نوع فایل
۳) فیلد اشاره‌گر ـ فیلد طول رکوردها، فلگها
۴) فیلد اشاره‌گر، فیلدطول رکورد، فیلد نوع رکورد، فلگها

۴۶- کدام مورد جزو معایب بلاک‌بندی است؟

۱) افزایش دفعات ورودی ـ خروجی
۲) ذخیره شدن رکورددر دو بلاک همجوار
۳) صرفه‌جویی در مصرف رسانه ذخیره‌سازی
۴) مصرف زیاد حافظه‌اصلی به دلیل بافرینگ

۴۷ـ در کدام تکنیک از تولید نسخه پشتیبان، عمل نوشتن در هر دو دیسک انجام می‌شود ولی عمل خواندن فقط از یک دیسک صورت می‌گیرد؟

۱) آینه‌سازی ✓
۲) استفاده از نیمه دوردیسک
۳) تولید دامپهای تدریجی (Incremental dump)
۴) I-node (indexnode)

۴۸ـ کاربری در یک برنامه به زبان پاسکال نوشته است:

```
seek (my file, 8)
read myfile, my Area
```

اگر بایت B=1000 و بایت R=500 و $RBA_{BOF} = 10$ باشد، RBA_{rw} عبارت است از:

۱) 3
۲) 13
۳) 23
۴) 24

۴۹ـ دید سیستم فایل منطقی نسبت به یک فایل کدام عبارت است؟

۱) فایل از تعدادی باکت تشکیل شده است که تقسیماتی مثل پاکت، خوشه در آن دیده می‌شود.
۲) قالبی است با یک ساختار مشخص که شامل تعدادی رکورد است و هر رکورد دارای طول معینی است.
۳) ✓ مجموعه‌ای از رکوردهای ذخیره شده می‌باشد که یک ساختار مشخص داشته و دستیابی به آن با یک شیوه مشخص است.
۴) مقدار داده‌ای است که در یک عمل I/O بین بیرون و درون ماشین مبادله می‌شود.

۵۰ـ در محاسبه فاکتور بلاک‌بندی روی یک رسانه ذخیره‌سازی طول بلاک 2000 byte و طول رکورد 100 byte است در صورتی که رکوردها بلاک‌بندی‌شان به صورت متغیر و دوپاره باشد مقدار B_f کدام است در صورتی که طول فیلد نشانه‌رو 10 بایت باشد؟

(آموزشکده‌های فنی ـ ۸۳)

۱) 18
۲) 20
۳) 22
۴) 23

۵۱ـ اگر طول رکورد در فایلی 30 بایت و طول سکتور 256 بایت و $B_f = 1$ باشد میزان واقعی استفاده از دیسک چند درصد است؟

(آموزشکده‌های فنی ـ ۸۳)

۱) 93
۲) 78
۳) 23
۴) 12

۵۲ـ کدام مورد کاهش زمانه استوانه‌جویی را باعث نمی‌شود؟ (آموزشکده‌های فنی ـ ۸۳)

۱) استفاده از دیسک‌های با بازوی ثابت
۲) استفاده از الگوریتم‌های مناسب برای حرکت دادن بازوی دیسک
۳) توزیع کردن فایل‌ها روی چند دیسک
۴) ✓ جای دادن بلاک‌ها روی شیار بدون رعایت ترتیب پردازش آن‌ها

۵۳ـ کدام سطح وظیفه تبدیل آدرس در برنامه پردازشگر (RBA) را به عهده دارد؟ (آموزکشده‌های فنی ـ ۸۳)

۱) سطح برنامه پردازشگر فایل
۲) سطح خارجی سیستم فایل
۳) سطح فیزیکی سیستم فایل
۴) ✓ سطح منطقی سیستم فایل

۵۴ـ کدام گزینه نشان‌دهنده اعمال اساسی در محیط فیزیکی برای یک سیستم فایل است؟ (کارشناسی ناپیوسته ـ دولتی ـ ۸۳)

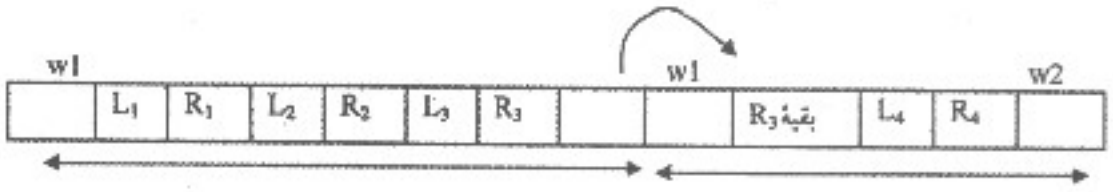
۱) مکان‌یابی، باز کردن و خواندن از رسانه

۲) مکان‌یابی، خواندن بلاک‌ها و نوشتن بر روش بلاک‌ها

۳) مکان‌یابی، خواندن از رسانه، نوشتن بر روی رسانه

۴) باز کردن فایل، خواندن فایل، نوشتن بر روی فایل

۵۵ـ شکل زیر نشان‌دهنده بلاک‌بندی رکوردها در بلاک‌های B1 و B2 است، فضای مقداری هر رکورد (طول بخش داده‌ای + طول بخش غیرداده‌ای) به صورت $R_1 = 10$ و $R_2 = 5$ ، $R_3 = 10$ ، $R_4 = 11$ است، در صورتی که طول فیلد نشانه‌رو 6 بایت و در هر بلاک تعداد 2 رکورد وجود داشته باشد، طول بلاک B عبارت است از: (کارشناسی ناپیوسته ـ دولتی ـ ۸۳)



۱) 90 ۲) 72 ۳) 36 ۴) 18

۵۶ـ کدام نوع بافرینگ به صورت چرخشی پیاده‌سازی می‌شود؟ (کارشناسی ناپیوسته ـ دولتی ـ ۸۳)

۱) Singel Buffering ۲) Double Buffering ۳) Multiple Buffering ۴) هیچکدام

۵۷ـ هنگامی که به دلیل وجود محدودیت در تخصیص حافظه نمی‌توان به فایل دو بافر اختصاص داد، کدام تکنیک در کاهش زمان درنگ دورانی بهینه است؟

۱) تغییر مکان نقطه آغاز شیارها ۲) دیسک‌های با بازوی ثابت

۳) پراکنده‌خوانی ۴) تداخل بلاک‌ها

۵۸ـ در سطح رسانه ذخیره‌سازی مثل دیسک در صورتی که رکورد مورد نظر رکورد 21ام باشد و ظرفیت رکورد متوسط $R = 500$ و $B = 1000$ (ظرفیت بلاک) شمارهٔ بلاک مورد نظر BLK # در حالتی که $\begin{cases} \text{RBA begin device} = 5 \\ b_i = 3 \end{cases}$ باشد، کدام است؟

۱) 1 ۲) 2 ۳) 10 ۴) 12

جواب تشریحی سوالات به همراه سوالات پیشنهادی دیگر در ۵۰ درصد دوم داده می‌شود.

ارزیابی پارامترهای رسانه (نوار ـ دیسک)

ارزیابی نوار

۱ـ ظرفیت نوار
- ۱ـ ظرفیت واقعی
- ۲ـ درصد استفاده واقعی از نوار
- ۳ـ ظرفیت اسمی

باتوجه به مطالبی که در فصل اول عنوان شد موارد ۱ و ۲و ۳به شرح زیر قابل محاسبه هستند:

$G =$ طول یا اندازه گپ $_{byte}$ $\quad B =$ طول یا اندازه بلاک $_{byte}$ $\quad D =$ چگالی نوار $_{byte/ft}$ $\quad L =$ طول نوار $_{ft}$

$$\text{ظرفیت اسمی} = L \times D$$

$$\text{درصد استفاده واقعی} = \frac{B}{B+G} \times 100$$

$$\text{درصد استفاده واقعی} * \text{ظرفیت اسمی} = \text{ظرفیت واقعی} \quad = L \times D \times \frac{B}{B+G}$$

نکته مهم: عامل اصلی کاهش میزان استفاده واقعی نوار ***گپ بین بلاک‌ها*** است، چون هرچه فضای گپ زیاد باشد، درصد استفاده واقعی کمتر و در نتیجه ظرفیت واقعی کمتر می‌شود در عین حال افزایش چگالی ـ طول نوار ـ طول بلاک باعث افزایش ظرفیت واقعی نوار می‌شود.

نکته: در محاسبه ظرفیت واقعی می‌توان به جای B از $B - W_B$ استفاده کرد، W_B فضای هرز به ازاء یک بلاک به جز گپ می‌باشد.

مثال: فایلی را در نظر می‌گیریم با 10000 رکورد 80 بایتی روی نواری با چگالی 1600 bpi وجود دارد، در نتیجه طول این فایل از رابطه زیر بدست می‌آید:

$$\text{اندازه فایل}_{byte} = \text{طول فایل}_{inch} \times \text{چگالی}_{byte/inch} \Rightarrow 80 \times 10000 = \text{طول فایل} \times 1600 \Rightarrow \text{طول فایل} = \frac{800000}{1600} = 500 \text{ inch}$$

اگر $B_f = 1$ و طول IBG برابر 0.5 اینچ باشد، برای گپ‌های نوار $10000 \times 0.5 = 5000$ اینچ مصرف می‌شود، و درصد استفاده واقعی $\frac{500}{5500} = 9.09\%$ می‌باشد.

■ نرخ انتقال واقعی نوار

پارامترهائی که در ارزیابی نرخ انتقال واقعی نوار دخالت دارند:

۱ـ طول گپ (G) ← بایت

۲ـ طول بلاک (B) ← بایت

۳ـ نرخ انتقال اسمی (t) ← بایت بر ثانیه

۴ـ نرخ انتقال واقعی (t′) ← بایت برثانیه

۵ ـ زمان حرکت / توقف (τ) ← میلی‌ثانیه

نکته مهم: عامل اصلی کاهش نرخ انتقال واقعی همان گپ است.

■ روش‌های (اسلوب) خواندن نوار

۱ـ اسلوب بلاکی: نوار پس از خواندن یک بلاک متوقف می‌شود.

۲ـ اسلوب جریانی: نوار پس از خواندن N بلاک متوقف می‌شود.

۱ـ اسلوب بلاکی

با آن‌که زمان خواندن یک بلاک $\frac{B}{t}$ (بدون در نظر گرفتن عوامل کاهنده)، اما زمان واقعی برابر است با:

$$\frac{B}{t} + \frac{G}{t} + \frac{\tau}{1000}$$

$\frac{B}{t}$ = زمان خواندن بلاک (ثانیه)

$\frac{G}{t}$ = زمان طی شدن گپ (ثانیه)

$\tau_{sec} = \frac{\tau_{ms}}{1000}$ = زمان حرکت / توقف (ثانیه)

از رابطه بالا می‌توانیم در نظر بگیریم:

$$k_{(1)} = \frac{\frac{B}{t}}{\frac{B}{t} + \frac{G}{t} + \frac{\tau}{1000}} < 1$$ درصد نرخ انتقال واقعی

(نرخ انتقال اسمی) $\times$ $k \times t = k_{(1)} \times t'$ (نرخ انتقال واقعی)

۲ـ اسلوب جریانی

در این اسلوب N بلاک را می‌خوانیم، بنابراین نرخ انتقال واقعی t' به صورت زیر بدست می‌آید:

زمان خواندن N بلاک به صورت غیرواقعی: $N \times \frac{B}{t}$

زمان خواندن N بلاک به صورت واقعی: $N \times \frac{B}{t} + N \times \frac{G}{t} + \frac{\tau}{1000}$

$\frac{\tau}{1000}$: زمان حرکت / توقف

$N \times \frac{G}{t}$: زمان طی شدن N گپ

$N \times \frac{B}{t}$: زمان خواندن N بلاک

$$k_{(N)} = \frac{N \times \frac{B}{t}}{N \times \frac{B}{t} + N \times \frac{G}{t} + \frac{\tau}{1000}} < 1$$ درصد نرخ انتقال واقعی

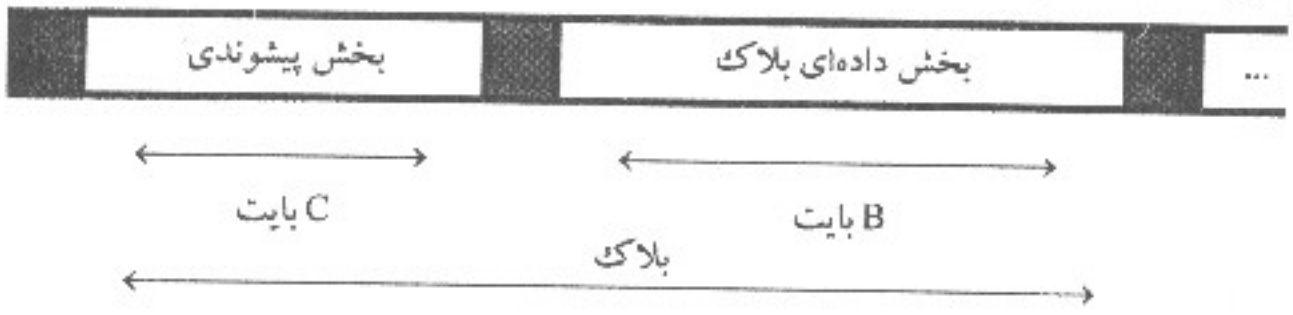
(نرخ انتقال اسمی) $\times$ $k_{(N)} \times t = k_{(N)}$ = نرخ انتقال واقعی

ارزیابی دیسک

■ ظرفیت واقعی

برای ارزیابی میزان استفاده واقعی از دیسک باید با توجه به نحوه فرمت‌بندی شیارها عمل کنیم،

با توجه به آن‌که برای هر بلاک یک بخش پیشوندی (کنترلی) در نظر گرفته می‌شود.



دو قسمت اصلی بلاک

اگر ظرفیت اسمی هر شیار را C_{N_T} بدانیم آن‌گاه:

$$T_f = \left\lfloor \frac{C_{N_T}}{C + B} \right\rfloor$$

T_f = فاکتور تراکینگ تعداد بلاک‌ها در شیار

در نتیجه:

ظرفیت بلاک‌های شیار

$$C_{E_T} = \frac{\overbrace{T_f \times B}}{\underbrace{C_{N_T}}} \times 100$$

درصد استفاده واقعی از شیار

ظرفیت اسمی شیار

و اگر دقیق‌تر بیان کنیم:

$$C_{E_T} = \frac{T_f \times (B - W_B)}{C_{N_T}}$$

W_B = میزان حافظه هرز درون بلاک

مثال: فرض کنید ظرفیت اسمی یک شیار $C_{N_T} = 1200$ بایت باشد. و طول بلاک $B = 60$ بایت و طول بخش پیشوندی $C = 36$ بایت باشد

مطلوبست محاسبه:

الف) T_f (تعداد بلاک‌ها در شیار)

$$T_f = \left\lfloor \frac{C_{N_T}}{C + B} \right\rfloor = \left\lfloor \frac{1200}{60 + 36} \right\rfloor = 12$$

ب) C_{E_T} (درصد استفاده واقعی از شیار)

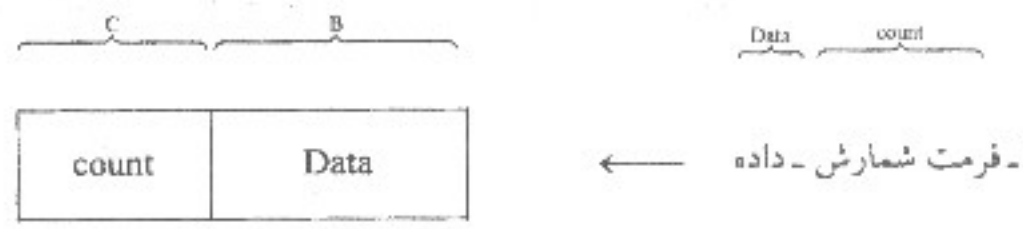
$$C_{E_T} = \frac{T_f \times B}{C_{N_T}} = \frac{12 \times 60}{1200} = \%60$$

نکته ۱: کاهش در ظرفیت واقعی عمدتاً ناشی از وجود گپ‌ها است. اما در عین حال به ***طول بلاک*** و ***ظرفیت شیار*** و ***نحوه فرمت‌بندی*** آن نیز بستگی دارد.

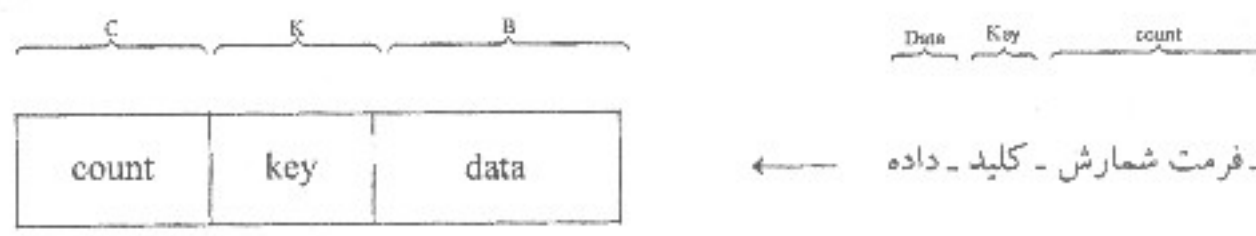
نکته ۲: میزان واقعی استفاده از شیار، اساساً به نحوه بلاک‌بندی و فرمت‌بندی شیار بستگی دارد. و علت کاهش ظرفیت واقعی نسبت به ظرفیت اسمی دیسک نحوه فرمت‌بندی دیسک است چون در هنگام فرمت‌بندی مقداری از فضای دیسک جهت ضبط اطلاعات پیشوندی رکوردها در نظر گرفته می‌شود که باعث کاهش ظرفیت واقعی نسبت به ظرفیت اسمی می‌شود.

■ فرمت‌های مختلف برای بلاک

فرمت‌بندی بلاک‌های هر شیار به دو روش زیر خواهد بود:



۱ـ فرمت شمارش ـ داده ←



۲ـ فرمت شمارش ـ کلید ـ داده ←

اگر T_f تعداد بلاک‌های شیار باشد:

$$T_f = \left\lfloor \frac{C_{N_1}}{C + B} \right\rfloor \qquad \text{روش 1}$$

$$T_f = \left\lfloor \frac{C_{N_2}}{C + k + B} \right\rfloor \qquad \text{روش 2}$$

و درنتیجه درصد استفاده واقعی از شیار:

$$\frac{T_f \times B}{C_{N_1}} \times 100$$

☑ دیسک‌های سکتوربندی شده: (مهم)

در صورتیکه تعداد درستی از رکوردها را در تعداد درستی از سکتورها جای دهیم، حافظهٔ هرزی انتهای سکتورها ایجاد نمی‌شود، در غیر این صورت فضای هرزی در انتهای سکتور ایجاد می‌شود. در عین حال در صورتیکه رکوردها بلاک‌بندی شوند، و بلاک‌ها روی سکتورها قرار گیرند، در صورتیکه رکوردها کوتاه و بلاک‌ها کوچک باشند درصد استفاده واقعی از دیسک سکتوربندی شده کاهش می‌یابد.

سکتور		سکتور		سکتور		سکتور		سکتور	
رکورد 1	رکورد 2	رکورد 3	رکورد 4	رکورد 5	رکورد 6	رکورد 7	رکورد 8	رکورد 9	رکورد 10

جای‌دهی تعداد درستی رکورد در تعداد درستی بلاک

■ **محاسبه درصد استفاده واقعی دیسک سکتوربندی: (مهم)**

$E = \frac{R \times B_f}{L_s \times N} \times 100$ درصد استفاده واقعی از دیسک

R = طول رکورد L_s = طول سکتور (پیش فرض 512 بایت)

$B_f = \left\lfloor \frac{B}{R} \right\rfloor$ = تعداد رکوردهای هر بلاک N = تعداد سکتور در بلاک (در صورت عدم بیان $N = \left\lceil \frac{R \times B_f}{L_s} \right\rceil$ محاسبه می‌شود)

مثال ۱: در یک فایل که طول هر رکورد در آن 30 بایت و $B_f = 1$ و طول هر سکتور 256 بایت است مطلوبست درصد استفاده واقعی از دیسک سکتوربندی شده؟

$$\left.\begin{array}{l} R = 30 \text{ byte} \\ B_f = 1 \\ L_s = 256 \text{ byte} \end{array}\right\} \Rightarrow N = \left\lceil \frac{R \times B_f}{L_s} \right\rceil = \left\lceil \frac{30 \times 1}{256} \right\rceil = 1$$

چون سکتورها باید بلاک‌ها را بپوشانند در نتیجه $R \times B_f$ اندازه بلاک است که برابر 30 بایت بوده و توسط یک سکتور 256 بایت پوشانده می‌شود.

$$E = \frac{R \times B_f}{L_s \times N} = \frac{30 \times 1}{256 \times 1} \times 100 = \%12$$

مثال ۲: فایلی را در نظر بگیرید که طول رکوردهای آن 160 بایت و طول سکتور آن 256 بایت باشد، اگر فاکتور بلاک بندی برابر 5 و تعداد 4 سکتور در هر بلاک باشد، درصد استفاده واقعی از دیسک چند درصد است؟

۱) 91 ۲) 85 ✓ ۳) 78 ۴) 50

$$\left.\begin{array}{l} R = 160 \\ L_s = 256 \\ B_f = 5 \\ N = 4 \end{array}\right\} \Rightarrow E = \frac{R \times B_f}{L_s \times N} = \frac{160 \times 5}{4 \times 256} = \%78$$

در صورتیکه N بیان نمی‌شد به راحتی از رابطه $N = \left\lceil \frac{R \times B_f}{L_s} \right\rceil = \left\lceil \frac{160 \times 5}{256} \right\rceil = 4$ بدست می‌آمد.

■ نرخ انتقال واقعی دیسک‌ها

نرخ انتقال واقعی در دیسک‌ها به عوامل زیر بستگی دارد:

۱ـ نوع بافرینگ

۲ـ شیوه دستیابی به بلاک (مستقیم ـ ترتیبی)

۳ـ زمان پردازش محتوی بلاک C_B

۴ـ طرز ذخیره‌سازی بلاک‌ها روی شیار

۵ ـ طرز پردازش رکوردها

۶ ـ طرز دستیابی برنامه به رکورد مورد نظر

■ **دستیابی مستقیم به بلاک: (مهم)**

هرگاه سیستم آدرس بلاک حاوی رکورد را داشته باشد، می‌تواند آن را مستقیم بخواند. *زمان خواندن مستقیم بلاک* $s + r + b_{tt}$ *است*

بنابراین نرخ انتقال به صورت زیر محاسبه می‌شود:

بایت زمان خواندن B بایت بلاک

$$\frac{\overbrace{s + r + b_{tt}}^{}}{\underbrace{1}} \quad \frac{\overbrace{B}^{}}{\underbrace{t}} \Rightarrow \boxed{t = \frac{B}{s + r + b_{tt}}} \xrightarrow{\text{دقیق‌تر}} \boxed{t = \frac{B - W_B}{s + r + b_{tt}}}$$

چند بایت در یک ثانیه خوانده می‌شود. در 1 ثانیه

W_B میزان حافظه هرز داخل بلاک می‌باشد.

مثال: زمان خواندن مستقیم یک بلاک 2 ثانیه است. در صورتیکه طول بلاک $B = 10000\ byte$ و مقدار حافظه هرز بلاکی 2000 بایت باشد نرخ انتقال واقعی چند $k\ byte/sec$ است؟ (کنکور ۸۲)

۱) 0.309 ۲) 3.9 ۳) 4.048 ۴) 4000

$s + r + b_{tt}$ = زمان خواندن مستقیم بلاک = 2 ثانیه

$B = 10000\ byte$

$W_B = 2000\ byte$

$$\rightarrow t = \frac{B - W_B}{s + r + b_{tt}} = \frac{8000}{2} = 4000\ byte/sec$$

$$t = 4000\ byte/sec = 4000/1024\ k\ byte/sec = 3.9\ k\ byte/sec$$

نکته ۱: مهم b_{tt} زمان انتقال بلاک است (block transfer time) و ebt یا b'_{tt} زمان انتقال بلاک به همراه گپ آن است. (effective block transfer time) درنتیجه همیشه $ebt > b_{tt}$ خواهد بود.

نکته ۲: مهم

اگر نرخ انتقال اسمی را با t و نرخ انتقال واقعی را با t' نمایش دهیم همیشه $t' < t$

■ **زمان خواندن بلاک‌ها و رکوردهای فایل به صورت تصادفی:**

زمان خواندن b بلاک به صورت تصادفی برابر است با:

$$T = \frac{b \times B(s + r + b_{tt})}{B - W_B}$$

و اگر $W_B = 0$ در نظر بگیریم:

$$T = b(s + r + b_{tt})$$

زمان کل پردازش فایل

به طور کلی دو حالت وجود دارد:

۱ـ پردازش رکوردی

۲ـ پردازش بلاکی

۱ـ پردازش رکوردی: در این نوع پردازش هربار که سیستم، بلاکی را می‌خواند، برنامه فایل پرداز فقط یک رکورد از بلاک را پردازش می‌کند.

زمان پردازش n رکورد، هر رکورد از یک بلاک

$$T_{pfile} = \underbrace{n\left(s + r - b_{tt}\right)}_{\downarrow} + \overbrace{n \times C_R}^{\uparrow}$$

زمان خواندن n بلاک برای پردازش n رکورد

pfile = process file

C_R = زمان پردازش یک رکورد.

۲ـ پردازش بلاکی: در این نوع پردازش هربار که سیستم بلاکی را می‌خواند برنامه فایل پرداز تمام رکوردهای آن بلاک را پردازش می‌کند.

زمان پردازش b بلاک، هر بلاک تمام رکوردها

$$T_{pfile} = \underbrace{b\left(s + r + b_{tt}\right)}_{\downarrow} + \overbrace{b \times C_B}^{\uparrow}$$

زمان خواندن b بلاک

C_B = زمان پردازش یک بلاک

مثال: پردازش یک فایل را دو حالت رکوردی و بلاکی مورد بررسی قرار داده‌ایم، زمان پردازش محتوای رکورد 1 ثانیه و زمان پردازش محتوای بلاک 3 ثانیه است، اگر زمان لازم برای خواندن بلاک 1 ثانیه باشد، فاکتور بلاک‌بندی عبارت است از:

۱) 8 ۲) 6 ۳) 3 ۴) 2

گزینه ۴ صحیح می‌باشد.

C_R = پردازش محتوی رکورد $= 1_{sec}$

C_B = پردازش محتوی بلاک $= 3_{sec}$

$s + r + b_{tt}$ زمان خواندن بلاک $= 1_{sec}$

پردازش بلاکی $= b\left(s + r + b_{tt}\right) + b_{C_B} = b \times 1 + b \times 3 = 4\,b$

پردازش رکوردی $= n\left(s + r + b_{tt}\right) + n_{C_R} = n \times 1 + n \times 1 = 2\,n$

می‌دانیم:

$$b = \frac{n}{B_f} \Rightarrow n = b \times B_f$$

حال اگر در پردازش رکوردی با زمان 2 n بخواهیم بلاکی عمل کنیم زمان 2n به $2 \times b \times Blf$ تبدیل می‌شود.

در نتیجه:

$$2 \times b \times B_f = 4\,b \Rightarrow B_f = 2$$

■ **دستیابی ترتیبی به بلاک‌ها (در پردازش انبوه فایل)**

منظور از دستیابی ترتیبی این است که با شروع از نقطه‌ای از فایل (BOF) بلاک‌ها به ترتیبی که ذخیره می‌شوند مورد دستیابی قرار گرفته و خوانده می‌شوند.

اول فایل

پردازش انبوه: برنامه تعدادی بلاک را پردازش کند، *نرخ انتقال در این وضعیت به C_B و نوع بافرینگ بستگی دارد.*

■ **انواع بافرینگ در دستیابی ترتیبی: (مهم ـ مهم)**

۱ـ بافرینگ ساده و مرتب‌خوانی
۲ـ بافرینگ ساده و درهم خوانی
۳ـ بافرینگ مضاعف و عدم وجود شرط کارائی
۴ـ بافرینگ مضاعف و شرط وجود کارائی

۱ـ بافرینگ ساده و مرتب‌خوانی

در بحث مربوط به روش کاهش زمان درنگ دورانی (تداخل بلاک‌ها) دیدیم زمانیکه سیستم یک بلاک را در بافر می‌خواند با شروع پردازش محتوی بافر آغاز بلاک بعدی در اثر دوران دیسک از زیر نوک خواندن / نوشتن رد می‌شود و برای خواندن آن، پردازنده ورودی / خروجی باید یک دور کامل دیسک انتظار بکشد.

زمان خواندن بلاک بعدی

$$\begin{array}{ccc} 2r + b_{tt} & B & \Rightarrow \boxed{t' = \dfrac{B}{2r + b_{tt}}} \\ 1 & t' & \end{array}$$

میزان بایتی که در 1 ثانیه انتقال داده می‌شود.

البته می‌توانیم B را به صورت $B - W_B$ بنویسیم (W_B = فضای هرز درون بلاکی).

دقت کنید در این‌جا فرض کرده‌ایم $C_B \leq 2r$ در غیر این صورت زمان انتظار پردازشگر 2 دور دیسک می‌شد. (4 r)

۲ـ بافرینگ ساده و درهم‌خوانی

در این وضعیت این محدودیت که باید بلاک‌ها را به ترتیب نشست آن‌ها بخوانیم وجود ندارد. به عبارت بهتر پردازشگر می‌خواهد پردازش روی همه رکوردها انجام دهد. ولی ترتیب خواندن آن‌ها مورد نظر نیست.

اگر $C_B \leq b_{tt}$ باشد در این صورت بلاک‌های شیار در 2 دور خوانده می‌شود.

$$\begin{array}{ccc} 4r & T_f \times B & \Rightarrow \boxed{t' = \dfrac{T_f \times B}{4r}} \\ 1 & t' & \end{array}$$

$T_f \times B$: ظرفیت بلاک های شیار

4r: زمان خواندن بلاک های شیار

t': میزان بایتی که در 1 ثانیه انتقال داده می‌شود.

اگر $C_B > b_{tt}$ باشد باتوجه به C_B تعداد کمتری بلاک در دیسک دور خوانده و پردازش می‌شود.

۳ـ بافرینگ مضاعف و شرط عدم کارائی

باتوجه به عدم وجود شرط کارائی در بافرینگ مضاعف $(C_B > b_u)$، در این وضعیت سیستم ابتدا دو بلاک را در دو بافر می‌خواند و پردازش می‌کند. باتوجه به عدم وجود شرط کارائی وقتی در همین دور دیسک، آغاز بلاک سوم به زیر نوک خواندن / نوشتن می‌رسد بافر اول هنوز اشغال است و لذا سیستم نمی‌تواند بلاک سوم را بخواند به این ترتیب برای خواندن دو بلاک بعدی یک دور دیسک لازم است و خواهیم داشت:

$$\begin{matrix} 2r + 2b_u & 2B \\ 1 & t' \end{matrix} \Rightarrow \boxed{t' = \frac{B}{r + b_u}}$$

$2B$: فضای دو بلاک

$2r + 2b_u$: زمان خواندن دو بلاک بعدی

t': میزان بایتی که در 1 ثانیه انتقال داده می‌شود.

۴ـ بافرینگ مضاعف و شرط کارائی

در صورتیکه در بافرینگ مضاعف شرط کارائی وجود داشته باشد $(C_B \le b_u)$، تمام بلاک‌های شیار در یک دور دیسک خوانده می‌شود، خواهیم داشت:

$$\begin{matrix} 2r & T_f \times B \\ 1 & t' \end{matrix} \Rightarrow \boxed{t' = \frac{T_f \times B}{2r}}$$

$T_f \times B$: فضای بلاک‌های شیار

$2r$: زمان خواندن بلاک‌های شیار

t': میزان بایتی که در 1 ثانیه انتقال داده می‌شود.

در صورتیکه حافظه هرز درون بلاکی را دخالت دهیم: $\boxed{t' = \frac{T_f\left(B - W_B\right)}{2r}}$

نکته: روش دیگر برای ارزیابی نرخ انتقال واقعی، آن است که محاسبه نرخ انتقال را براساس تعداد رکوردهای شیار انجام دهیم (تعداد رکوردهای شیار $= T_f \times B_f$) در نتیجه:

$$\text{زمان خواندن یک شیار} = \frac{\overbrace{T_f \times B_f\left(R + W_R\right)}^{\text{فضای شیار براساس ظرفیت رکورد}}}{\underbrace{\quad\quad}_{\text{نرخ انتقال}}}$$

اما در این مدت فقط $T_f \times B_f \times R$ بایت واقعاً منتقل می‌شود. پس:

فضای یک شیار (رکوردی) زمان خواندن یک شیار (رکورد و گپ)

$$\frac{\overbrace{T_f \times B_f \times (R + W_R)}^{\uparrow}}{t} \qquad \overbrace{T_f \times B_f \times R}^{\uparrow} \qquad \Rightarrow \quad \boxed{t' = t \times \frac{R}{R + W_R}}$$

$$1 \qquad\qquad t'$$

میزان بایتی که در 1 ثانیه انتقال داده می‌شود.

حداکثر نرخ انتقال

دیدیم که در حالت مختلف بافرینگ و با توجه به مقدار C_B و عوامل دیگر نرخ انتقال بدست آمده متفاوت بود.

نکته: زمان درنگ دورانی (r) تأثیری زیاد در نرخ انتقال دارد. همیشه بهینه آن است که سیستم بتواند در یک دور دیسک تمام بلاک‌های شیار را بخواند در این حالت حداکثر نرخ انتقال بدست می‌آید.

$$\max(t') = \frac{\text{Track size}}{2r} = \frac{\text{ظرفیت شیار}}{2r}$$

مثال: در یک دیسک track size = 1500 byte و متوسط زمان درنگ دورانی r = 2.5 ms مطلوبست حداکثر نرخ انتقال:

$$\max(t') = \frac{\text{track size}}{2r} = \frac{1500\ (\text{byte})}{2 \times 2.5\ (\text{ms})} = 300\ {}^{\text{byte}}\!/_{\text{ms}}$$

ارزیابی دقیق زمان درنگ دورانی

در صورتیکه r متوسط زمان درنگ دورانی باشد باتوجه به آن RPM واحد چرخش دیسک در دقیقه است همیشه:

RPM = دور در دقیقه

r = میلی ثانیه

$$r_{ms} = \frac{1}{2} \times \frac{60 \times 1000}{RPM}$$

حال اگر بخواهیم مقدار دقیق زمان درنگ دورانی را پیدا کنیم، می‌توانیم از رابطه زیر استفاده کنیم:

$$r' = r\left(2 - \frac{1}{T_f}\right)$$

r = متوسط زمان درنگ دورانی

T_f = تعداد بلاک‌ها در شیار (فاکتور تراکنش)

سیستم و ساختار فایل

مقدمه

اهداف اصلی هر سیستم فایل عبارتند از:

۱ـ سرعت عملیاتی (در بازیابی و ذخیره‌سازی)

۲ـ صرفه‌جویی در حافظه

برای رسیدن به این دو هدف اصلی، تلاش بر این است که در طراحی سیستم‌های ذخیره و بازیابی، ضابطه‌های اساسی زیر در نظر گرفته شوند:

۱ـ حداقل بودن میزان افزونگی (برای کاهش میزان حافظه مصرفی و کاهش هزینه بهنگام‌سازی)

۲ـ دستیابی سریع (برای داشتن سرعت مطلوب در بازیابی و ذخیره‌سازی)

۳ـ سهولت در عملیات بهنگام‌سازی (تا اطلاعات با کمترین هزینه بهنگام درآیند)

۴ـ سهولت نگاهداری سیستم

۵ ـ قابلیت اطمینان بالا

به علاوه جنبه‌هایی همچون حفاظت داده‌ها، ایمنی داده‌ها (Data Security) و اشتراکی شدن داده‌ها (Data Sharing)، انعطاف‌پذیری، جابجایی‌پذیری (Portability) و ... نیز در سیستم‌های جدید مورد نظرند.

ساختارهای مختلف، در دو عمل اساسی، یعنی بازیابی و ذخیره‌سازی، رفتار متفاوت دارند. در این فصل به مطالعه ساختارهای مبنایی زیر می‌پردازیم و ساختارهای دیگر را در فصل‌های بعد بررسی می‌کنیم.

۱) فایل با ساختار برهم Pile (heap)، (بی‌نظم)

۲) فایل با ساختار ترتیبی (Sequential)

در مطالعه این ساختارها، ابتدا به معرفی و بیان خصوصیات و شرح جنبه‌های تکنیکی هر یک می‌پردازیم و پس از بیان موارد کاربرد هر ساختار، کارایی آن را به طور تحلیلی مورد ارزیابی قرار می‌دهیم.

برای ارزیابی کارایی (Performance Evaluation (PE)) ضوابطی لازم است که در زیر بررسی می‌کنیم.

ضوابط ارزیابی کارآیی

این ضوابط عبارتند از:

۱ـ اندازه رکورد (متوسط حافظه لازم برای ذخیره‌سازی یک رکورد): R

۲ـ زمان لازم برای واکشی (Fetch) یک رکورد دلخواه از فایل: T_F

۳ـ زمان لازم برای بازیابی «رکورد بعدی(Get Next)»: T_N

۴ـ زمان لازم برای بهنگام‌سازی از طریق درج (Insert) یک رکورد: T_I

۵ـ زمان لازم برای بهنگام‌سازی (Update) از طریق ایجاد تغییر در یک رکورد: T_U

۶ـ زمان لازم برای خواندن تمام فایل ((Exhaustive read (READ ALL): T_X

۷ـ زمان لازم برای سازماندهی مجدد (Restructuring) فایل: T_Y

اساساً شش نوع عمل روی فایل‌ها، توسط سیستم فایل انجام می‌شود:

الف ـ واکشی رکورد

ب ـ بازیابی رکورد بعدی

ج ـ درج رکورد جدید (بهنگام‌سازی از طریق درج)

دـ تغییر در یک رکورد موجود (بهنگام‌سازی)

هـ ـ خواندن تمام فایل

وـ سازماندهی مجدد

عملیات درج، بهنگام‌سازی، حذف و سازماندهی مجدد را عملیات تغییر دهنده محیط فیزیکی ذخیره‌سازی می‌نامیم. عمل حذف (Delete) رکورد، به شرحی که خواهیم دید، حالت خاصی از عمل بهنگام‌سازی است.

نکته: انجام عملیات ششگانه، نهایتاً منجر به انجام سه عمل اساسی در محیط فیزیکی ذخیره‌سازی، می‌شود. این سه عمل عبارتند از:

الف ـ مکان‌یابی (پیگرد)

ب ـ خواندن فیزیکی

ج ـ نوشتن فیزیکی

متوسط اندازه رکورد

در این ارزیابی، تنها در نظر گرفتن بخش داده‌ای رکورد کافی نیست. زیرا به طوری که دیده شد، در پیاده‌سازی رکوردها در یک ساختار مشخص، فیلد یا فیلدهای دیگری نیز وجود دارند، (بخش غیرداده‌ای). پس عوامل دخیل در این ارزیابی عبارتند از: بخش داده‌ای رکورد، بخش غیرداده‌ای رکورد و در یک ارزیابی دقیق‌تر، W_R با در نظر داشتن همه «هرزها»، متراکم یا غیرمتراکم بودن فایل، پدیده افزونگی و استفاده یا عدم استفاده از تکنیک‌های فشرده‌سازی.

نکته ۱:

فایل متراکم به فایلی گفته می‌شود که تمام مقادیر همه صفات خاصه، در تمام رکوردهایش مشخص باشند.

فایل غیرمتراکم فایلی است که برخی از مقادیر بعضی از صفات خاصه، در برخی از رکوردها، موجود نباشند.

نکته ۲: (مهم)

اگر فایل را با رکوردهایی با قالب غیرثابت مکان طراحی کنیم، حالت غیرمتراکم پدید نمی‌آید. پس هنگامی فایل غیرمتراکم است که رکوردهای با طول ثابت و قالب ثابت مکان داشته باشیم و طبعاً حافظه هرز ایجاد می‌شود که باید در محاسبه متوسط اندازه رکورد، منظور گردد.

تعریف افزونگی

فایلی را دارای *افزونگی* گوییم که مقادیر بعضی از صفات خاصه‌اش بیش از یک‌بار در محیط فیزیکی، ذخیره‌سازی شده باشند این تکرار ذخیره‌سازی یا افزونگی دو حالت دارد:

۱ـ افزونگی طبیعی Natural Redundancy

۲ـ افزونگی تکنیکی Technical Redundancy

نکته: در افزونگی طبیعی، صفت خاصه چنان است که یک مقدار مشخص از آن در تعدادی از نمونه رکوردها وجود دارد. مثلاً در فایل ثبت‌نام دانشجویان، شماره یک درس مشخص، در رکورد تمام دانشجویانی که آن درس را اخذ کنند ذخیره می‌شود. زمانیکه صفت خاصه تکرارشونده چند مقداری باشد، افزونگی تشدید می‌شود.

نکته: برای کاهش مصرف حافظه در حالت وجود افزونگی طبیعی، در اساس دو تدبیر متصور است:

۱) طراحی فایل با ساختار کاراتر

۲) استفاده از یک تکنیک فشرده‌سازی

در مورد ساختار کاراتر، به عنوان مثال به ساختار چند حلقه‌ای می‌توان اشاره کرد.

الف: نمونه‌ای از یک فایل متراکم

	Student			Current			
	No.	Class	Credits	Incompletes	Work	Age	Grade
1	721	S	43	5	12	20	PF
2	843	S	51	0	15	21	Reg
3	1019	F	25	2	12	19	Reg
4	1021	F	26	0	12	19	Reg
5	1027	F	28	0	13	18	Reg
6	1028	F	24	3	12	19	PF
7	1029	F	25	0	15	19	Reg
8	1031	F	15	8	12	20	Aud
9	1033	E	23	0	14	19	PF
10	1034	F	20	3	10	19	Reg

ب: نمونه‌ای از یک فایل غیرمتراکم

		Coures taken					
	Student						Exp.
	No.	CS101	CS102	Bus3	EE5	IE103	Years
1	721	F72	F73		W73		
2	843	F72	W73				1
3	1019		S72	S73			
4	1021		S72			F73	
5	1027	F73		S73			
6	1028				W73		1
7	1029	F73	W73				
8	1031	F73					
9	1033						3
10	1034					F73	

ج: نمونه‌ای از یک فایل دارای افزونگی

	Prerequisite	Student No.	When Taken	Years Exp.	Acc. Credits	.....	Graoe
1	CS102	721	F73		43	.....	PF
2	CS102	843	W73		51	.....	Reg
3	CS102	1019	S72		25	.....	Reg
4	CS102	1021	S72		26	.....	Reg
5	CS102	1029	W73		25	.....	Reg
6	Bus3	1019	S73		25	.....	Reg
7	Bus3	1027	S73		28	.....	Reg
8	EE5	721	W73		43	.....	PF
9	EE5	1027	W73		28	.....	Reg
10	IE103	1021	F72		26	.....	Reg
11	IE103	1034	F73		20	.....	Reg
12	Exp.	1019		3	25	.....	Reg
13	Exp.	1028		1	24	.....	PF
14	Exp.	1030		1	23	.....	PF
15	None	1031			20	.....	Aud

فایل متراکم، فایل غیرمتراکم و فایل دارای افزونگی

تکنیک‌های فشرده‌سازی

در موارد زیر به‌کار بردن تکنیک فشرده‌سازی توصیه می‌شود:

۱) کمبود حافظه وجود داشته باشد.

۲) میزان افزونگی بالا باشد (طبیعی یا تکنیکی).

۳) حجم انتقال اطلاعات در محیط شبکه زیاد باشد.

نکته: فشرده‌سازی باعث بروز فزونکاری (Over head) در سیستم می‌شود زیرا به روال‌های فشرده‌ساز و نافشرده‌ساز نیاز است، چون کاربر پایانی در نهایت داده‌ها را به صورت نافشرده می‌خواهد.

تکنیک‌های فشرده‌سازی

۱) حذف بلانک‌های اضافی

۲) حذف صفرهای بی‌معنا در داده‌های عددی

۳) برش از آغاز (Front truncation)

در این تکنیک کاراکترهای مشابه در آغاز فقره داده‌های مرتب پی‌درپی، حذف می‌شوند و به جای آن‌ها، عددی نشان دهنده تعداد کاراکتر مشابه حذف شده، ذخیره می‌شود.

۴) برش از انتها (Rare truncation) که شبیه تکنیک قبلی است اما عمل برش از انتها صورت می‌گیرد.

۵) انتخاب کد کاراکتر کوتاهتر

۶) انتخاب کد نمایش بهتر برای داده‌های عددی

۷) عدم ذخیره‌سازی داده‌های عددی که به کمک یک فرمول مشخص قابل تولید هستند.

۸) نمایش مقادیر صفت خاصه به کمک کدگذاری مناسب به جای ذخیره‌سازی صریح آن مقادیر، البته وقتی که مقادیر از مجموعه‌ای محدود باشند. مثلاً به جای ذخیره‌سازی صریح اسم گیاهان که معمولاً طولانی است، کد متناظر با هر اسم ذخیره شود.

