*** به نام خداوند جان و خرد ***

جـــزوه آمـــوزشـــی

ذخیره و بازیابی اطلاعات

فصل اول

وسایل ذخیره و بازیابی اطلاعات

دیسک و نوار

حافظه: هر دستگاهی که قادر به نگهداری اطلاعات باشد به نحوی که استفاده از آن بتواند به اطلاعات مورد نیازش در هر لحظه که لازم باشد دستیابی داشته باشد.

خصوصیات حافظه:

1) **خواندن و نوشتن** : هر حافظه باید این قابلیت را داشته باشد که در آن بتواند اطلاعات را نوشت (درج کرد) و یا بتوان آن اطلاعات را خواند (بازیابی کرد) مانند هارد دیسک.
2) **قابلیت آدرس پذیری** : (به هر قسمتی از حافظه بتوانیم دستیابی داشته باشیم) هر حافظه را می توان با استفاده از مکانیزم آدرس دهی مورد دستیابی قرار داد. دستیابی ممکن است به منظور خواندن یا نوشتن در حافظه صورت گیرد. مدت زمانی را بین لحظه دستور خواندن یا نوشتن در حافظه و زمانی که داده دستیابی می شود را زمان دستیابی یا access time گفته می شود.
3) **آدرس پذیر بودن**: هر حافظه مجهز به یک مکانیزم آدرس است به تعبیر دیگر می توان به کمک این قابلیت به مکانی از حافظه دستیابی پیدا کرد.
4) **نرخ انتقال یا سرعت انتقال**: به میزان اطلاعاتی که در واحد زمان در حافظه قابل انتقال است نرخ انتقال گفته می شود.
5) **ظرفیت**: هر حافظه دارای ظرفیتی است که با بیت یا بایت یا با آدرس پارامترهای وابسته بیان می شود.
6) **مانا یا نامانا بودن اطلاعات**: برخی از حافظه ها اطلاعات را به صورت دائمی نگهداری می کنند (مانا) و برخی حافظه ها اطلاعات را به صورت موقت در خود نگهداری می کنند (نامانا).

سلسله مراتب حافظه:

حافظه های درون ماشین : ثبات ، حافظه نهان ، Ram

حافظه های برون ماشین : دیسک سخت ، دیسک نوری ، دیسک مغناطیسی

هرچه از پایین به بالا می رویم سرعت دستیابی افزایش و هزینه یک بیت نیز افزایش پیدا می کند.

هر چه از بالا به پایین می آییم ظرفیت ها بیشتر می شود.

نوار مغناطیسی :

رسانایی است از جنس پلاستیک یا غشای مغناطیسی ؛ اطلاعات بر روی شیارهایی که بر سطح نوار قرار گرفته اند ذخیره می شود. این نوار توسط دستگاه نوار خوان به حرکت در می آید . هر دستگاه نوار خوان دارای یک نوک (head) خواندن و نوشتن می باشد. نوار مغناطیسی اصولا برای پردازش ترتیبی یا پی در پی اطلاعات استفاده می گردد.

- اطلاعات در عرض نوار قرار می گیرند.

چگالی نوار (D):

تعداد بیت های قابل ضبط در هر اینچ را چگالی نوار می گویند و آن را با واحد بیت بر اینچ (bpi) تعریف می کند.

: GAP

فضای بلا استفاده (هرز- wast) بین دو رکورد یا دو بلاک که برای تنظیم سرعت حرکت هد خواندن و نوشتن استفاده می‌شود. به فضای هرز بین دو رکورد IRG (Inter Record Gap) . به فضای هرز بین دو بلاک IBG (Inter) Block Gap) گفته می شود.

پارامترهای اصلی نوار

سرعت : مسافت طی شده در واحد زمان (اینچ بر ثانیه)

چگالی : میزان اطلاعات ذخیره شده در ایچ

طول نوار : بر حسب ایچ یا فوت

نرخ انتقال : میزان اطلاعات منتقل شده در واحد زمان

ظرفیت اسمی : میزان گنجایش اعلام شده توسط کارخانه سازنده نوار که روی بسته قید شده است.