۹) انتخاب مقادیر عددی با اجزاء بی‌معنا (فاقد بار اطلاعاتی) به عنوان مقادیر کلید خارجی رکورد به جای مقادیر با اجزاء با معنا (کدگذاری با اجزاء بامعنا). کدگذاری با اجزاء بامعنا، به ویژه وقتی که تعداد رکوردها زیاد باشد.

۱۰) کوتاهتر کردن داده‌های عددی با نمایش ممیز شناور وقتی که دقت بالا مورد نظر باشد.

۱۱) حذف صفرهای بامعنا و ذخیره کردن عددی نشان دهنده تعداد آن‌ها.

۱۲) استفاده از کد هافمن (Huffman Code)

ایده اصلی در این نوع کدگذاری استفاده از کد کاراکتر با طول متغیر است (به جای طول ثابت). به این ترتیب که کوتاهترین کد به کاراکتری داده می‌شود که بسامد نمود آن در داده متنی، (Text data)، بالاتر است. این کدگذاری نیاز به منطق خاصی دارد که اساس آن همان درخت هافمن (Huffman Tree) است.

۱۳) استفاده از ماتریس بیتی (Bit matrix)

تکنیک ماتریس بیتی

این تکنیک هنگامی کاربرد دارد که اولاً فقره اطلاع تکرار شونده (صفت خاصه چند مقداری) داشته باشیم و ثانیاً مقادیر صفت خاصه، از مجموعه‌ای محدود برگرفته شده باشند. در این صورت هم طول رکوردها متغیر و هم افزونگی طبیعی تشدید می‌شود. تکنیک ماتریس بیتی یکی از روش‌های کاهش این افزونگی است.

مثال: فرض می‌کنیم فایلی حاوی اطلاعاتی در مورد دانشجو ـ درس را می‌خواهیم ذخیره کنیم.

روش اول: بدون استفاده از ماتریس بیتی: برای هر نمونه دانشجو، رکوردی با طول متغیر در نظر می‌گیریم مطابق شکل:

	Student number	Course number
R_1	54381	177, 179, 184, 185, 187
R_2	54407	177, 178, 181, 183, 187, 191
R_3	54408	176, 184, 189, 191
R_4	54503	181, 185, 188
R_5	54504	178, 183, 185, 188, 191
⋮		

رکوردهای فایل دانشجو ـ درس با فرمت خطی

در این روش، رکوردهای فایلی بدون به‌کارگیری تکنیک فشرده‌سازی ذخیره شده‌اند.

روش دوم: استفاده از ماتریس بیتی: این روش در شکل زیر نشان داده شده است:

Student number	Courses number															
	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
54381	0	1	0	1	0	0	0	0	1	1	0	1	0	0	0	0
54407	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
54408	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
54503	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0
54504	0	0	1	0	0	0	0	1	0	1	0	0	1	0	0	1

ذخیره‌سازی رکوردهای فایل دانشجو ـ درس به کمک ماتریس بیتی

اگر N تعداد عناصر مجموعه‌ای باشد که مقادیر صفت خاصه درس از آن برگرفته می‌شوند، در روش ماتریس بیتی برای ذخیره‌سازی تمام درس‌ها، به N بیت حافظه نیاز داریم. اگر هر دانشجو C درس داشته باشد و $C = P.N, P \leq 1$.

افزونگی تکنیکی

عبارتست از تکرار بعضی یا تمام مقادیر یک (یا چند) صفت خاصه در محیط فیزیکی ذخیره‌سازی (خود فایل داده‌ای یا فایل کمکی آن)، به خاطر ایجاد یک شیوه دستیابی کاراتر برای فایل.

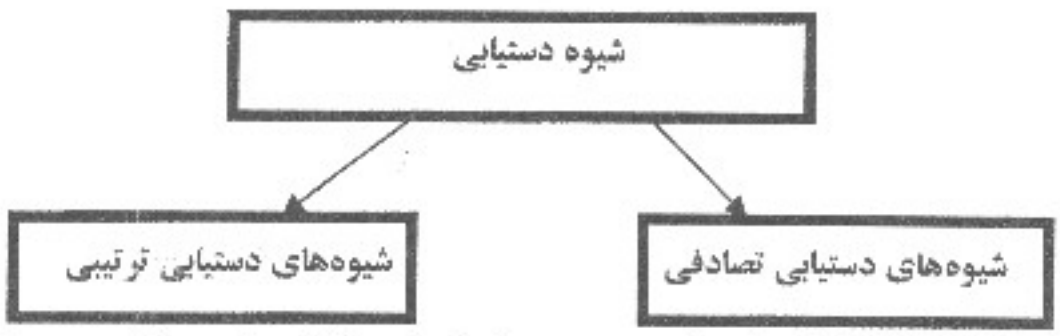
مثال: به عنوان مثالی از این نوع افزونگی، می‌توان از شاخص‌بندی نام برد. وقتی که برای یک فایل، روی یک صفت خاصه، شاخص ایجاد می‌کنیم، مقادیر آن صفت خاصه (بعضی یا تمام) در محیط ذخیره‌سازی تکرار می‌شوند.

شرح اصول عملیات ششگانه

واکشی رکورد دلخواه

این عمل اساساً عملی محتوایی است، یعنی رکوردی باید واکشی شود که مقدار یکی از صفات خاصه‌اش (گاه بیش از یک صفت خاصه) به عنوان نشانوند جستجو داده شده است (محتوای یک یا بیش از یک فیلد.) لازمه این عمل جستجو کردن در فایل، دستیابی به بلاک حاوی رکورد مورد نظر و خواندن آن است.

برای دستیابی به بلاک حاوی رکورد، در اساس دو دسته شیوه دستیابی وجود دارد:



در شیوه دستیابی ترتیبی، سیستم باید تعدادی بلاک را به طور پی‌درپی بخواند تا به بلاک حاوی رکورد مورد نظر برسد. مثلاً در یک فایل با ساختار نامنظم، سیستم باید از آغاز فایل به طور متوسط نصف بلاک‌های فایل را بخواند و بررسی کند. اصطلاحاً می‌گوییم باید جستجوی خطی (Linear Search) انجام دهد.

نکته مهم: تاکیداً یادآوری می‌شود که در ارزیابی زمان عملیات ششگانه روی فایل، هرگاه بلاکی با شیوه دستیابی ترتیبی خوانده شود. زمان خواندن $\frac{B}{t}$ است و چنانچه با شیوه دستیابی تصادفی (مستقیم) خوانده شود، زمان خواندن $s+r+b_{fr}$ است.

نکته مهم: روش‌های تنظیم درخواست واکشی

این روش‌ها بیشتر در محیط‌های DMS یا DBMS به کار می‌روند. در محیط FS (سیستم فایل) معمولاً روش اول، از روش‌های زیر، مورد استفاده قرار می‌گیرد:

۱) درخواست ساده (Single request)

۲) درخواست طیفی (Range request)

۳) درخواست محاسباتی (Functional request)

۴) درخواست بولی (Boolean request)

۵) درخواست مرکب (Composite request)

درخواست ساده

درخواستی که در آن یک نشانوند جستجو داده می‌شود و جواب آن در صورت وجود، یک رکورد است. مثلاً «مشخصات دانشجو به شماره X را بدهید.» اگر نشانوند جستجو، همان کلید اصلی فایل باشد می‌گوییم واکشی از طریق کلید اصلی است، یعنی شناسه واحد یک نوع موجودیت، و در فایل، شناسه واحد یک نمونه از یک نوع رکورد. ممکن است نشانوند جستجو، صفت خاصه‌ای غیر از کلید اصلی از جمله کلید ثانوی و یا هر صفت خاصه‌ای هم باشد.

درخواست طیفی

درخواستی که در آن طیفی از مقادیر کلید اصلی داده شود. مثلاً مشخصات دانشجویان از شماره X تا Y. در این‌جا نیز نشانوند جستجو، لزوماً همیشه کلید اصلی نیست. گاه ممکن است کلید ثانوی باشد و یا هر صفت خاصه دیگر، باید دید که ساختار، در پاسخ تقاضا، چگونه و با چه حدی از کارایی عمل می‌کند.

درخواست محاسباتی

درخواستی است که لازمهٔ پاسخ دادن به آن، انجام محاسبه توسط سیستم است، مانند درخواست بازیابی معدل. در حالی که خود «معدل» به عنوان یک فقره داده، لزوماً در فایل وجود ندارد.

درخواست بولی

درخواستی است که پاسخ به آن با انجام عملیات بولی و با استفاده از عملگرهای AND، OR و XOR به دست می‌آید.
مثلاً اسامی دانشجویانی که قدشان X و وزنشان Y باشد را بدهید.

درخواست مرکب

درخواستی است که در آن مقدار چند صفت خاصه داده شود و حالت خاصی از پرس و جوی بولی تلقی می‌گردد.

بازیابی رکورد بعدی

در بحث لوکالیتی دیدیم که رکورد بعدی منطقی، رکوردی است که براساس یک نظم خاص مورد نظر پردازشگر فایل، بعد از رکورد فعلی باید بازیابی شود.
ساختار فایل باید به گونه‌ای باشد که امکان بازیابی رکورد بعدی را بدهد، یعنی سیستم با بازیابی فعلی، بتواند بعدی را به دست آورد.
نکته: موقعیت رکورد بعدی نسبت به رکورد فعلی، به طور کلی به یکی از سه صورت زیر است:

۱ـ رکورد بعدی همجوار فیزیکی رکورد فعلی است.

۲ـ رکورد فعلی به رکورد بعدی نشانه‌رو دارد (مستقیم یا به طور غیرمستقیم که خواهیم دید).

۳ـ هیچ ارتباطی بین رکورد فعلی و بعدی وجود ندارد.

در تمام ساختارهایی که بررسی می‌کنیم منظور از رکورد بعدی، همان رکورد بعدی مقداری است. یعنی بعدی براساس نظم صعودی مقادیر نشانوند جستجو که در اکثر ساختارها همان کلید اصلی یا ثانوی است.
نکته دیگر این که در حالت سوم، یعنی وقتی که هیچ ارتباط ساختاری بین فعلی و بعدی وجود نداشته باشد، بازیابی رکورد بعدی اساساً ناممکن است.

بهنگام‌سازی از طریق درج

منظور از این عمل، درج یک رکورد جدید، بعد از لود اولیه فایل، در فایل است و ما از این پس فقط آن را عمل درج می‌نامیم. می‌دانیم که این عمل جزء عمل‌های تغییر دهنده محیط فیزیکی ذخیره‌سازی است.

نکته۱: نکته اساسی این است که رکورد باید در بلاکی وارد شود. این بلاک باید یافته و خوانده شود. اما این‌که این بلاک، کدام بلاک باشد، بستگی به ساختار فایل دارد. به طور کلی دو حالت وجود دارد:

۱ـ رکورد باید در یک بلاک خاصی درج شود. به این بلاک اصطلاحاً نقطه منطقی درج (Logical Point of Insert) می‌گوییم. این حالت عملیات خاصی لازم دارد که خواهیم دید.

۲ـ بلاک خاصی مطرح نیست، رکورد در هر جای فایل می‌تواند درج شود. در این حالت معمولاً رکورد در آخرین بلاک فایل (با فرض جادار بودن) وارد می‌شود و اصطلاحاً می‌گوییم رکورد را در انتهای فایل‌الحاق (Append) می‌کنیم و نشانگر پایان فایل تغییر می‌کند.

نکته۲: اصول عملیات در عمل درج چنین است:

۱ـ یافتن و خواندن بلاکی که رکورد باید در آن درج شود.

۲ـ جا دادن رکورد در بلاک که اینک در بافر است.

۳ـ بازنویسی بلاک (Rewrite)

۴ـ در بعضی از ساختارها، عملیات دیگری هم لازم است که به آن‌ها عملیات پس از درج می‌گوییم. این عملیات اساساً برای تنظیم ارتباط ساختاری رکورد درج شده با رکورد(های) دیگر انجام می‌شود و عمدتاً شامل ایجاد یا اصلاح نشانه‌رو(ها) است.

بهنگام‌سازی از طریق تغییر محتوای رکورد

این عمل یعنی تغییر مقدار یک یا بیش از یک صفت خاصه در یک رکورد مشخص.

نکته۱: رکورد بهنگام درآمده، ممکن است در مکان قبلی‌اش باز نوشته شود و یا اساساً در مکانی جدید. در حالت اول می‌گوییم بهنگام‌سازی، درجا (Inplace) صورت می‌گیرد (عمل جایگزینی (Replace)) و در حالت دوم، بهنگام‌سازی برون ازجا (Outplace). بهنگام‌سازی درجا همیشه امکان‌پذیر نیست. مثلاً اگر طول رکورد در اثر بهنگام‌سازی تغییر کند و دیگر امکان بازنویسی آن در جای قبلی وجود نداشته باشد.

نکته۲: اگر رکورد بهنگام درآمدنی دو پاره باشد، نحوه بهنگام‌سازی آن، با حالتی که یکپاره است، متفاوت خواهد بود.

نکته۳: اگر افزونگی در فایل موجود باشد، در تمام نقاطی که مقدار صفت (صفات) خاصه مشمول بهنگام‌سازی ذخیره شده است. باید عمل بهنگام‌سازی صورت پذیرد و اصطلاحاً می‌گوییم «بهنگام‌سازی منتشر شونده» (Propagating update) باید انجام شود و طبعاً زمان بهنگام‌سازی افزایش خواهد یافت. اگر بهنگام‌سازی منتشر شونده انجام نشود، فایل از نظر داده‌ای ناسازگار می‌شود (Inconsistent).

اصول عملیات بهنگام‌سازی یک رکورد:

۱ـ واکشی رکورد بهنگام درآمدنی

۲ـ کار در بافر (ایجاد نسخه جدید).

۳ـ بازنویسی نسخه جدید (در جای قبلی)، در بهنگام‌سازی درجا

۴ـ بازنویسی نسخه قدیم با نشانگر «حذف شده» در بهنگام‌سازی برون ازجا و درج نسخه جدید در جای دیگر.

۵ ـ در برخی از ساختارها انجام عملیات پس از بهنگام‌سازی به منظور تنظیم ارتباط ساختاری بین رکورد با رکورد(های) دیگر فایل. زیرا عمل بهنگام‌سازی نیز، مثل عمل درج، محیط فیزیکی ذخیره‌سازی را تغییر می‌دهد.

خواندن تمام فایل

در موارد زیر خواندن تمام فایل انجام می‌شود:

۱) پیرو درخواست کاربر، مثل لیست‌گیری‌ها، انجام یک پردازش خاص روی تمام رکوردها

۲) ایجاد نسخه‌ای (Copy File) دیگر از فایل (پیرو درخواست کاربر یا بنا بر یک نیاز سیستمی)

۳) در سازماندهی مجدد

۴) در ایجاد یک استراتژی دستیابی برای فایل.

نکته۱: در این عمل، تمام بلاک‌های فایل باید خوانده شوند. وجود حافظه‌های هرزی که به هر دلیل پدید می‌آیند، سبب افزایش زمان این عمل می‌شود.

نکته۲: خواندن تمام فایل به دو صورت ممکن است انجام شود:

به صورت پی‌درپی، یعنی بلاک به بلاک از آغاز فایل تا انتهای آن. (sequential)

به صورت سریال یعنی براساس نظم صعودی مقادیر یکی از صفات خاصه، معمولاً کلید. (Serial)

سازماندهی مجدد

هر فایلی، پس از لود اولیه، دوره حیاتی دارد که طی آن عملیاتی را اعم از بازیابی یا ذخیره‌سازی تحمل می‌کند. در اثر تغییراتی که در فایل ایجاد می‌شود، ممکن است به تدریج فایل کارایی اولیه‌اش را از دست بدهد و لذا باید آن را سازماندهی مجدد کرد.

نکته۱: دلایل کاهش تدریجی کارایی فایل به طور کلی عبارتند از:

۱ـ از بین رفتن نظم (یا وضع) ساختاری آغازین.

۲ـ بروز فضای هرز در فایل

۳ـ بروز وضعیت نامطلوب در استراتژی دستیابی (Access Strategy) (در فایل‌های شاخص‌دار خواهیم دید)

نکته۲: می‌توان دلایل سازماندهی مجدد را به صورت زیر بیان داشت:

۱ـ احیاء نظم یا وضع ساختاری آغازین

۲ـ بازپس‌گیری فضای هرز(Space desallocation)

۳ـ اصلاح استراتژی دستیابی

نکته۳: به طور کلی اصول عملیات چنین است:

۱ـ خواندن تمام فایل (پی‌درپی یا سریال)

۲ـ بلاک‌بندی مجدد (Reblocking) رکوردها ضمن خارج کردن رکوردهای حذف شدنی

۳ـ بازنویسی رکوردهای فعال (Active record) (براساس ساختار فایل)

۴ـ بازسازی ساختار مربوط به استراتژی دستیابی (در صورت وجود).

زمان بازنویسی بلاک

در عملیات درج، بهنگام‌سازی و سازماندهی مجدد، خواندن تعدادی بلاک، انجام عملیاتی در آن‌ها، و بالاخره بازنویسی آن‌ها لازم است. زمان عمل نوشتن بلاک را، که همان انتقال محتوای بافر به دیسک است. در یک ارزیابی کلی، می‌توان مساوی زمان عمل خواندن بلاک در نظر گرفت یعنی: $S+R+b_{tt}$ و یا در حالت نوشتن انبوه $\frac{B}{t'}$.

$$T_{RW} = \underbrace{2r - b_{tt}}_{\swarrow} + \underbrace{b_{tt}}_{\searrow} = 2r$$

زمان انتقال از بافر به رسانه (نوشتن فیزیکی) زمان انتظار برای رسیدن مجدد آغاز بلاک به زیر نوک R/W

اگر عملیات در بافر به موقع صورت نگیرد، سیستم یک دور دیگر از دست می‌دهد و:

$$\boxed{T_{Rw} = 4r}$$

البته به طور پیش‌فرض همواره $T_{RW} = 2r$ است.

شرح ساختارهای مبنایی

۱) ساختار پایل (برهم)

۲) ساختار ترتیبی (دنباله‌ای)

فایل با ساختار «پایل» (برهم)

معرفی ساختار

این فایل ساختاری دارد فاقد هرگونه نظم، یعنی رکوردها براساس مقادیر هیچ صفت خاصه‌ای مرتب نیستند.

نکته۱: در بهترین حالت، نظم بین رکوردها، نظمی است زمانی ((Entry Sequential: ES(Chronologic) انگار رکوردها بر یکدیگر «پشته» شده باشند، این ساختار فاقد هرگونه استراتژی دستیابی کمکی است و جستجو به صورت خطی از ابتدای فایل انجام می‌شود.

نکته۲:(مهم)

رکوردها قالب غیرثابت مکان و طول متغیر دارند. تعداد صفات خاصه و نیز مکان فیلدهای مربوط به صفات خاصه، در نمونه‌های مختلف رکوردها، متفاوت است. ساختار رکورد در این فایل به صورت زیر است:

$$\boxed{A_1 = V_1, A_2 = V_2, \ldots}$$

ساختار رکوردی در فایل پایل

A_i: اسم صفت خاصه V_i: مقدار صفت خاصه

مثال: به عنوان مثال در یک فایل حاوی اطلاعاتی در مورد دانشجویان، رکورد یک دانشجو می‌تواند به صورت زیر باشد:

.... = شماره، .... = وزن، .... = قد، .... = اسم

و رکورد دانشجوی دیگر به صورت زیر:

... = اسم، .... = قد، .... = شماره

دو نمونه از رکورد دانشجو

نکته۳: چون رکوردها قالب از پیش تعیین شده‌ای ندارند. به ناچار جفت اسم و مقدار صفت خاصه در تمام رکوردها برای تمام صفات خاصه، باید ذخیره شود که خود سبب بروز افزونگی در ذخیره‌سازی اسم صفات خاصه در تعدادی از رکوردها و مصرف حافظه، می‌شود. از سوی دیگر چون برای اطلاع نهست (فیلدهای مربوط به اطلاعات پیشوندی رکورد همانند: طول رکورد، فیلد حذف و ...) فیلد در نظر گرفته نمی‌شود، از این رو صرفه‌جویی در حافظه داریم.

☐ موارد استفاده

۱) در محیط‌هایی که در آن‌ها، داده‌ها نظم‌پذیر نباشند و پیش‌پردازشی (Preprocess) روی داده‌ها انجام نشده باشد و فایل اساسا برای بایگانی (Archive File) ایجاد شود (فعال نباشد).

۲) در محیط‌های با داده‌های استراتژیک وقتی که ایمنی داده‌ها مورد نظر باشد، بی‌نظمی می‌تواند ایمنی فایل را افزایش دهد. البته باز هم فایل باید برای بایگانی باشد.

۳) مبنایی است برای مطالعه و درک بهتر ساختارهای دیگر و نیز طراحی ساختارهای کاراتر.

■ ارزیابی کارایی

☑ متوسط اندازه رکورد

مفروضات:

۱) فایل در لود اولیه، n رکورد دارد.

۲) کل تعداد صفات خاصه در نظر گرفته شده در محیط عملیاتی را a می‌نامیم.

۳) متوسط تعداد صفات خاصه در یک رکورد را با a' نشان می‌دهیم. (متوسط تعداد فقره اطلاع‌ها در یک رکورد).

۴) متوسط حافظه لازم برای ذخیره‌سازی اسم صفت خاصه را، A بایت در نظر می‌گیریم.

۵) متوسط حافظه لازم برای ذخیره‌سازی مقدار صفت خاصه را V بایت فرض می‌کنیم.

با این مفروضات و باتوجه به قالب رکورد می‌توان نوشت:

$$R = a'(A + V + 2)$$

که در آن یک بایت برای علامت انتساب و یک بایت برای علامت جداساز منظور شده است.

مثال: در یک فایل پایل متوسط حافظه لازم برای ذخیره‌سازی اسم صفت خاصه 20 بایت و متوسط تعداد صفات خاصه برای رکورد 4 می‌باشد. در صورتیکه تعداد رکوردها 5 و فضای مقداری هر یک از رکوردها به شکل زیر باشد، طول متوسط رکورد کدام است؟

$$V(R_1) = 10\ ,\ V(R_2) = 5\ ,\ V(R_3) = 4\ ,\ V(R_4) = 4\ ,\ V(R_5) = 7$$

۱) 28 ۲) 112 ۳) 208 ۴) 352

گزینه ۲ صحیح می‌باشد.

$R = a'(A + V + 2)$ متوسط اندازه رکورد در فایل پایل

$$\left.\begin{array}{l} A = \text{متوسط اسم صفت خاصه} = 20\ \text{byte} \\ a' = \text{متوسط تعداد صفت خاصه} = 4 \\ V = \dfrac{V_1 + V_2 + V_3 + V_4 + V_5}{5} = \dfrac{10 + 5 + 4 + 4 + 7}{5} = 6 \end{array}\right\} \quad \begin{array}{l} R = 4(20 + 6 + 2) \\ = 112 \end{array}$$

☑ واکشی رکورد T_F

نشانوند جستجو در درخواست به صورت $A_i = V$ داده می‌شود (یعنی اسم یک صفت خاصه و مقدارش).

عملیات لازم:

خواندن بلاک حاوی رکورد مورد نظر با جستجوی خطی، اما رکورد مورد نظر ممکن است در اولین بلاک فایل باشد یا مثلاً در آخرین (و یا هر بلاک دیگر)، بنابراین به طور متوسط نصف بلاک‌های فایل باید خوانده و وارسی شود. اگر تعداد بلاک‌های فایل b و اندازه بلاک B بایت باشد، زمان واکشی از این رابطه به دست می‌آید:

$$T_F = \frac{1}{2} b \frac{B}{t'} \quad \text{یا} \quad T_F = \frac{1}{2} n \frac{R}{t'}$$

البته برای رفتن به BOF زمان $s + r$ نیز سپری می‌شود که به علت کوچک بودن در مقابل زمان بالا می‌توان از آن صرفنظر کرد.

☑ بازیابی رکورد بعدی (مهم) T_N

رکورد بعدی، در این ساختار مفهومی ندارد، زیرا هیچگونه ارتباط ساختاری بین رکورد فعلی و بعدی آن برقرار نیست. اگر کاربر خود نشانوند جستجوی رکورد بعدی را بدهد، این عمل تبدیل به عمل واکشی تک رکورد می‌شود و داریم:

$$T_N = T_F$$

که بسیار ناکاراست.

عمل درج T_I

عملیات لازم:

چون فایل فاقد هرگونه نظم است، لذا رکورد جدید به انتهای فایل‌الحاق می‌شود و برای این منظور:

۱ـ خواندن آخرین بلاک فایل که سیستم آدرس آن را دارد،

۲ـ **کار در بافر (که زمانش را در ارزیابی دخالت نمی‌دهیم)**: انتقال رکورد از ناحیه کاری برنامه کاربر به بلاک که در بافر است.

۳ـ بازنویسی بلاک

لازم است.

$$T_I = s + r + b_{fr} + T_{RW}$$

که در آن، T_{RW} زمان بازنویسی بلاک است و می‌توان آن را با $2r$ برابر گرفت.

$$T_I = s + 3r + b_{fr}$$

سیستم در این عمل کارا است.

☑ **عمل بهنگام‌سازی T_U**

در این ساختار عمل بهنگام‌سازی در حالت کلی، به صورت برون ازجا انجام می‌شود.

عملیات لازم:

۱ـ واکشی رکورد بهنگام درآمدنی

۲ـ ضبط نشانگر «حذف شده» در بخش پیشوندی نسخه قدیم } کار در بافر

۳ـ ایجاد نسخه جدید

۴ـ بازنویسی نسخه قدیم (که اینک نشانگر "حذف شده" دارد)

۵ـ درج نسخه جدید در انتهای فایل

پس داریم:

$$T_U = T_F + T_{RW} + T_I$$

که در آن:

T_F: واکشی رکورد بهنگام درآمدنی

T_{RW}: بازنویسی همین رکورد با نشانگر «حذف شده»

T_I: درج نسخهٔ جدید

☑ **عمل حذف: (مهم) T_{Delete}**

حالت خاصی است از بهنگام‌سازی که در آن درج نسخه جدید انجام نمی‌شود، و زمان آن برابر است با:

$$T_{U_{delete}} = T_F + T_{RW}$$

مثال: در یک فایل پایل (pile) اگر $T_{Next} = 2_{ms}$ و تعداد دور دیسک در دقیقه 3000 باشد، زمان حذف (T_{Delete}) چند ms است؟

۱) 10 ۲) 12

۳) 20 ۴) 22

گزینه ۴ صحیح می‌باشد.

می‌دانیم در فایل پایل $T_N = T_F$

$$\begin{cases} T_N = 2\,ms \\ RPM = 3000 \text{ دور در دقیقه} \Rightarrow r = \frac{1}{2} \times \frac{60 \times 1000}{RPM} = 10\ ms \Rightarrow 2r = 20\ ms \end{cases}$$

$$T_{delete} = T_F + T_{Rw} = T_N + 2r = 2_{ms} + 20_{ms} = 22_{ms}$$

☑ **خواندن تمام فایل** T_X

به سادگی می‌توان دریافت که:

Seq: پی در پی $\boxed{T_{x_{seq}} = 2T_r}$

زیرا سیستم باید تمام بلاک‌ها را بخواند.

خواندن سریال این فایل ناممکن است زیرا بازیابی رکورد بعدی ناممکن است. می‌توان فایل را روی صفت خاصه مورد نظر کاربر مرتب کرد. طبعاً فایل حاصله دیگر پایل نیست و البته به تمامی ترتیبی هم نیست زیرا ممکن است صفت خاصه مورد نظر در تمام رکوردها موجود نباشد و در نتیجه تنها یک بخش از فایل مرتب خواهد بود.

در این صورت داریم:

$$\boxed{T_{x_{ser}} = T_{sort}(n) + T_{x_{seq}}}$$

زمان مرتب کردن فایل بستگی به الگوریتم مرتب سازی خارجی دارد. در عین حال اگر بخواهیم بدون عمل sort زمان خواندن سریال را بدست آوریم از رابطه زیر استفاده می‌کنیم:

$$\boxed{T_{X_{ser}} = n\,T_F}$$

در این حالت برای خواندن سریال زمان خواندن هر رکورد به صورت منظم با زمان T_F بدست می‌آید و برای خواندن سریال n رکورد زمان $n \times T_F$ نیاز است.

☑ **سازماندهی مجدد** T_Y

اگر فرض کنیم تعداد رکوردهای درج شده در یک دوره از حیات فایل (یعنی از لود اولیه تا درست قبل از سازماندهی مجدد) o باشد و d رکورد حذف شده باشند (نشانگر «حذف شده» خورده باشند) زمان سازماندهی مجدد چنین خواهد بود:

$$\boxed{T_Y = (n+o)\frac{R}{t'} + (n+o-d)\frac{R}{t'}}$$

که در آن:

$(n+o)\frac{R}{t'}$: زمان خواندن کل فایل

$(n+o-d)\frac{R}{t'}$: زمان بازنویسی کل فایل

توجه داریم که در ارزیابی زمان بازنویسی کل فایل، زمان لازم برای نوشتن یک رکورد را $\frac{R}{t'}$ در نظر گرفتیم، یعنی همان زمان انتقال یک رکورد در خواندن انبوه فایل.

فایل با ساختار ترتیبی

معرفی ساختار

فایل ترتیبی بر دو نوع است:

۱) فایل ترتیبی کلیدی

۲) فایل ترتیبی زمانی

در فایل ترتیبی زمانی، رکوردها به ترتیب وارد سیستم ذخیره می‌شوند و نوع خاصی از فایل پایل است که در آن رکوردها معمولاً قالب ثابت مکان دارند.

فایل ترتیبی کلیدی

این فایل نسبت به فایل پایل دو بهبود ساختاری دارد:

۱) در لود اولیه، تمام نمونه رکوردها براساس مقادیر یکی از صفات خاصه منظم هستند و این نظم با همجواری فیزیکی رکوردها پیاده‌سازی می‌شود.

صفت خاصه نظم معمولاً همان کلید اصلی است که طبعاً می‌تواند مرکب هم باشد. وجود کلید با خاصیت یکتایی، تضمین است زیرا در بدترین حالت با ترکیب تمام صفات خاصهٔ یک رکورد، می‌توان به مقدار مرکب یکتا رسید.

۲) تمام نمونه رکوردها قالب از پیش طراحی شده‌ای دارند، به عبارت دیگر، فیلدهای قالب رکورد هر یک مربوط به صفت خاصه مشخصی هستند (رکورد دارای قالب ثابت مکان است). لذا در ذخیره‌سازی فایل نیازی نیست که اسم صفت خاصه هر بار ذخیره شود و همان مقدار کافی است.

ساختمان رکورد چنین است:

V_1	V_2	.....	V_n

ساختمان رکورد در فایل ترتیبی

این دو بهبود ساختاری مزایا و معایبی (نسبت به فایل پایل) در بردارد.

مزایا:

۱ـ صرفه‌جویی در مصرف حافظه به خاطر عدم ذخیره‌سازی اسم صفت خاصه در نمونه رکوردها.

۲ـ ساده‌تر بودن قالب رکورد، به نحوی که رکورد ذخیره شده عملاً نگاشتی است از آن‌چه که در برنامه پردازشگر اعلان می‌شود.

۳ـ نرم‌افزار ساده‌تر برای ایجاد، مدیریت و پردازش فایل.

۴ـ وجود یک استراتژی دستیابی: نظم خود نوعی استراتژی دستیابی است که در آن شیوه دستیابی ترتیبی است بنابراین می‌توان الگوریتم جستجوی بهتری نسبت به جستجوی خطی، در واکشی رکورد اعمال کرد (مثلاً الگوریتم جستجوی دودویی).

۵ـ پردازش سریال رکوردها تسریع و تسهیل می‌شود به ویژه اگر فایل کاملاً ترتیبی باشد.

معایب:

۱) مصرف حافظه بیشتر به خاطر در نظر گرفتن فیلد برای اطلاع نهست (قالب ثابت مکان)

۲) وجود پدیده عدم تقارن(Asymmetry): زیرا فقط یک صفت خاصه (صفت خاصه نظم) در عملیات روی فایل نقش دارند و سایر صفات خاصه نقشی ندارند. در واقع استراتژی دستیابی، متکی به همان صفت خاصه نظم است.

۳) کاهش انعطاف‌پذیری ساختار.

۱ـ ثابت بودن طول رکوردها (عدم امکان تغییر طول رکورد)

۲ـ در عملیات تغییر دهنده (درج، حذف و بهنگام‌سازی) فاقد انعطاف‌پذیری است.

در صورتیکه بخواهیم طول رکورد را تغییر دهیم باید فایل را دوباره طراحی و ایجاد کنیم.

در عملیات تغییر دهنده مانند درج، رکورد درج شدنی باید در نقطه منطقی خودش درج شود، تا ساختار ترتیبی بماند و این عمل نیاز به شیفت دارد که در فایل‌های بزرگ هزینه بالائی دارد. برای رفع مشکلات فوق فایل اصلی را ترتیبی و به صورت خواندنی در نظر گرفته و تمام تغییرات را در فایل دیگری به نام فایل ثبت تراکنش (T . L . F) انجام می‌دهیم این فایل از نظر ساختاری تمام یک فایل پایل است.

راه حل مشکل نوشتن یا تغییرات در فایل ترتیبی

۱ـ فایل ترتیبی را خواندنی ایجاد می‌کنیم.

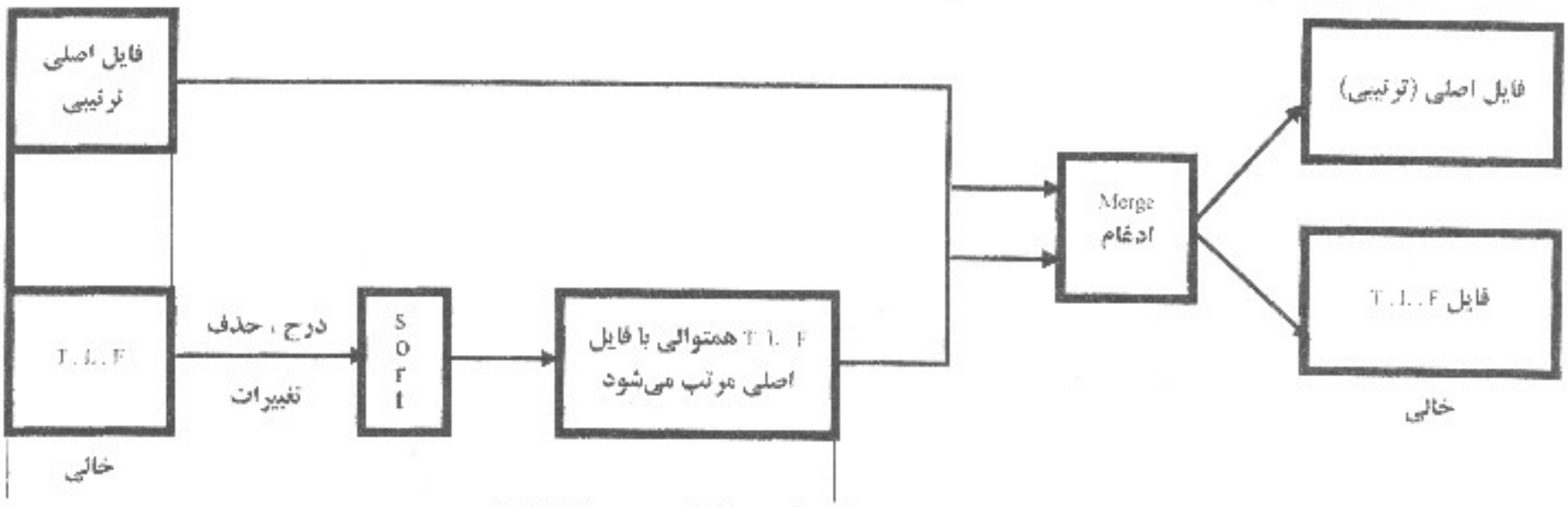
۲ـ تمام عملیات تغییر دهنده (درج ـ حذف ـ تغییرات) در فایل دیگری به نام فایل *ثبت تراکنش* (T.L.F) انجام می‌شود.

در نتیجه همیشه دو فایل داریم:

الف) فایل اصلی (master) که همیشه ترتیبی بوده و فقط خواندنی است.

ب) فایل T.L.F (Transactionlog File) که یک فایل پایل بوده و ترتیبی زمانی است.

نکته: در زمان سازماندهی مجدد، فایل تراکنش (T.L.F) مرتب شده و سپس با فایل اصلی ادغام می‌شود. و یک فایل ترتیبی جدید با رکوردهای اضافه شده از فایل تراکنش (T.L.F) ایجاد شده که تمام تغییرات از فایل T.L.F نیز روی آن فایل اصلی اعمال می‌شود.



روند کلی عملیات ادغام فایل ثبت تراکنش T . L . F با فایل اصلی

نکته: قبل از ادغام دو فایل اصلی (ترتیبی) و فایل ثبت تراکنش (T.L.F) فایل ثبت تراکنش باید حتماً همتوالی با فایل اصلی مرتب شود (منظور از همتوالی این است که فایل تراکنش براساس همان صفتی مرتب شود که فایل اصلی نیز براساس آن مرتب شده است)

کاربرد فایل ترتیبی

۱ـ در کاربردهای تجاری وقتی که رکوردها را با سیستم دسته‌ای (batch) به صورت یکجا پردازش می‌کنیم.

مانند: گزارش لیست حقوقی کارمندان ـ گزارش لیست حقوقی دانشجویان

۲ـ واکشی سریع تک *رکوردها مطرح نباشد*

۳ـ تغییر طول رکوردها *در ساختار مطرح نباشد.*

۴ـ پردازش سریال فایل *مطرح باشد.* مخصوصاً در فایل پایل ابتدا فایل را مرتب کرده خواندن سریال فایل به راحتی انجام می‌شود.

ارزیابی کارایی

متوسط اندازه رکورد

با توجه به ساختار رکورد می‌توان نوشت:

$$R = a.V$$

و داریم:

$$S_{file} = n.a.V$$

اما دیدیم که عملیات در این فایل پس از لود اولیه، در عمل در فایل ثبت تراکنش‌ها انجام می‌شود. لذا برای ارزیابی دقیق‌تر در لود اولیه باید ظرفیت آن فایل را نیز دخالت داد:

$$s_{file} = (n+o).a.V$$

فرض کرده‌ایم که فایل تراکنش‌ها ظرفیت o رکورد را دارد.

☑ **واکشی رکورد T_F**

دو حالت در نظر می گیریم:

الف ـ نشانوند جستجو غیر از صفت خاصه نظم باشد.

ب ـ نشانوند جستجو همان صفت خاصه نظم (کلید) باشد.

حالت الف: در این حالت، فایل عملاً تبدیل می‌شود به حالت خاصی از پایل و با جستجوی خطی، خواهیم داشت:

$$T_F = \frac{1}{2}.n.\frac{R}{t'}$$

و اگر فرض کنیم که ظرفیت T.L.F به تعداد o رکورد باشد و در لحظه واکشی، o' رکورد در آن وجود داشته باشد ($o' \le o$ و o' به تدریج به سمت o میل می کند) ارزیابی دقیق‌تر چنین خواهد بود:

$$T_F = \frac{1}{2}(n+o')\frac{R}{t'}$$

می‌بینیم که در این حالت، زمان بالا است و فایل کارآیی لازم را ندارد.

مثال: در یک فایل ترتیبی اگر تعداد صفات خاصه 4، متوسط حافظه لازم برای ذخیره‌سازی مقدار صفت خاصه 10 بایت و تعداد کل رکوردها (n=20) و مقدار 2 K byte /sec انتقال وجود داشته باشد، مطلوبست T_F در صورتی که نشانگر جستجو غیر از صفت خاصه منظم باشد.

۱) 1.95 ثانیه ۲) 195 ثانیه ۳) 195 میلی ثانیه ۴) 19.5 میلی ثانیه

گزینه ۳ صحیح می‌باشد.

چون جستجو براساس صفت خاصه غیرمنتظم است فایل را به صورت پایل در نظر گرفته و در نتیجه $T_F = \frac{n}{2} \times \frac{R}{t'} = \frac{b}{2} \times \frac{B}{t'}$

$$\left.\begin{array}{l} a = 4 \\ V = 10 \end{array}\right\} \Rightarrow R = aV = 40 \text{ byte}$$

$$n = 20$$

$$t' = 2^{\,k\,byte/sec} = 2 \times 1024^{\,byte/sec} = 2^{11\,byte/sec}$$

$$\left.\begin{array}{l} T_F = \frac{n}{2} \times \frac{R}{t'} \\ \quad = \frac{20}{2} \times \frac{40}{2^{11}} \\ \quad = 0.195 \text{ sec} \\ \quad = 195 \text{ ms} \end{array}\right.$$

حالت ب: در این حالت با توجه به صفت خاصه‌ای که فایل براساس آن منظم شده روش‌های زیر وجود دارد:

۱ـ جستجوی دودوئی
۲ـ جستجوی باپرش بلاکی
۳ـ جستجو با تخمین و کاوش

۱) جستجوی دودویی

جستجوی دودویی، در یک محیط منظم خارجی باید در دو سطح انجام گیرد. در سطح اول، جستجویی در فایل داریم تا بلاک مورد نظر پیدا شود (واحد جستجو (Search unit) در این سطح، بلاک است). برای این کار طبعاً باید بلاک‌ها خوانده شوند. در سطح دوم برای هر بلاک که به بافر آورده می‌شود، یک جستجوی دودویی درون بلاکی داریم. این هر دو جستجو در ارزیابی زمان دخالت دارند.

$$T_{F_{\text{every walk}}} = \text{Log}_2\left(n\frac{R}{B}\right)(s+r+b_{rl}+c_B)+T_{F_o}$$

$$T_{F_o} = \frac{1}{2}o'\frac{R}{t'}$$

یادآوری می‌شود که T.L.F دارای نظم زمانی است (فایل پایل). ممکن است رکورد در فایل اصلی نباشد، یعنی رکورد، درجی باشد. (و در فایل T.L.F ظاهر شده باشد) لذا دخالت دادن T_{F_o} لازم است.

نکته: چون واحد جستجو در سطح خارجی بلاک است، $\text{Log}_2\left(n\frac{R}{B}\right)$، دفعات مراجعه به فایل خواهد بود و در هر بار مراجعه، یک بلاک (بلاک میانی) به طور مستقیم خوانده می‌شود. در بررسی محتوای یک بلاک، نیز با روش جستجوی دودویی عمل می‌شود تا مشخص شود که رکورد مورد نظر در بلاک هست یا نه. و اگر در بلاک وجود نداشت، بلاک میانی دیگر باید خوانده شود. زمان C_B در رابطه T_F، صرف همین بررسی محتوای بلاک می‌شود. و اما الگوریتم جستجوی دودویی، تنها روش نیست. روش‌های دیگر هم برای جستجو در یک محیط منظم خارجی وجود دارند. از جمله روش جستجو با پرش بلاکی و جستجو با روش تخمین کاوش (Probing) که ذیلاً به آن می‌پردازیم.

تعداد دفعات مراجعه به فایل $\log_2\frac{nR}{B}$ است که از رابطه زیر بدست می‌آید:

$$\log_2\frac{nR}{B} = \log_2\frac{n}{B} = \log_2\frac{n}{B_f} = \log_2^b$$

مثال: در یک فایل ترتیبی (Sequential File) در صورتیکه ضریب بلاک بندی 2 و تعداد رکوردها 16 باشد، دفعات مراجعه به فایل در عمل جستجو عبارتست از؟ (سراسری ۸۳)

۱) 3 ۲) 5 ۳) 16 ۴) 64

$B_f = 2$
$n = 16$
تعداد مراجعات دودوئی $= \log_2 b = \log_2\frac{n}{B_f} = \log_2\frac{16}{2} = 3$

۲) جستجو با پرش بلاکی (Skipped block search)

این روش اساساً همان روش جستجوی خطی است با این تفاوت که در جستجوی درون بلاک، کلید رکورد مورد نظر با کلید آخرین رکورد هر بلاک مقایسه می‌شود، چنانچه این کلید از کلید رکورد مورد نظر بزرگتر یا کوچکتر باشد. محتوای بلاک خوانده شده، (به صورت خطی). در غیر این صورت سیستم با پرش از بلاک، بلاک بعدی را می‌خواند و به طور متوسط، نصف بلاک‌ها و در هر بلاک به طور متوسط نصف رکوردها باید خوانده شوند.

$$\text{متوسط تعداد مقایسات} = \frac{b}{2}+\frac{B_f}{2}$$

$\frac{b}{2}$: نصف بلاک‌ها

$\frac{B_f}{2}$: نصف رکوردهای هر بلاک

نکته مهم: در این روش مناسب‌ترین اندازه برای بلاک (مقدار بهینه B_f) برابر است با $\sqrt{n}$ (n تعداد رکوردهای فایل است). به عبارت دیگر:

$$B_f = \sqrt{n}$$

مثال: اندازه بهینه فاکتور بلاک‌بندی در ساختار ترتیبی با واکشی پرش بلاکی در فایلی با تعداد رکورد 10^4 و طول رکورد 200 کدام است؟

۱) 10 ۲) 100 ۳) 200 ۴) 300

گزینه ۲ صحیح می‌باشد.

$$\begin{cases} B_f = \sqrt{n} = \sqrt{10^4} = 100 \\ n = \text{تعداد رکوردها} = 10^4 \end{cases}$$

نکته: به عبارت دیگر اگر $B_f = \sqrt{n}$ در نظر گرفته شود، تعداد رکوردهایی که باید بررسی شوند تا رکورد مورد نظر به دست آید، به حداقل می‌رسد، این حداقل مقایسه $(\sqrt{n})$ است.

حداقل تعداد مقایسه $= \sqrt{n}$

۳ـ جستجو با تخمین و کاوش

در این روش ابتدا آدرس تقریبی بلاک حاوی رکورد مورد نظر را تخمین زده، به ابتدای آن بلاک رفته و از آن‌جا جستجوی خطی را تا انتهای فایل انجام می دهیم تا این‌که رکورد مورد نظر پیدا شود. محاسبه زمان واکشی در این وضعیت به طور دقیق امکان‌پذیر نیست:

$$T_F = s + r + b_{tt} - k\left(\frac{B}{t'}\right) \qquad k \geq 1$$

$s + r + b_{tt}$: زمان لازم برای رسیدن به ابتدای بلاک که آدرس آن را تخمین زده‌ایم.

$k\left(\frac{B}{t'}\right)$: زمان خواندن k بلاک بعد از آدرس تخمین‌زده شده به صورت خطی.

اگر رکورد در مرحله اول کاوش خطی بدست نیاید دوباره آن را تکرار می کنیم.

نکته: کارائی این روش بستگی به چگونگی توزیع رکوردها دارد.

☑ **بازیابی رکورد بعدی T_N**

در ساختار ترتیبی با خواندن هر بلاک B_f رکورد بدست می‌آید که در داخل بلاک که رکورد بعدی برای هر رکورد بعد از آن است. بنابراین برای محاسبه زمان T_N، زمان خواندن بلاک برای رکورد جاری را بین تمام رکوردهای بلاک B_f تقسیم می کنیم.

$$T_N = \frac{\frac{B}{t'}}{B_f} = \frac{\frac{B}{t'}}{\frac{B}{R}} = \frac{R}{t'}$$

نکته: به احتمال $\frac{1}{B_f}$ ممکن است رکورد جاری، رکورد آخر بلاک باشد. که در این صورت رکورد بعدی در بلاک بعدی بوده و باید آن بلاک را بخوانیم همچنین ممکن است رکورد بعدی در فایل T.L.F باشد که در این صورت باید به صورت خطی فایل تراکنش (T.L.F) جستجو شود.

☑ **عمل درج T_I**

رکورد درج شدنی را نمی‌توان به انتهای فایل اصلی الحاق کرد. زیرا باید در نقطه خاصی درج شود و این کار در فایل‌های بزرگ زمان‌گیر است. در عمل رکورد در T.L.F درج می‌شود تا در سازماندهی مجدد، فایل براساس نظم مورد نظر، بازآرایی (Reordering) شود. برای ارزیابی زمان درج دو حالت را در نظر می‌گیریم:

۱) حالت اول: درج در فایل کوچک (مهم)

می‌توان در فایل کوچک، نقطهٔ درج را یافت و رکورد را در آن نقطه درج کرد. در این صورت بقیه رکوردها باید به سمت پایان فایل (EOF) شیفت داده شوند. چون معلوم نیست که نقطهٔ درج کجا است، دقیقاً مشخص نیست چه تعداد بلاک باید شیفت داده شود. به طور متوسط ، نصف بلاک‌های فایل شیفت داده می‌شوند، پس عملیات لازم در این حالت عبارتست از:

۱ـ یافتن نقطه منطقی درج

۲ـ درج رکورد در بلاک مورد نظر (کار در بافر)

۳ـ شیفت دادن بلاک‌ها از نقطه منطقی درج به سمت EOF. (به طور متوسط نصف بلاک‌ها شیفت داده می‌شوند.)

و خواهیم داشت:

$$T_I = T_F + \frac{1}{2} b \left(\frac{B}{t'} + T_{RW} \right)$$

T_F: یافتن نقطه منطقی درج $\qquad \frac{B}{t'} + T_{RW}$: زمان شیفت یک بلاک

مثال: در یک فایل ترتیبی در عمل درج در فایل‌های از نوع کوچک باید نقطهٔ منطقی درج پیدا شده و رکورد در آن درج شود، در صورتیکه که نشانگر جستجو غیر از صفت خاصه نظم باشد، با توجه به اطلاعات زیر زمان درج کدام است؟

4 = تعداد صفات خاصه

16 = فضای متوسط داده‌ای صفات خاصه

$240 \frac{bit}{sec}$ = نرخ انتقال

30 = تعداد رکوردها

3 = تعداد بلاک‌ها

2 sec = زمان شیفت بلاکی

۱) 10 $\qquad$ ۲) 32 $\qquad$ ۳) 35 $\qquad$ ۴) 38

گزینه ۳ صحیح می‌باشد.

$$\begin{matrix} a = 4 \\ V = 16 \end{matrix} \Rightarrow R = aV = \boxed{64 \text{ byte}}$$

$$t' = 240 \frac{bit}{sec} = \frac{240}{8} \text{ byte}/\text{sec} = \boxed{30 \text{ byte}/\text{sec}}$$

$$n = 30 \quad b = 3 \quad \text{زمان شیفت بلاکی} = \frac{B}{t'} + T_{RW} = 2 \text{ sec}$$

زمان درج در فایل کوچک ترتیبی $T_I = T_F + \frac{b}{2}\left(\frac{B}{t'} + T_{RW}\right)$ که چون صفت خاصه جستجو غیرمنظم است فایل ترتیبی به صورت پایل جستجو می‌شود، از طرفی در فایل پایل $T_F = \frac{n}{2} \times \frac{R}{t'}$ پس:

$$T_I = \frac{n}{2} \times \frac{R}{t'} + \frac{b}{2}\left(\frac{B}{t'} + T_{RW}\right)$$
$$= \frac{30}{2} \times \frac{64}{30} + \frac{3}{2}\left(2 \text{ sec}\right) = 35 \text{ sec}$$

مثال: در یک فایل ترتیبی اگر B=2100 byte طول بلاک و نرخ انتقال 240 bit/sec و rpm=2000 باشد، زمان شیفت بلاک کدام است؟

۱) 70.03 ۲) 85.03 ۳) 95.03 ۴) 100.03

گزینه ۱ صحیح می‌باشد.

$$\text{زمان شیفت} = \frac{B}{t'} + T_{RW} = \frac{B}{t'} + 2r$$

$$\left.\begin{matrix} B = 2100 \text{ byte} \\ t' = 240 \text{ bit}/\text{sec} = \frac{240}{8} = 30 \text{ byte}/\text{sec} \\ RPM = 2000 \Rightarrow r = \frac{1}{2} \times \frac{60 \times 1000}{RPM} = 15 \text{ ms} = 0.015 \text{ sec} \end{matrix}\right\} \text{زمان شیفت} = \frac{2100}{30} + 0.03 = 70.03 \text{ sec}$$

دور در دقیقه

$2r = 0.03$

۲) حالت دوم: درج در حالت کلی (فایل بزرگ)

رکورد درج شدنی در آخرین بلاک فایل ثبت تراکنش‌ها درج می‌شود. و زمان این عمل همان زمان درج در پایل است و به علاوه زمانی دیگر نیز باید دخالت داده شود که $\frac{T_Y}{o}$ است.

$$\boxed{T_I = s + 3r + b_n + \frac{T_Y}{o}}$$

نکته مهم: توضیح در مورد $\frac{T_Y}{o}$: هرچند عمل سازماندهی مجدد، همزمان با عمل درج صورت نمی‌گیرد، ولی به هر حال این کار در فایل ترتیبی به خاطر احیاء نظم آغازین انجام می‌شود، با این هدف که o رکورد درج شده در فایل تراکنش‌ها، در محل منطقی خود جای داده شوند. سازماندهی مجدد o رکورد، زمانی را مصرف می‌کند که این زمان باید بین o رکورد درج شده در فایل تراکنش‌ها سرشکن شود، زیرا فقط پس از سازماندهی مجدد است که هر رکورد، در نقطه منطقی‌اش، درج می‌شود.

☑ **عمل بهنگام‌سازی** T_U

نکته ۱: اگر مقدار کلید عوض نشود، می‌توان رکورد را به صورت درجا (Inplace) بهنگام آورد.

ولی برای بهنگام سازی معمولاً به صورت برون ازجا (Outplace) عمل کرده و از فایل تراکنش استفاده می‌شود. به این ترتیب که رکورد بهنگام درآمدنی واکشی شده و در بافر بهنگام می‌شود و همراه با یک رکورد کوچک دیگر، همزمان در فایل تراکنش‌ها درج می‌شود.

نکته ۲:

عملیات لازم:

۱) واکشی رکورد بهنگام درآمدنی (T_F)

۲) عمل بهنگام سازی در بافر (ایجاد نسخه جدید) (0)

۳) درج رکورد بهنگام درآمده، همراه با یک رکورد کوچک منظم به آن در فایل تراکنش (T_I).

$$T_U \approx T_F + T_I$$

☑ **خواندن تمام فایل** T_X

خواندن پی‌درپی به روش معمول انجام می‌شود و زمان خواندن به طور پی‌درپی برابر است با:

$$T_{X_{seq}} = (n+o')\frac{R}{t'}$$

برای خواندن سریال، فایل تراکنش‌ها باید مرتب شود. زمان چنین برآورد می‌شود:

$$T_{X_{sor}} = T_{sort}(o') + (n+o')\frac{R}{t'}$$

البته گاه در عمل، به ویژه اگر پریود خواندن سریال طولانی باشد فایل اصلی را با T.L.F ادغام می‌کنند، یعنی عملاً سازماندهی مجدد انجام می‌شود و سپس فایل اصلی جدید که کاملاً ترتیبی است. خوانده می‌شود.

☑ **سازماندهی مجدد** T_Y

برای سازماندهی مجدد این فایل، باید عملیات زیر انجام شود:

۱ـ مرتب کردن فایل تراکنش (تا با فایل اصلی همتوالی شود).

۲ـ خواندن فایل اصلی

۳ـ خواندن فایل تراکنش

۴ـ بلاک‌بندی مجدد رکوردها (ادغام (Merge) آن‌ها طبق نظم) ضمن خارج کردن رکوردهای «حذف شدنی».

۵ ـ بازنویسی کل فایل.

زمان برابر است با:

$$T_Y = T_{sort}(o) + n.\frac{R}{t'} + o.\frac{R}{t'} + (n+o-d)\frac{R}{t'}$$

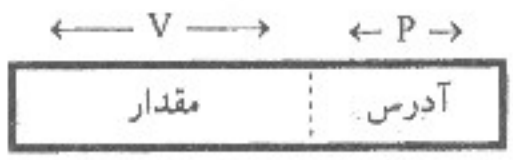
ساختار شاخص

شاخص‌بندی

جزء شیوه‌های دستیابی تصادفی است و دستیابی تصادفی از طریق آدرس رکورد امکان‌پذیر می‌شود. این آدرس طبعاً بستگی به مکان رکورد در فایل داده‌ای دارد و در رکورد شاخص یا مدخل شاخص (Index entry) (درایهٔ نمایه) نگهداری می‌شود.

شاخص

شاخص یا فایل شاخص مجموعه‌ای است از تعدادی مدخل، هر مدخل شاخص به شکل زیر است:



در فیلد مقدار، مقدار هر صفت خاصه‌ای (ساده یا مرکب (Single index , Composite index)) می‌تواند قرار گیرد و در حالت خاص مقدار کلید اصلی در این فیلد گذاشته می‌شود.

فیلد آدرس، حاوی یک نشانه‌رو به یک رکورد (شاخص متراکم) یا گروهی از رکوردها (شاخص غیرمتراکم) در فایل داده‌ای است.

نکته ۱: هر مدخل شاخص را می‌توان یک رکورد کوچک دانست با طول ثابت P + V بایت.

نکته ۲: در ساختار شاخص‌دار، دو مجموعه رکورد داریم: مجموعه رکوردهای داده‌ای و مجموعه رکوردهای شاخص.

به مجموعه اول فایل شاخص‌بندی شده (Indexed File) و به مجموعه دوم فایل شاخص (Index File) می‌گوییم. گاه به فایل شاخص‌بندی شده، مجموعه داده‌ها (Data set) و به فایل شاخص، مجموعه شاخص (Index set) نیز گفته می‌شود. مجموعه شاخص ناظر است به مجموعه داده‌ها.

نمایش ساده شده شاخص‌بندی

نکته ۳: هرچند ساختار شاخص‌بندی شده در سطح منطقی از دو مجموعه تشکیل شده است، اما می‌تواند در سطح فیزیکی، به صورت یک یا دو فایل پیاده‌سازی شود.

نکته ۴: فایل داده‌ای می‌تواند بیش از یک مجموعه شاخص باشد. (چند شاخصی)، اما در ساده‌ترین گونه این ساختار، یک مجموعه شاخص روی صفت خاصه کلید اصلی وجود دارد (ساختار تک شاخصی (Mono Indexed)).

شاخص اصلی (Primary Index) و شاخص (های) ثانوی (Secondary Index)

روشن است که در انتخاب صفت خاصه شاخص، اولویت با کلید اصلی است و در این صورت می‌گوییم که شاخص اصلی داریم. اما هر صفت خاصه دیگری نیز می‌تواند به عنوان صفت خاصه شاخص انتخاب شود و در این حالت می‌گوییم شاخص (های) ثانوی داریم. اگر صفت خاصه دیگری هم خاصیت کلید بودن را داشته باشد (کلید ثانوی) و روی آن شاخص ایجاد کنیم، شاخص روی کلید ثانوی داریم.

لنگرگاه (نقطهٔ اتکاء (Anchor.point))

نقطه‌ای از فایل داده‌ای که مدخل شاخص به آن نشانه می‌رود، لنگرگاه یا نقطه اتکاء شاخص نامیده می‌شود.

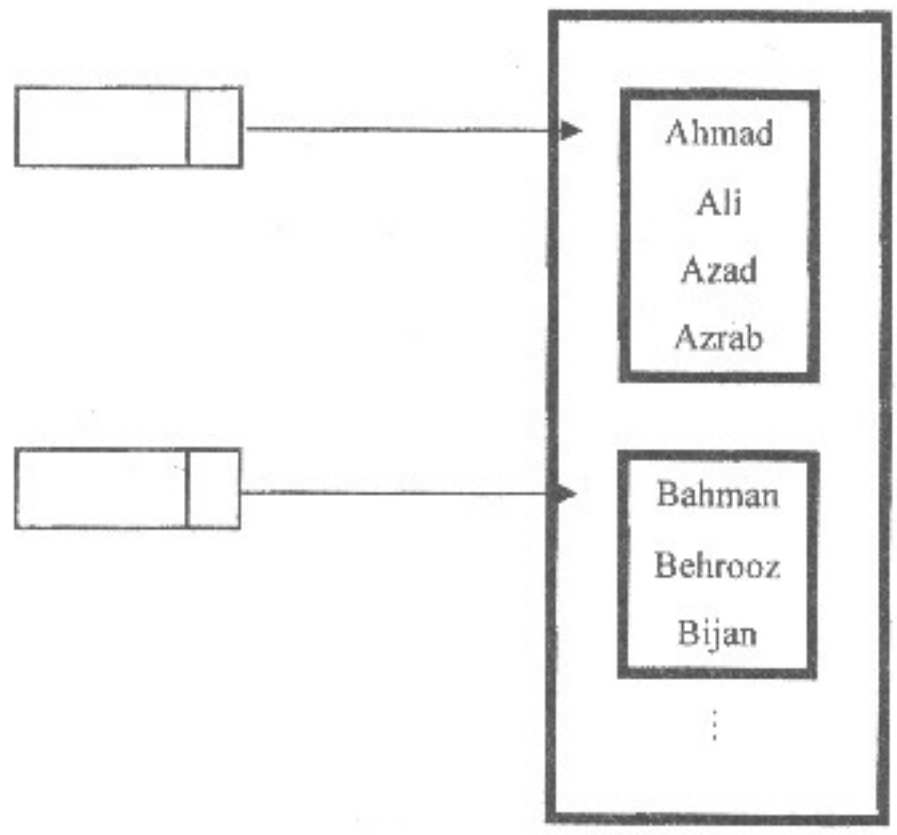
شاخص متراکم: اگر لنگرگاه، رکورد باشد، شاخص را متراکم (Dense Index) گویند.

شاخص غیرمتراکم: اگر لنگرگاه گروهی از رکوردها باشد شاخص را غیر متراکم (Non Dense Index) گویند.

در واقع در شاخص متراکم، واحد شاخص‌پذیر رکورد (Indexable unit) و در شاخص غیرمتراکم، گروهی از رکوردها است.

نکته مهم: در شاخص غیرمتراکم، فایل داده‌ای باید روی مقادیر صفت خاصه شاخص مرتب باشد، تا بتوان رکوردها را گروه‌بندی کرد ولی در شاخص متراکم لزومی ندارد که فایل داده‌ای مرتب باشد و روی فایل نامرتب هم ایجاد می‌شود.

مثال:



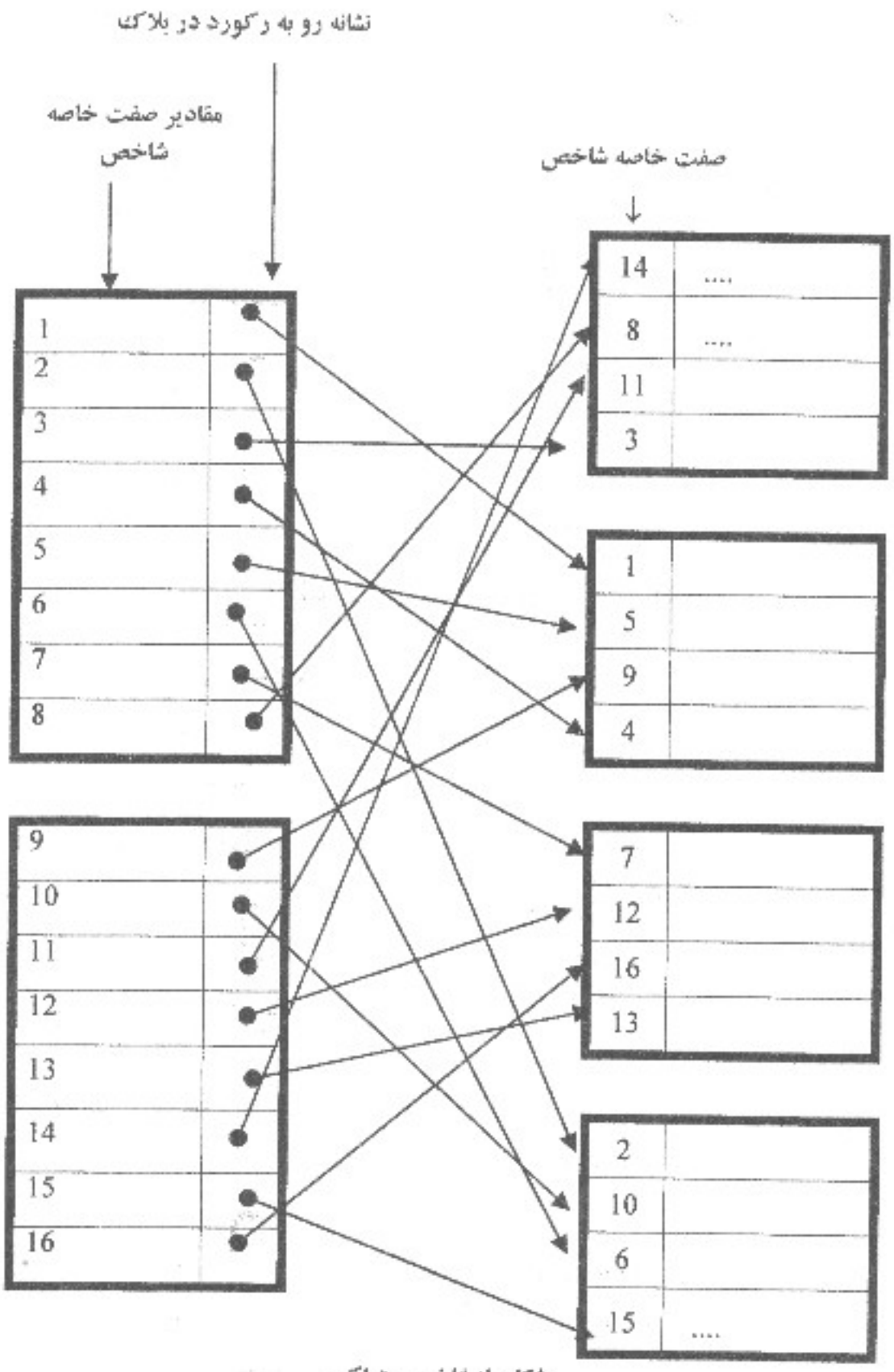
فایل داده‌ای مرتب

شاخص غیرمتراکم هر مدخل به یک بلاک گروهی از رکوردها نشانه می‌رود.

نکته: در هر دو حالت شاخص متراکم و شاخص غیرمتراکم، فایل داده‌ای می‌تواند بلاک‌بندی شده باشد و نیز مدخل‌های شاخص نیز در بلاک‌ها جای داده می‌شوند. در یک پیاده‌سازی مشخص، اندازه بلاک شاخص و بلاک داده‌ای، یکسان است. ولی در این‌جا مثالی از شاخص متراکم ارائه می‌کنیم که در آن فایل داده‌ای، نامرتب است.

در حالت شاخص غیرمتراکم، مقدار در فیلد مقدار مدخل شاخص می‌تواند کوچکترین مقدار صفت خاصه شاخص در رکوردهای گروه و یا بزرگترین مقدار آن باشد.

مثالی از شاخص متراکم (روی فایل نامرتب)



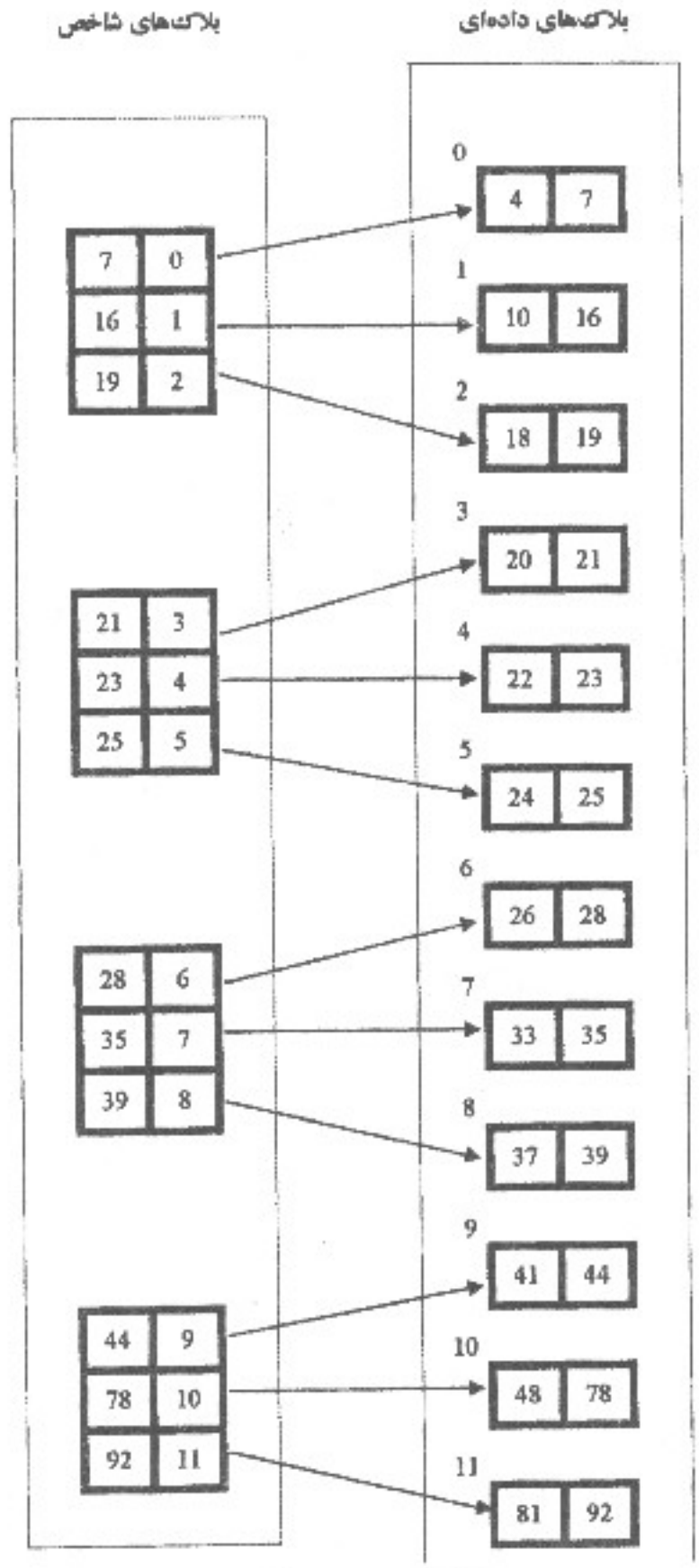
شکل ۱: شاخص متراکم

گروه در شاخص غیرمتراکم

بلاک مثلاً باکت یا با هر نام

دیگر. در این حالت می‌گوییم شاخص نرم‌افزاری(Software Index) داریم. گروه معادل است با یک یا بیش از یک [illegible] یعنی شیار، استوانه و گاه حتی خود دیسک (در حالتی که فایل روی چند دیسک ذخیره شده باشد). مثال بیان شده در شاخص متراکم (روی فایل نامرتب) نوعی شاخص نرم‌افزاری است. در شکل ۲ مثالی از شاخص نرم‌افزاری دیده می‌شود که در آن مقدار در فیلد مقدار مدخل شاخص بزرگترین مقدار صفت خاصه در رکوردهای گروه است، و گروه، بلاک با دو رکورد است. در قسمت نمایش منطقی شاخص سخت‌افزاری مثالی از شاخص سخت‌افزاری Hardware Index خواهیم دید.

مثالی از شاخص نرم‌افزاری (نامتراکم روی فایل مرتب)

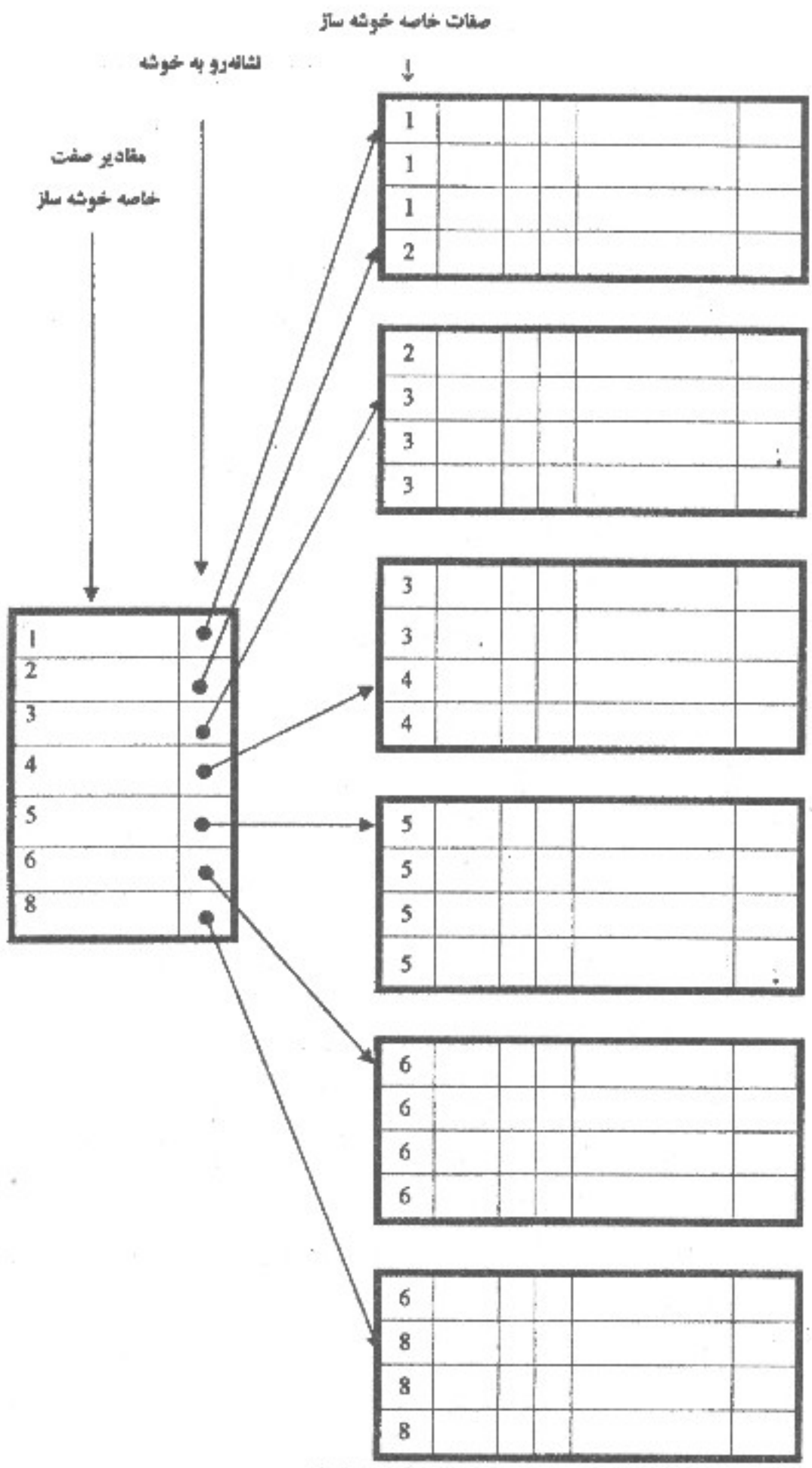


شکل ۲: شاخص نرم‌افزاری

شاخص خوشه‌ساز (Clustering Index)

دیدیم که در شاخص غیرمتراکم، فایل داده‌ای باید مرتب باشد، تا بتوان رکوردها را در گروه‌هایی جای داد. اما گاه ممکن است مقادیر صفت خاصه تکراری باشند (یعنی شاخص روی صفت غیرکلید ایجاد شود. البته در ساختار ترتیبی شاخص‌دار، شاخص روی کلید اصلی که همان صفت نظم است، ایجاد می‌شود). صفت خاصه‌ای که مقادیرش در فایل تکراری است، امکان می‌دهد تا رکوردها در خوشه‌هایی جای گیرند. چنین صفت خاصه‌ای، به صفت خاصه خوشه‌ساز (Clustering attribute) موسوم است. شاخص ایجاد شده روی چنین صفت خاصه‌ای، شاخص خوشه‌ساز نام دارد.

مثال شاخص خوشه ساز (غیرمتراکم)

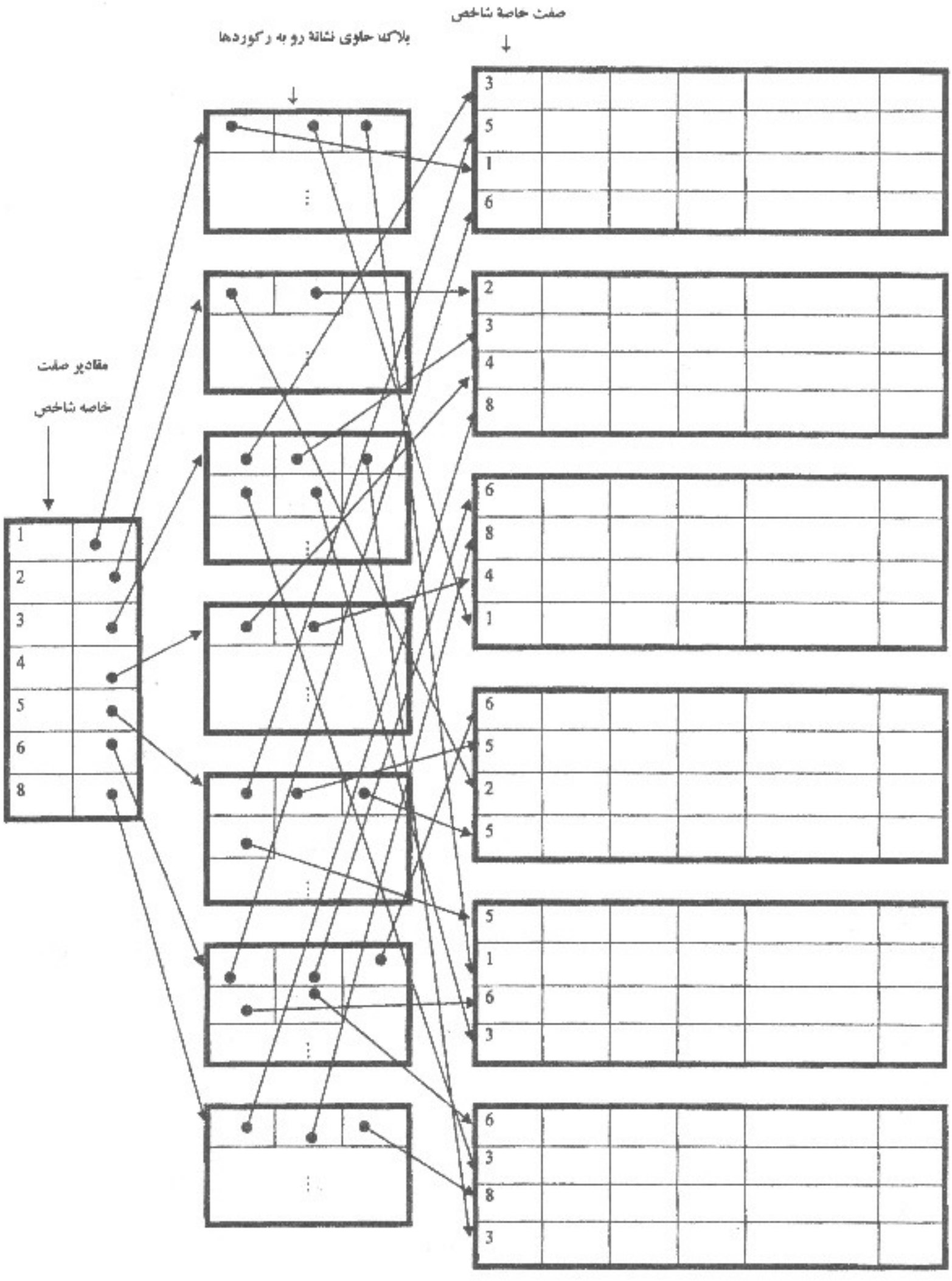


شکل ۳: شاخص خوشه‌ساز

شکل ۳: شاخص خوشه‌ساز

نکته: توجه داریم که صرف تکراری بودن مقادیر صفت خاصه ایجاب نمی‌کند که شاخص حتماً غیرمتراکم و خوشه‌ساز باشد. می‌توان شاخص متراکم هم ایجاد کرد و برای اجتناب از تکرار مقدار شاخص در مدخل‌های شاخص، گاه از یک سطح آدرس‌دهی غیرمستقیم هم استفاده می‌شود.

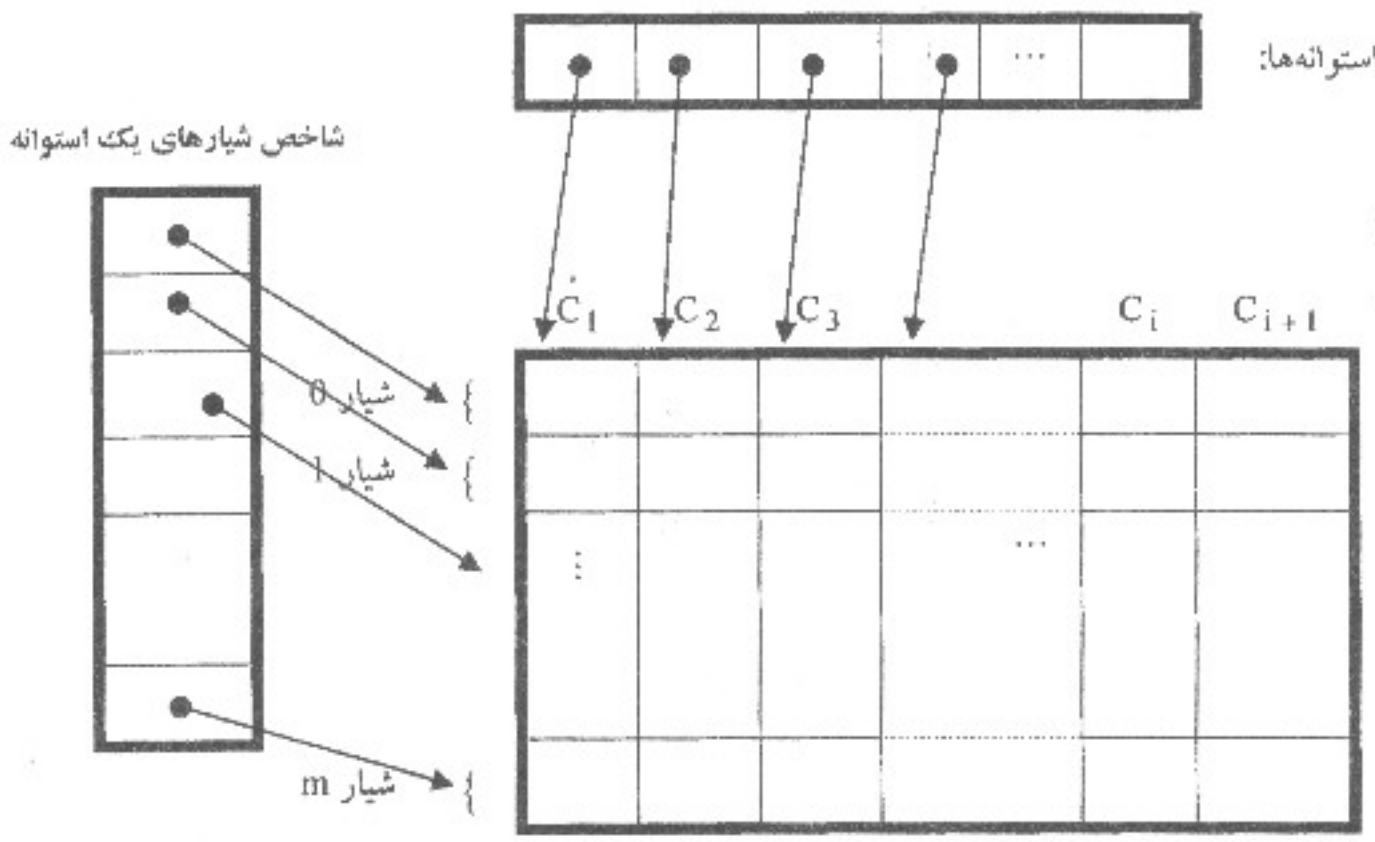
مثال شاخص غیرخوشه‌ساز (متراکم)



شکل ۴: شاخص متراکم و غیر خوشه‌ساز

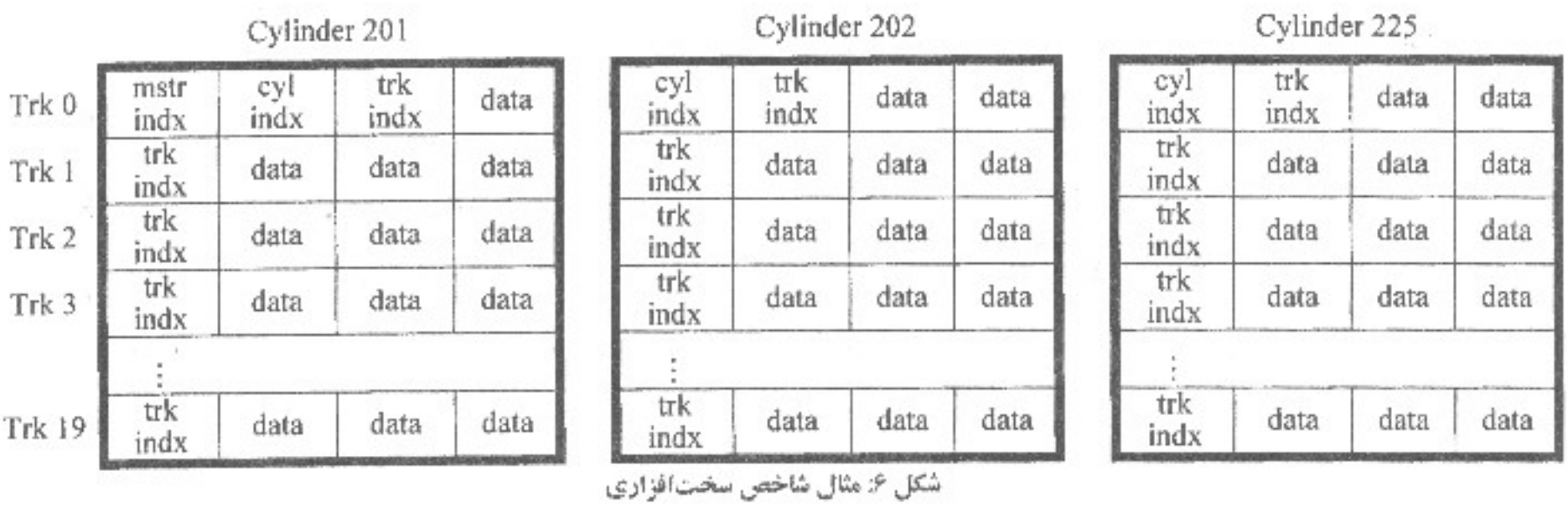
نمایش منطقی شاخص سخت‌افزاری

در شکل ۵ نمایش منطقی شاخص سخت‌افزاری دیده می‌شود . در این‌جا دو سطح شاخص داریم: شاخص استوانه (Cylinder Index) و شاخص شیار (Track Index).در هر مدخل از شاخص استوانه، کوچکترین مقدار کلید رکوردهای آن استوانه، شماره استوانه و آدرس شاخص شیارهای همان استوانه قرار دارد. در هر مدخل از شاخص شیارهای هر استوانه، کوچکترین مقدار کلید رکوردهای همان شیار و شماره شیار قرار داده می‌شود.



شکل ۵: نمایش منطقی شاخص سخت‌افزاری

مثالی از شاخص‌بندی وابسته به سخت‌افزار

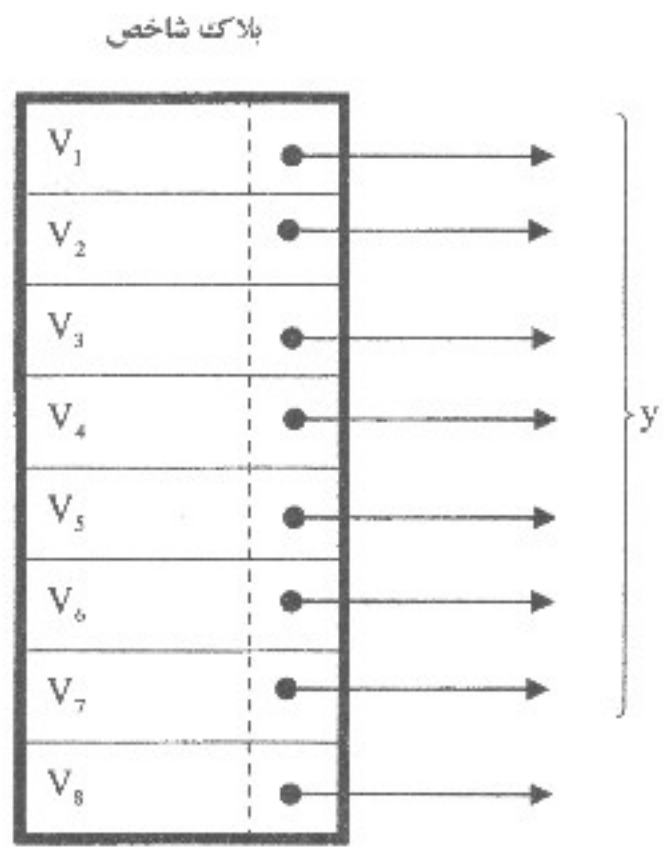


شکل ۶: مثال شاخص سخت‌افزاری

در این مثال، فایل از استوانه شماره 201 شروع و تا استوانه شماره 225 ادامه دارد. در هر شیار چهار بلاک وجود دارد. در اولین بلاک از شیار شماره صفر از استوانه 201، سرشاخص ذخیره شده است. بعد از آن، شاخص همین استوانه وجود دارد و بعد از آن، شاخص هر شیار و بلاک‌های داده‌ای ذخیره شده‌اند. غیر از شیار شماره صفر از استوانه شماره 201 که در آن یک بلاک حاوی شاخص شیار و یک بلاک داده‌ای داریم، در بقیه شیارها، یک بلاک شاخص شیار و سه بلاک داده‌ای ذخیره شده اند. در این مثال، هم مجموعه داده‌ای و هم مجموعه شاخص، در یک فایل پیاده‌سازی شده‌اند. *مثالی از این نوع شاخص‌بندی، سیستم فایل ISAM از IBM است.*

ظرفیت نشانه‌روی بلاک شاخص (Fan out)

گفتیم که مدخل‌های شاخص در بلاک‌هایی جای داده می‌شوند. تعداد مدخل‌های یک بلاک شاخص را ظرفیت نشانه‌روی آن بلاک می‌نامیم و آن را با y نمایش می‌دهیم.