دیسک مغناطیسی :

نوار مغناطیس دارای محدودیت هایی است ، یکی از آن ها نوع قرار گرفتن داده ها بر روی نوار به صورت ترتیبی می باشد که امکان دسترسی مستقیم به اطلاعات را مقدور نمی سازد. دیسک مغناطیس رسانه ای است با امکان دستیابی مستقیم به داده های ذخیره شده.

دیسک مغناطیس شامل صفحه ای است دوار حول یک محور عمودی ، بر روی این صفحه دوایر با شیارهای متحدالمرکزی وجود دارد که داده ها به صورت سریالی در این شیارها قرار می گیرند. اطلاعات بر روی این شیارها توسط بازوی خواندن و نوشتن نوشته و یا از آن خوانده می شود.

تقسیمات دیسک:

استوانه یا سیلندر: شیارهای هم شعاع بر روی صفحه های هم مرکز بر روی رویه های مختلف را سیلندر گویند.

شیار یا track : به دوائر متحدالمرکز که اطلاعات بر روی آن ها در قالب صفر و یک ذخیره می شود شیار گفته می شود.

سکتور یا قطاع : تقسیمات از شیار به اندازه های مساوری را سکتور می گویند.

دو نوع سکتور وجود دارد:

1- **سکتور سخت افزاری** : توسط خود سیستم تعیین می شوند و ما دسترسی به آن ها نداریم.
2- **سکتور نرم افزاری** : ضریب صحیحی از سکتورهای سخت افزاری هستند که توسط کاربر و برنامه نویس تعیین می‌شود.

دسته بندی دیسک ها

از نظر جا به جا شدن : دیسک های ثابت و دیسک های جا به جا شدنی.

از نظر ثابت بودن یا متحرک بودن : هد خواندن و نوشتن که به دو دسته ی دیسک ها با نوک ثابت و دیسک ها با نوک متحرک تقسیم می شوند.

از نظر تعداد صفحات که روی صفحه عمود قرار می گیرند به دو دسته تقسیم می شوند : دیسک های تک صفحه ای و دیسک های چند صفحه ای ، به دیسک های چند صفحه ای اصطلاحا پک می گویند.

یک پک pack با n صفحه رویه دارای 2n رویه است . که 2- 2n رویه برای ذخیره سازی و دو رویه بالایی و پایینی برای حافظت و کنترل بیت ها استفاده می شود.

از نظر جنس صفحه : به دو دسته دیسک سخت (hard disk) و دیسک انعطاف پذیر یا نرم (floppy disk) تقسیم می شوند.

پارامترهای دیسک

زمان استوانه جوییS(seek time) : زمانی است که جهت رسیدن نوک خواندن و نوشتن و قرار گرفتن بر روی استوانه ای که اطلاعات مورد نظر ما بر روی آن قرار دارد صرف می شود و آن را با S نشان می دهند و واحد آن میلی ثانیه است.

زمان انتظار دورانی: زمانی است که پس از رسیدن نوک خواندن و نوشتن بر روی استوانه مورد نظر باید سپری شود تا داده‌ی مورد نظر به زیر نوک خواندن و نوشتن برسد و واحد آن میلی ثانیه است.

متوسط آن را باr نشان می دهیم که نصف زمان لازم برای یک دور دیسک 2r می باشد.$(0\leq r<2r)$

سرعت گردش دیسک : مقدار دورهای دیسک در دقیقه می باشد و باrpm(Rotation per minute) نمایش داده می شود و واحد آن دور به دقیقه است .