شکل ۷: ظرفیت نشانه‌روی شاخص

باتوجه به طول مدخل شاخص، V + P و اندازهٔ بلاک، B، داریم:

$$y = \left\lfloor \frac{B}{V + P} \right\rfloor$$

از y در ایجاد سطوح مختلف استفاده می‌شود.

شاخص چند سطحی (Multi level index) (مهم)

وقتی که تعداد مدخل‌های شاخص زیاد باشد، جستجو در شاخص برای یافتن مدخل شاخص مورد نظر، زمان‌گیر می‌شود. برای تسریع عمل جستجو در شاخص، آن را در چند سطح ایجاد می‌کنند. *تعداد سطوح شاخص را عمق شاخص می‌نامند و با x نمایش می‌دهند.*

نکته ۱:

در حالت x = 1، شاخص را خطی می‌گویند.

نکته ۲:

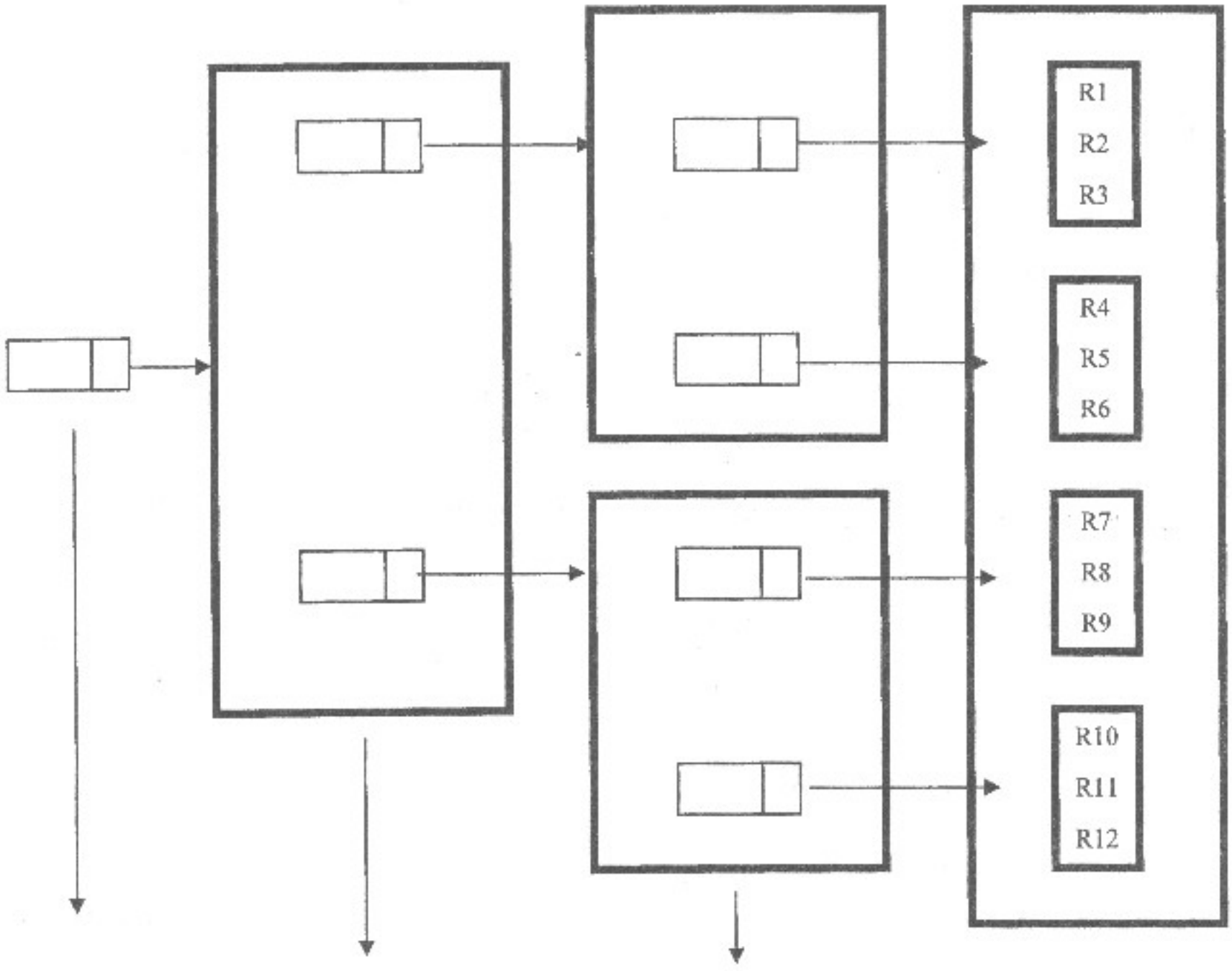
در شاخص چند سطحی، شاخص‌های سطح دوم به بعد، غیرمتراکم هستند.

چون از سطح دوم به بعد مدخل‌های شاخص بلاک بندی شده، بنابراین مدخل‌های هر سطح به بلاک‌های حاوی مدخل‌های شاخص در سطح بعدی اشاره می‌کنند و در نتیجه شاخص غیرمتراکم خواهد شد.

مثال:

$e_1 = 4$ تعداد مدخل‌های سطح اول

$e_2 = 2$ تعداد مدخل‌های سطح سوم



در این مثال شاخص‌ها در سطح اول شاخص غیرمتراکم و در سطوح دوم و سوم نیز غیرمتراکم است.

e_1 = تعداد بلاک‌های فایل داده‌ای = تعداد مدخل‌های سطح اول = 4

e_2 = تعداد بلاک‌های سطح اول = تعداد مدخل‌های سطح دوم = 2

e_3 = تعداد بلاک‌های سطح دوم = تعداد مدخل‌های سطح سوم = 1

نکته ۳: بالاترین سطح شاخص (آخرین سطح) را سرشاخص (Master index یا Top index) می‌نامند و اندازه آن معمولاً یک بلاک است و گاه آن را در حافظه اصلی نگهداری می‌کنند. بلاک‌های سایر سطوح شاخص روی دیسک، در شیارهای استوانه‌های آغازین و یا در شیارهای آغازین هر استوانه از فضای فایل، ذخیره می‌شود.

نکته ۴: سطح اول در شاخص چند سطحی می‌تواند متراکم یا غیرمتراکم باشد. در صورتیکه متراکم باشد تعداد مدخل‌های سطح اول $e_1 = n$ به تعداد رکوردهای فایل و در صورتیکه غیر متراکم باشد $e_1 = b$ به تعداد بلاک‌های فایل است.

محاسبهٔ ژرفای شاخص (Index deep) (عمق شاخص = تعداد سطوح شاخص)

اگر تعداد مدخل‌های سطح اول شاخص را e_1 بنامیم، با توجه به مفهوم y، می‌توان نوشت:

$$y^{x-1} \le e_1 \le y^x$$

و ژرفای شاخص برابر است با:

$$\boxed{x = \left\lceil \log_y^{e_1} \right\rceil}$$

دقت کنید تعداد مدخل‌های سطح اول (e_1) در صورتیکه شاخص غیرمتراکم باشد به تعداد بلاک‌های فایل است یعنی:

$$e_1 = b = \left\lceil \frac{n}{B_f} \right\rceil = \left\lceil \frac{n}{\left\lfloor \frac{B}{R} \right\rfloor} \right\rceil$$

و در صورتیکه شاخص متراکم باشد $e_1 = n$ به تعداد رکوردهای فایل خواهد بود.

چون فایل داده‌ای مرتب است معمولاً شاخص را غیرمتراکم فرض کرده و $e_1 = b$ در نظر گرفته می‌شود.

مثال: تعداد سطوح فایل شاخص غیرمتراکم برای یک فایل ترتیبی با 10^5 رکورد با طول رکورد 100 byte و طول بلاک 2000 byte در صورتیکه طول هر مدخل فایل شاخص 20 بایت باشد کدام است؟

$$\text{عمق شاخص یا تعداد سطوح} = X = \log_y e_1$$

$$\left.\begin{array}{r} R = 100 \\ B = 2000 \\ \text{طول مدخل} \quad = V + P = 20 \end{array}\right\} \Rightarrow \begin{array}{l} B_f = \left\lfloor \frac{B}{R} \right\rfloor = 20 \\ y = \left\lfloor \frac{B}{V+P} \right\rfloor = 100 \end{array}$$

از طرفی چون فایل شاخص غیرمتراکم است بنابراین تعداد مدخل‌های سطح اول (e_1) برابر تعداد بلاک‌های فایل داده‌ای است.

$$e_1 = b = \left\lceil \frac{n}{B_f} \right\rceil = \frac{10^5}{20} = 50000$$

$$\Rightarrow X = \left\lceil \log_{100} 50000 \right\rceil = 2$$

شاخص چند سطحی، ساختاری درختی دارد و معمولاً براساس $B-Tree$ و یا B^+-Tree ساخته می‌شود تا حالت پویا داشته باشد، یعنی همروند با عملیات ذخیره‌سازی در فایل، قابل تنظیم باشد. این نوع شاخص را در بحث ساختار چند شاخصی توضیح خواهیم داد.

پیاده‌سازی ساختار یک فایل ترتیبی شاخص‌دار در صورتیکه

$$\left\{\begin{array}{r} n = \text{تعداد رکوردهای فایل داده‌ای} \\ B = \text{طول بلاک} \\ R = \text{طول رکورد} \\ V = \text{فیلد مقدار شاخص} \\ P = \text{فیلد اشاره‌گر شاخص} \\ V + P = \text{طول مدخل شاخص} \end{array}\right.$$

باشد، با فرض آن‌که شاخص در سطح اول غیرمتراکم باشد. می‌خواهیم

e_i = تعداد مدخل‌های سطح iام

b_i = تعداد بلاک‌های شاخص سطح iام

s_i = حافظه مصرفی سطح iام

را محاسبه کنیم. در عین حال این عمل را می‌توانیم با در نظر گرفتن تعداد سطوح شاخص (x) انجام دهیم.

$$\text{در سطح اول شاخص} \begin{cases} e_1 = b = \left\lceil \frac{n}{B_f} \right\rceil \\ b_1 = \left\lceil \frac{e_1}{y} \right\rceil \\ s_1 = B \times b_1 \end{cases}$$

$$\text{از سطح دوم به بعد} \begin{cases} e_i = b_{i-1} \\ b_i = \left\lceil \frac{e_i}{y} \right\rceil \\ s_i = B \times b_i \end{cases}$$

البته در سطح آخر اگر نیازی به بلاک‌بندی نباشد برای محاسبه s_i می‌توانیم از $s_i = e_i \times (V + P)$ استفاده کنیم.

مثال: فایلی با مشخصات زیر را در نظر می‌گیریم، مطلوبست طراحی شاخص:

✓ $n = 10^6$ تعداد رکوردها، $R = 200$ byte، $B = 2000$ byte، $V = 14$ byte، $P = 6$ byte

$$B_f = \frac{2000}{200} = 10 \qquad V + P = 14 + 6 = 20 \qquad y = \left\lfloor \frac{2000}{20} \right\rfloor = 100$$

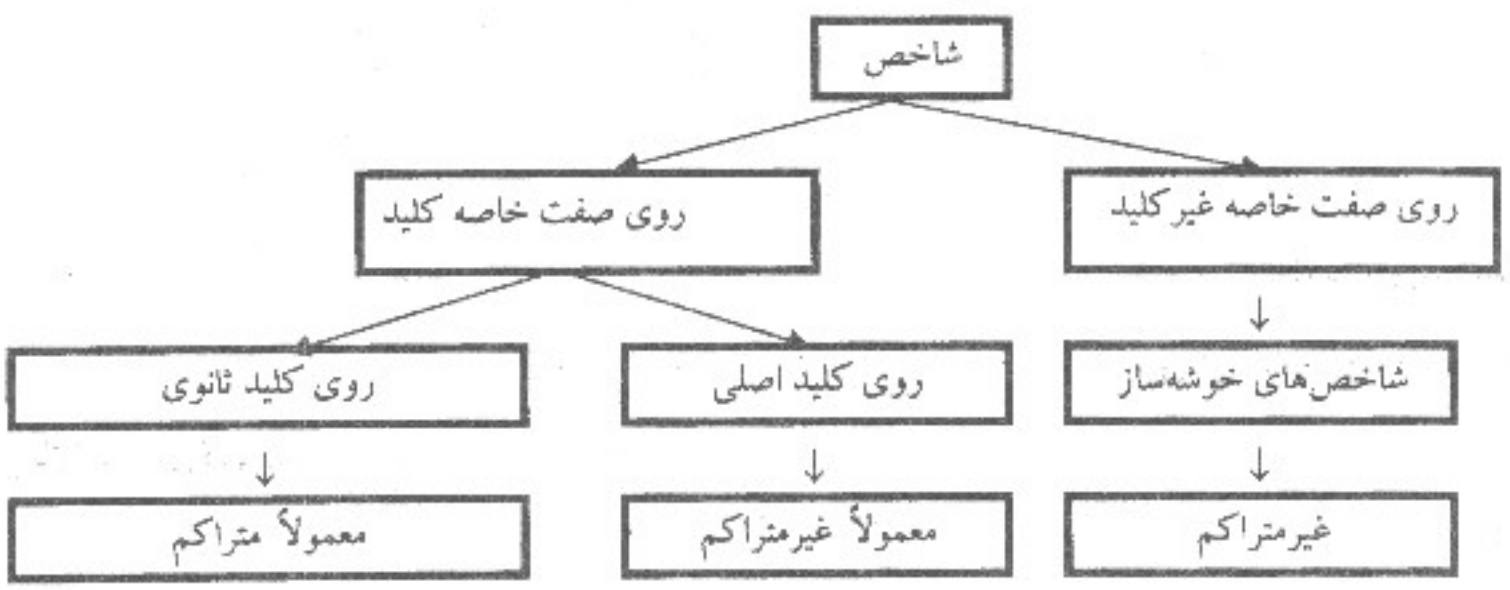
$$\text{سطح اول} \begin{cases} e_1 = \frac{10^6}{10} = 10^5 \\ b_1 = \left\lceil \frac{10^5}{100} \right\rceil = 1000 \\ s_1 = 2000 \times 1000 = 2 \times 10^6 \end{cases} \quad \text{سطح دوم} \begin{cases} e_2 = b_1 = 1000 \\ b_2 = \left\lceil \frac{1000}{100} \right\rceil = 10 \\ s_2 = 2000 \times 10 = 2 \times 10^4 \end{cases} \quad \text{سطح سوم} \begin{cases} e_3 = b_2 = 10 \\ s_3 = 10 \times 20 \\ \quad = 200 \end{cases}$$

عمق شاخص:

$$X = \lceil \log_{100} 10^5 \rceil = 3$$

جمع‌بندی: انواع شاخص

باتوجه به مباحث گذشته، می‌توان شاخص را در انواع زیر رده‌بندی کرد:



معایب شاخص‌بندی

معایب ساختارهای شاخص‌دار به ویژه در حالت چند شاخصی عبارتند از:

۱ـ مصرف حافظه برای ایجاد شاخص‌ها

۲ـ فزونکاری (Over head) در عملیات ذخیره‌سازی به ویژه در شاخص پویا (Dynamic index) که خواهیم دید.

ساختار ترتیبی شاخص‌دار (Indexed Sequential (IS))

معرفی ساختار

این ساختار برای تسریع واکشی تک رکورد از یک فایل ترتیبی طراحی و ایجاد می‌شود. اجزاء تشکیل دهنده آن عبارتند از:

۱ـ فایل ترتیبی که به آن اصطلاحاً ناحیه اصلی (Main area (Primary area)) می‌گویند.

۲ـ ناحیه سرریزی (Overflow area) برای انجام عملیات ذخیره‌سازی پس از لود اولیه.

۳ـ نشانه‌روها

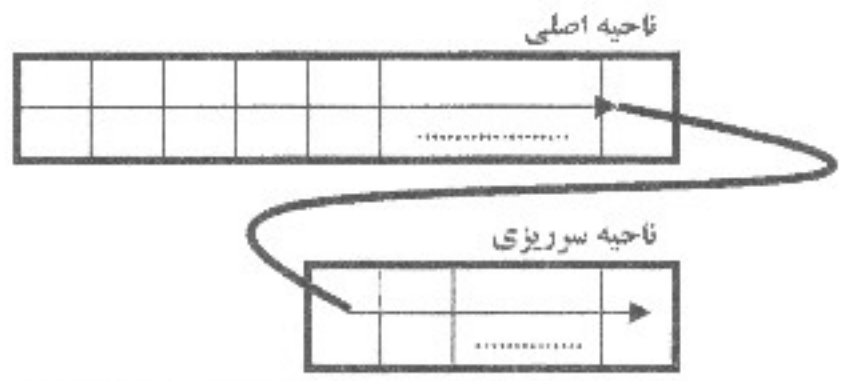
۴ـ مجموعه شاخص

فایل ترتیبی روی صفت خاصه کلید، مرتب است. ناحیه سرریزی به شرحی که خواهیم دید به نحوی انتخاب می‌شود که پردازش سریال رکوردها تسهیل شود.

شاخص در این ساختار می تواند سخت‌افزاری یا نرم‌افزاری باشد و به صورت غیرمتراکم ایجاد می‌شود و فقط ناظر است به رکوردهای ناحیه اصلی و از این نظر فاقد پویایی است و می‌گوییم شاخص ایستا (Static index) است.

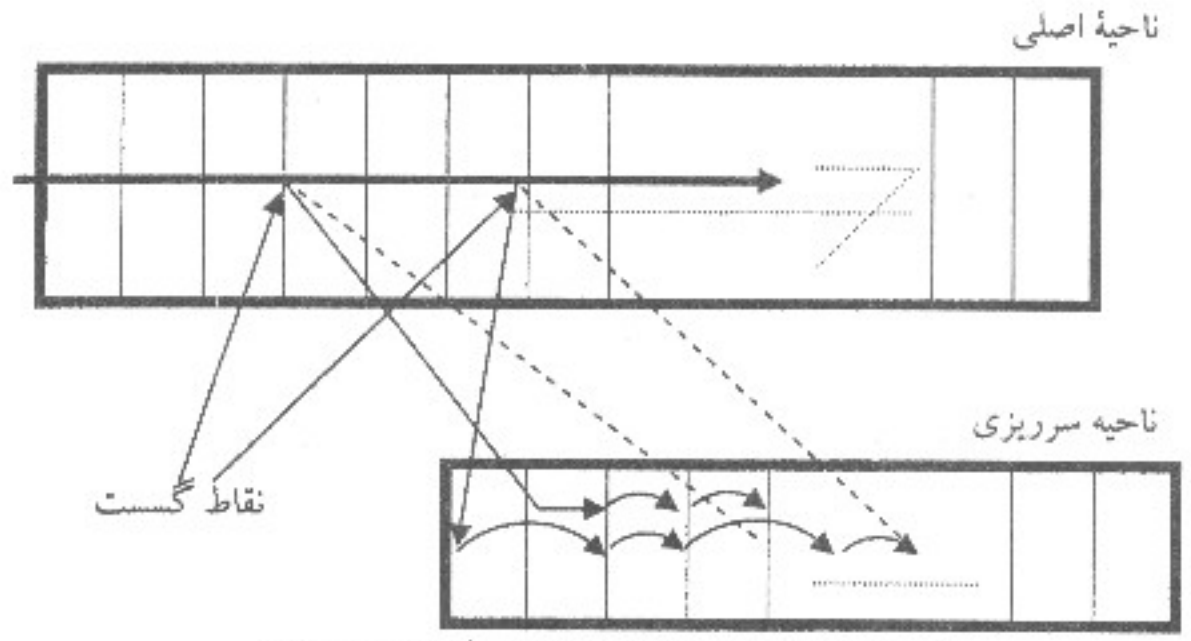
از شاخص برای تسریع واکشی رکوردها استفاده می‌شود و در عملیاتی مثل خواندن تمام فایل به طور پی‌درپی یا سریال کاربرد ندارد. نحوه انجام عملیات در این ساختار چنین است:

الف) خواندن پی در پی: سیستم، ناحیه اصلی و ناحیه سرریزی را بلاک به بلاک می‌خواند و چون رکوردهای ناحیه سرریزی مرتب نیستند، پس در این نحوه خواندن، رکوردها به طور سریال خوانده نمی‌شوند.



شکل ۸: نمایش منطقی مسیر خواندن پی‌درپی تمام فایل

ب) خواندن فایل به صورت سریال: بلاک‌های ناحیه اصلی روی کلید خوانده می‌شوند تا به بلاک یا رکوردی برسیم که دارای یک نشانه‌رو به ناحیه سرریزی است. از این پس، به ناحیه سرریزی رفته، خواندن را با طی کردن زنجیره سرریزی‌ها، دنبال می‌کنیم تا به پایان زنجیره برسیم و مجدداً به ناحیه اصلی بازگشت می‌کنیم (به نقطه گسست) و بدین نحو خواندن سریال ادامه می‌یابد.

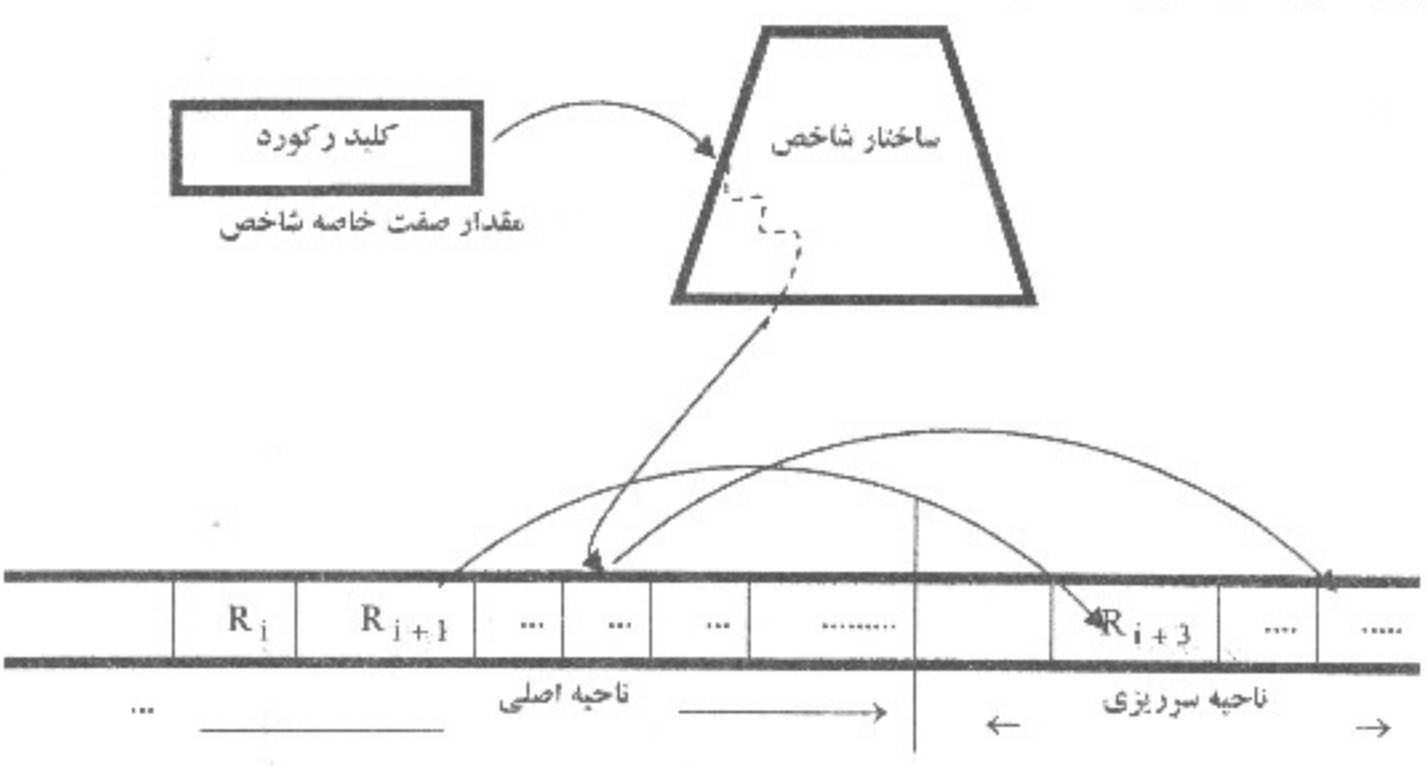


شکل ۹: نمایش منطقی مسیر خواندن تمام فایل به طور سریال

به نحوی که خواهیم دید. از ناحیه اصلی، تعداد زنجیره (Chain) داریم ادامه یافته در ناحیه سرریزی و همین زنجیره‌ها، پردازش سریال را امکان‌پذیر می‌سازند و نیز در واکشی یک رکورد، چنانچه رکورد مورد نظر در ناحیه سرریزی باشد، این زنجیره‌ها پیمایش می‌شوند.

ج) **واکشی رکورد از روی کلید:** برای واکشی رکورد، باید در فایل شاخص جستجو کرد و با یافتن مدخل مربوطه در فایل شاخص، قسمتی از ناحیه اصلی و در صورت لزوم، زنجیره سرریزی‌های آن قسمت باید خوانده شود تا رکورد مورد نظر در صورت وجود، واکشی شود.

در شکل زیر نمایش منطقی نحوه عمل در بازیابی تک رکورد دیده می‌شود.



شکل ۱۰: نمایش منطقی بازیابی تک رکورد از طریق شاخص

دیدیم که شاخص، خود یک فایل یا بخشی از فایل داده‌ای است با ساختار درونی خاص خود و رکوردهایی به طول V + P بایت. معمولاً شاخص بالاترین سطح، موسوم به سرشاخص در حافظه اصلی نگاهداری می‌شود و بلاک‌های سایر سطوح شاخص روی دیسک، در شیارهای آغازین هر استوانه و یا به تمامی در استوانه‌های آغازین ذخیره می‌شوند. در مثال شاخص سخت‌افزاری دیدیم که شاخص هر شیار در همان شیار ذخیره می‌شد.

بررسی مشکل سرریزی

در بررسی مشکل سرریزی باید به سؤالات زیر پاسخ داد:

۱) فضای لازم برای درج رکوردهای سرریزی چگونه انتخاب شود (طرح تفصیص منطقی)؟

پاسخ:

در پاسخ به سؤال اول، سه راه حل به نظر می‌رسد:

الف) در نظر گرفتن جا در هر بلاک در لود اولیه (چگالی اولیه کمتر از صددرصد باشد)

ب) ایجاد یک فایل جداگانه (مثل راه‌حل فایل ترتیبی)

ج) در نظر گرفتن ناحیه‌ای جداگانه در همان فایل داده‌ای.

راه حل الف: هر چند به نظر می‌رسد که از نظر قوی بودن لوکالیتی رکوردها، راه خوبی باشد، زیرا رکوردهای سرریزی در همان بلاک رکوردهای اصلی درج می‌شوند. ولی اگر توزیع سرریزی‌ها نایکنواخت باشد، برخی بلاک‌ها با کمبود جا برای درج سرریزی‌ها مواجه می‌شوند (زیرا پر می‌شوند) و برخی دیگر پر نشده باقی می‌مانند و برای بلاک‌های پر شده باز هم مسئله سرریزی‌ها مطرح خواهد بود.

راه حل ب: راه حل زمان‌گیری است، زیرا عملیات اساساً در فایل جداگانه‌ای انجام می‌شود و با توجه به این‌که باید بین رکوردهای فایل اصلی و رکوردهای سرریزی ارتباط ایجاد شود (به کمک نشانه‌روها)، ایجاد این ارتباط بین دو فایل جداگانه، از ایجاد ارتباط بین رکوردهای یک فایل واحد، دشوارتر خواهد بود و ضمناً لوکالیتی رکوردهای سرریزی ضعیف می‌شود.

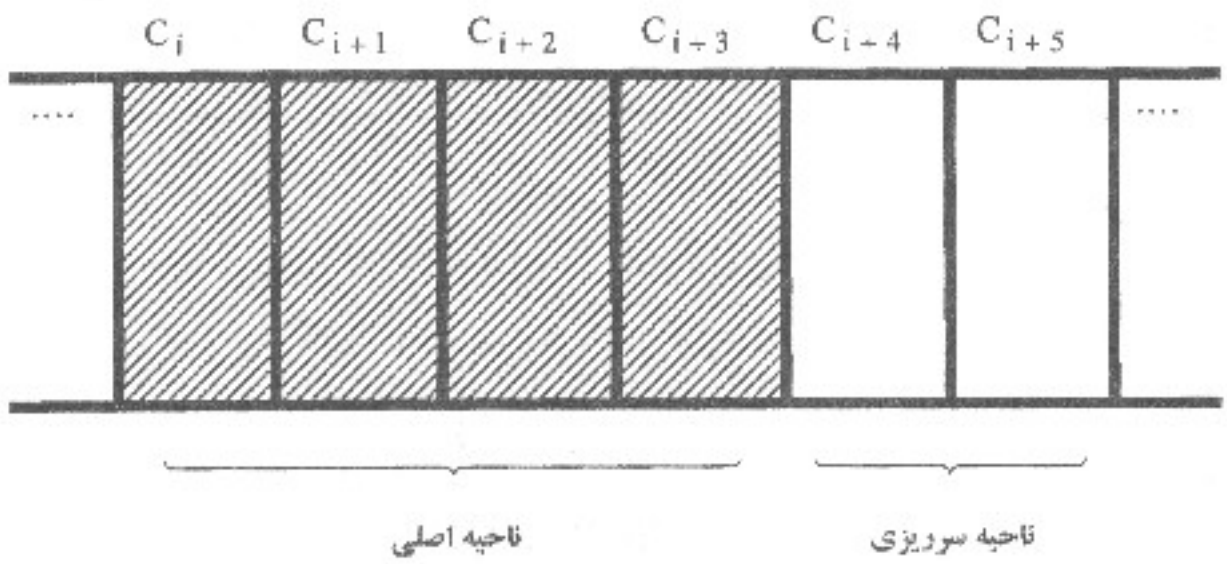
راه حل ج: مناسب‌ترین راه حل در این ساختار است. رکوردهای سرریزی با استفاده از تکنیک‌هایی که خواهیم دید، درج خواهند شد.

۲) فضای انتخاب شده، چگونه در محیط فیزیکی (روی دیسک) به فایل تخصیص یابد (طرح تخصیص فیزیکی)؟

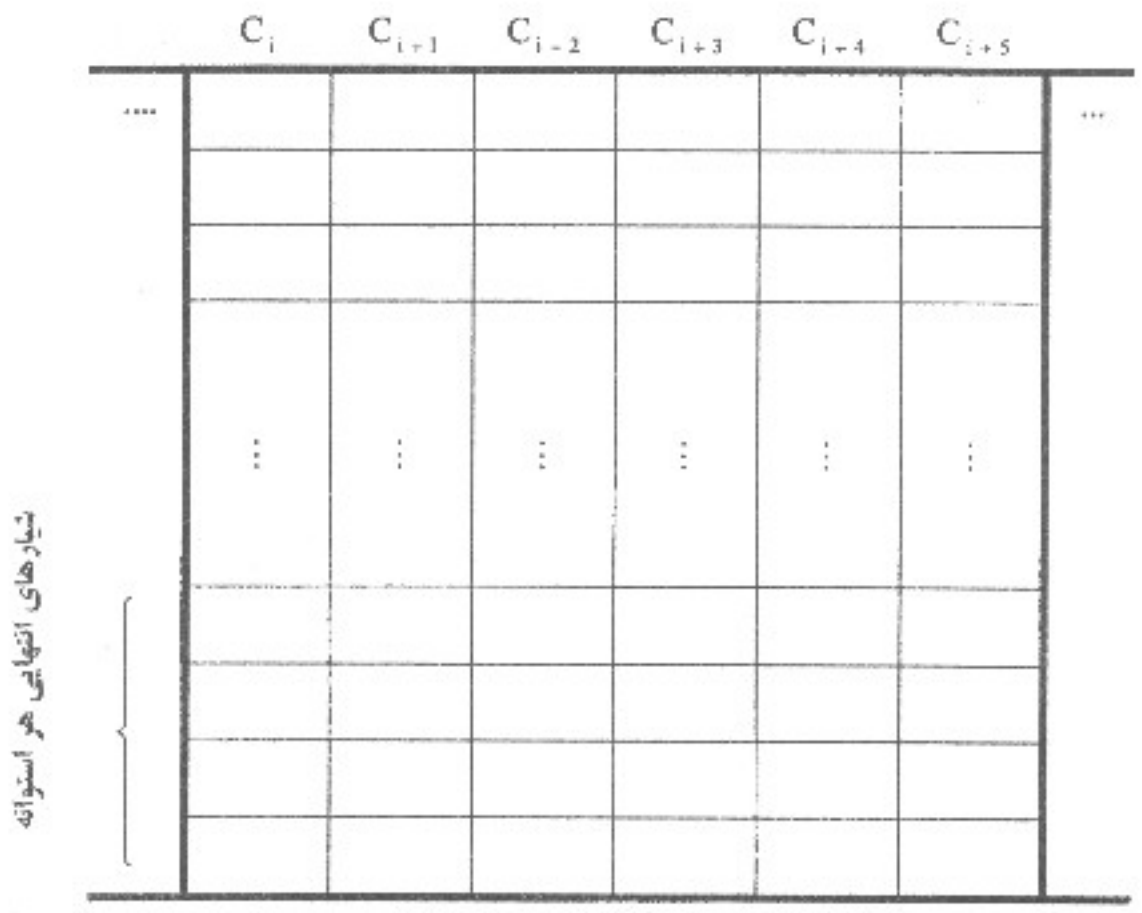
در پاسخ به سؤال دوم، دو راه وجود دارد:

۱) تخصیص استوانه‌هایی در انتهای فایل، برای ایجاد ناحیه جداگانه.

۲) تخصیص شیارهایی در انتهای هر استوانه، به عنوان ناحیه سرریزی استوانه. (Cylinder Overflow area)

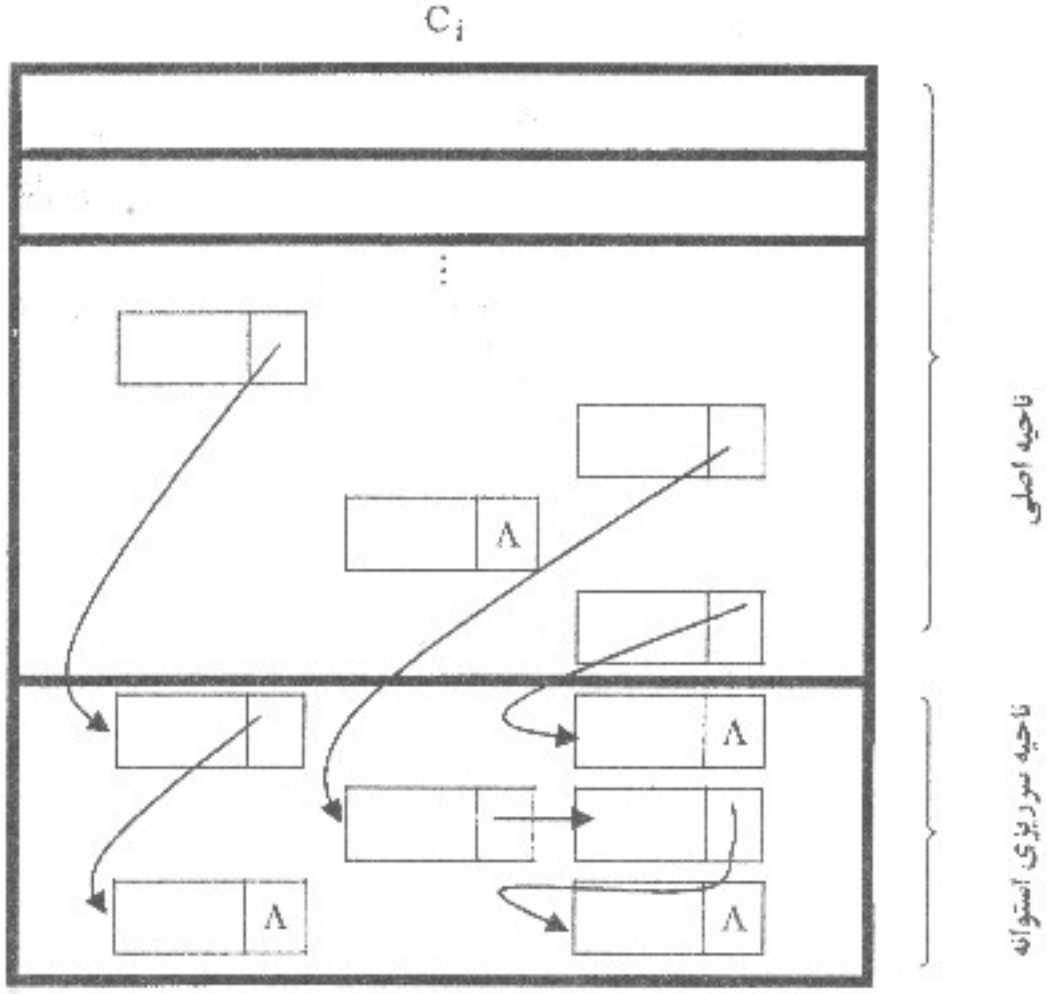


شکل ۱۱: ناحیه جداگانه روی استوانه‌های انتهایی فایل



شکل ۱۲: ناحیه سرریزی در شیارهای انتهایی هر استوانه

راه حل اول در شکل ۱۱ نمایش داده شده است. این راه حل مناسب نیست، زیرا سبب می‌شود که لوکالیتی رکوردهای سرریزی ضعیف شود و در نتیجه متوسط زمان استوانه‌جویی افزایش یابد. راه حل دوم در شکل ۱۲ دیده می‌شود، این راه حل، متوسط زمان استوانه‌جویی را کاهش می‌دهد؛ زیرا رکوردهای سرریزی هر استوانه در همان استوانه جای دارند. البته وقتی که ناحیه سرریزی یک استوانه پر شود، ناحیه دیگری برای درج سرریزی‌ها باید ایجاد کرد (ناحیه سرریزی ثانویه (Secondary Overflow area)) و یا این‌که فایل را سازماندهی مجدد کرد. شکل ۱۳ وضعیت کلی رکوردهای یک استوانه، و رکوردهای سرریزی آن‌ها را نشان می‌دهد.



شکل ۱۳: وضعیت کلی استوانه و رکوردهای اصلی و سرریزی

در راه‌حل اخیر این سؤال مطرح می‌شود که اندازه ناحیه سرریزی در هر استوانه، چه باید باشد و آیا در تمام استوانه‌ها، این اندازه یکسان است؟ منطقاً نیازی به مساوی بودن اندازه ناحیه سرریزی در هر استوانه نیست و بستگی به حجم عملیات درج در هر استوانه دارد. اگر بتوان حجم این عملیات را ارزیابی کرد، آن‌گاه ممکن است متناسب با آن، اندازه ناحیه را در هر استوانه انتخاب کرد. ولی معمولاً اندازه ناحیه سرریزی در تمام استوانه‌ها یکسان گرفته می‌شود و البته امکان دارد در صورت نایکنواخت بودن توزیع درجی‌ها، بعضی از استوانه‌ها با کمبود جا مواجه شوند و به ناحیه سرریزی ثانوی نیاز داشته باشند، در حالی که در برخی دیگر از استوانه‌ها، ناحیه سرریزی پر نشود و یا این‌که اساساً درجی در آن‌ها صورت نگیرد.

۳) عمل درج سرریزی‌ها با چه تکنیکی انجام شود؟

در پاسخ به سؤال سوم، تکنیک‌های موجود را مطرح می‌کنیم:

تکنیک‌های درج سرریزی

دو تکنیک وجود دارد:

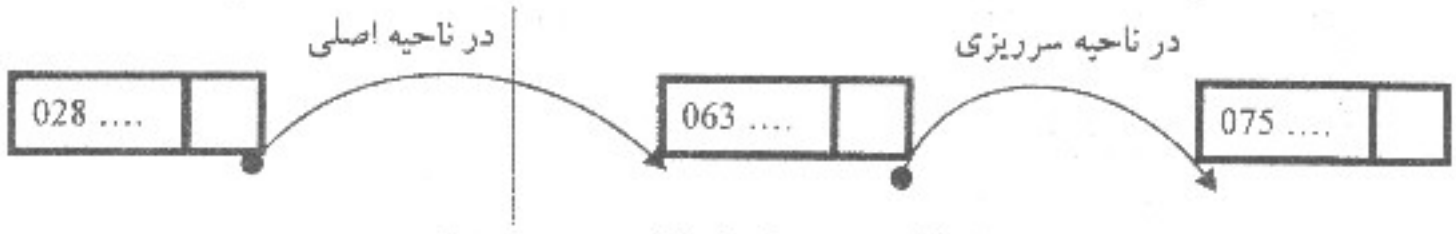
۱) درج در اولین بلاک جادار در ناحیه سرریزی

۲) درج با جابجایی (Push through)

درج در اولین بلاک‌جا دار در ناحیه سرریزی

در این تکنیک رکورد جدید، مستقیماً وارد بلاکی از ناحیه سرریزی می‌شود و در اولین مکان آزاد جای می‌گیرد (در اولین بلاک جادار). سپس از رکورد منطقاً پیشین به رکورد درج شده نشانه‌رو ایجاد می‌شود و بدین ترتیب زنجیره رکوردهای سرریزی پدید می‌آید. در این تکنیک برای هر رکورد از ناحیه اصلی و ناحیه سرریزی یک فیلد نشانه‌رو وجود دارد. البته ممکن است در بعضی از رکوردها، محتوای این فیلد NULL باشد.

در فیلد نشانه‌رو آخرین رکورد هر زنجیره، NULL داریم به معنای این‌که پایان زنجیره است. پس در این روش، زنجیره‌هایی داریم که مبداء آن‌ها، رکوردهایی از ناحیه اصلی است و در ناحیه سرریزی ادامه می‌یابند و می‌گوییم زنجیره سرریزی‌های رکورد (Record Overflow Chain) داریم.



شکل ۱۴: زنجیره در تکنیک اول درج سرریزی‌ها

تکنیک درج با جابه‌جایی

در این تکنیک سعی می‌شود که نظم رکوردها در بلاک‌های ناحیه اصلی حفظ شود. رکورد جدید در بلاک مربوط در ناحیه اصلی، در محلی که منطقاً باید جای گیرد یعنی از رکورد منطقاً قبلی‌اش وارد می‌شود، و رکوردهای بعدی همان بلاک (البته غیر از اولین رکورد بلاک) به سمت انتهای بلاک شیفت داده می‌شوند و در نتیجه رکورد آخر بلاک جابجا شده، به اولین بلاک جادار در ناحیه سرریزی، منتقل می‌شود. البته زنجیره رکوردهای سرریزی هم متناسباً ایجاد یا اصلاح می‌گردد. در این تکنیک، زنجیره سرریزی‌های بلاک (Block Overflow Chain) داریم.

در این روش، برای هر بلاک از ناحیه اصلی یک نشانه‌رو داریم (و نه برای هر رکورد). ضمناً ممکن است، رکوردهای به هم زنجیر شده، در بدترین حالت، هر یک در یک بلاک از ناحیه سرریزی باشد، *طول زنجیره در این روش از روش قبلی بیشتر است، ولی در عوض پردازش سریال فایل تسهیل می‌گردد.*

نکته: در هیچ یک از دو تکنیک، فایل شاخص تغییر نمی‌کند، زیرا در این ساختار شاخص فقط ناظر است به ناحیه اصلی به عبارت دیگر، ساختار شاخص، حالت پویا ندارد. هر یک از این دو تکنیک، مزایا و معایبی دارند.

فرمول‌های مهم فایل ترتیبی شاخص‌دار:

☑ y = ظرفیت نشانه روی

$$y = \left\lfloor \frac{B}{v + P} \right\rfloor$$

(تعداد مدخل‌های بلاک شاخص را ظرفیت نشانه رو می‌گوئیم)

☑ x = عمق یا ژرفای شاخص

$$y^{x-1} \le e_1 \le y^x \quad *$$

$$\begin{cases} x = \lceil \log_y^{e_1} \rceil \\ e_1 = \left\lceil \frac{n}{B_f} \right\rceil = b \end{cases} \qquad B_f = \left\lfloor \frac{B}{R} \right\rfloor$$

تعداد مدخل‌های سطح اول — تعداد بلاک‌های فایل داده‌ای

سطح اول	$e_1 = \left\lceil \frac{n}{B_f} \right\rceil = b$	
	$b_1 = \left\lceil \frac{e_1}{y} \right\rceil$	
	$s_1 = B \times b_1$	

سطح iام (از سطح دوم به بعد)	تعداد مدخل‌های سطح iام	$e_i = b_{i-1}$
	تعداد بلاک‌های سطح iام	$b_i = \left\lceil \frac{e_i}{y} \right\rceil$
	حافظه مصرفی سطح iام	$s_i = B \times b_i$

موارد استفاده ساختار:

این ساختار در کاربردهایی استفاده می‌شود که در آن‌ها پردازش سریال فایل برحسب مقادیر فقط یک صفت خاصهٔ (کلید) مطرح بوده، به علاوه واکشی تک رکوردها از طریق مقدار کلید آن‌ها عمل رایجی باشد. در اغلب سیستم‌های داده‌پردازی تجاری ـ مدیریتی، این ساختار مورد استفاده قرار می‌گیرد.

ارزیابی کارایی

- سطح اول شاخص غیر متراکم است.
- فایل شاخص همتوالی با فایل داده‌ای.

- بلاک‌های شاخص در یک استوانه جای دارند و ناحیه سرریزی استوانه داریم.
- تکنیک درج، درج با جابجایی است.
- ساختار شاخص، ایستا است.
- بعد از سازمان‌دهی مجدد، بلاک‌های شاخص و ناحیه اصلی پر و بلاک‌های ناحیه سرریزی خالی هستند.
- رکوردهای حذف شدنی، بلافاصله به طور فیزیکی محو نمی‌شوند، بلکه نشانگر «حذف شده» می‌خورند تا در سازماندهی مجدد، واقعاً حذف گردند.
- هم فایل داده‌ای و هم فایل شاخص، بلاک‌بندی شده‌اند (طول هر بلاک B بایت).

متوسط اندازه رکورد

برای محاسبه متوسط اندازه رکورد، باید عوامل زیر را در نظر گرفت:

۱) حافظه لازم برای یک رکورد از ناحیه اصلی.

۲) حافظه مصرف شده برای ناحیه سرریزی به ازاء یک رکورد از ناحیه اصلی.

۳) حافظه مصرف شده برای شاخص به ازاء یک رکورد از ناحیه اصلی.

$$R = R_{data} + R_{over} + R_{index} \qquad R_{data} = aV + \frac{P}{B_f}$$

$$R_{over} = \frac{o}{n+o} \cdot R' \quad , \quad R' = aV + P$$

$$R_{index} = \frac{\text{کل حافظه مصرف شده برای شاخص}}{n+o} = \frac{S_I}{n+o}$$

واکشی رکورد T_F

۱ـ بررسی سرشاخص (که در حافظه اصلی است).

۲ـ جستجو در سطوح شاخص تا رسیدن به مدخل مربوطه در سطح اول. برای این منظور، یک بلاک شاخص در هر سطح باید خوانده و بررسی شود.

۳ـ خواندن بلاکی از ناحیه اصلی که آدرس آن از مدخل مربوطه در سطح اول شاخص بدست می‌آید.

۴ـ به احتمالی، رفتن به ناحیه سرریزی و جستجو در زنجیره سرریزی‌ها.

بنابراین، برای ارزیابی زمان واکشی یک رکورد، زمان‌های زیر را باید در نظر گرفت:

۱) $T_{F_{main}}$: زمان واکشی رکورد از ناحیه اصلی

۲) $T_{F_{over}}$: زمان یافتن رکورد از ناحیه سرریزی

زمان دوم با یک ضریب احتمالاتی دخالت داده می‌شود. روند نمای واکشی در ادامه آمده است.

$$T_F = T_{F_{main}} + T_{F_{over}}$$

محاسبه $T_{F_{min}}$:

$$T_{F_{min}} = C_B + (x-1)(s+r+b_R) + s + r + b_R$$

که در آن داریم:

C_B: زمان بررسی سرشاخص.

$(x-1)(s+r+b_R)$: زمان خواندن بلاک‌های شاخص در $x-1$ سطح.

$(s+r+b_R)$: خواندن بلاک در ناحیه اصلی.

با فرض به این‌که بلاک‌های شاخص در یک استوانه جای دارند، خواهیم داشت:

$$T_{F_{min}} = C_B + 2s + x(r+b_R)$$

$$T_F = C_B + 2s + \left(x + \frac{1}{2}\frac{o'}{n+o'} + \frac{1}{2}\frac{o'^2}{n(n+o')}B_f\right)(r+b_R)$$

بازیابی رکورد بعدی T_N

برای بازیابی رکورد بعدی، باید از آخرین رکورد واکشی شده، شروع کنیم و ببینیم آیا رکورد بعدی در بلاکی از ناحیه اصلی است یا در بلاکی از ناحیه سرریزی. بنابراین در یک ارزیابی تقریبی، می‌توان گفت که اگر رکورد بعدی در بلاکی از ناحیه اصلی باشد، با زمان $\frac{1}{B_f}.b_R$ بدست می‌آید. احتمال این‌که رکورد بعدی در ناحیه سرریزی باشد $\frac{o'}{n+o'}$ است.

پس می‌توان نوشت:

$$\boxed{T_N = \frac{1}{B_f}b_R + \frac{o'}{n+o'}(r+b_R)}$$

اما برای ارزیابی دقیقتر این زمان، باید مکان رکورد بعدی را دقیقتر مشخص کرد، شش حالت وجود دارد و هر یک از حالات، به احتمالی ممکن است بروز کند. پس از در نظر گرفتن احتمالات و بیان آن‌ها به کمک مقدار Pro و ساده کردن خواهیم داشت:

$$T_N = \left(\frac{1-Pro}{B_f} + Pro\right)(r+b_R)$$

و یا

$$\boxed{T_N = \frac{n+o'.B_f}{(n+o')B_f}(r+b_R)}$$

حالات شش‌گانه:

۱ـ رکورد فعلی در بلاکی از ناحیه اصلی است و رکورد بعدی نیز در همان بلاک و بلاک در بافر است.

۲ـ رکورد فعلی آخرین رکورد بلاک است از ناحیه اصلی و رکورد بعدی در بلاک بعدی است از همان استوانه.

۳ـ رکورد فعلی آخرین رکورد بلاک از آخرین بلاک استوانه است و رکورد بعدی در بلاک بعدی است از استوانه دیگر.

۴ـ رکورد فعلی آخرین رکورد بلاک است و رکورد بعدی در بلاکی از ناحیه سرریزی است.

۵ ـ رکورد فعلی در بلاکی از ناحیه سرریزی است و رکورد بعدی هم در بلاکی از ناحیه سرریزی و از همان استوانه.

۶ ـ رکورد فعلی در بلاکی از ناحیه سرریزی است و رکورد بعدی در بلاکی از ناحیه اصلی.

عمل درج T_I

برای درج یک رکورد جدید، عملیات زیر باید انجام شود: (با توجه به درج با تکنیک جابجایی)

۱ـ یافتن بلاکی که رکورد باید در آن درج شود.

۲ـ وارد کردن رکورد در این بلاک ضمن خارج کردن آخرین رکورد بلاک و قرار دادنش در بافر کمکی.

۳ـ بازنویسی این بلاک.

۴ـ خواندن بلاکی از ناحیه سرریزی.

۵ ـ وارد کردن رکورد خارج شده از بلاک اصلی، در این بلاک.

۶ ـ بازنویسی همین رکورد.

$$T_I = T_F + T_{RW} + r + b_{rl} + T_{RW}$$

$$\boxed{T_I = T_F + 5r + b_{rl}}$$

عمل بهنگام‌سازی T_U

ابتدا حالتی را در نظر می‌گیریم که در آن مقدار کلید در اثر بهنگام‌سازی عوض نشود و طول رکورد نیز تغییر نکند، در این حالت می‌توان رکورد را در جا بهنگام در آورد.

عملیات لازم:

۱ـ واکشی رکورد بهنگام درآمدنی

۲ـ ایجاد نسخه جدید (در بافر)

۳ـ بازنویسی نسخه جدید

$$T_{U\ inplace} = T_F + T_{RW}$$

$$\boxed{T_{U\ inplace} = T_F + 2r}$$

و اگر بخواهیم رکورد را حذف کنیم، زمان برابر است با همان زمان $T_{U\ inplace}$.

اما در حالت کلی، نسخه قدیم رکورد نشانگر حذف می‌خورد و رکورد بهنگام درآمده (نسخه جدید که باید در بافر ساخته شود) درج می‌شود و در واقع بهنگام‌سازی برون ازجا انجام می‌شود.

$$T_{U\ outplace} = T_F + T_{RW} + T_I$$

$$\boxed{T_{U\ outplace} = 2T_F + 7r + b_{rl}}$$

خواندن تمام فایل T_X

۱ـ در حالت سریال:

اولین رکورد واکشی می‌شود و بقیه رکوردها طی یک سلسله عملیات بازیابی رکورد بعدی، خوانده می‌شوند.

$$T_{X_{ser}} = T_F + (n + o' - 1)T_N$$

اگر تکنیک درج را درج با جابجائی فرض کنیم، زمان خواندن سریال را به طرز دیگری هم می‌توان ارزیابی کرد. در این تکنیک، رکوردهای اولین بلاک ناحیه اصلی همیشه مرتب است. پس سیستم می‌تواند رکوردهای این بلاک را بخواند و سپس بقیه رکوردها را طی یک سلسله عملیات بازیابی بعدی به دست آورد و داریم:

$$\boxed{T_{N_{ser}} = s + r + b_n + (n + o' - B_f)\, T_N}$$

۲ـ در حالت پی‌درپی:

طبق روش معمول عمل می‌شود:

$$T_{X_{seq}} = (n + o')\frac{R}{t'}$$

سازماندهی مجدد: T_y

در این ساختار وقتی که ناحیه سرریزی (اولیه و در صورت وجود، ثانوی) پر شود، می‌توان فایل را سازماندهی مجدد کرد. (یا طول زنجیره‌ها طولانی شوند)

۱ـ خواندن سریال فایل (تا بتوان مجدداً ناحیه اصلی جدید را به صورت ترتیبی ایجاد کرد).

۲ـ بلاک بندی رکوردها ضمن حذف رکوردهای حذف شدنی.

۳ـ بازنویسی نسخه جدید فایل

۴ـ بازسازی ساختار شاخص که بلاک‌هایش به تدریج در بافر ساخته می‌شوند.

برای تسریع عملیات، بهتر است سیستم برای هر سطح شاخص حداقل یک بافر و برای تولید نسخه جدید فایل حداقل دو بافر داشته باشد.

$$T_Y = T_{X_{ser}} + (n + o - d)\frac{R}{t'} + \frac{S_1}{t'}$$

$\frac{S_1}{t'}$: زمان بازنویسی بلاک های شاخص

در پایان معایب عمده این ساختار را یادآور می‌شویم:

۱ـ عدم تقارن.

۲ـ ایستا بودن شاخص.

۳ـ مسئله درج سرریزی‌ها (زنجیره‌های طولانی کارائی سیستم را کاهش می‌دهند).

فایل چند شاخصی

این ساختار برای رفع معایبی که در ساختار ترتیبی شاخص‌دار دیده شد، طراحی شده است به عبارت دیگر، ساختار چنان است که پدیده عدم تقارن در آن وجود ندارد، زیرا روی تعدادی، حتی تمام صفات خاصه، می‌توان شاخص داشت. مسئله رکوردهای سرریزی، به صورتی که در ساختار سوم مطرح بود، در اینجا وجود ندارد، یعنی درج رکوردهای جدید آسان‌تر و پویاتر است و بالاخره خود ساختار شاخص حالت پویا دارد و همروند با تغییرات در فایل داده‌ای، قابل تنظیم است.

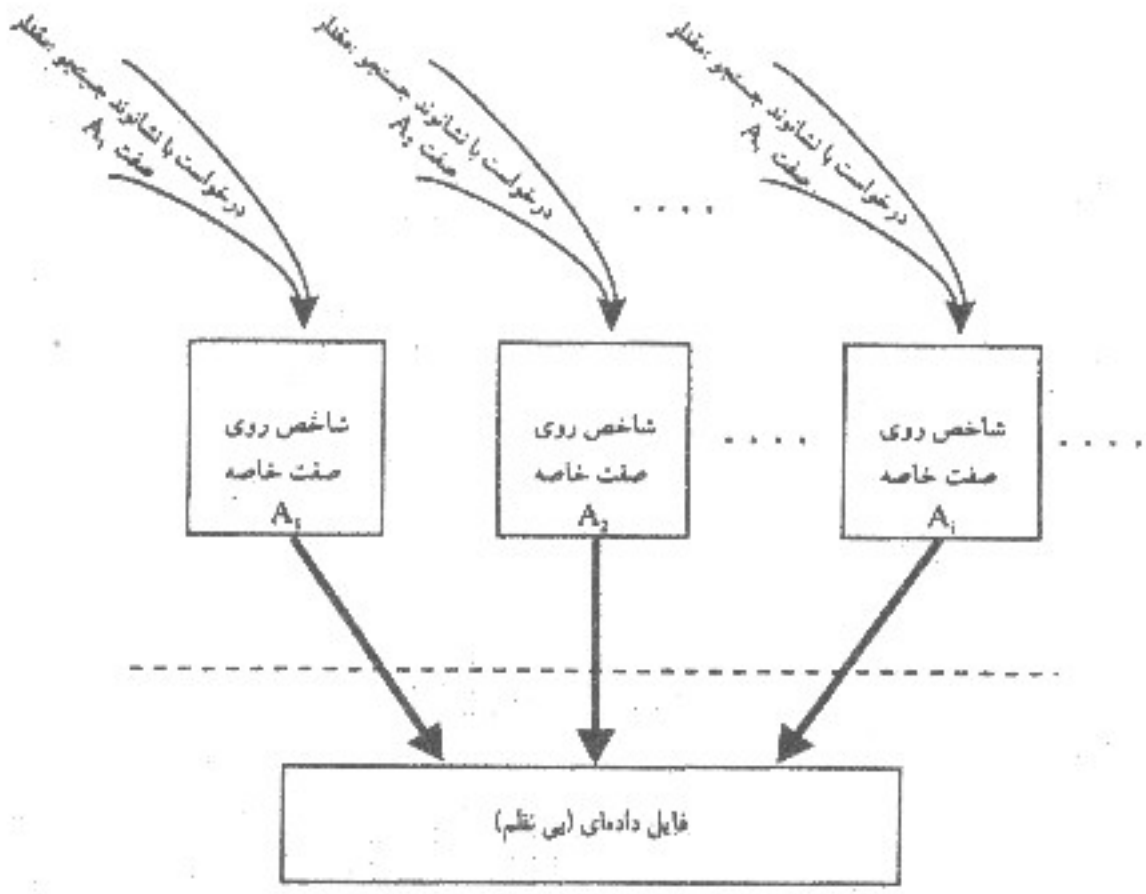
اجزاء اصلی ساختار

۱) فایل داده‌ای، که در بی‌نظم‌ترین شکلش می‌تواند حتی پایل باشد.

۲) چندین فایل شاخص، به تعداد صفات خاصه می‌توان فایل شاخص داشت و بدین ترتیب کاربر می‌تواند، هر یک از صفات خاصه را به عنوان نشانوند جستجو در پرس و جوهایش به کار ببرد.

اگر a تعداد صفات خاصه در فایل باشد، حداکثر a فایل شاخص می‌توان داشت. البته ایجاد شاخص روی ترکیب‌های مختلف صفات خاصه نیز امکان‌پذیر است. حتی می‌توان روی اجزایی از صفت خاصه هم شاخص ایجاد کرد (حداقل از لحاظ تئوریک). از آن‌جا که در یک رکورد به طور متوسط، a′ صفت خاصه وجود دارد، لذا به یک رکورد، a′ شاخص ناظر است. پس این ساختار در اساس از نظر فایل داده‌ای، همان پایل است، اما مجهز به یک استراتژی دستیابی قوی، پویا و سریع.

نکته: کاربر می‌تواند برای هر تعداد از صفات خاصه‌ای که در فایل دارد، درخواست ایجاد شاخص کند و برای واکشی سریع تک رکوردها، الزامی ندارد که حتماً از کلید اصلی به عنوان نشاوند جستجو استفاده نماید. در شکل زیر نمایش منطقی این ساختار دیده می‌شود.



نمایش منطقی ساختار چند شاخصی

چون فایل داده‌ای پایل است، لذا درج رکوردهای جدید، مثل درج در پایل، انجام می‌شود، و البته شاخص‌ها باید متناسباً تنظیم شوند.

■ **تعریف فایل وارون (Inverted File)**

وقتی که روی تمام صفات خاصه شاخص داشته باشیم، اصطلاحاً فایل را کاملاً وارون می‌گویند.

■ **نشست رکوردها روی رسانه**

چون فایل داده‌ای نظم ندارد، نشست رکوردها روی رسانه محدودیت ساختاری ندارد.

■ **ضابطه انتخاب صفات خاصه شاخص**

گفتیم که حداکثر، a ساختار شاخص می‌توان داشت، لذا لزومی ندارد که روی تمام صفات خاصه، شاخص ایجاد شود. می‌توان بین صفات خاصه قائل به اولویت شد و آن صفاتی را برگزید که در بیشترین درخواست‌ها، به عنوان نشاوند جستجو به کار برده می‌شوند.

متوسط تعداد مدخل‌ها در سطح اول شاخص

اگر n' تعداد مدخل‌ها در سطح اول شاخص روی صفت خاصه A_i و n' متوسط تعداد مدخل‌ها در سطح اول یک شاخص باشد، با فرض وجود a شاخص، می‌توان نوشت:

اما n' از رابطه دیگری هم به دست می‌آید:

$$\boxed{n' = \frac{a'}{a} n}$$

در این رابطه فرض بر این است که روی تمام صفات خاصه شاخص داریم.

(n تعداد رکوردهای فایل داده‌ای است). سه حالت ممکن است وجود داشته باشد:

۱) $a' = a$ یعنی تمام صفات خاصه در تمام نمونه رکوردها موجود باشند و برای هر صفت خاصه در هر رکورد، یک مقدار معلوم داشته باشیم. در این صورت: $n' = n$

۲) $a' < a$ در بعضی از نمونه رکوردها، بعضی از صفات خاصه را نداریم، در این صورت $n' < n$.

۳) $a' > a$ این حالت هنگامی بروز می‌کند که پدیده گروه اطلاع تکرار شونده و یا فقره اطلاع تکرار شونده (صفت خاصه ساده یا مرکب چند مقداری) داشته باشیم، به عبارت دیگر به ازاء یک صفت خاصه، چندین مقدار، هر یک در یک فیلد، در نمونه‌هایی از رکوردها موجود باشد. در این صورت $n' > n$.

مثال ۱: فایلی حاوی اطلاعاتی در مورد کارمندان داریم، با 20000 رکورد، در این فایل سوابق شغلی کارمندان را ذخیره کرده‌ایم. یک کارمند به طور متوسط 2.5 شغل در سابقه شغلی‌اش دارد. روی صفت خاصه شغل شاخص ایجاد می‌کنیم. این صفت خاصه تکرار شونده است.

$a = 1 \quad , \quad a' = 2.5 \quad , \quad n = 20000$

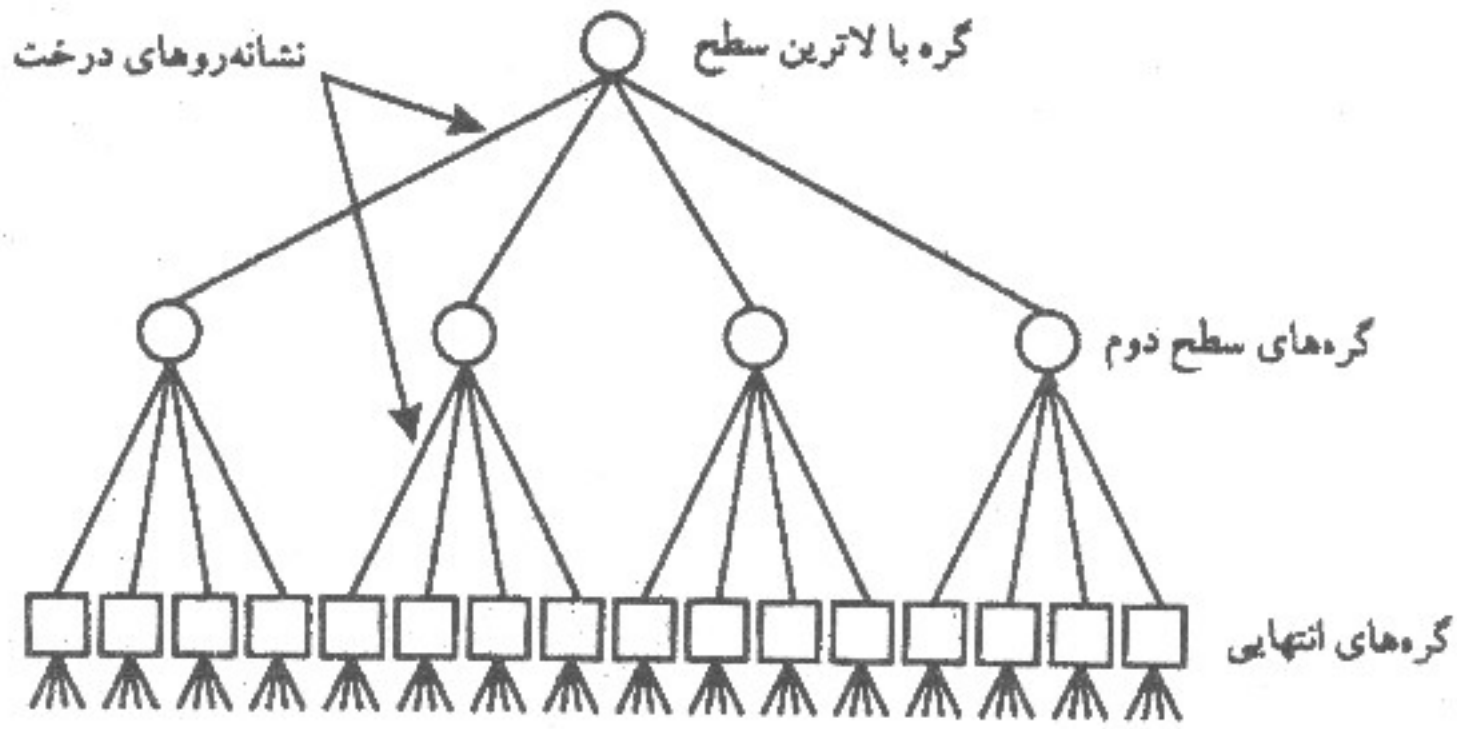
$n' = \frac{2.5}{1} \times 20000 = 50000$ مدخل

توجه داشته باشیم که در این محاسبه، a تعداد صفات خاصه‌ای است که روی آن‌ها شاخص داریم و در اینجا $a = 1$ است.

ساختار شاخص

تنها استراتژی دستیابی در این فایل، همان شاخص است. یعنی تعدادی فایل شاخص ناظر بر رکوردها داریم. فایل شاخص باید ساختاری داشته باشد که بتوان آن را، همروند با عملیات روی فایل داده‌ای، به طور پویا به هنگام درآورد. راه حل مطروحه در فایل ترتیبی شاخص‌دار یعنی ایجاد شاخص ایستا فاقد کارایی است.

راه حل اساسی این است که در ایجاد فایل شاخص از یک ساختار داده‌ای استفاده کنیم که رفتاری پویا داشته باشد. یک ساختار رایج B – Tree (Balanced Tree) (درخت متعادل) است. یادآور می‌شویم که B – Tree درختی است که در آن ژرفای تمام شاخه‌ها، از ریشه تا گره‌های انتهایی Terminal node یکسان است. شکل زیر وضع کلی درخت شاخص را نشان می‌دهد.

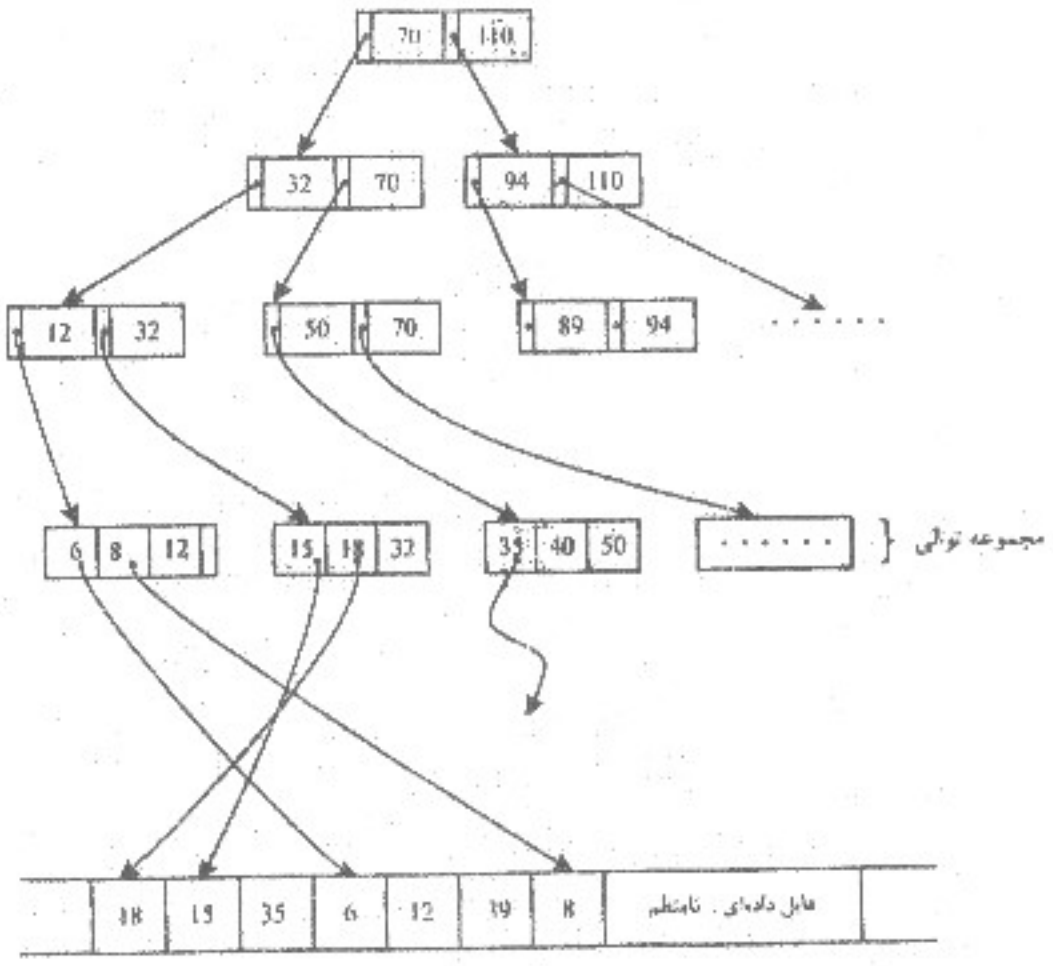


وضع کلی یک درخت شاخص

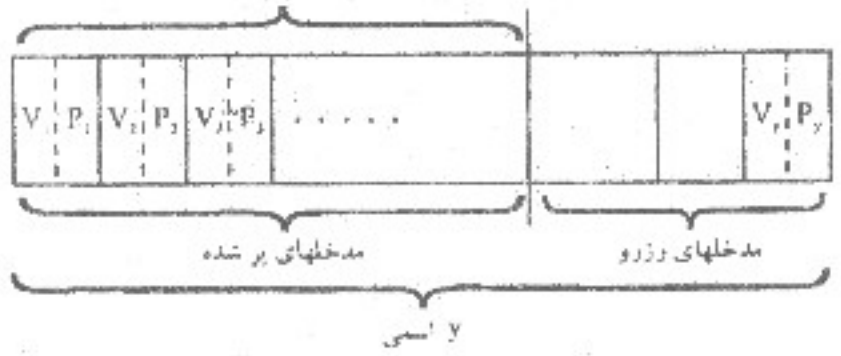
پایین‌ترین سطح شاخص‌دار، دارای نظم است و به آن مجموعه توالی (Sequence set) می‌گویند.

نحوه پیاده‌سازی فایل شاخص با استفاده از B – Tree

هر بلاک شاخص (یا تعدادی بلاک با نامی دیگر) گرهی از درخت در نظر گرفته می‌شود. یک گره، در لود اولیه، به تمامی پر نمی‌شود، به عبارت دیگر چگالی آن در لود اولیه کمتر از صددرصد است. ساختمان درونی یک گره (با فرض این‌که هر گروه یک بلاک باشد) به صورتی است که در شکل زیر دیده می‌شود.



مثالی ساده از یک فایل داده‌ای و درخت شاخص آن



ساختمان درونی یک گره

اگر طول هر مدخل $V + P$ بایت باشد، y اسمی برابر است با: $y = \left\lfloor \frac{B}{V + P} \right\rfloor$

این ظرفیت نشانه‌روی؛ در لود اولیه به تمامی پر نمی‌شود، بلکه تعدادی مدخل به عنوان رزرو منظور می‌گردد. ظرفیت واقعی نشانه روی (Effective fanout) در لود اولیه که آن را با y_{eff} نشان می‌دهیم حداکثر برابر با y و حداقل $\frac{y}{2}$ است:

$$\frac{y}{2} \le y_{eff} \le y$$

یعنی در لود اولیه، اقلاً نیمی از مدخل‌های یک گره درخت پر است. معمولاً $y_{eff} = 0.69\ y$ در نظر گرفته می‌شود.

در درج

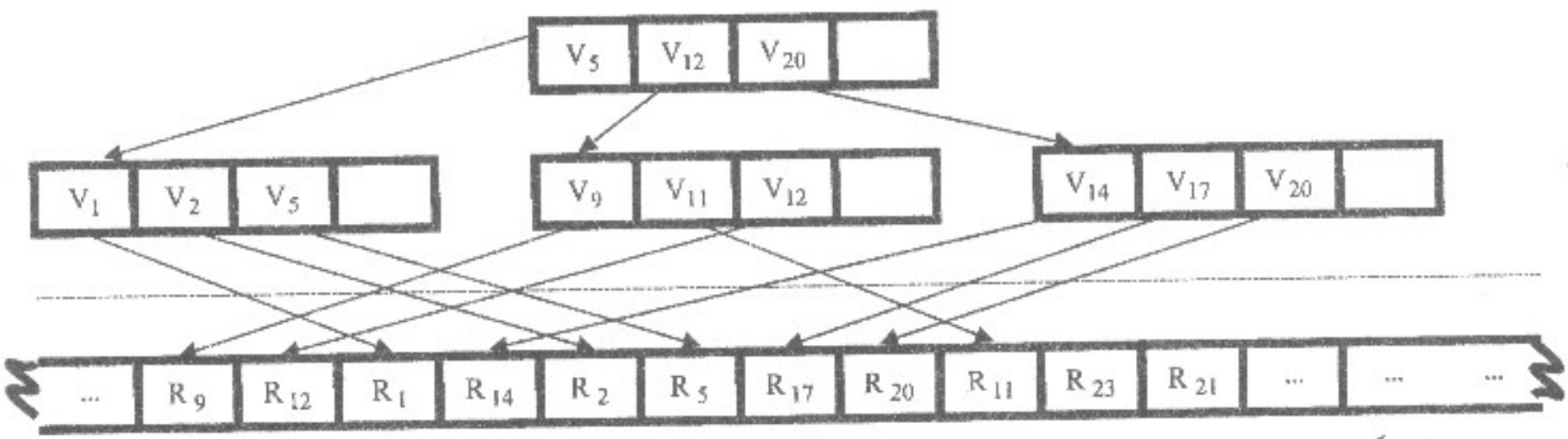
برای درج رکورد آن را در فایل داده‌ای (که فرض کردیم پایل است) اضافه می‌کنیم. پس اردرج، باید ارتباط ساختاری رکورد بلافاصله برقرار شود، یعنی مدخل شاخص مربوطه در سطح اول شاخص ایجاد گردد. این مدخل طبعاً باید در بلاک شاخص مربوطه (باتوجه به مقدار صفت خاصه شاخص و محل منطقی آن در مجموعه توالی) به وجود آید.

چون تعدادی مدخل رزرو در هر بلاک شاخص داریم، لذا ایجاد مدخل شاخص ناظر به رکورد درج شده به آسانی انجام می‌پذیرد (آدرس مکان درج رکورد در فیلد نشانه‌رو گذاشته می‌شود). تا زمانی که مدخل خالی در بلاک شاخص موجود باشد، تنظیم درخت شاخص با ایجاد مدخل جدید، پایان می‌پذیرد و عمل دیگری لازم نیست. به عبارت دیگر، تغییر عمده‌ای در درخت پدید نمی‌آید. اما اگر برای ایجاد مدخل در بلاک شاخص مربوطه در سطح اول دیگر جا نباشد یعنی y_{eff} برابر با y شده باشد، در این صورت بلاک شاخص پر شده را باید تقسیم (Splitting) (شکسته) کرد.

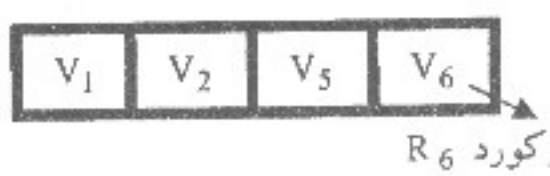
نحوه عمل چنین است:

یک بلاک شاخص خالی، موسوم به بلاک همراه (Partner) به فایل شاخص اختصاص داده می‌شود و نیمی از مدخل‌های بلاک پر شده، با رعایت ترتیب، به این بلاک انتقال می‌یابند. این بلاک همیشه در سمت راست بلاک پر شده ایجاد می‌شود و بدین‌سان گرهی جدید، در همان سطح گره پر شده، دردرخت پدید می‌آید. این گره جدید طبعاً باید با گرهی در سطح بلافاصله بالاتر مرتبط شود، به عبارت دیگر در گرهی از سطح بالاتر باید مدخلی جدید ایجاد شود تا به بلاک همراه نشانه رود. پس در عمل تقسیم بلاک پر شده اقلاً سه بلاک شاخص باید ایجاد و یا به هنگام درآیند. اگر بلاک سطح بالاتر هم پر باشد، خود نیاز به تقسیم شدن دارد و بلاکی همراه باید برای آن ایجاد شود. بدین‌ترتیب عمل تقسیم بلاک‌های شاخص، در یک شاخه از درخت، ممکن است حتی تا ریشه منتشر شود و در این صورت، ژرفای درخت از x به $x + 1$ تغییر کند. در شکل زیر مثالی می‌بینیم که نحوه عمل را نشان می‌دهد.

مثال ۱:



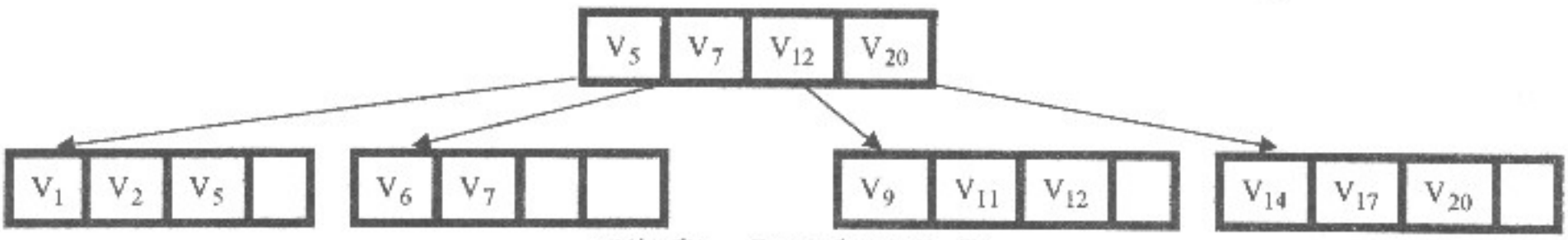
در این مثال، یک درخت شاخص، درست پس از لود اولیه با $Y_{eff} = 0.75$ y نشان داده شده است. پس از درج R6 خواهیم داشت:



حال اگر رکورد R 7 را درج کنیم، بلاک دیگر جا ندارد، لذا باید، به صورت زیر، تقسیم شود:

وضع جدید بلاک پر شده — بلاک همراه

باید مدخلی در بلاک سطح بالاتر ایجاد شود مطابق شکل زیر:



مثالی از درج رکورد و تقسیم بلاک شاخص

موارد استفاده (مهم)

اساساً در محیط‌هایی به کار می‌رود *که واکنش سریع تک رکوردها مورد نظر بوده*، داده‌ها مرتباً دست‌خوش تغییرات شوند، به عبارت دیگر فایل حالت نامانا داشته باشد و به علاوه کاربر بخواهد از طریق صفات خاصه مختلف (ساده یا مرکب)، به رکوردها دستیابی داشته باشد. به عنوان مثال در *سیستم رزرواسیون ها در خطوط هوایی* که در آن اطلاعات مربوط به جا مرتباً تغییر می‌کند، استفاده از چنین ساختاری مطلوب خواهد بود.

بهبود کارایی شاخص‌ها

۱) کاهش هزینه دستیابی (Access Cost) به شاخص

۲) کاهش تداخل (Interference) در B – Treeها

۳) بهبود زمان پاسخ‌دهی در بهنگام‌سازی B – Tree

کاهش هزینه دستیابی به شاخص

برای این منظور، دو تکنیک وجود دارد:

۱) نگاهداری سرشاخص در حافظه اصلی

۲) **اعمال ملاحظات لوکالیتی:** جای‌دهی بلاک های شاخص روی رسانه به نحوی که زمان استوانه‌جویی کاهش یابد.

نگاهداری سرشاخص در حافظه اصلی: که پس از باز شدن فایل صورت می‌گیرد، سبب می‌شود تا ژرفای ساختار شاخص یک سطح کاهش یابد، و روشن است که از این رهگذر هزینه دستیابی کاهش می‌یابد.

اعمال ملاحظات لوکالیتی: ایجاد شاخص سخت‌افزاری

همان‌طور که در ساختار ترتیبی شاخص‌دار گفتیم، در این روش ساختار شاخص با توجه به تقسیمات سخت‌افزاری رسانه، به بخش‌هایی تقسیم و به نحوی روی رسانه جای داده می‌شود که باعث کاهش زمان استوانه‌جویی می‌گردد.

روش‌های تسریع در جستجوی مدخل شاخص

روش‌های جستجو مدخل شاخص عبارتند از:

۱) جستجوی خطی

۲) جستجوی دودویی

۳) جستجوی با پرش

۴) ایجاد درخت جستجوی دودویی

۵) ایجاد B - Tree و پیمایش خطی گره‌ها

۶) ایجاد B - Tree و جستجوی با پرش در هر گره

۷) ایجاد B - Tree و جستجوی دودویی در گره‌ها

۸) ایجاد درخت نامتعادل (Unbalanced Tree)

۹) استفاده از بافرینگ چندتایی

۱۰) استفاده از تابع در هم‌ساز (Hashing Function)

فایل مستقیم (فایل در هم) (Hashed)

■ شیوه‌های دستیابی

به طور کلی شیوه‌های دستیابی به اطلاعات یک فایل به دو دسته تقسیم می‌شوند:

۱ـ ترتیبی

۲ـ تصادفی

دستیابی تصادفی: منظور از دستیابی تصادفی این است که رکورد مورد نظر با یک بار و یا با تعداد کمی دستیابی تصادفی به دست می‌آید.

برای دستیابی تصادفی دو روش وجود دارد:

۱ـ **شاخص بندی:** مقدار کلید رکورد همراه آدرس آن در مدخل شاخص ذخیره می‌شود.

V	P

۲ـ **تبدیل کلید به آدرس (KAT = Key To Address Transformation):** آدرس هر رکورد با پردازشی روی مقدار کلید آن رکورد بدست می‌آید.

فایل مستقیم

ساختار فایل مستقیم شکل بهبود یافته ساختارهای قبلی نیست، بلکه ساختار جدائی است که درج و واکشی رکوردها در آن با استراتژی خاصی انجام می‌شود.

ایجاد فایل در لود اولیه

یکی از صفات خاصه را به عنوان کلید در نظر گرفته و مقادیر این کلید را به سیستم فایل داده (تابعی به نام تابع hash) تا پردازشی روی آن انجام دهد، حاصل پردازش آدرس تصادفی است که رکورد باید در آن آدرس قرار گیرد. این آدرس به آدرس طبیعی یا حفره طبیعی رکورد (Natrual Address)، معروف است.

دسترسی به رکوردهای ذخیره شده

در واکشی رکورد ذخیره شده، کلید رکورد دوباره به سیستم فایل (تابع hash) داده می‌شود و سیستم فایل (تابع hash) همان پردازش هنگام ذخیره‌سازی را انجام داده و آدرس نشست رکورد را پیدا می‌کند و آن را واکشی می‌کند. به این استراتژی دستیابی، استراتژی مستقیم می‌گویند.

فضای آدرس فایل مستقیم

فضای آدرسی یک فایل مستقیم را با m آدرس از 1 تا m یا از 0 تا m − 1 در نظر می‌گیرند که هر آدرس مربوط به یک حفره و هر حفره مکان ذخیره‌سازی یک رکورد است.

در این فضای m آدرسی مربوط به فایل مستقیم، باید n رکورد در لود اولیه درج شوند و البته باید $m \geq n$ باشد، تا به اندازهٔ کافی فضای آدرس برای n رکورد وجود داشته باشد.

نکته: نسبت $\frac{n}{m}$ فاکتور لود نامیده می‌شود که $\frac{n}{m} \leq 1$ است.منظور از فاکتور لود درصد حفره‌هائی از فایل است که در لود اولیه پر می‌شود.

نتیجه‌گیری:

در فایل مستقیم برای ذخیره‌سازی و دسترسی به اطلاعات، یک تبدیل کلید به آدرس انجام می‌شود که در بار اول تبدیل برای ذخیره‌سازی رکورد یک آدرس تصادفی را بدست آورده و آن را ذخیره می‌کند از بار دوم به بعد برای دسترسی به رکورد مورد نظر همان تبدیل انجام شده و دوباره همان آدرس را بدست آورده و دسترسی به آن رکورد به صورت مستقیم انجام می‌شود.

تعداد حالات قرار گرفتن n رکورد در فضای m آدرسی

تعداد حالات جای دادن n رکورد در یک فضای m آدرسی m^n حالت است، که در آن هر کدام از n رکورد m انتخاب برای قرار گرفتن دارند یعنی:

$$\underbrace{m \times m \times \ldots \times m}_{n} = m^n$$

اما در هر حفره فقط می‌توان یک رکورد قرار داد در نتیجه تعداد حالات یک به یک برای قرار دادن n رکورد در m حفره:

$$\overbrace{m}^{\text{انتخاب رکورد اول}} \times \underbrace{(m-1)}_{\text{انتخاب رکورد دوم}} \times \ldots \times \underbrace{m-(n-1)}_{\text{انتخاب رکورد nام}} = \boxed{\frac{m!}{(m-n)!}}$$

دیده می‌شود که زمانیکه رکورد اول m حفره را برای انتخاب دارد، رکورد دوم $m-1$ حالت و ... و رکورد nام، $m-(n-1)$ حالت برای انتخاب دارد تا برای هر رکورد فقط و فقط یک حفره باشد اما چون نشست رکوردها به صورت تصادفی است امکان این‌که برای دو رکورد متفاوت آدرس یکسانی بدست بیاید وجود دارد.

نکته مهم: از آنجائیکه تابع مبدل برای یک کلید رکورد یک آدرس تصادفی در فضای m آدرسی پیدا می‌کند امکان دارد برای دو کلید رکورد متفاوت آدرس یکسانی بدست بیاید در این حالت می‌گوئیم پدیده تصادم (collision) اتفاق افتاده است به عبارت بهتر:

$$\text{تصادم یا collision} \Rightarrow \underbrace{A_i = A_j}_{\text{آدرس یکسان}} \quad , \quad \underbrace{k_i \neq k_j}_{\text{برای دو کلید رکورد متفاوت}}$$

در وضعیت بروز تصادم سیستم فایل باید رکوردی که برای آن تصادم اتفاق افتاده به نحوی در فایل ذخیره کند که در ادامه روش‌های آن بررسی خواهد شد.

■ **ویژگی‌های ساختار مستقیم: (مهم)**

۱ـ با توجه به نشست تصادفی اطلاعات ساختار فایل اساساً بی‌نظم است.

۲ـ با توجه به ساختار بی‌نظم امکان پردازش سریال وجود ندارد.

۳ـ رکوردها با طول ثابت در نظر گرفته می‌شود. (می‌توان رکوردهایی با طول متغیر نیز ایجاد کرد)

۴ـ چون براساس یک صفت خاصه کلید در نظر گرفته شده و از روی آن آدرس بدست می‌آید، فایل دارای عدم تقارن است.

۵ـ اگر نشاوند جستجو صفت خاصه‌ای، به جز صفت خاصه‌ای باشد که براساس آن آدرس تولید شده، امکان دسترسی مستقیم دیگر وجود ندارد. و فایل مانند فایل پایل خواهد بود. چون فایل براساس تنها یک صفت خاصه آدرس‌دهی شده و فقط براساس آن صفت خاصه امکان دسترسی مستقیم به رکوردها وجود دارد.

۶ـ به علت امکان وجود تصادم هنگام نشست رکوردها امکان بروز حافظه هرز در جای جای حافظه وجود دارد، بنابراین نشست یکنواخت رکوردها از اهمیت به سزائی برخوردار است.

۷ـ هر چه رکوردهای تصادفی کمتر باشد، واکشی رکوردها سریعتر انجام می‌شود.

۸ـ رکوردهای غیرتصادفی با یک بار دستیابی واکشی می‌شود اما رکوردهای تصادفی با حداقل یک بار دستیابی واکشی می‌شوند. چون ممکن است در محل تصادف نباشند و مجبور به جستجو در محل‌های دیگر شویم.

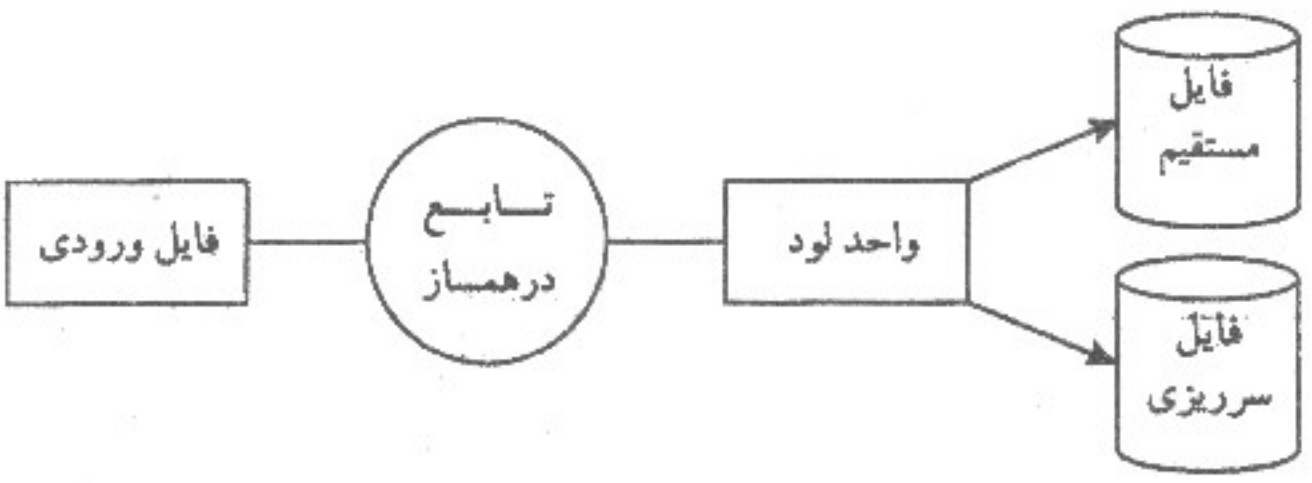
لود کردن فایل

در اساس دو روش برای لود کردن فایل وجود دارد:

۱) لود مستقیم

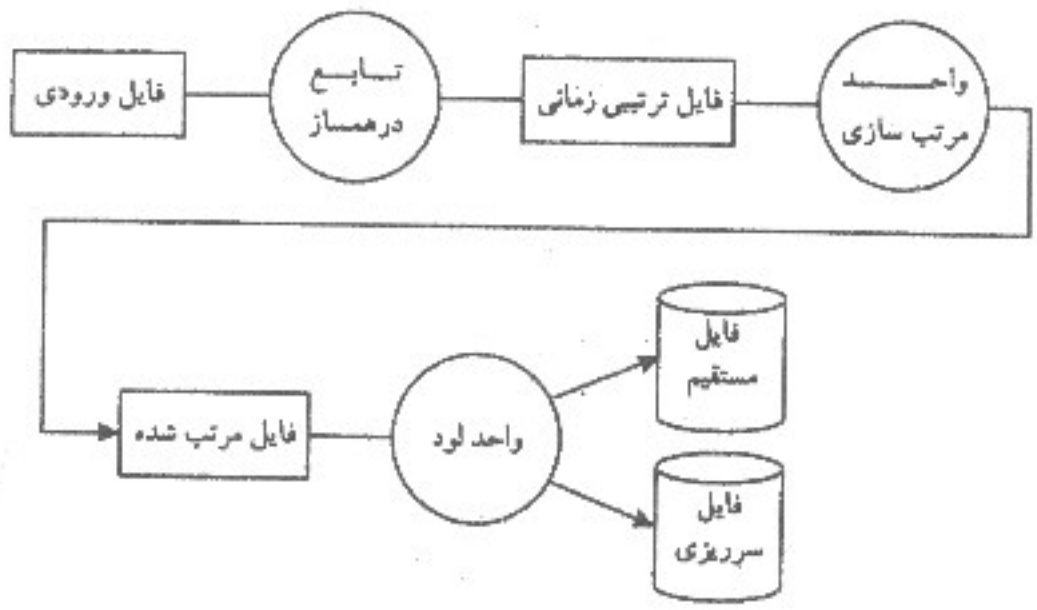
۲) لود ترتیبی

در روش لود مستقیم، هر رکورد از یک فایل ورودی خوانده می‌شود. تابع مبدل روی مقدار کلید اصلی رکورد اعمال می‌شود. اگر آدرس طبیعی رکورد خالی باشد رکورد در آن آدرس درج می‌شود. در غیر این صورت، رکورد در یک فایل دیگر که نظم زمانی دارد و گاه به آن فایل سرریزی می‌گویند، نوشته می‌شود. این عملیات برای تمام رکوردهای فایل ورودی انجام می‌شود. در مرحله بعدی، رکوردهای موجود در فایل سرریزی براساس یکی از تکنیک‌های درج رکوردهای تصادفی (سرریزی‌های) در فایل مستقیم درج می‌شوند.(شکل ۱)



شکل ۱: لود مستقیم

در روش لود ترتیبی، هر یک از رکوردهای فایل ورودی خوانده شده تابع مبدل روی کلید اصلی آن اعمال می‌شود. آدرس به دست آمده برای رکورد در فیلد جدیدی از رکورد وارد شده و رکورد در یک فایل ترتیبی زمانی نوشته می‌شود. سپس تمام رکوردهای این فایل براساس مقادیر صعودی مقدار فیلد آدرس آن‌ها مرتب می‌شوند و این رکوردهای مرتب شده، در یک فایل دیگر نوشته می‌شوند (فایل مستقیم). رکوردهای غیرسرریزی در همان آدرس طبیعی خود در این فایل درج می‌شوند و رکوردهای سرریزی در یک فایل سرریزی وارد می‌شوند. در مرحله آخر این رکوردهای سرریزی براساس یکی از تکنیک‌های سرریزی در فایل مستقیم لود می‌شوند. (شکل ۲)



شکل ۲: لود ترتیبی

نکته مهم:

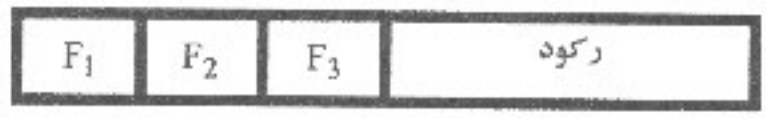
برای استفاده از روش‌های کاراتر حل مشکل تصادف، برای هر رکورد باید فیلد نشانه‌رو در نظر گرفت که در ابتدا با NULL مقدارگذاری می‌شود.

هر حفره باید دارای یک بخش پیشوندی باشد. این بخش فیلدهایی دارد از جمله:

۱) فیلد حاوی کد وضع (State Code) حفره (خالی یا پر): F_1

۲) فیلد حاوی وضع رکورد در حفره (فعال یا غیرفعال: "D"دار) : F_2

۳) فیلد حاوی این اطلاع که رکورد سر زنجیره است (در آدرس طبیعی خود است) یا نه و اگر نه، چندمین رکورد زنجیره است: F_3



حفره

توابع مبدل

به توابعی گفته می‌شود که به هنگام ایجاد فایل مستقیم برای تبدیل کلید به آدرس استفاده می‌شوند.

انواع توابع مبدل:

۱ـ توابع دارای قطعیت

۲ـ توابع احتمالاتی

۱ـ توابع دارای قطعیت

این توابع به گونه‌ای هستند که به ازاء هر مقدار کلید رکورد، آدرس منحصر به فردی تولید می‌کنند که با آدرس هیچ کلیدی یکسان نخواهد بود به عبارت بهتر:

آدرس متفاوت وجود دارد. $k_i \neq k_j \longrightarrow a_i \neq a_j$ به ازاء کلیدهای متفاوت

این توابع بیشتر جنبه تئوری دارند، چون در زمانیکه تعداد رکوردهای فایل (n) نسبت به فضای آدرسی m زیاد شود ناخودآگاه امکان بروز تصادم ($a_i = a_j$, $k_i \neq k_j$) پیش می‌آید.

۲ـ توابع احتمالاتی

در این توابع برعکس توابع دارای قطعیت امکان بروز تصادم وجود داشته و این توابع خود بر ۲ دسته هستند:

توابع احتمالاتی
۱) توابع احتمالاتی قائل به نظم
۲) توابع احتمالاتی در هم‌ساز (Hashing – Randomize – Scrambling)

۱ـ توابع قائل به نظم

به توابع احتمالاتی که در آن اگر $k_i < k_j$ (مقدار کلید i از مقدار کلید j کوچکتر باشد) آن‌گاه در صورت عدم بروز تصادف $a_i < a_j$ (مقدار آدرس i از مقدار آدرس j کوچکتر باشد) باشد، توابع قائل به نظم می‌گویند به عبارت بهتر:

تابع مبدل یا تولید آدرس قائل به نظم است. $\longrightarrow$ $a_i < a_j$, $k_i < k_j$

نکته: این توابع به ازاء nهای کوچک (تعداد رکوردهای کم) به کار برده می‌شود و برای تعداد رکورده‌ای زیاد کاربرد ندارد.

۲ـ توابع درهم‌ساز (Hashing)

در این توابع آدرس مربوط به کلید رکوردها به صورت کاملاً تصادفی در فضای آدرسی بدست آمده و در نتیجه رکوردها به صورت نامنظم در فضای آدرسی قرار می‌گیرند، چون این توابع در سیستم‌های موجود، برای ایجاد فایل مستقیم استفاده می‌شوند، برای همین به فایل مستقیم، فایل درهم نیز گفته می‌شود.

توابع درهم‌ساز و توابع مولد اعداد تصادفی

مسئله یافتن تابع در هم‌ساز با مسئله یافتن تابع مولد اعداد تصادفی ارتباط نزدیکی دارد با این تفاوت که تابع درهم‌ساز باید قابلیت تکرار شدنی داشته باشد. این به آن مفهوم است در صورتیکه تابع در هم‌ساز بار اول برای ذخیره سازی رکورد یک آدرس تصادفی با توجه به کلید رکورد مشخص کند، در بارهای بعدی نیز برای کلید رکورد مورد نظر باز هم باید همان آدرس را تولید کند تا رکورد مورد دستیابی مستقیم واقع شود.

نکته: برای هر کلید رکورد دلخواه همیشه آدرس تصادفی تولید می‌شود که هیچ ارتباطی با آدرس رکوردهای دیگر ندارد. در چنین وضعیتی اگر آدرس دو کلید متفاوت یکسان باشد با پدیده تصادف روبرو می‌شویم.

انواع توابع درهم‌ساز

توابع مختلفی برای ایجاد فایل مستقیم وجود دارد که بروز تصادف، در همه آن‌ها، کم و بیش محتمل است. این توابع عبارتند از:

الف: انتخاب ارقام میانی مربع کلید (Mid Square)

ب: تقسیم کردن (Division)

ج: شیفت دادن (Shifting)

د: تازدن (Folding)

هـ: تحلیل ارقام (Digit analysis)

و: تبدیل مبنا (Radix Conversion)

ز: تقسیم چند جمله‌ای (Polynomial division)

ح: روش X – OR

ط: روش انتخاب ارقام کلید، در گونه‌های مختلف از جمله:

- انتخاب ارقام توان پایین کلید
- انتخاب ارقام توان بالای کلید
- انتخاب ارقام میانی کلید
- انتخاب ارقام کلید به طور تصادفی

ی: خرد کردن و جمع کردن (Chopping and adding)

ضوابط انتخاب تابع در هم‌ساز

برای انتخاب تابع بهتر، ضوابطی وجود دارد. ضوابط انتخاب تابع بهتر عبارتند از:

۱ـ تابع باید چنان باشد که بتوان آن را روی تمام اجزاء کلید (تمام ارقام کلید) اعمال کرد.

۲ـ توزیع یکنواخت‌تر رکوردها

۳ـ کمتر بودن تعداد تصادفی‌ها و در نتیجه کمتر بودن متوسط تعداد عملیات ورودی / خروجی لازم برای واکشی یک رکورد دلخواه.

مشکل تصادف

شرایط بروز پدیده تصادف

اگر شرایط زیر وجود داشته باشند، در این صورت تصادف بروز کرده است:

۱ـ $k_i \neq k_j$ ، $a_i = a_j$ (به ازاء دو کلید متفاوت k_i , k_j، آدرس‌های آن‌ها a_i , a_j یکسان باشد)

۲ـ محتوای a_i رکوردی فعال باشد (یعنی نشانگر حذف شده نداشته باشد).

سیستم فایل پس از تشخیص پدیده تصادف، باید روال متصدی تصادف (Collision handling routine) را فرابخواند.

باکت‌بندی

در صورتیکه در فایل مستقیم به جای آدرس حفره با استفاده از باکت‌بندی کردن حفره‌ها از آدرس باکت استفاده کنیم، آدرس‌دهی می‌تواند برای باکت‌ها از 0 تا M – 1 برای M باکت باشد.

واضح است که آدرس باکت باید به آدرس بلاک روی دیسک تبدیل شود.

■ Bk_f **فاکتور باکت‌بندی:** (Bucketing factor)

تعداد حفره‌های موجود در هر باکت را با Bk_f یا فاکتور باکت‌بندی معرفی می‌کنند.

$$Bk_f = \frac{\text{تعداد حفره‌ها}}{\text{تعداد باکت‌ها}} = \frac{m}{M}$$

■ **فاکتور لود:**

فاکتور لود به صورت $\frac{n}{m}$ را با توجه به Bk_f، می‌توان به صورت زیر نشان داد:

$$\text{فاکتور لود} = \frac{n}{m} = \boxed{\frac{n}{Bk_f \times M}}$$

مزایای باکت بندی

۱) تسهیل و آسان شدن در حل مشکل تصادف

۲) کوتاه‌تر شدن طول آدرس‌ها

۳) امکان ایجاد فایل مستقیم با رکوردهایی با طول متغیر

۱) تسهیل و آسان شدن حل مشکل تصادف:(مهم)

تا زمانیکه یک باکت حاوی چند حفره پر نشده باشد، رکوردهای تصادفی آن باکت را می‌توان در آن باکت جای داد، در این صورت نیازی به استفاده از روال‌های حل مشکل تصادف نداریم.

در لود اولیه $\frac{n}{m} < 1$ فایل پر است. در نتیجه $\left(1 - \frac{n}{m}\right)$ فایل خالی است. در باکت‌بندی $\left(1 - \frac{n}{m}\right) Bk_f$ حفره در یک باکت خالی خواهد بود

در نتیجه هر چه Bk_f بزرگتر باشد تعداد حفره‌های باکت بزرگتر شده در نتیجه درصد سرریزی‌ها کاهش پیدا می‌کند.

نکته مهم: هرچه اندازه رکوردها کوچکتر باشد، Bk_f بزرگتر شده و در نتیجه کارائی فایل مستقیم بیشتر می‌شود و تصادفی‌ها کمتر شده.

■ **روش‌های کاهش تصادفی‌ها: (مهم)**

۱) انتخاب بهترین تابع در هم‌ساز (hash)

۲) افزایش Bk_f

۳) رشد خطی فضای آدرس (Linear Growth)

۲) کوتاهتر شدن طول آدرس (تعداد بیت‌های لازم برای ساختن آدرس)

برای ساختن m آدرس حداقل به $\log_2 m$ بیت و برای ساختن M آدرس حداقل به $\log_2 M$ بیت نیاز است:

$$\log_2 M = \log_2 \frac{m}{Bk_f} = \log_2 m - \log_2 Bk_f$$

در محاسبه بالا با استفاده از باکت‌بندی با M باکت بیت‌های مورد نیاز نسبت به m حفره کاهش پیدا می‌کند و باعث صرفه‌جوئی در حافظه می‌شود.

راه‌حل‌های مشکل تصادف

شش راه‌حل، برای مشکل تصادف و درج سرریزی‌ها وجود دارد:

۱ـ ایجاد یک فایل جداگانه و درج تصادفی‌ها در این فایل

۲ـ در نظر گرفتن ناحیه‌ای جداگانه در خود فایل و درج تصادفی‌ها در این ناحیه

۳ـ کاوش خطی (Linear probing) و درج تصادفی در اولین باکت جادار

۴ـ احتمالاتی کردن مجدد ((Rerandomizing (Rehashing) (باز در همسازی)

۵ـ ایجاد زنجیره بدون جایگزینی (Replacement)

۶ـ ایجاد زنجیره با جایگزینی

راه حل اول: ایجاد یک فایل جداگانه

در این راه حل: که ساده‌ترین است، اساساً فایلی جداگانه در نظر گرفته می‌شود و رکوردهای تصادفی در آن وارد می‌شوند.

معایب:

۱) فایل اصلی و فایل تصادفی‌ها دو فایل جدا هستند و سیستم باید دو فایل را پردازش کند.

۲) در فایل اصلی حفره‌های هرز پدید می‌آید، به ویژه اگر فایل اصلی بزرگ باشد، تعداد حفره‌های هرز (بلااستفاده) به طور قابل ملاحظه‌ای فزونی خواهد گرفت.

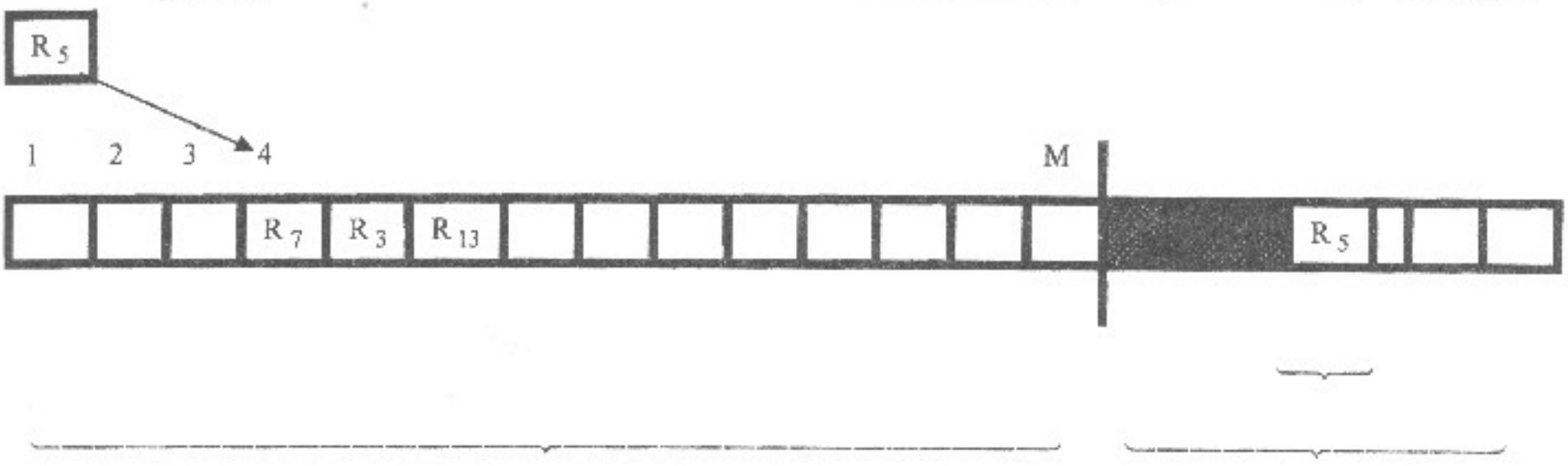
نکته: این راه حل چندان در عمل پیاده‌سازی نمی‌شود.

راه حل دوم: در نظر گرفتن ناحیه‌ای جداگانه در خود فایل

در این راه حل، ناحیه‌ای جداگانه داخل فایل برای درج تصادفی‌ها در نظر گرفته می‌شود. رکورد تصادفی در اولین باکت جادار در این ناحیه درج می‌گردد و زنجیره تصادفی‌های یک آدرس، ایجاد می‌شود. می‌توان ناحیه جداگانه را، در انتهای هر استوانه در نظر گرفت، تا زمان مکان‌جویی کاهش یابد.

معایب:

در این روش ممکن است باکت‌هایی از ناحیهٔ جداگانه، پر شونددر حالی که باکت‌هایی از ناحیه اصلی، دارای حفره‌های خالی باشد.



درج تصادفی‌ها در ناحیه جداگانه سرریزی

مزایا:

در این راه حل، زنجیره‌هایی از ناحیه اصلی به ناحیه جداگانه و به طول‌های متفاوت، داریم که واکشی تصادفی‌ها را تسریع می‌کنند.

راه حل سوم: کاوش خطی و درج در اولین باکت جادار

در این روش، جایابی با کاوش خطی انجام می‌شود و رکورد تصادفی در اولین باکت جادار، با شروع ازنقطه تصادف به طرف انتهای فایل، در واقع در نزدیکترین باکت جادار، به نقطه تصادف درج می‌شود. این باکت جادار ممکن است نزدیک انتهای فایل باشد و حتی امکان دارد باکت جادارای به طرف انتهای فایل یافت نشود و در این صورت جستجو برای یافتن باکت جادار، می‌تواند به طور چرخشی از ابتدای فایل شروع شود.

معایب:

۱) به تدریج که فاکتور لود به یک نزدیک می‌شود، فایل در بعضی از باکت‌ها دچار انبوهی درج می‌شود و جستجوی خطی برای واکشی رکورد مورد نظر (که تصادفی باشد) به درازا می‌انجامد.

۲) مشکل دیگری که این روش دارد این است که رکوردها را نمی‌توان به آسانی حذف کرد. زیرا در صورت حذف یک رکورد، حفره‌ای خالی می‌شود و چون کاوش خطی برای یافتن یک رکورد مورد نظر، به محض برخورد با یک حفره خالی پایان می‌پذیرد، سیستم قادر به یافتن رکورد مورد نظر نخواهد بود.

نکته: برای اجتناب از بروز چنین وضعیتی، سیستم فایل باید رکورد حذف شونده را با ضبط نشانگر «حذف شده» حذف کند ولی حفره آن را آزاد اعلام ننماید، تا زمانی که رکوردی جدید در این حفره درج شود. در این صورت جستجوی خطی به دنبال رکورد دچار گسست نخواهد شد و سیستم رکورد را واکشی خواهد کرد. راه حل دیگر این است که تمام فایل جستجو شود تا بالاخره رکورد مورد نظر واکشی گردد که البته کار بسیار زمان‌گیری خواهد بود.

نکته: این روش درج تصادفی‌ها، ازآن‌جا که هر باکت پر شده، از حفره‌های خالی نزدیکترین باکت استفاده می‌کند، اصطلاحاً، به روش «همسایگی بد» (Bad neighborhood) موسوم است.

فرض کنید رکوردها با آدرس‌های زیر توسط تابع hash تولید شده‌اند و قرار است، آن‌ها را در 8 خانه قرار دهیم.

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
hash :	3	4	7	4	2	7	8	4

در روش وارسی خطی می‌توانیم به ترتیب درج رکوردها از R_1 به صورت زیر عمل کنیم.

R_7	R_5	R_1	R_2	R_4	R_8	R_3	R_6
1	2	3	4	5	6	7	8

راه حل چهارم: احتمالاتی کردن مجدد

در این روش کلید رکورد تصادفی به تابعی دیگر داده می‌شود. اگر اولین تابع به صورت m_1 mod (مقدار کلید) = A باشد (m_1 نزدیکترین عدد اول به m و A آدرس حاصله از اعمال تابع است)، تابع مشابه دیگری به صورت $m_2 + 1$ mod (مقدار کلید) = D روی مقدار کلید

اعمال می‌شود. m_2 عدد اول دیگری است بلافاصله کوچکتر از m_1. سپس مقدار D به عنوان جابجایی (Displacement) برای آدرس A به کار می‌رود، و آدرس جدید A + D خواهد بود.

نکته: روش‌های سوم و چهارم اصطلاحاً به نشانی‌دهی باز (Open addressing) موسومند.

راه حل پنجم: ایجاد زنجیره بدون جایگزینی

رکوردهای غیر دخیل:

در روش سوم، یعنی جایابی با کاوش خطی، هنگامی که فاکتور لود افزایش می‌یابد، زمان جستجوی رکورد تصادفی بالا می‌رود، زیرا رکوردهایی مورد وارسی قرار می‌گیرند که با رکورد مورد نظر تصادف ندارند و اساساً ارتباطی هم بین این رکوردها و رکورد مورد جستجو وجود ندارد. ما این رکوردها را رکوردهای غیردخیل در بازیابی رکورد مورد نظر می‌نامیم.

برای اجتناب از وارسی رکوردهای غیردخیل، روش ایجاد زنجیره بدون جایگزینی طراحی شده است. در این روش با زنجیر کردن رکوردهای تصادفی در یک آدرس، واکشی آن‌ها تسریع می‌شود. در بازیابی یک رکورد تصادفی، کافی است زنجیره تصادفی‌های آدرس مربوطه پیمایش شود و بدین‌ترتیب از رکوردهای غیردخیل پرش می‌شود. این‌که رکورد تصادفی در کجا جای داده می‌شود، خود مسئله‌ای است جدا از این روش، در واقع محل درج رکورد تصادفی را می‌توان با روش سوم (و حتی دوم) مشخص کرد. فرض کنیم که روش سوم را برای جایابی رکورد تصادفی به کار برده باشیم. در روش ایجاد زنجیره بدون جایگزینی، امکان دارد رکوردهای تصادف کننده روی آدرس‌های متفاوت در یک زنجیره جای گیرند. به عبارت دیگر، مثلاً دو زنجیره با مبداء متفاوت، یکی شوند و اصطلاحاً می‌گوییم با یکدیگر ائتلاف کنند. (Coalesce)

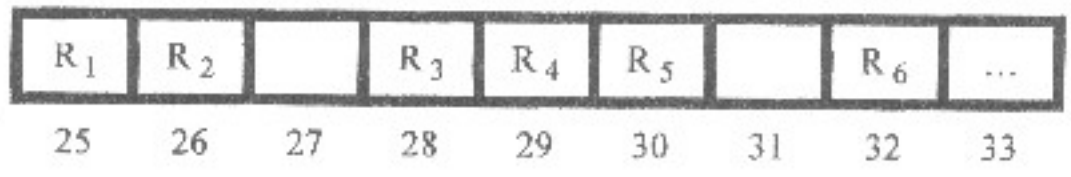
معایب:

پدیدهٔ ائتلاف سبب می‌شود تا، به تدریج که فاکتور لود بالا می‌رود، طول زنجیره‌ها طولانی‌تر شود و در این صورت فایل را باید سازماندهی مجدد کرد. وگرنه کارایی فایل در عملیات درج و حذف کاهش می‌یابد.

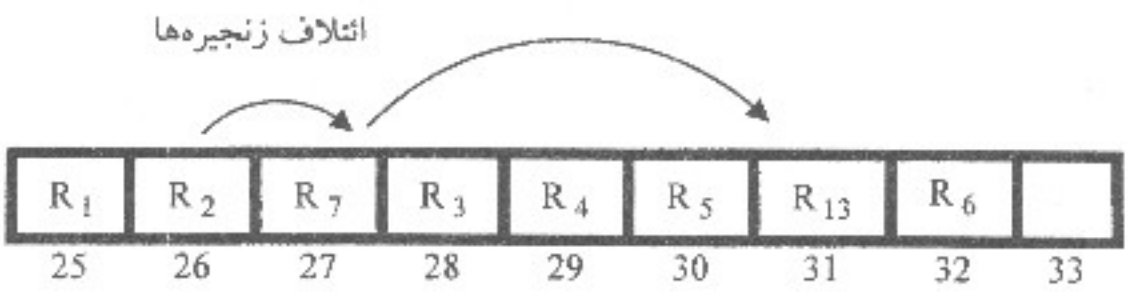
نکته: در این روش، جستجو برای یافتن یک رکورد زمانی پایان می‌پذیرد که به انتهای زنجیره رسیده باشیم. آخرین رکورد زنجیره نشانه‌رو NULL دارد. در ایجاد فایل باید توجه شود که تمام رکوردها دارای فیلد نشانه‌رو بوده، و در ابتدا با NULL مقدارگذاری شده باشند.

نکته: این روش، هر چند از روش‌های سوم و چهارم کاراتر است، اما در عمل حذف مشکلاتی دارد. نخست این‌که پس از حذف یک رکورد، باید دقت شود تا زنجیره گسسته نشود و برای این منظور باید فیلد نشانه‌روها متناسباً تنظیم شود (این مشکل، که چندان مهم نیست، در هر ساختار زنجیره‌ای وجود دارد و خاص این روش نیست).

مشکل دوم که حائز اهمیت است، وقتی بروز می‌کند که رکورد حذفی شدنی، خود سر زنجیره باشد. اگر چنین رکوردی حذف شود، سیستم در بازیابی یک رکورد، از زنجیره، ابتدا به آدرسی می‌رسد که خالی است و پیام «رکورد یافت نشد» خواهد داد.



در صورتیکه رکورد جدید R_7 بخواهد درج شود و آدرس آن 26 باشد به علت تصادف با R_2 در اولین محل خالی یعنی 27 درج می‌شود و از R_2 به R_7 نشانه می‌رود حال اگر رکورد جدید R_{13} با آدرس 27 ایجاد شود، چون حفره طبیعی آن توسط R_7 که با R_2 تصادف کرده بود پر شده باید در اولین حفره خالی بعدی یعنی 31 درج شود که این یعنی ائتلاف زنجیره تصادفی‌ها در این حالت از R_7 باید به R_{13} نشانه‌رو داشته باشیم.



راه حل ششم: ایجاد زنجیره با جایگزینی

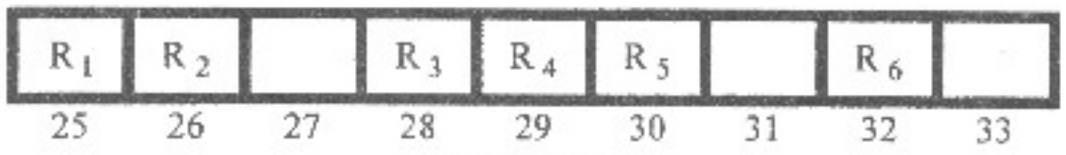
روش پنجم، هر چند از نظر کاهش زمان جستجو، نسبت به روش‌های سوم و چهارم، کاراتر است، اما دیدیم که در عمل حذف دشواری داشت. برای رفع این مشکل، روش ایجاد زنجیره‌ها با جایگزینی مطرح شده است. این روش علاوه بر رفع مشکل حذف، زمان جستجو را باز هم کوتاه‌تر می‌کند. در واقع این روش شکل کامل‌تر روش پنجم است و تفاوت اساسی آن با روش پنجم این است که موقع درج رکورد جدید، چنانچه آدرس طبیعی آن، اشغال باشد. رکورد موجود در حفره مربوط به رکورد درج شدنی از حفره برداشته می‌شود و رکورد جدید در حفره طبیعی خود جای می‌گیرد. این تغییر ساده در تکنیک درج رکوردها، مزایای قابل توجهی دربر دارد.

مزایا:

مزیت اول: این است که احتمال این‌که یک رکورد در حفره مربوط به خودش (در آدرس طبیعی‌اش) جای گیرد، افزایش می‌یابد. رکوردهای تصادفی در حفره‌هایی جای می‌گیرند که حفره طبیعی هیچ رکوردی نیست.

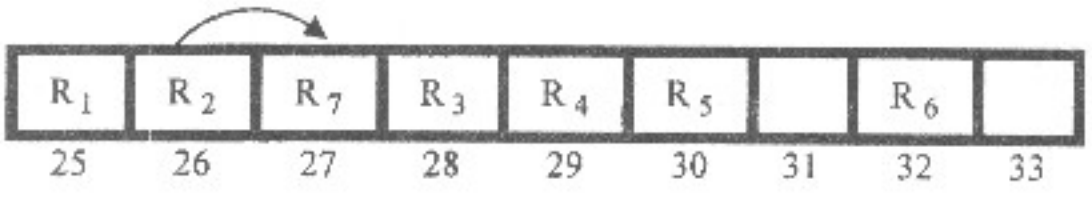
مزیت دوم: این‌که پدیده ائتلاف زنجیره‌ها بروز نمی‌کند، زیرا تمام تصادفی‌های یک رکورد، اعضاء زنجیره مستقلی هستند که سرزنجیره مشخص خود را دارد، و هیچ یک از این اعضاء سرزنجیرهٔ زنجیره دیگری نیست. نتیجه این‌که، طول زنجیره‌ها کوتاه‌تر می‌شود و از این رهگذر، زمان جستجوی یک رکورد مورد نظر کاهش می‌یابد.

مثال:

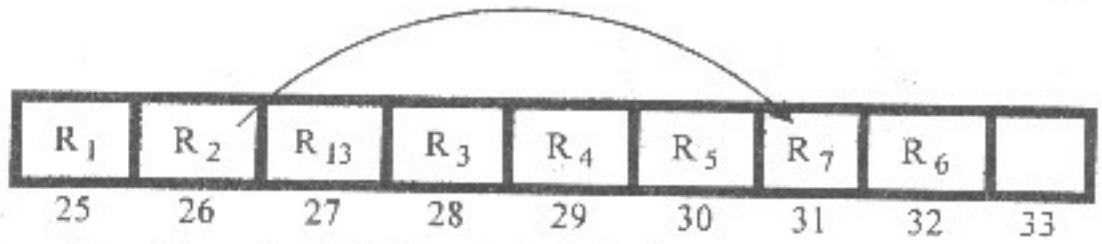


(فایل در لود اولیه)

در مرحله اول که R_7 با آدرس 26 می‌خواهد درج شود به علت تصادف با R_2 در 27 درج می‌شود. و از R_2 به آن اشاره می‌کنیم.



حال اگر بخواهیم R_{13} با آدرس 27 را درج کنیم چون حفره طبیعی R_{13} توسط یک رکورد تصادفی R_7 که با 26 تصادف کرده پر شده ابتدا R_7 را به اولین جای خالی دیگر برده 31 سپس R_{13} در جای خودش درج می‌شود و R_2 به جای جدید R_7 اشاره می‌کنیم و پدیدهٔ انتلاف به وجود نمی‌آید.



نکته: صرف‌نظر از دو روش اول و دوم که چندان رایج نیستند، راه‌حل‌های اصلی مشکل تصادف در فایل مستقیم عبارتند از:

- روش‌های نشانی‌دهی باز
  ۱ـ کاوش خطی
  ۲ـ احتمالاتی کردن مجدد
- ایجاد زنجیره بدون جایگزینی
- ایجاد زنجیره با جایگزینی

که کاملترین روش، ایجاد زنجیره با جایگزینی می‌باشد.

معایب فایل مستقیم مبنایی

۱ـ بروز حافظه هرز، جای جای در فایل و نایکنواخت شدن توزیع رکوردها در فضای آدرسی.

۲ـ عدم تقارن

۳ـ محدودیت ثابت بودن طول رکوردها

۴ـ پدیده تصادف

۵ ـ به علت درهم بودن فایل، بازیابی رکورد بعدی ناممکن است و در نتیجه امکان پردازش سریال رکوردها وجود ندارد.

موارد استفاده فایل

در محیط‌هایی به کار می‌رود که ماهیت پردازش، ترتیبی نباشد و دستیابی سریع به رکوردها مورد نظر باشد و رکوردها طول ثابت و کوچک داشته باشند و ترجیحاً نرخ عملیات درج، پایین باشد.

متوسط اندازه رکورد

ابتدا کل حافظه مصرفی برای فایل را محاسبه کرده، سپس R را به دست می‌آوریم.

$$S_{total} = (m + o)(aV + P)$$

$$\boxed{R = \frac{m + o}{n}(aV + P)}$$

واکشی رکورد T_F

این زمان بستگی دارد به این احتمال که رکورد مورد نظر تصادفی باشد یا نه و این‌که چندمین رکورد تصادفی در یک آدرس مشخص است و تکنیک درج تصادفی چیست.

عملیات لازم:

۱) اعمال تابع و یافتن آدرس (که زمان آن را با C_h نشان می‌دهیم)

۲) خواندن باکتی که در آدرسش به دست آمده است.

۳) بررسی محتوای باکت.

۴) اگر رکورد در باکت نباشد، رکورد تصادفی است و باید آن را در رکوردهای تصادفی جستجو کرد.

$$T_F = C_h + s + r + b_{tt} + K_F\left(s + r + b_{tt}\right)$$

که در آن K_F، تعداد دستیابی لازم برای یافتن رکورد بین رکوردهای تصادفی است.

بازیابی رکورد بعدی T_N

رکورد بعدی در این ساختار مفهومی ندارد، زیرا فایل درهم است و هیچ ارتباطی، برای تامین توالی منطقی رکوردها، بین آن‌ها موجود نیست. اگر کاربر خود نشاوند جستجو را بدهد، داریم:

$$T_N = T_F$$

درج رکورد T_I

عملیات لازم چنین است:

۱) اعمال تابع مبدل

۲) خواندن باکتی که آدرسش به دست آمده است.

۳) درج رکورد در باکت (در صورت وجود جا).

۴) اگر برای درج رکورد جا نبود، باید آن را براساس یکی از روش‌ها درج کرد.

$$T_I \approx K_I\left(s + r + b_{tt}\right) + K_I'' . T_{RW}$$

که در آن K_I تعداد بلاک‌هایی است که در جایابی برای رکورد تصادفی، باید خوانده شونده و مقدار آن، بستگی به تکنیک حل مشکل تصادف دارد و K_I'' تعداد دفعات بازنویسی است که حداقل یک است و در صورت ایجاد زنجیره می‌تواند بیش از یک هم باشد.

عمل بهنگام‌سازی T_U

در این ساختار، بهنگام‌سازی به صورت درجا انجام می‌شود و شرطش این است که کلید رکورد عوض نشود. عملیات لازم به شرح زیر است:

۱) واکشی رکورد

۲) کار در بافر و ایجاد نسخه جدید

۳) بازنویسی رکورد

$$T_{U_{\text{inplace}}} = T_F + T_{RW}$$

اگر کلید رکورد عوض شود، نسخه قدیم باید با نشانگر «حذف شده» بازنویسی شود و نسخه جدید، که اساساً رکوردی دیگر تلقی می‌گردد باید درج شود، در این صورت:

$$T_U = T_F + T_{RW} + T_I$$

خواندن تمام فایل T_X

این فایل، جای جای دارای حافظه هرز است که باید خوانده شود، در واقع تمام حفره‌های فایل باید خوانده شوند.

$$T_{X_{seq}} = (m + o')\frac{R}{t'}$$

ویا:

$$T_{X_{seq}} = (n + o')\frac{R + W_R}{t'}$$

در این‌جا نیز فرض بر این است که از ناحیه جداگانه سرریزی برای درج تصادفی‌ها استفاده می‌شود، در غیر این صورت دخالت دادن o' لزومی ندارد.

$$T_{X_{ser}} = n\,T_F$$

سازماندهی مجدد T_Y

این فایل از نظر احیاء نظم آغازین نیازی به سازماندهی مجدد ندارد. اگر ناحیه جداگانه سرریزی پر شود و یا زنجیره تصادفی ها بیش از حد طولانی گردد، فایل را باید سازماندهی مجدد کرد.

عملیات لازم:

۱) خواندن تمام فایل

۲) لود کردن فایل با رکوردهای فعال (رکوردهای حذف شدنی دیگر وارد فایل نمی‌شوند).

$$T_y = T_x + T_{Load}(n)$$

برای لود کردن فایل مستقیم، ساده‌ترین روش (و نه لزوماً کاراترین) این است که رکوردها را، رکورد به رکورد، در فضای آدرسی بنویسیم، در این صورت خواهیم داشت:

$$T_{Load}(n) = n\,T_I$$

که n تعداد رکوردهای فعال است. بدیهی است زمان T_I با افزایش $\frac{n}{m}$، تغییر خواهد کرد. روش دیگر این است که رکوردها را، برحسب مقدار آدرس آن‌ها، که در اثر اعمال تابع روی کلید به دست می‌آید، مرتب می‌کنیم در حالی که هر رکورد یک آدرس با خود دارد، سپس این فایل مرتب شده را، با رکوردهای بدون آدرس، در فضای آدرسی فایل مستقیم بازنویسی می‌کنیم. حفره‌هایی که آدرس آن‌ها در فایل مرتب شده موجود نیست، پرش می‌شوند و سپس رکوردهای تصادفی، براساس یکی از روش‌های درج در فایل درج می‌شوند.

زمان لود برابر است با:

$$T_{Load} = C_{h_{load}} + T_{sort} + T_{X_{seq}}$$

از بین ساختارهایی که تاکنون مطالعه شد، «فایل مستقیم» سریعترین است.

معایب ساختار مستقیم:

- محدودیت ثابت بودن طول رکوردها
- **عدم تقارن:** نشانوند جستجو صفت خاصه‌ای باید باشد که کلید فایل است.
- عدم امکان پردازش سریال فایل
- بروز پدیده تصادف

درخت B

به کمک درخت B می‌توان شاخص‌هایی چند سطحی پدید آورد که مشکل هزینه خطی درج و حذف کردن کلید را ندارند. امروزه درخت‌های B روش استاندارد جهت شاخص سازی به حساب می‌آیند. هر گره درخت B در واقع یک رکورد شاخص است و هنگامی که یک گره پر می‌شود لازم نیست عملیات شیفت را انجام داد، بلکه کافی است گره مذکور را به دو گره نیمه پر تقسیم کرد. هنگام حذف نیز هنگامی که گره خالی می‌شود نیازی به عمل شیفت نیست، بلکه کافی است گره‌های مناسبی را با هم ترکیب کنیم. پس عملیات درج و حذف کلید به جای شیفت دادن رکوردها به تقسیم و ترکیب شدن گره‌ها تبدیل می‌شوند.

هر گره درخت B از تعداد یکسانی جفت (کلید ـ آدرس) تشکیل یافته است. تعداد این جفت کلید آدرس را مرتبه درخت B می‌نامند.در هر گره باید حداقل نصف جفت کلید ـ آدرس استفاده شده باشند . مثلاً یک درخت B از مرتبه ۱۰۰، در هر رکورد (گره) حداقل دارای ۵۰ کلید و حداکثر دارای ۱۰۰ کلید است . تنها استثناء گره ریشه است که می تواند حداقل دو کلیدداشته باشد .

اضافه کردن یک کلید جدید به رکورد شاخصی که پر نیست بسیار ساده بوده و فقط کافی است همان رکورد شاخص را اصلاح کنید . ولی اگر کلید جدید بزرگترین کلید موجود در گره باشد ، کلید گره بالاتر آن نیز خواهد بود و بنابراین گره سطح بالا نیز باید به هنگام شود . هنگامی که اضافه کردن به یک رکورد شاخص باعث سرریز شدن آن گردد ، رکورد مذکور را به دو رکورد تقسیم می کنیم ، به نحوی که هر کدام شامل نصف کلیدها گردند . از آنجا که یک گره شاخص جدید در این سطح پدید آمده ، لازم است بزرگترین کلید در این گره جدید به سطح بالاتر اضافه گردد که به این کار اصطلاحاً ارتقای کلید گفته می شود . این ارتقای کلید ممکن است در سطوح بالاتر نیز ادامه پیدا کند . اگر این سرریز شدن تا ریشه ادامه پیدا کند آنگاه یک سطح دیگر به شاخص چند سطحی درخت B اضافه می گردد بنابراین درخت B از برگ‌ها به سمت ریشه رشد می کند .

مثال از ساختن درخت B

در این جا با یک مثال نحوه درج کلید جدید در درخت B و ساختن آن را نشان می دهیم .

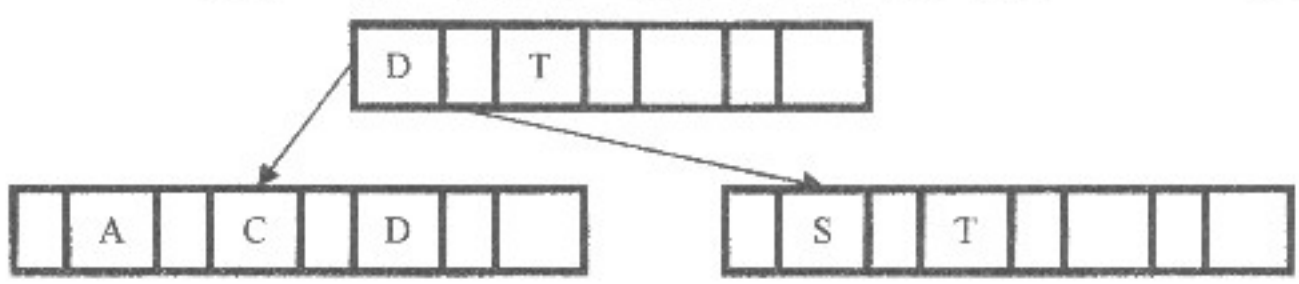
مثال ۱: کلیدهای زیر به ترتیب از چپ به راست وارد سیستم شده‌اند و می‌خواهیم شاخص B – Tree مربوط به آن‌ها را ترسیم کنیم:

C S D T A M P I B W N G U R K E H O L J Y Q Z F X V

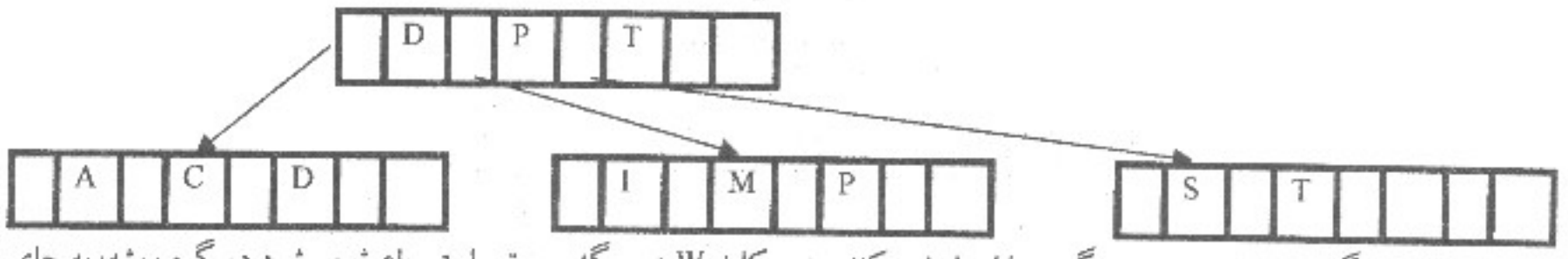
فرض کنید از درخت B مرتبه 4 استفاده کنیم یعنی در هر گره حداکثر 4 جفت کلید ـ آدرس و حداقل 2 جفت کلید ـ آدرس وجود داشته باشد. شکل زیر مراحل ساخت این درخت B را نشان می‌دهد.

۱ـ درج کلیدهای C , S , D , T در گره ریشه

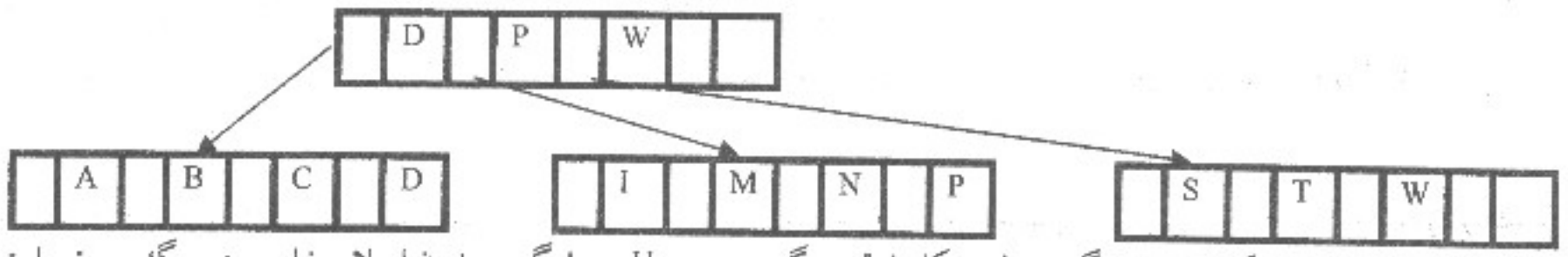
۲ـ با درج گره A، گره ریشه شکسته شده و بزرگترین کلید در هر گرهٔ برگ در گره ریشه قرار داده می‌شود:



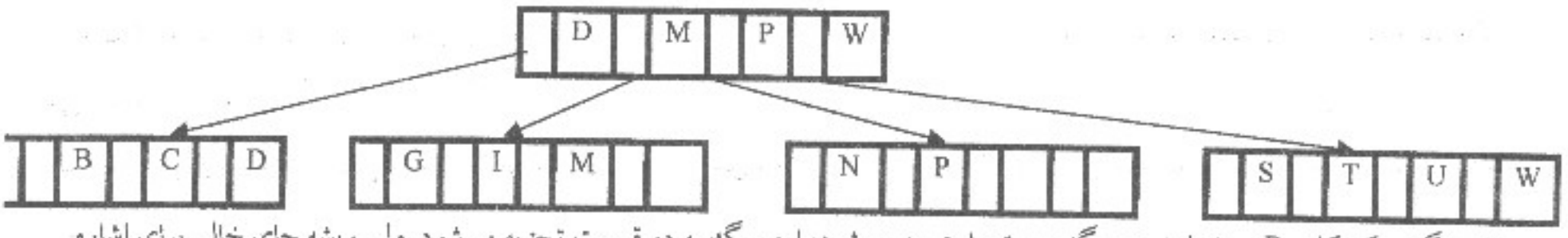
۳ـ ابتدا M ، P در برگ سمت راستی شکل فوق درج می‌شوند، سپس درج I باعث شکسته شدن گرهٔ سمت راستی می‌گردد:



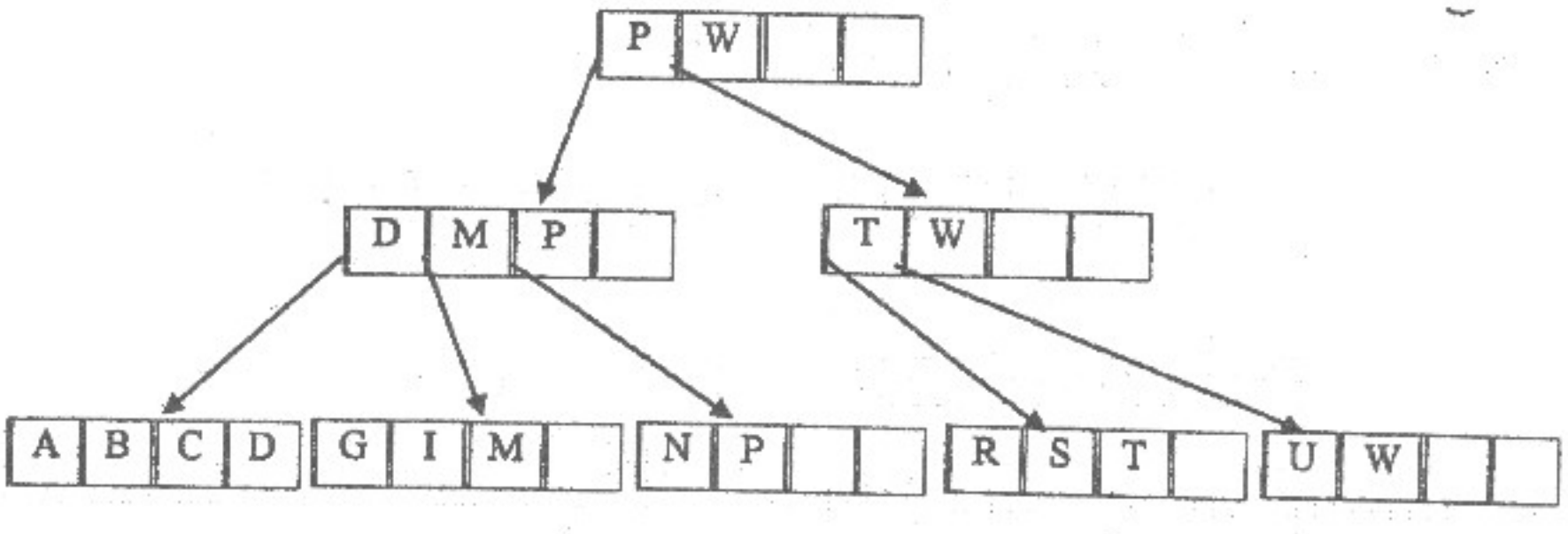
۴ـ درج B در برگ سمت چپی تغییر دیگری را ایجاد نمی‌کند. درج کلید W در برگ سمت راستی باعث می‌شود در گره ریشه، به جای T کلید W نوشته شود. درج کلید N در برگ وسطی نیز نیاز به اصلاح دیگری ندارد.



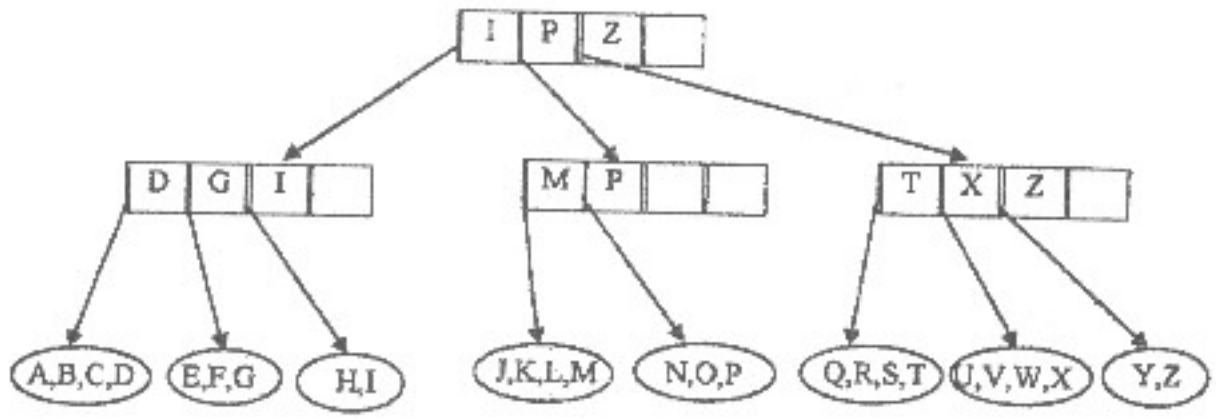
۵ـ درج کلید G باعث شکسته شدن برگ وسطی شکل فوق می‌گردد. سپس U به سادگی و بدون اصلاح خاصی در برگ سمت راستی ذخیره می‌شود.



۶ـ هنگامی که کلید R می‌خواهد در برگ سمت راستی درج شود، این برگ به دو قسمت تجزیه می‌شود. ولی ریشه جای خالی برای اشاره کردن به این گره جدید را ندارد، لذا عمل تجزیه باید در سطح بالا (یعنی ریشه) نیز انجام گیرد.



۷ـ روند فوق را ادامه دهید تا به شکل نهایی زیر برسید:



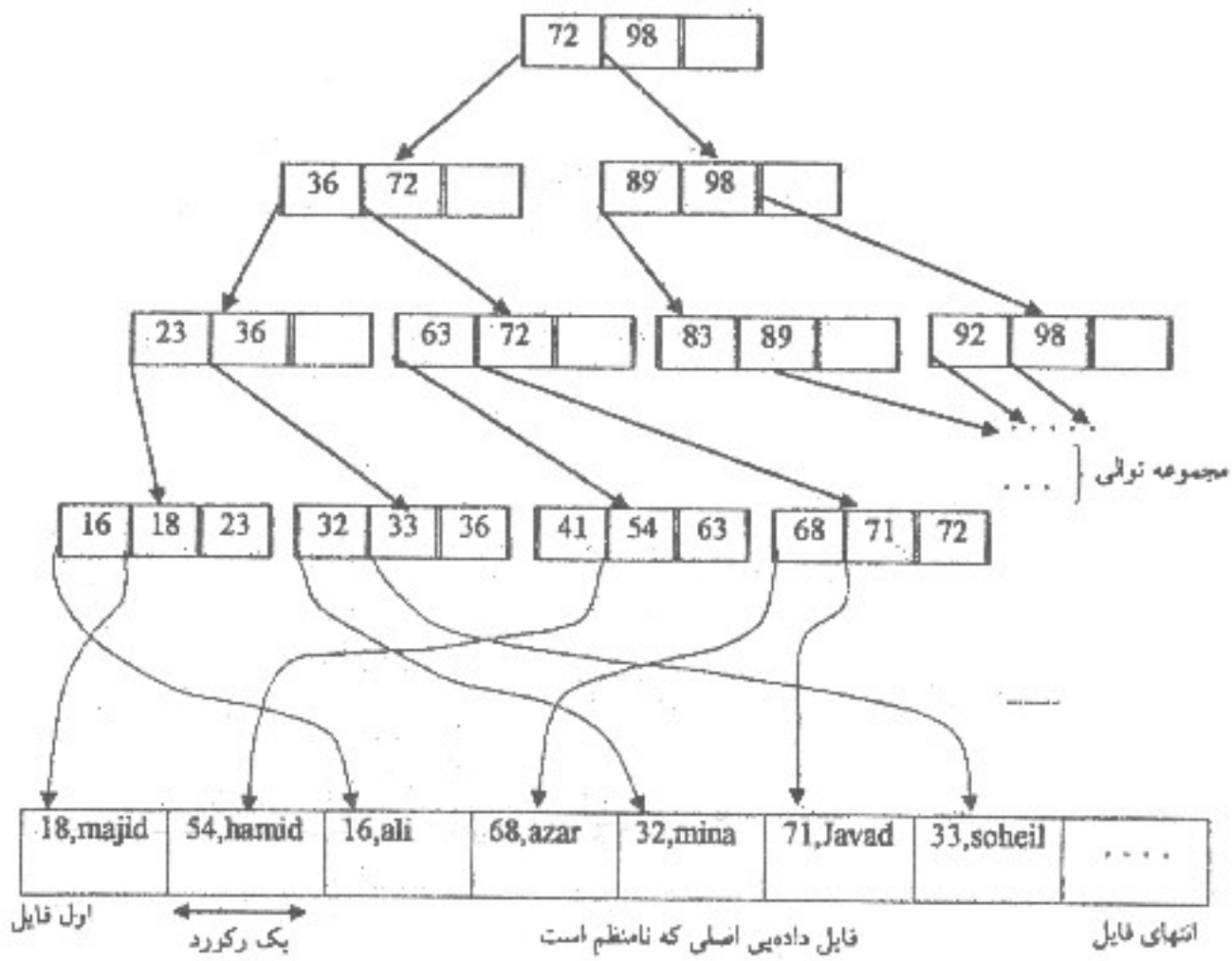
پس درخت B نهایی دارای 3 سطح است، به عبارتی دیگر ارتفاع آن برابر 3 می‌باشد.

اگر ساخت درخت جستجویی دودویی (BST) را از درس ساختمان داده‌ها به یاد دارید، آن را با داده‌های فوق ترسیم کرده و BST حاصل را با درخت B فوق مقایسه کنید.

توجه کنید که همه کلیدها در سطح آخر (سطح برگ‌ها) از چپ به راست مرتب شده هستند. پس پائین‌ترین سطح درخت B (یعنی برگ‌ها) همواره دارای نظم بوده و به این دلیل به آن مجموعه توالی (Sequence set) نیز می‌گویند. همچنین توجه کنید که درخت B در هر مرحله متوازن است. به عبارتی دیگر در B – Tree ارتفاع تمام شاخه‌ها از ریشه تا برگ‌ها یکسان است.

درخت B را هم می‌توان برای شاخص اولیه و هم برای شاخص ثانویه به کار برد. هنگامی که درخت B برای پیاده‌سازی شاخص اولیه به کار می‌رود، فید آدرس در برگ‌ها به آدرس رکورد متناظر آن کلید اشاره می‌کند.

مثال ۲: شکل زیر نحوه استفاده درخت B مرتبه 3 را برای شاخص اولیه فایل کد کارمندان نشان می‌دهد. کلید اصلی شماره یکتای 2 رقمی است که به هر کارمند داده می‌شود.



تعریف رسمی درخت B

با توجه به توضیحات فوق می توان خواص اصلی درخت B از مرتبه m را به صورت زیر بیان کرد:

۱ـ هر گره حداکثر m فرزند دارد.

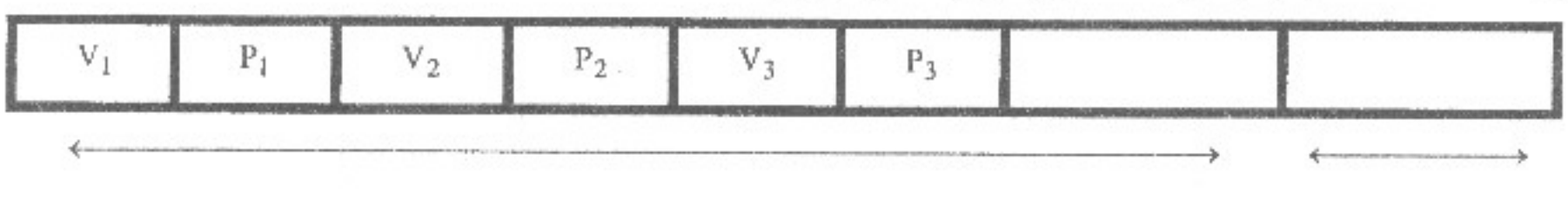
۲ـ هر گره به جز ریشه حداقل $\left\lceil \frac{m}{2} \right\rceil$ فرزند دارد.

۳ـ ریشه حداقل دو فرزند دارد (مگر آن که برگ باشد)

۴ـ تمام برگ‌ها در یک سطح واقع شده‌اند.

۵ـ سطح برگ‌ها، یک شاخص مرتب و کامل از فایل داده‌یی اصلی را پدید می‌آورد.

در لود اولیه درخت شاخص B، حداقل نیمی از مدخل‌های هر گره درخت پر می‌باشد. بخش‌هایی از هر گره نیز احتمالاً جهت توسعه آینده رزرو شده‌اند. لذا ساختار هر گره درخت B در لود اولیه به شکل کلی زیر است:



همان‌طور که می‌دانید، اگر V + P طول هر مدخل و اندازه هر مدخل برابر یک بلاک (B بایت) باشد، آن‌گاه ظرفیت نشانه روی اسمی هر گره برابر است با:

$$\text{اسمی} \quad y = \left\lfloor \frac{B}{V + P} \right\rfloor$$

این ظرفیت اسمی در لود اولیه به طور کامل استفاده نمی‌شود. اگر ظرفیت واقعی نشانه روی (Effective fanout) را با y_{eff} نشان دهیم آن‌گاه:

$$\frac{y}{2} \le y_{eff} \le y$$

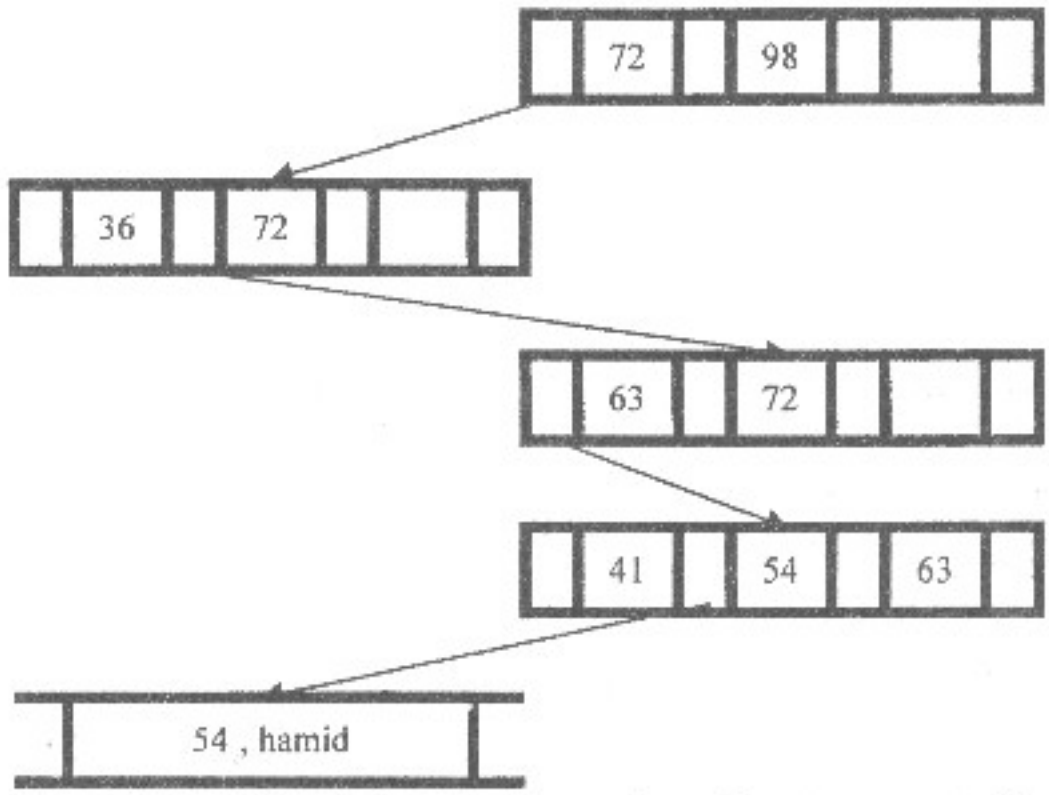
چرا که در لود اولیه حداقل نیمی از مدخل‌های هر گره پر می‌باشد. محققین نشان داده‌اند که اغلب ضریب $y_{eff} = 0.69\ y$ برای لود اولیه مقدار مناسبی است.

پیاده سازی دقیق الگوریتم‌های کار با درخت B نظیر جستجو، درج و حذف در کتاب‌های ساختمان داده‌ها (با زبان‌های مختلف) آورده شده است. مثلاً در فصل روش‌های جستجو از کتاب ساختمان‌داده‌ها در C نوشته تننباوم می‌توانید لیست این توابع را به زبان C مشاهده کنید. ما در این‌جا این الگوریتم‌ها را به صورت کلی بررسی می‌کنیم.

جستجو در درخت B

فرض کنید که اندازه هر گره برابر یک بلاک است و خواندن هر گره به یک دستیابی (پیگرد) دیسک نیاز دارد.

مثال ۳: به شکل مثال ۲ نگاه کنید، فرض کنید نام کارمندی با شماره 54 را بخواهیم. بلاک ریشه را می‌خوانیم و در آن به دنبال اولین مدخلی می‌گردیم که بزرگتر یا مساوی 54 باشد. بدین ترتیب مدخل شماره 72 پیدا شده و از طریق فیلد آدرس این مدخل به سراغ بلوک سمت چپی در سطح دوم می‌رویم. به همین ترتیب عملیات را ادامه می‌دهیم تا به فایل داده‌یی اصلی برسیم. شکل زیر این روند جستجو را نشان می‌دهد:



اگر گره ریشه در حافظه اصلی نگهداری شود برای واکشی رکورد (54 , hamid) به 4 پیگرد در دیسک نیاز خواهیم داشت.بدیهی است که زمان جستجو به تعداد پیگردها بستگی دارد. تعداد پیگردها نیز به ارتفاع درخت و ارتفاع درخت به اندازه گره‌ها وابسته است. هنگام کار با درخت B می‌بایست بتوانیم به پرسش زیر پاسخ دهیم:

«در بدترین حالت، برای پیدا کردن یک کلید در درخت B، حداکثر به چه تعداد پیگرد در دیسک نیاز داریم؟»

در یک درخت B از مرتبه m، حداقل تعداد فرزندان گره ریشه 2 عدد می‌باشد، لذا سطح دوم درخت تنها 2 گره دارد. هر کدام از این گره‌های سطح دوم، حداقل $\left\lceil \frac{m}{2} \right\rceil$ فرزند خواهند داشت. لذا در سطح سوم $2 \times \frac{m}{2}$ گره داریم. از آن‌جا که هر کدام از این گره‌ها هم حداقل $\left\lceil \frac{m}{2} \right\rceil$ فرزند دارند، رابطه میان سطح و حداقل فرزندان به صورت زیر خواهد بود:

سطح	حداقل تعداد فرزندان
1 (ریشه)	2
2	$2 \times \left\lceil \frac{m}{2} \right\rceil$
3	$2 \times \left\lceil \frac{m}{2} \right\rceil^2$
4	$2 \times \left\lceil \frac{m}{2} \right\rceil^3$
⋮	⋮

پس در حالت کلی در هر سطح d از درخت B، حداقل تعداد فرزندان آن سطح برابر است با:

$$\text{حداقل تعداد فرزندان در سطح d} = 2 \times \left\lceil \frac{m}{2} \right\rceil^{d-1}$$

اگر فایل شاخص N کلید باشد و ارتفاع درخت B برابر H باشد (یعنی حداکثر d) آن‌گاه بدیهی است که بین N و H رابطه زیر برقرار است:

$$N \geq 2 \times \left\lceil \frac{m}{2} \right\rceil^{H-1}$$

با حل نامعادله فوق برای محاسبه H خواهیم داشت:

$$H \leq 1 + \log_{\left\lceil \frac{m}{2} \right\rceil} \left(\frac{N}{2} \right)$$

مثال ۴: حداکثر عمق درخت B از مرتبه 512 که شامل یک میلیون رکورد است را بدست آورید.

حل:

$$H \leq 1 + \log_{256} 500000 \rightarrow H \leq 3.37$$

یعنی یک درخت B از مرتبه 512 با داشتن یک میلیون رکورد عمقی بیشتر از 3 سطح نخواهد داشت.

تذکر: ما جهت سادگی ترسیم شکل‌ها در مثال‌های 1 , 2 مرتبه درخت را 4 , 3 در نظر گرفتیم ولی در عمل مرتبه درخت مانند مثال فوق می‌تواند 512 یا بیشتر باشد.

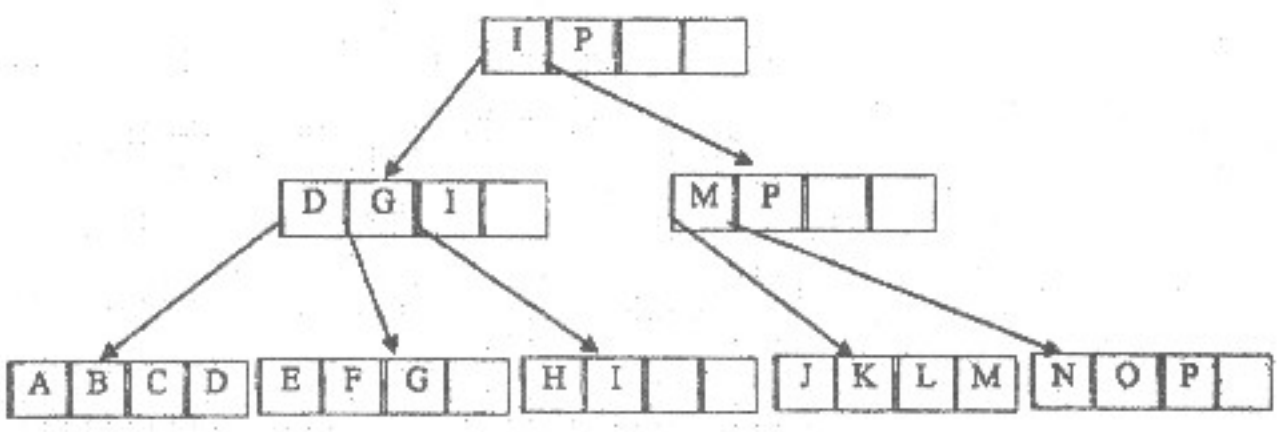
درج در درخت B

ما قبلاً در مثال ۱ مراحل درج کلیدهای جدید در درخت B و حالت‌های مختلفی که می‌تواند رخ بدهد. را شرح داده‌ایم. جهت درج ابتدا با جستجویی که تا سطح برگ ادامه می‌یابد، محل درج پیدا می‌شود. پس از پیدا شدن محل درج، عمل اضافه کردن، تشخیص سرریزی و عمل تقسیم کردن (Splitting) از پائین به سمت بالا پیش می‌رود. اگر در عمل درج نیاز به تقسیم کردن گره باشد، حداقل سه گره باید ایجاد یا اصلاح شوند. اگر هنگام تقسیم کردن، گره سطح بالایی نیز پر باشد، خود آن گره سطح بالا نیز باید تقسیم گردد. بنابراین عمل تقسیم گره‌ها ممکن است حتی تا ریشه ادامه پیدا کرده و در این صورت عمق درخت از x به x + 1 تغییر یابد. یکبار دیگر به روند مثال ۱ نگاه کنید.

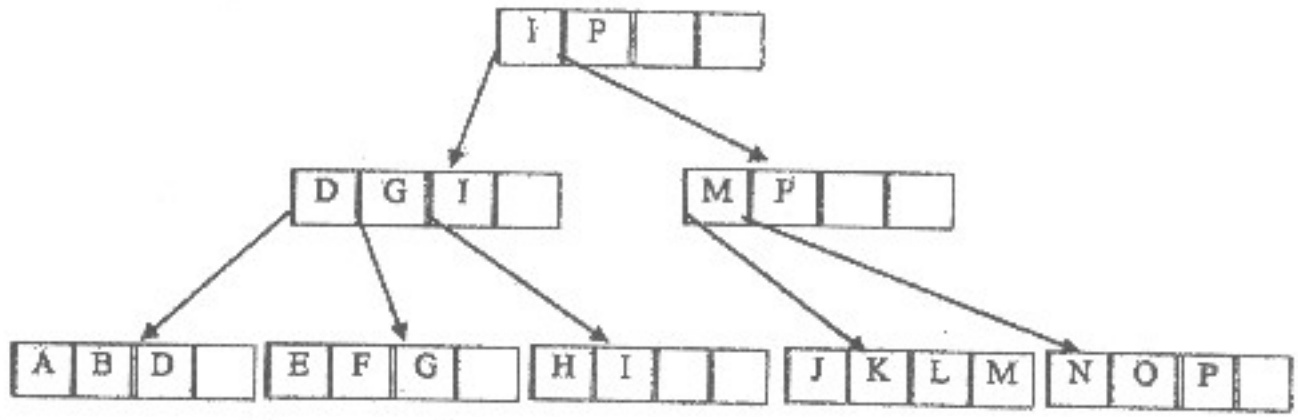
حذف از درخت B

ابتدا با یک مثال دستی نشان می‌دهیم که عمل حذف یک کلید از درخت شاخص B می‌تواند منجر به حالت‌های متفاوتی گردد.

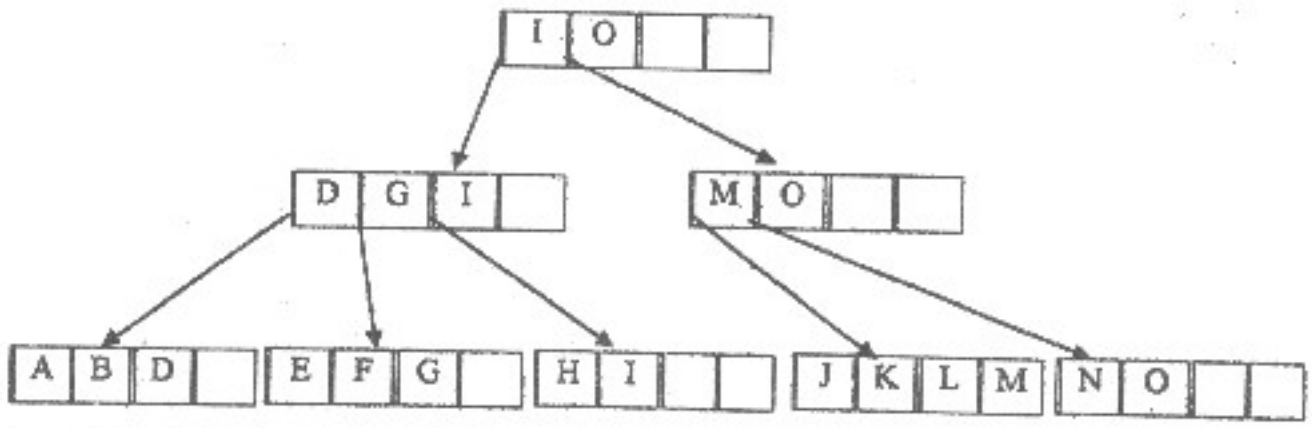
مثال ۵: در شکل زیر به ترتیب کلیدهای H , P , C را حذف کرده و در هر مرحله شکل حاصل را ترسیم کنید.



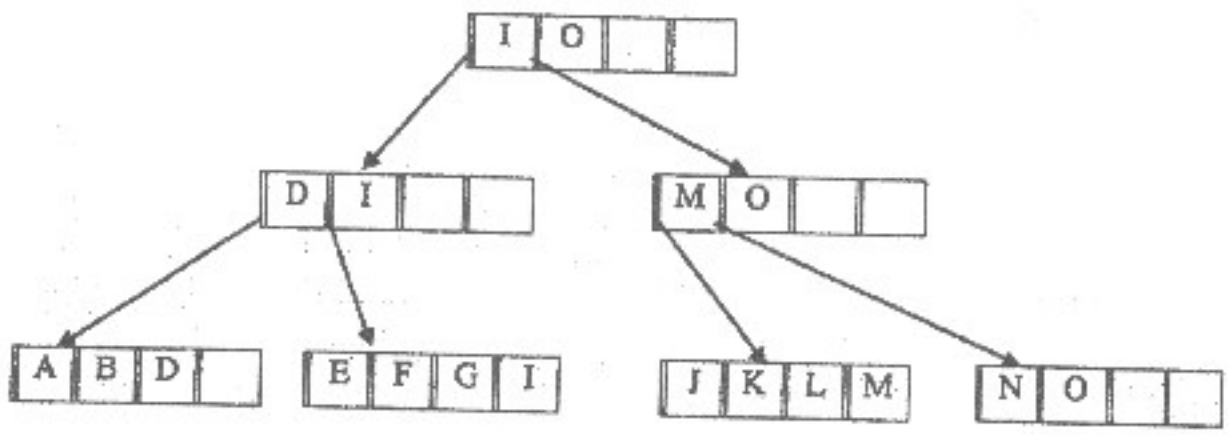
حل: حذف کلید C ساده‌ترین حالت حذف بوده که بر اثر آن، گره از حد مجاز خالی‌تر نمی‌شود و همچنین بزرگترین کلید موجود در گره نیز تغییری نمی‌کند. در این حالت کافی است به سادگی کلید C را پاک کنیم و هیچ اصلاح دیگری لازم نیست. بنابراین با حذف C شکل زیر حاصل می‌شود:



حذف کلید P باعث خالی شدن غیرمجاز گره برگ حاوی آن نمی‌شود. ولی چون بزرگترین کلید برگ را تغییر می‌دهد لذا سطح (یا سطوح) بالایی آن می‌بایست به طور متناسب اصلاح گردد.



حذف کردن کلید H در شکل فوق باعث می‌شود برگ حاوی آن بیش از حد مجاز خالی شود. در این حال تنها کلید باقی مانده در این گره (یعنی I) به همسایه سمت چپی‌اش که جای خالی دارد اضافه می‌شود. بدین ترتیب در هنگام حذف ممکن است به عملیات ادغام نیاز داشته باشیم. پس از انجام عمل ادغام، سطح بالایی می‌بایست متناسباً اصلاح گردد. این اصلاحات ممکن است تا ریشه درخت ادامه پیدا کند. اگر در آخر، ریشه فقط یک کلید و یک فرزند داشته باشد می‌توان آن را حذف کرد و تنها فرزندش را جایگزین ریشه نمود. بدین ترتیب تعداد سطوح درخت یک واحد کاهش می‌یابد. بنابراین بعد از حذف کلید H شکل درخت به صورت زیر در می‌آید:



با توجه به مثال فوق می‌تواند قواعد حذف کلید K از یک گره n را در یک درخت B به صورت زیر بیان کرد:

۱ـ اگر K بزرگترین کلید در گره n نیست و تعداد کلیدهای گره n بیشتر از حداقل کلیدهای مجاز می‌باشد، فقط کافی است K را از گره n حذف کنیم.

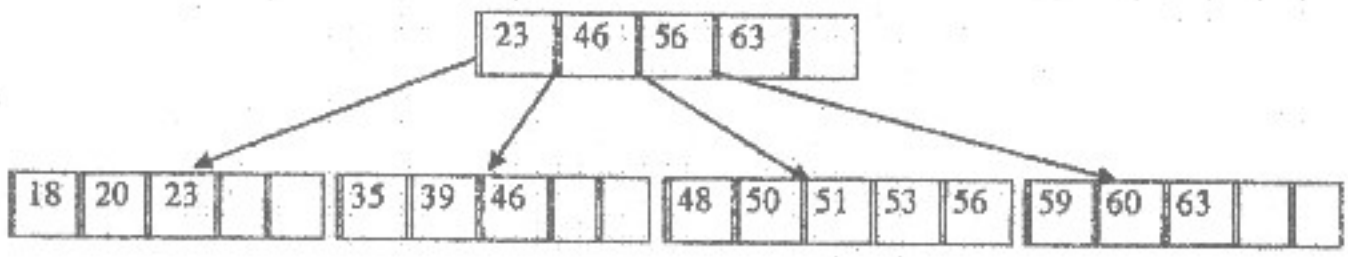
۲ـ اگر K بزرگترین کلید در گره n است و تعداد کلیدهای گره n بیشتر از حداقل کلیدهای مجاز می‌باشد، آنگاه K را حذف کرده و کلیدهای سطح و یا سطوح بالاتر را متناسب با بزرگترین کلید جدید در گره n تغییر می‌دهیم.

۳ـ اگر تعداد کلیدهای گره n دقیقاً برابر حداقل کلیدهای مجاز می‌باشد و یکی از برادرهای گره n به اندازه کافی خالی است، آنگاه گره n را با برادرش ادغام کرده و یک کلید از گره مادر را حذف می‌کنیم.

۴ـ اگر تعداد کلیدهای گره n دقیقاً برابر حداقل کلیدهای مجاز می‌باشد و یک از برادرهای گره n کلیدهای زیادی دارد، آن‌گاه با انتقال بعضی از کلیدها از یک برادر به گره n، کلیدها را دوباره توزیع کرده و کلیدهای سطوح بالاتر را متناسب با بزرگترین کلیدهای جدید گره‌های دستکاری شده، اصلاح می‌کنیم.

تذکر ۱: ممکن است در حذف یک کلید، هر دو عمل 3 , 4 امکان‌پذیر باشد.

مثال ۶: در درخت B مرتبه 5 شکل زیر، حداقل تعداد کلیدها در هر گره می‌بایست 3 عدد باشد:



جهت حذف کلید 20 باید از روش ادغام استفاده کنیم و امکان توزیع دوباره وجود ندارد چرا که اگر مثلاً از گره برادر سمت راستی اش بخواهیم عدد 35 را وارد این گره کنیم خود آن گره 2 کلیده خواهد شد و این غیر مجاز است .

جهت حذف کلید 60 نمی توانیم از روش ادغام برویم چرا که گره برادر سمت چپی اش کاملاً پر است ، لذا در این حالت باید از روش توزیع دوباره رفته و مثلاً یک یا دو کلید (56 و یا 53) را از گره برادر سمت چپی ، به گره مذکور انتقال دهیم .

جهت حذف کلید 39 ، دو انتخاب وجود دارد . هم می توان گره مذکور را در برادر سمت چپی ادغام کرد و هم می توان آن را با گره برادر راستی اش توزیع دوباره نمود .

تذکر ۲: توزیع دوباره با تقسیم و ادغام کردن از این جهت تفاوت دارد که هیچگاه باعث تغییر تعداد زیادی از گره ها نمی گردد و اثرات آن محلی می باشد .

تذکر۳: هنگام در ج در درخت B الزامی به استفاده از تکنیک توزیع دوباره نیست و عمل تقسیم کردن همواره می تواند حالت سرریزی را برطرف سازد . ولی گاهی اوقات استفاده از روش توزیع دوباره در هنگام درج نیز مطلوب می باشد ، چرا که به کمک توزیع دوباره می توان از ایجاد گره های جدید جلوگیری کرد و یا پدید آمدن آنها را به تعویق انداخت . تحقیقات نشان داده اند که استفاده از تکنیک توزیع دوباره در هنگام درج (به جای تقسیم کردن) بهره وری استفاده از فضا را می تواند تا 85 درصد بهبود بخشد .

تذکر۴: در فایلی که چندین درخت شاخص دارد ، ممکن است جهت بهنگام سازی لازم باشد تعدادی یا همه درخت های شاخص اصلاح گردند . در این حالت می توان عملیات تنظیم را به صورت تدریجی انجام داد یعنی با تعیین اولویت بین شاخص ها ، در ابتدا مهمترین درخت شاخص را اصلاح کرد و سپس در مواقعی مناسب درخت های دیگر را به هنگام در آورد .

مجموعه سوالات کنکوری

۱ـ در دیسکی با مشخصات زیر زمان لازم برای خواندن سه سکتور به صورت تصادفی چند ms است؟

ظرفیت سکتور = 512 Byte ، r=8 ms ، S =10 ms ، نرخ انتقال = 2MB/sec

۱) 18.7 ۲) 37.5 ۳) 54.7 ۴) 109.4

۲ـ استفاده واقعی از دیسکی سکتوربندی شده با مشخصات زیر، چند درصد است؟

فاکتور بلاک‌بندی=4، اندازه رکورد=320 byte، اندازه سکتور = 512 byte

۱) 45 ۲) 68 ۳) 75 ۴) 83

۳ـ با در نظر گرفتن مفروضات زیر، زمان خواندن کل فایل به طور تصادفی چند ms است؟ با توجه به اینکه در هر بار خواندن تنها یک رکورد خوانده می‌شود. (اندازه بلاک برابر باکت است)

2800 B/S : نرخ انتقال انبوه همراه گپ‌ها، 3000B/S : نرخ انتقال اسمی، 200 byte : اندازه رکورد، 18000 : تعداد رکوردها، 2400 : اندازه بلاک، 6.5 ms : (r) زمان درنگ دورانی، 10 ms : زمان استوانه‌جویی

۱) 312120 ۲) 311400 ۳) 26010 ۴) 25950

۴ـ در کدامیک از موارد زیر سازماندهی مجدد فایل لازم نیست؟

۱) تغییر ساختار فایل شاخص
۲) تغییر استراتژی دستیابی
۳) برگرداندن ساختار فایل به حالت اولیه
۴) خارج کردن حافظه‌های هرز از فایل

۵ـ در فایل پایل زمان بدست آوردن رکورد بعدی چگونه محاسبه می‌شود؟

۱) $T_N = 2T_F$ ۲) $T_N = T_F$ ۳) $T_N = \frac{T_F}{2}$ ۴) $T_N = \frac{T_F}{4}$

۶ـ در فایلی از نوع پایل که بر روی دیسک ذخیره شده است طبق مفروضات زیر، زمان خواندن کل فایل به صورت ترتیبی چقدر است؟

5000: تعداد رکوردها، 2800 kB/S: زمان انتقال انبوه، 50 Byte : اندازه رکورد

۱) 223.2s ۲) 111.6s ۳) 178 ms ۴) 89ms

۷ـ کدامیک از موارد زیر بیان کننده لنگرگاه فایل می‌باشد؟

۱) هر یک از رکوردهای ناحیه اصلی که اشاره‌گری از فایل شاخص به آن وجود دارد.
۲) هر یک از مداخل ورودی فایل شاخص که به مدخل دیگری اشاره دارد.
۳) هر یک از مداخل ورودی فایل شاخص.
۴) هر یک از رکوردهای ناحیه اصلی.

۸ـ یک فایل ترتیبی داریم که روی دیسک ذخیره شده است. زمان خواندن رکوردهای این فایل به ترتیب معکوس (از آخرین رکورد) به صورت ترتیبی کدام است؟

250000B/S : زمان انتقال انبوه، 100B : اندازه رکورد، 400 : تعداد رکوردها

۱) 320 ms ۲) 160ms ۳) 32s ۴) 16s

۹ـ زمان بدست آوردن رکورد بعدی در روش Push-Through چگونه محاسبه می‌شود؟

۱) $T_N = \frac{n + O'B_F}{(n+O')B_T}\left(s + b_{tt}\right)$ ۲) $T_N = \frac{O'}{n+O'}\left(s+r+b_{tt}\right)$

۳) $T_N = \left(\frac{1-Pro}{B_F} + Pro\right)\left(r + b_{tt}\right)$ ۴) $T_N = Pro\left(s+r+b_{tt}\right)$

۱۰ ـ کدامیک از موارد زیر جزء حالات شتگانه قرارگیری رکورد بعدی در روش Push –Through نیست؟

۱) رکورد فعلی آخرین رکورد بلاک ناحیه اصلی است و رکورد بعدی در بلاک بعدی از همان استوانه

۲) رکورد فعلی آخرین رکورد بلاک ناحیه اصلی است و رکورد بعدی در بلاکی از ناحیه سرریزی

۳) رکورد فعلی در بلاکی از ناحیه سرریزی است و رکورد بعدی در بلاکی از ناحیه اصلی

۴) رکورد فعلی آخرین رکورد از آخرین بلاک ناحیه اصلی است و رکورد بعدی در آخرین بلاک ناحیه سرریز

۱۱ ـ برای انجام عمل درج یک رکورد جدید در روش Push Through چه اعمالی باید انجام شود؟

۱) اضافه کردن رکورد جدید به انتهای فایل ـ تنظیم اشاره‌گر رکورد

۲) افزودن رکورد جدید به آخرین بلاک ـ تنظیم اشاره‌گر رکورد قبلی به رکورد جدید ـ بازنویسی بلاک

۳) خواندن بلاک آخر برای اضافه کردن رکورد جدید ـ واکشی رکورد قبلی و تنظیم اشاره‌گر آن ـ بازنویسی بلاک ـ ایجاد اشاره‌گر در فایل شاخص

۴) خواندن بلاکی که رکورد باید در آن درج شود ـ بازنویسی این بلاک ـ واکشی رکورد منطقاً قبلی و تنظیم اشاره‌گر ـ بازنویسی همین رکورد

۱۲ ـ کدامیک از موارد زیر جزء معایب روش Push Through محسوب نمی‌شود؟

۱) ایجاد شاخص برای سرریزیها ۲) پویا نبودن شاخص

۳) عدم تقارن ۴) مسئله درج سرریزیها

۱۳ ـ فایلی داریم شاخص‌بندی شده دارای 100000 رکورد، تعداد مدخلها در سطح اول شاخص و حافظه مصرفی آن به ترتیب کدام است؟

2000B : طول بلاک، V:20B ، P:6B ، 380 B : طول رکورد

۱) 20000، 400000B ۲) 20000 ، 520000B

۳) 25000، 650000B ۴) 16666، 99996B

۱۴ ـ کدام فایل دارای افزونگی اطلاعات است؟

۱) فشرده نشده باشد.
۲) بعضی از رکوردها تکراری باشد.
۳) بعضی از صفات خاصه‌اش بیش از یک بار تکرار شده باشد.
۴) اطلاعات اضافی در فایل وجود داشته باشد.

۱۵ ـ فایل داده‌ای (اصلی) در ساختار چند شاخصی چه ساختاری دارد؟

۱) مستقیم
۲) ترتیبی
✓ ۳) پایل
۴) ترکیبی پایل و مستقیم

۱۶ ـ کدام عبارت نادرست است؟

۱) برای ایجاد مکان ذخیره شاخص در حافظه ثانویه باید از شاخص‌های چند سطحی استفاده نمود.
✓ ۲) در ساختار ترتیبی شاخص‌دار فایل شاخص ممکن است همتوالی با فایل داده‌ایی نباشد.
۳) در عملیات درج در ساختار ترتیبی شاخص‌دار فایل شاخص Update نمی‌گردد.
۴) یکی از مشکلات ساختار ترتیبی شاخص‌دار مسئله درج سرریزی‌ها است.

۱۷ ـ کدام تعریف برای فایل وارون صحیح است؟

۱) فایلی با ساختار ترتیبی شاخص‌دار که تمام صفات خاصه‌اش مرتب باشند.
✓ ۲) فایلی با ساختار چندشاخصی که روی تمام صفات خاصه‌اش شاخص وجود داشته باشد.
۳) فایلی با ساختار ترتیبی اگر فایل تراکنش وجود داشته باشد.
۴) فایلی با ساختار چند شاخصی اگر فقط یک شاخص داشته باشد.

۱۸ ـ کدام گزینه در مورد «ساختار ترتیبی شاخص‌دار» نادرست است؟

✓ ۱) امکان خواندن تمام رکوردها به صورت ترتیبی بدون استفاده از شاخص امکان‌پذیر نیست.
۲) چون ناحیه اصلی باید ترتیبی باشد برای سادگی عمل حذف و درج یک ناحیه سرریزی ایجاد می‌شود.
۳) در شاخص متراکم واکشی تک رکوردها سریعتر از شاخص غیرمتراکم انجام می‌گیرد.
۴) واکشی تک رکوردها با استفاده از شاخص سریعتر انجام می‌گیرد.

۱۹ ـ فایلی را در نظر بگیرید که طول رکوردهای آن 160 بایت و طول سکتور آن 256 بایت باشد، اگر فاکتور بلاک‌بندی برابر 5 و تعداد 4 سکتور در هر بلاک باشد، استفاده واقعی از دیسک چند درصد است؟

۱) 91
۲) 85
۳) 78
۴) 50

۲۰ ـ درصد استفاده واقعی نواری با مشخصات زیر کدام است؟

سرعت: 125inch/sec $t_0 = 4\,msec$

چگالی: 1600bpi $B = 1200Byte$

۱) 70
۲) 60
۳) 50
۴) 40

۲۱ ـ علت کاهش ظرفیت واقعی یک دیسک مغناطیسی نسبت به حالت اسمی چیست؟

۱) فرمت‌بندی دیسک
۲) یکسان نبودن اندازه سکتورها
۳) یکسان نبودن اندازه تراک‌ها
۴) یکسان نبودن چگالی تراک‌ها

۲۲ ـ پدیده تصادف (Collision) هنگامی رخ می‌دهد که به ازای دو کلید ..... آدرس ..... تولید شود.

۱) $A_1 < A_2 \ , \ K_1 < K_2$
۲) $A_1 > A_2 \ ، K_1 < K_2$
۳) مختلف، مختلف
۴) مختلف ، یکسان

۲۳ ـ ساختار شاخص در فایل غیرترتیبی چند شاخصی کدام است؟

۱) درخت برپا شده در حافظه RAM
۲) درخت با گره‌هایی به اندازه بلاک
۳) شاخص توزیع شده در چند فایل
۴) فایل شاخص است.

۲۴ ـ عمل واکشی رکورد در کدام ساختار سریع‌تر انجام می‌شود؟

۱) مستقیم
۲) ترتیبی شاخص‌دار
۳) ترتیبی با واکشی باینری
۴) ترتیبی با واکشی پرش بلاکی

۲۵ ـ اندازه بهینه فاکتور بلاک‌بندی در ساختار ترتیبی با واکشی پرش بلاکی، در فایلی با تعداد رکورد 10^4 و طول رکورد 200 کدام است؟

۱) 10
۲) 100
۳) 200
۴) 300

۲۶ ـ پردازش ترتیبی در کدام ساختار، با توجه به نوع استراتژی دستیابی غیرممکن است؟

۱) ساختار ترتیبی شاخص‌دار
۲) ساختار غیرترتیبی شاخص‌دار
۳) ساختار مستقیم
۴) هر دو گزینه ۱ و ۲

۲۷ ـ کدام روش به منظور برطرف کردن مسئله تصادف در ساختار مستقیم مناسب‌تر است؟

۱) ایجاد زنجیره با جایگزین
۲) ایجاد زنجیره بدون جایگزین
۳) تصادفی کردن مجدد
۴) درج در اولین بلاک جادار

۲۸ ـ تعداد سطوح فایل شاخص غیرمتراکم برای یک فایل ترتیبی با 10^5 رکورد با طول رکورد 100 و طول بلاک 2000 بایت در صورتی که طول هر مدخل فایل شاخص برابر 20 بایت باشد، کدام است؟

۱) 1
۲) 2
۳) 3
۴) 4

۲۹ ـ از کدام تابع، می‌توان به عنوان یک تابع درهم ساز برای کلید Key در محدوده آدرس $0 \le A \le 100$ استفاده کرد؟ (تابع Rand (n) عددی اعشاری شانسی بین 0 تا n تولید می‌کند.)

۱) $\mathrm{Int}(\mathrm{Key} * \mathrm{Rand}(1))$
۲) $\mathrm{Int}(\mathrm{Rand}(\mathrm{Key}))$
۳) Key MOD 100
۴) Key MOD 101

۳۰ـ علت کاهش نرخ انتقال واقعی دیسک مغناطیسی کدام است؟

۱) گپ بین بلاکها
۲) زمان استوانه‌جویی
۳) زمان درنگ دورانی
۴) هر دو گزینه ۲ و۳

۳۱ـ کدام گزینه، از معایب ساختار ترتیبی شاخص‌دار نمی‌باشد؟

✓۱) اتلاف حافظه
۲) ایستا بودن شاخص
۳) عدم تقارن
۴) مسئله درج سرریزی‌ها

۳۲ـ یک فایل ترتیبی برای کدامیک از کاربردهای زیر مناسب است؟

۱) حقوق و دستمزد BATCH
۲) رزرواسیون جا در هواپیما
۳) صدور صورتحساب مشتریان در هر ماه
۴) موارد ۱ و۳

۳۳ـ علت کاهش ظرفیت واقعی یک دیسک مغناطیسی نسبت به حالت اسمی چیست؟

۱) یکسان نبودن اندازه تراک‌ها
۲) یکسان نبودن اندازه سکتورها
۳) فرمت‌بندی دیسک
۴) یکسان نبودن چگالی تراک‌ها

۳۴ـ یک طرح فایل بطور کلی روی کدامیک از موارد زیر کنترل دارد؟

۱) زمان CPU و زمان صف I/O
۲) زمان I/O روی فایل و فضای اشغال شده روی دیسک
۳) فضای اشغال شده روی حافظه اصلی و زمان CPU
۴) هیچکدام

۳۵ـ کدام فایل دارای افزونگی اطلاعات است؟

۱) بعضی از صفات خاصه‌اش بیش از یکبار تکرار شده باشد.
۲) فشرده نشده باشد.
۳) بعضی از رکوردها تکراری باشد.
۴) اطلاعات اضافی در فایل وجود داشته باشد.

۳۶ـ کدامیک از موارد زیر، در مورد سیستم فایلهای VSAM در IBM صحت دارد؟

✓۱) پس از ذخیره شدن تعدادی رکورد در یک فایل VSAM ممکن است هم دستیابی مستقیم و هم دستیابی ترتیبی وقت کمتری بگیرد.
۲) دستیابی ترتیبی نسبت به دستیابی مستقیم تعداد بیشتری INDEX BUFFER نیاز دارد.
۳) محدوده یک CONTROL AREA می‌بایست یک سیلندر باشد.
۴) موارد ۱ و۳

۳۷ـ در یک فایل سریال اندازه هر بلاک رکورد 4 کیلوبایت، طول هر رکورد 200 بایت و سرعت انتقال داده‌ها بین دیسک و حافظه اصلی 100 کیلوبایت در ثانیه می‌باشد. اگر فایل دارای 100 بلاک باشد زمان متوسط دستیابی به یک رکورد در این فایل چقدر است؟

۱) 3 ثانیه
۲) 40 میلی‌ثانیه
۳) 1 ثانیه
۴) 20 میلی‌ثانیه

۳۸ـ یک فایل به اندازه 2 مگابایت دارای اندازه بلاک برابر 4 کیلوبایت و اندازه نشانگر 32 بایت می‌باشد. تعداد بلاکهای ایندکس در این فایل چقدر است؟

۱) 2
۲) 8
۳) 4
۴) 1

۳۹ـ کدام یک از موارد، جزو دلایل نیاز به سازماندهی مجدد نمی‌باشد؟

۱) احیاء نظم ساختاری آغازین
۲) اصلاح استراتژی دستیابی
۳) ایجاد نسخه‌ای دیگر از فایل
۴) بازپس گرفتن حافظه‌های هرز

۴۰ـ فایلی با 5000 رکورد 40 بایتی داریم می‌خواهیم این فایل را روی نواری با چگالی 800bpi ذخیره کنیم طول این نوار برابر کدام است؟

۱) 160in
۲) 250in
۳) 200000in
۴) 32000in

۴۱ـ در یک نوار طول هر بلاک 400 بایت و طول گپ بین بلاکها 50 بایت می‌شود. اگر در مدت 10 ثانیه 18000 بایت اطلاعات از روی نوار خوانده شود نرخ مفید انتقال اطلاعات از نوار چند بایت بر ثانیه است؟

۱) 1600
۲) 1800
۳) 2000
۴) 2025

۴۲ـ تصادف (Collision) در ساختار فایل‌های مستقیم چه مفهومی دارد؟

۱) برای دو رکورد متفاوت یک آدرس برای ذخیره کردن آنها بدست آید.
۲) عدم تقارن ساختار
۳) وجود رکورد سرریز در ساختار
۴) یکسان شدن آدرس دو رکورد متفاوت و عدم تقارن ساختار

۴۳ـ در یک فایل از نوع Pile زمان به دست آوردن رکورد بعدی برابر کدام است؟

(B: اندازه بلاک، N تعداد بلاک‌ها یا رکوردها، 't نرخ واقعی انتقال)

۱) $\frac{1}{3}N\frac{B}{t'}$
۲) $\frac{1}{2}N\frac{B}{t'}$
۳) $2N\frac{B}{t'}$
۴) $N\frac{B}{t'}$

۴۴ـ در یک فایل ترتیبی شاخص‌دار که تعداد رکوردهای آن 1000000 و B=2000 بایت و R=200 بایت و V=14 بایت و P=6 بایت است تعداد سطوح برابر با کدام گزینه است؟

۱) 3
۲) 4
۳) 5
۴) 6

۴۵ ـ پدیدهٔ Collision در فایل‌های مستقیم چه مفهومی دارد؟

۱) عمل بازنویسی مجدد رکوردها

۲) عمل ایجاد فایل جدید برای رکوردهای به هنگام شده

۳) اگر خروجی تابع Hashing مقادیر مشابه را تولید نماید Collision رخ می‌دهد.

۴) اگر خروجی تابع Hashing مقادیر مشابه را تولید ننماید Collision رخ می‌دهد.

۴۶ ـ کدام یک از ساختارهای زیر سریع‌تر از سایر ساختارها می‌باشد؟

۱) ساختار ترتیبی شاخص‌دار

۲) ساختار ترتیبی شاخص‌دار مجهز به شاخص‌های ثانویه

۳) ساختار پایل (برهم)

۴) ساختار مبنایی فایل مستقیم

۴۷ ـ فایل وارون، فایلی است که.....

۱) فقط یک شاخص دارد.

۲) روی برخی از صفات خاصه آن شاخص دارد.

۳) روی تمام صفات خاصه آن شاخص دارد.

۴) روی هیچکدام از صفات خاصه آن شاخص ندارد.

۴۸ ـ در یک فایل پایل (pile) عملیات لازم برای انجام عمل درج به ترتیب کدام است؟

۱) انتقال رکورد از ناحیه کاری برنامه به بلاک ـ خواندن آخرین بلاک فایل که سیستم آدرس آن را دارد ـ بازنویسی بلاک

۲) بازنویسی بلاک ـ انتقال رکورد ازناحیه کاری برنامه به بلاک ـ خواندن آخرین بلاک فایل که سیستم آدرس آن را دارد.

۳) خواندن آخرین بلاک فایل که سیستم آدرس آن را دارد ـ انتقال رکورد از ناحیه کاری برنامه به بلاک ـ بازنویسی بلاک

۴) خواندن آخرین بلاک فایل که سیستم آدرس آن را دارد ـ بازنویسی بلاک ـ انتقال رکورد از ناحیه کاری برنامه به بلاک

۴۹ ـ در کدام روش بازیابی اطلاعات در محیط‌های DMS و DBMS در خواست به صورت مقدار صفت خاصه کلیدی است؟

۱) درخواست بولی

۲) درخواست ساده

۳) درخواست طیفی

۴) درخواست محاسباتی

۵۰ ـ در یک فایل شاخص‌بندی شده تعداد 3×10^5 رکورد داریم. تعداد مدخل‌ها در سطح اول شاخص و حافظه مصرفی آن به ترتیب از راست به چپ کدام است؟

$v = 40byte, p = 10byte$ و طول بلاک =4000 byte

طول رکورد =400 byte

۱) 3×10^4 byte ,15×10^5

۲) 15×10^5 byte , 20×10^4

۳) 15×10^5 byte,10

۴) 15×10^5 byte ,3×10^4

۵۱ ـ فایلی را در نظر بگیرید با 10000 رکورد 80 بایتی روی نواری به چگالی 1600bpi ، اگر $B_f = 50$ و طول IBG = 0.5 باشد، میزان واقعی استفاده از نوار کدام است؟

۱) 0.83

۲) 0.86

۳) 0.87

۴) 0.90

۵۲ـ زمان خواندن مستقیم یک بلاک 2 ثانیه است. در صورتی که طول بلاک B=10000 byte و مقدار حافظه هرز بلاکی 2000 byte باشد، نرخ انتقال واقعی چند K byte/sec می‌باشد؟

۱) 0.309 ۲) 3.9

۳) 4.048 ۴) 4000

۵۳ـ در عمل درج در یک فایل شاخص‌دار ترتیبی، چند عمل بازنویسی انجام می‌شود؟

۱) 1 ۲) 2

۳) 3 ۴) به دلیل مرتب بودن فایل عمل بازنویسی نداریم.

۵۴ـ دید سیستم فایل منطقی نسبت به یک فایل کدام عبارت است؟

۱) فایل از تعدادی باکت تشکیل شده است که تقسیماتی مثل پاکت، خوشه در آن دیده می‌شود.

۲) قالبی است با یک ساختار مشخص که شامل تعدادی رکورد است و هر رکورد دارای طول معینی است.

۳) مجموعه‌ای از رکوردهای ذخیره شده می‌باشد که یک ساختار مشخص داشته و دستیابی به آن با یک شیوه مشخص است.

۴) مقدار داده‌ای است که در یک عمل I/O بین بیرون و درون ماشین مبادله می‌شود.

سوالات دولتی ۸۳

۱ ـ گزینه نادرست کدام است؟

۱) جستجو در سطح فایل T.I.F براساس Binary Search است.

۲) عمل جستجو روی صفات خاصه غیر کلید به صورت جستجوی خطی است.

۳) فایل ترتیبی دارای عدم تقارن است.

۴) T.I.F نوعی فایل برهم است.

۲ ـ در بررسی که از سطح دیسک انجام شد موارد زیر مشاهده شده است: 4000 byte = طول بلاک، 90 ms = متوسط زمان استوانه‌جوئی، $b_{\mu} = 100$ m sec و rpm = 3000، نرخ انتقال واقعی چند کیلوبایت برثانیه است؟

۱) 10 ۲) 20 ۳) 40 ۴) 200

۳ ـ در محاسبه فاکتور بلاک‌بندی روی یک رسانه ذخیره‌سازی طول بلاک 2000 byte و طول رکورد 100 byte است در صورتیکه رکوردها بلاک بندی‌شان به صورت متغیر و دوپاره باشد مقدار B_f کدام است در صورتیکه طول فیلد نشانه‌رو 10 بایت باشد؟

۱) 18 ۲) 20 ۳) 22 ۴) 23

۴ ـ اگر سرعت حس نوک پک نوار برابر 150 اینچ برثانیه و زمان رسیدن به این سرعت 2 میلی ثانیه باشد، IBG چند اینچ است؟

۱) 0.3 ✓ ۲) 0.75 ۳) 7.5 ۴) 300

۵ ـ اگر طول رکورد در فایلی 30 بایت و طول سکتور 256 بایت و $B_f = 1$ باشد میزان واقعی استفاده از دیسک چند درصد است؟

۱) 93 ۲) 78 ۳) 23 ۴) 12

۶ ـ در کدام نوع از حافظه‌ها تعداد استوانه‌ها برابر یک است؟

۱) دیسک پک (DISK PACK) ۲) دیسک مغناطیسی

۳) طبله مغناطیسی (DRUM) ۴) دیسک نوری

۷ ـ کدام مورد کاهش زمانه استوانه‌جویی را باعث نمی‌شود؟

۱) استفاده از دیسک‌های با بازوی ثابت

۲) استفاده از الگوریتم‌های مناسب برای حرکت دادن بازوی دیسک

۳) توزیع کردن فایل‌ها روی چند دیسک

۴) جای دادن بلاک‌ها روی شیار بدون رعایت ترتیب پردازش آن‌ها

۸ ـ کدام سطح وظیفه تبدیل آدرس در برنامه پردازشگر (RBA) را به عهده دارد؟

۱) سطح برنامه پردازشگر فایل ۲) سطح خارجی سیستم فایل

۳) سطح فیزیکی سیستم فایل ۴) سطح منطقی سیستم فایل

۹ ـ بهترین ساختمان داده برای ایجاد فایل شاخص در مدل Multi Indexed کدام است؟

۱) Heap tree
۲) Linked list
۳) B ـ Tree
۴) AVL tree

۱۰ ـ فایلی داریم با یک میلیون رکورد اگر داشته باشیم: بایت P = 6، بایت V = 14، بایت B = 2000، بایت R = 200 حافظه مصرفی برای سطح اول شاخص (در فایل با ساختار ترتیبی شاخص‌دار) چند بایت است؟

۱) 5×10^4
۲) 16×10^4
۳) 20×10^5
۴) 8×10^6

۱۱ ـ تعداد کل سطوح در فایلی با مشخصات زیر کدام است؟ $P = 4 , V = 16 , n = 10^6 , B = 2000 , R = 200$

۱) ۱
۲) ۲
۳) ۳
۴) ۴

۱۲ ـ در یک فایل مستقیم باکت‌بندی شده که اندازه باکت معین و مشخص است درچه صورت کارایی بهتر می‌شود؟

۱) طول رکوردها برابر باکت باشد.
۲) طول رکوردها تا حد ممکن کوچکتر باشد.
۳) طول رکوردها تا حد ممکن بزرگتر باشد.
۴) طول رکوردها متغیر باشد.

۱۳ ـ حداقل و حداکثر تعداد کلیدهای یک نود درخت B (به جز ریشه درخت) از مرتبه m به ترتیب کدام است؟

۱) $m + 2 , m - 2$
۲) $2 * m , m / 2$
۳) $m , m - 2$
۴) $m , m - 2$

۱۴ ـ فایل پایلی شامل 450 بلاک بوده و هر بلاک برابر 2400 بایت است. چنانچه نرخ انتقال 3000 بایت در ثانیه باشد، زمان بدست آوردن رکورد بعدی (TN) چند دقیقه خواهد بود؟

۱) 2
۲) 3
۳) 4
۴) 6

۱۵ ـ یک فایل ترتیبی شامل 900 رکورد می‌باشد، هر بلاک در این مجموعه شامل چندین رکورد باشد تا جستجوی بلاکی با سرعت بهتری انجام شود؟

۱) 30
۲) 90
۳) 100
۴) 440

۱۶ ـ در فایل‌های شاخص‌دار ترتیبی زمان بازنویسی بلوک‌های شاخص کدام است؟

N طول ناحیه اصلی (رکوردها)
t' نرخ انتقال
T_N زمان بازیابی رکورد بعدی
O طول ناحیه سرریزی (رکوردها)
R فضای متوسط رکورد
Tser زمان خواندن فایل به صورت سریالی
d رکوردهای حذف شده
T_F زمان واکشی

۱) $\frac{sindex}{t'}$
۲) $T_F + (n + o' - 1)T_N$
۳) $(n + o')\frac{R}{t'} + \frac{sindex}{t'}$
۴) $Tser + (n + o - d)\frac{R}{t'} + \frac{sindex}{t'}$

۱۷ ـ در عمل بازیابی رکورد بعدی در فایل‌های شاخص‌دار ترتیبی کدام حالت دارای لوکالیتی بالاتر است؟

۱) رکورد فعلی در بلاکی از ناحیه سرریزی و رکورد بعدی در بلاکی از ناحیه اصلی قرار دارد.

۲) رکورد فعلی در بلاکی از ناحیه اصلی و رکورد بعدی در همان بلاک و بلاک در بافر است.

۳) رکورد فعلی در بلاکی از ناحیه سرریزی و رکورد بعدی هم در بلاکی از ناحیه سرریزی از همان استوانه است.

۴) رکورد فعلی آخرین رکورد بلاک از آخرین بلاک استوانه و رکورد بعدی در بلاک بعدی از استوانه دیگر است.

۱۸ ـ کدام گزینه نشان دهنده اعمال اساسی در محیط فیزیکی برای یک سیستم فایل است؟

۱) مکان‌یابی، باز کردن و خواندن از رسانه

۲) مکان‌یابی، خواندن بلاک‌ها و نوشتن بر روی بلاک‌ها

۳) مکان‌یابی، خواندن از رسانه، نوشتن بر روی رسانه

۴) باز کردن فایل، خواندن فایل، نوشتن بر روی فایل

۱۹ ـ در صورتیکه امکان همروندی عملیات CPU و عملیات پردازش ورودی و خروجی نداشته باشد و با شروع پردازش محتوای بافر آغاز بلاک بعدی در اثر دوران دیسک از زیر نوک R / W رد شود نرخ انتقال واقعی (t') عبارتست از:

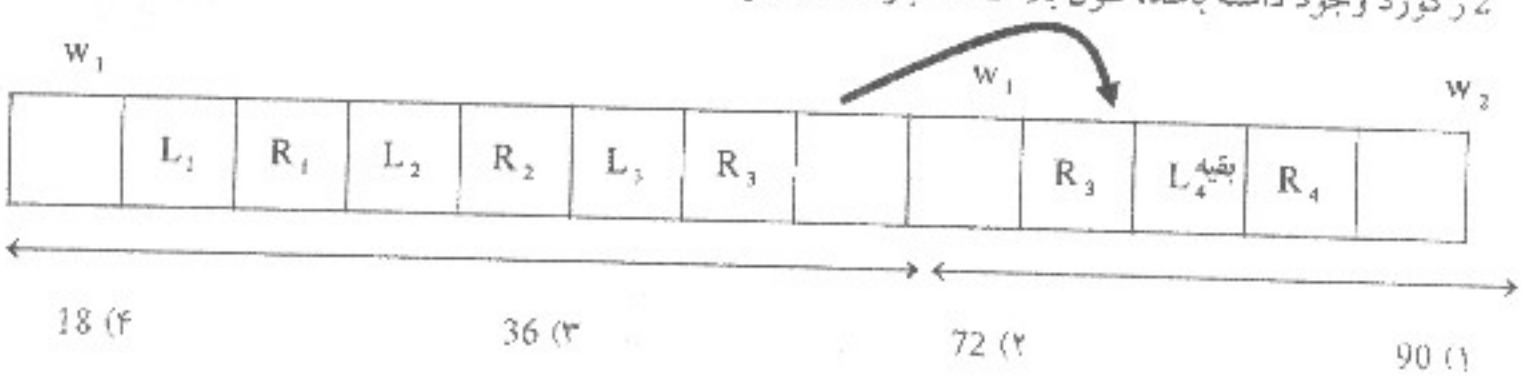
۱) $t' = \frac{B}{2r + b_g}$ (r دورنگ دورانی، B طول بلاک، b_g انتقال بلاکی)

۲) $t' = \frac{T_F \cdot B}{4r}$ (r دورنگ دورانی، B طول بلاک، T_F فاکتور تراکنش)

۳) $t' = \frac{B}{s + r + b_g}$ (r دورنگ دورانی، B طول بلاک، s متوسط زمان استوانه‌جویی، b_g انتقال بلاک)

۴) $t' = \frac{1}{2} \times \frac{B + G}{b_g}$ (G طول گپ، b_g انتقال بلاک، B طول بلاک)

۲۰ ـ شکل زیر نشان‌دهنده بلاک‌بندی رکوردها در بلاک‌های B_1 و B_2 است، فضای مفداری هر رکورد (طول بخش داده‌ای + طول بخش غیرداده‌ای) به صورت $R_1 = 10 , R_2 = 5 , R_3 = 10 , R_4 = 11$ است، در صورتیکه که طول فیلد نشانه‌رو 6 بایت و در هر بلاک تعداد 2 رکورد وجود داشته باشد، طول بلاک B عبارت است از:



۱) 90 ۲) 72 ۳) 36 ۴) 18

۲۱ ـ کدام نوع بافرینگ به صورت چرخشی پیاده‌سازی می‌شود؟

۱) single Buffering ۲) Double Buffering ۳) Multiple Buffering ۴) هیچکدام

۲۲ ـ در یک نوار مغناطیسی در صورتیکه ظرفیت اسمی نوار 2500 بایت و چگالی نوار $250\frac{byte}{ft}$ باشد در صورتیکه فضای داده‌ای بلاک 200 بایت و مقدار IBG = 50 بایت باشد، مزان واقعی استفاده از نوار کدام است؟

۱) 2000 ۲) 2500 ۳) 50000 ۴) 500000

۲۳ ـ هنگامی که به دلیل وجود محدودیت در تخصیص حافظه نمی‌توان به فایل دو بافر اختصاص داد، کدام تکنیک در کاهش زمان درنگ دورانی بهینه است؟

۱) تغییر مکان نقطه آغاز شیارها ۲) دیسک‌های با بازوی ثابت

۳) پراکنده خوانی ۴) تداخل بلاک‌ها

۲۴ ـ در یک فایل ترتیبی (sequential File) در صورتیکه که ضریب بلاک‌بندی 2 و تعداد رکوردها 16 باشد، دفعات مراجعه به فایل در عمل جستجو عبارت است از:

۱) 3 ۲) 5 ۳) 16 ۴) 64

۲۵ ـ شاخص غیرمتراکم عبارت است از:

۱) مدخل شاخص که به رکورد اشاره می کند.

۲) مدخل شاخص که به بزرگترین مقدار صفت خاصه اشاره می کند.

۳) مدخل شاخص که به مجموعه‌ای از رکوردها اشاره می کند.

۴) مدخل شاخص که به کوچکترین مقدار صفت خاصه اشاره می کند.

۲۶ ـ یک فایل چند شاخصی (multi indexed file) در حالت خاص کدام نوع فایل است؟

۱) پایل ۲) ترتیبی ۳) سریالی ۴) شاخص دار

۲۷ ـ در سطح رسانه ذخیره‌سازی مثل دیسک در صورتیکه که رکورد مورد نظر رکورد 21ام باشد و ظرفیت رکورد متوسط $B = 1000 , R = 500$ (ظرفیت بلاک) شمارهٔ بلاک مورد نظر BLK # در حالتی که $\begin{cases} RBA\ begin\ of\ device = 5 \\ b_i = 3 \end{cases}$ باشد کدام است؟

۱) 1 ۲) 2 ۳) 10 ۴) 12

پاسخ تشریحی مجموعه سوالات کنکوری

۱ ـ گزینه ۳ صحیح می‌باشد.

۲ ـ گزینه ۴ صحیح می‌باشد.

۳ ـ هیچکدام 14697s

۴ ـ گزینه ۱ صحیح می‌باشد.

۵ ـ گزینه ۲ صحیح می‌باشد.

۶ ـ گزینه ۴ صحیح می‌باشد.

۷ ـ گزینه ۱ صحیح می‌باشد.

۸ ـ گزینه ۳ صحیح می‌باشد.

۹ ـ گزینه ۳ صحیح می‌باشد.

۱۰ ـ گزینه ۴ صحیح می‌باشد.

۱۱ ـ گزینه ۴ صحیح می‌باشد.

۱۲ ـ گزینه ۱ صحیح می‌باشد.

۱۳ ـ گزینه ۲ صحیح می‌باشد.

۱۴ ـ گزینه ۳ صحیح می‌باشد.

۱۵ ـ گزینه ۳ صحیح می‌باشد.

۱۶ ـ گزینه ۲ صحیح می‌باشد.

۱۷ ـ گزینه ۲ صحیح می‌باشد.

۱۸ ـ گزینه ۱ صحیح می‌باشد.

۱۹ ـ گزینه ۳ صحیح می‌باشد.

۲۰ ـ گزینه ۲ صحیح می‌باشد.

۲۱ ـ گزینه ۱ صحیح می‌باشد.

۲۲ ـ گزینه ۴ صحیح می‌باشد.

۲۳ ـ گزینه ۲ صحیح می‌باشد.

۲۴ ـ گزینه ۱ صحیح می‌باشد.

۲۵ ـ گزینه ۲ صحیح می‌باشد.

۲۶ ـ گزینه ۳ صحیح می‌باشد.

۲۷ ـ گزینه ۱ صحیح می‌باشد.

۲۸ ـ گزینه ۲ صحیح می‌باشد.

۲۹ ـ گزینه ۴ صحیح می‌باشد.

۳۰ ـ گزینه ۴ صحیح می‌باشد.

۳۱ ـ گزینه ۱ صحیح می‌باشد.

۳۲ ـ گزینه ۴ صحیح می‌باشد.

۳۳ ـ گزینه ۳ صحیح می‌باشد.

۳۴ ـ گزینه ۲ صحیح می‌باشد.

۳۵ ـ گزینه ۱ صحیح می‌باشد.

۳۶ ـ گزینه ۱ صحیح می‌باشد.

۳۷ ـ گزینه ۱ صحیح می‌باشد.

۳۸ ـ گزینه ۳ صحیح می‌باشد.

۳۹ ـ گزینه ۳ صحیح می‌باشد.

۴۰ ـ گزینه ۲ صحیح می‌باشد.

۴۱ ـ گزینه ۱ صحیح می‌باشد.

۴۲ ـ گزینه ۱ صحیح می‌باشد.

۴۳ ـ گزینه ۲ صحیح می‌باشد.

۴۴ ـ گزینه ۱ صحیح می‌باشد.

۴۵ ـ گزینه ۳ صحیح می‌باشد.

۴۶ ـ گزینه ۴ صحیح می‌باشد.

۴۷ ـ گزینه ۴ صحیح می‌باشد.

۴۸ ـ گزینه ۳ صحیح می‌باشد.

۴۹ ـ گزینه ۳ صحیح می‌باشد.

۵۰ ـ گزینه ۲ صحیح می‌باشد.

۵۱ ـ گزینه ۴ صحیح می‌باشد.

۵۲ ـ گزینه ۱ صحیح می‌باشد.

۵۳ ـ گزینه ۲ صحیح می‌باشد.

۵۴ ـ گزینه ۲ صحیح می‌باشد.

۵۵ ـ گزینه ۳ صحیح می‌باشد.

پاسخنامه تشریحی ۸۳

۱ ـ گزینه ۱ صحیح می‌باشد.

۲ ـ گزینه ۲ صحیح می‌باشد.

۳ ـ گزینه ۱ صحیح می‌باشد.

۴ ـ گزینه ۱ صحیح می‌باشد.

۵ ـ گزینه ۱ صحیح می‌باشد.

۶ ـ گزینه ۳ صحیح می‌باشد.

۷ ـ گزینه ۴ صحیح می‌باشد.

۸ ـ گزینه ۴ صحیح می‌باشد.

۹ ـ گزینه ۳ صحیح می‌باشد.

۱۰ ـ گزینه ۳ صحیح می‌باشد.

۱۱ ـ گزینه ۴ صحیح می‌باشد.

۱۲ ـ گزینه ۲ صحیح می‌باشد.

۱۳ ـ گزینه ۳ صحیح می‌باشد.

۱۴ ـ گزینه ۲ صحیح می‌باشد.

۱۵ ـ گزینه ۱ صحیح می‌باشد.

۱۶ ـ گزینه ۱ صحیح می‌باشد.

۱۷ ـ گزینه ۲ صحیح می‌باشد.

۱۸ ـ گزینه ۳ صحیح می‌باشد.

۱۹ ـ گزینه ۱ صحیح می‌باشد.

۲۰ ـ گزینه ۳ صحیح می‌باشد.

۲۱ ـ گزینه ۳ صحیح می‌باشد.

۲۲ ـ گزینه ۱ صحیح می‌باشد.

۲۳ ـ گزینه ۴ صحیح می‌باشد.

۲۴ ـ گزینه ۱ صحیح می‌باشد.

۲۵ ـ گزینه ۳ صحیح می‌باشد.

۲۶ ـ گزینه ۱ صحیح می‌باشد.

۲۷ ـ گزینه ۲ صحیح می‌باشد.

پاسخ تشریحی ۵۰٪ اول

۱ـ گزینه ۲ صحیح می‌باشد.

۲ـ گزینه ۳ صحیح می‌باشد.

۳ـ گزینه ۲ صحیح می‌باشد.

$$\left.\begin{array}{l} v_0 = 60\ in/sec \\ IBG = 0.15_{\ in} \end{array}\right\} \Rightarrow t_0 = \frac{IBG}{v_0} \Rightarrow t_0 = \frac{0.15_{\ in}}{60\ in/sec} = 0.0025\ sec = 2.5\ ms$$

۴ـ گزینه ۳ صحیح می‌باشد.

$$\text{درصد استفاده واقعی} = \frac{B}{B+G} \times 100$$

$$G = 0.6_{\ in} \Rightarrow G = 0.6_{\ in} \times 1800\ byte/in = 1080\ byte$$

$$D = 1800\ bpi$$

$$B = 1400\ byte \Rightarrow \text{درصد استفاده واقعی} = \frac{1400}{1400 + 1080} \times 100 = \%\ 56$$

۵ـ گزینه ۱ صحیح می‌باشد.

۶ـ گزینه ۱ صحیح می‌باشد.

۷ـ گزینه ۳ صحیح می‌باشد.

۸ـ گزینه ۲ صحیح می‌باشد.

۹ـ گزینه ۱ صحیح می‌باشد.

$$\left.\begin{array}{l} n = 1000 \\ R = 80\ byte \end{array}\right\} \Rightarrow \text{ظرفیت فایل} = n \times R = 1000 \times 80 = 80000\ byte$$

$$\text{ظرفیت اسمی نوار} = L \times D \Rightarrow 80000\ byte = L \times 1600\ \frac{byte}{in} \Rightarrow L = 500_{\ in}$$

(D: چگالی، L: طول نوار)

۱۰ـ گزینه ۱ صحیح می‌باشد.

۱۱ـ گزینه ۲ صحیح می‌باشد.

۱۲ـ گزینه ۱ صحیح می‌باشد.

۱۳ـ گزینه ۴ صحیح می‌باشد.

۱۴ـ گزینه ۳ صحیح می‌باشد.

۱۵ـ گزینه ۳ صحیح می‌باشد.

۱۶ ـ گزینه ۴ صحیح می‌باشد.

۱۷ ـ گزینه ۴ صحیح می‌باشد.

۱۸ ـ گزینه ۴ صحیح می‌باشد.

۱۹ ـ گزینه ۱ صحیح می‌باشد.

ظرفیت فایل = طول فایل × چگالی

$$= \frac{10000 \times 80}{1600} = 500 \text{ inch}$$

۲۰ ـ گزینه ۴ صحیح می‌باشد.

۲۱ ـ گزینه ۲ صحیح می‌باشد.

۲۲ ـ گزینه ۳ صحیح می‌باشد.

۲۳ ـ گزینه ۴ صحیح می‌باشد.

۲۴ ـ گزینه ۱ صحیح می‌باشد.

$B = B_f \times R = 50 \times 80 = 4000 \text{ byte}$

$G_{byte} = G_{inch} \times \text{چگالی} = 0.5 \times 1600 = 800 \text{ byte}$

$$\text{درصد واقعی از نوار} = \frac{B}{B + G} \times 100 = \frac{4000}{4000 + 800} \times 100 = \frac{40}{48} \times 100 = \frac{5}{6} \times 100 = \%83$$

۲۵ ـ گزینه ۳ صحیح می‌باشد.

۲۶ ـ گزینه ۱ صحیح می‌باشد.

چنانچه رکورد بعدی در استوانه فعلی باشد، زمان استوانه‌جوئی برای یافتن بلاک بعدی صفر خواهد بودولی در سایر گزینه‌ها $s > 0$ (زمان استوانه‌جوئی) است.

۲۷ ـ گزینه ۴ صحیح می‌باشد.

شرط کارائی بافرینگ مضاعف $C_B < B_{tt}$ (زمان پردازش محتوی بافر از زمان انتقال بلاک به بافر کمتر باشد) است، با توجه به آنکه زمان و سرعت رابطه معکوس با یکدیگر دارند، سپس هرچه سرعت بیشتر باشد زمان کمتر خواهد بود و بالعکس.

۲۸ ـ گزینه ۱ صحیح می‌باشد.

زمان استوانه‌جوئی ارتباطی با طول گپ ندارد.

۲۹ ـ گزینه ۲ صحیح می‌باشد.

۳۰ ـ گزینه ۴ صحیح می‌باشد.

۳۱ ـ گزینه ۱ صحیح می‌باشد.

۳۲ـ گزینه ۴ صحیح می‌باشد.

هرچه لوکالیتی بیشتر باشد، میزان همسایگی فیزیکی رکوردهائی که منطقاً همجوار هستند بیشتر شده و زمان پردازش سریال کمتر می‌شود.

۳۳ـ گزینه ۳ صحیح می‌باشد.

۳۴ـ گزینه ۳ صحیح می‌باشد.

۳۵ـ گزینه ۳ صحیح می‌باشد.

۳۶ـ گزینه ۳ و ۴ صحیح می‌باشد.

۳۷ـ گزینه ۴ صحیح می‌باشد.

$L_d =$ چگالی لود اولیه (پیش فرض 1 در نظر گرفته می‌شود.)

در صورتیکه:

$b =$ (تعداد بلاک‌ها در فایل) $\Rightarrow \left\lceil \frac{n}{B_f \times L_d} \right\rceil$

$n =$ تعداد رکوردهای فایل

$B_f =$ ضریب بلاک‌بندی = تعداد رکوردهای هر بلاک

در این مسئله $n = 10^4$ رکورد و $B_f = 10$ است اما چون $L_d = \%\ 60$ است پس B_f واقعی $10 \times 60\ \% = 6$ می‌باشد. پس خواهیم داشت:

$$b = \left\lceil \frac{n}{B_f \times L_d} \right\rceil = \left\lceil \frac{10^4}{10 \times \%\ 60} \right\rceil = 1667$$

۳۸ـ گزینه ۲ صحیح می‌باشد.

در روش SSTF ، حرکت نوک خواندن / نوشتن همیشه در جهت شیاری است که کمترین فاصله را با شیار جاری داشته باشد. بنابراین به صورت زیر عمل می‌شود.

17 (جاری) $\overline{\ :\ }$ 16 , 15 , 7 , 3 , 29

۳۹ـ گزینه ۱ صحیح می‌باشد.

۴۰ـ گزینه ۲ صحیح می‌باشد.

۴۱ـ گزینه ۴ صحیح می‌باشد.

۴۲ـ گزینه ۳ صحیح می‌باشد.

۴۳ـ گزینه ۳ صحیح می‌باشد.

۴۴ـ گزینه ۱ صحیح می‌باشد.

۴۵ـ گزینه ۴ صحیح می‌باشد.

۴۶ـ گزینه ۴ صحیح می‌باشد.

۴۷ـ گزینه ۱ صحیح می‌باشد.

۴۸ ـ گزینه ۲ صحیح می‌باشد.

$$rba_{rec} = RBA_{BOF} + \left\lfloor \frac{(i-1)R}{B} \right\rfloor = 10 + \left\lfloor \frac{(8-1)\,500}{1000} \right\rfloor = 13$$

۴۹ ـ گزینه ۳ صحیح می‌باشد.

۵۰ ـ گزینه ۱ صحیح می‌باشد.

$$B_f = \frac{B - P}{R + P} = \frac{2000 - 10}{100 + 10} = \frac{1990}{110} = 18$$

۵۱ ـ گزینه ۱ صحیح می‌باشد.

$$B = B_f \times R = 1 \times 30 = 30 \text{ byte}$$

تعداد بلوک‌هایی که در یک سکتور جا می‌شوند $S_f = \left\lfloor \frac{\text{طول سکتور}}{\text{طول بلاک}} \right\rfloor = \left\lfloor \frac{256}{30} \right\rfloor = 8$

۵۲ ـ گزینه ۴ صحیح می‌باشد.

۵۳ ـ گزینه ۴ صحیح می‌باشد.

۵۴ ـ گزینه ۳ صحیح می‌باشد.

۵۵ ـ گزینه ۳ صحیح می‌باشد.

$$L_1 = L_2 = L_3 = L_4 = P = 6 \text{ byte}$$

$$\text{متوسط طول رکورد} = \frac{10 + 5 + 10 + 11}{4} = \frac{36}{4} = 9$$

$$\text{طول بلاک بدون احتساب فضاهای هرز} = 2(R + P) + P = 2(9 + 6) + 6 = 36$$

۵۶ ـ گزینه ۳ صحیح می‌باشد.

۵۷ ـ گزینه ۴ صحیح می‌باشد.

۵۸ ـ گزینه ۲ صحیح می‌باشد.

$b_i = 3$ یعنی هر شیار کامل 3 بلاک است.

$$RBA_{rec} = \left\lfloor \frac{(i-1)R}{B} \right\rfloor = \left\lfloor \frac{(21-1) \times 500}{1000} \right\rfloor = \frac{20}{2} = 10$$

$$BLK\# = \left(RBA_{rec} - RBA_{\text{begin of device}}\right) \bmod b_i = (10 - 5) \bmod 3 = 2$$

پیشنهاد ما برای شما: **DVD منابع کامپیوتر**

قیمت : ۵,۰۰۰ تومان

با خرید« DVD منابع کامپیوتر» در وقت و هزینه خود صرفه جویی کنید.

* بسته آموزشی منابع و تست های سال های گذشته همراه با حل تشریحی سوالات
* اسلایدهای آموزشی(PowerPoint) دروس مختلف جهت یادگیری بهتر دروس
* جزوات آموزشی پارسه دروس مختلف
* تمام منابع درسی و دانشگاهی فارسی همراه با جزوات اساتید دانشگاه های معتبر
* تمام منابع انگلیسی دروس دانشگاهی و کمیاب
* همراه با صدها تست از هر درس شامل تست های کنکور کاردانی به کارشناسی و کارشناسی ارشد سراسری و آزاد سال های گذشته
* همراه با کارنامه نفرات برتر و پذیرفته شدگان سال های گذشته
* نرم افزار کنکور آزمایشی، شبیه ساز آزمون کارشناسی ارشد برای مدیریت وقت و کاهش اضطراب و استرس
* به همراه صدها عنوان کتاب و مقالات آموزشی مختلف ...

در هر صورت شما برنده اید.

شما با خرید این محصول در وقتتان و هزینه تان صرفه جویی می کنید:

» چون جستجو و دانلود و جمع آوری این منابع در اینترنت به زمان و هزینه زیادی نیاز دارد.

» با فرض اینکه شما خط اینترنت پر سرعت هم داشته باشد حداقل چند ماه طول می کشد تا شما این منابع را دانلود کنید.(صرف نظر از مبالغی که باید برای هزینه اینترنت بپردازید)

» قیمت کتاب های منبع هم نیازی به یادآوری ندارد و شما با کمترین هزینه ، زحمت و نگرانی مجموعه کاملی از تمام کتابهای فارسی و انگلیسی را در اختیار دارید که حتی در صورت نیاز، هزینه چاپ و پرینت تمام صفحات یک کتاب بسیار کمتر از قیمت آن در فروشگاه های کتاب خواهد شد.

» شما مجموعه ایی از سوالات و تست های کنکور را در اختیار دارید که احتمال تکرار همان سوال ها و یا با کمی تغییر در آزمون های بعدی وجود خواهد داشت.

» شما با توجه به منابع و سوالات، بهتر خواهید توانست برای خود برنامه ریزی کنید و از وقت خود در بهترین حالت، یعنی یادگیری و تست زنی استفاده خواهید کرد.

برای سفارش و کسب اطلاعات بیشتر و مشاهده لیست تمام کتاب ها و منابع می توانید به آدرس اینترنتی زیر مراجعه فرمایید.

www.joyandeh.com

خرید پستی؛ ابتدا محصول را درب منزل دریافت کرده و سپس مبلغ سفارش را پرداخت نمائید.