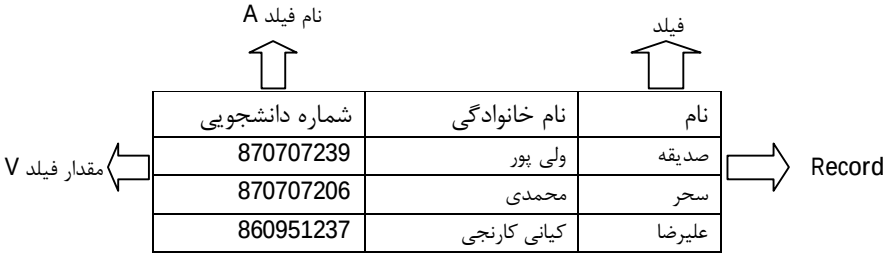
نرخ انتقال : تعداد بایتی است که در یک ثانیه به رسانه منتقل و یا از رسانه به سیستم منتقل می شود.

سرعت خواندن دسترسی مستقیم با نرخ انتقال دارد.

فصل دوم

رکورد ، بلاک و روش های بلاک بندی

همه ی اطلاعات در کامپیوتر به صورت فایل ذخیره می شوند.



دو نوع ذخیره سازی برای رکوردها(هر سطر یک رکورد نام دارد.) وجود دارد:

رکورد با طول ثابت

طول رکورد = تعداد فیلدها * متوسط اندازه مقادیر فیلدها

R=a*v

رکورد با طول متغییر :

برای ذخیره سازی هر فیلد باید نام آن فیلد را بیاوریم مثال (نام=مریم، نام خانوادگی = محمدی ، شماره دانشجویی = 870707239)

R=(A+V+2)*n

2 در این فرمول برای ذخیره هر فیلد یک = و یک ، جداکننده داریم.

دلایل متغییر شدن طول رکورد

1) ممکن است که طول برخی از فیلدها متغییر باشد . مثلا فیلد مربوط به نام و نام خانوادگی یا آدرس که می تواند طول متغییر داشته باشد.
2) ممکن است که برخی از فیلدها دارای چندین صفت خاصه باشند که در نمونه های مختلف آن تعداد مقادیر متفاوتی داشته باشد به عنوان نمونه دروس انتخاب شده در یک ترم در رکورد اطلاعات دانشجو ، که تعداد دروس برای فیلد ثبت نام متفاوت می باشد.
3) ممکن است برای رکورد ، حالات مختلفی از فیلدها داشته باشیم(تعداد فیلدها متغییر باشد.) به عنوان نمونه در رکورد اطلاعاتی مربوط به کارمندان یک سازمان ، فیلدهای کارمندان رسمی و کارمندان قرار دادی با هم متفاوت اند.

رکورد در محیط فیزیکی

بخش داده اي	بخش غير داده اي

- بخش غیر داده ای :به هر رکورد خود سیستم اطلاعاتی می دهد که اگر حذف شوند سیستم نمی داند چه رکوردی است ، بخش داده ای معنی است که برای کاربر می دهد و بخش غیر داده ای معنی است که برای سیستم می دهد.

در بخش داده ای : فایل ما قرار می گیرد.

در بخش غیر داده ای :

طول رکورد : هنگامی که رکوردها دارای طول متغییر هستند از این فیلد برای نگهداری و مشخص کردن محدوده یا طول بخش داده ای (اطلاعات مورد نظر ما جهت ذخیره سازی) می باشد ولی در رکوردهایی با طول ثابت به این فیلد نیازی نمی‌باشد.

نوع رکورد : برای فایل هایی که دارای چندین نوع رکورد می باشد از این فیلد جهت مشخص کردن نوع رکورد استفاده می‌شود.

پرچم های عملیاتی (flag) : یک نشانگر برای فیلدهاست ، عملیاتی که قرار است بر روی آن انجام گیرد یا انجام شده است را نگهداری می کند. مثلا در یک فایل ، عمل حذف به صورت منطقی انجام می گیرد. فلک مربوطه را علامت گذاری می کنیم تا بعد در هنگام یکپارچه سازی رکوردهای حذف شده منطقی را به صورت فیزیکی حذف کنیم.

اشاره گر ها: از این فیلد جهت ارتباط منطقی بین رکوردهای فایل استفاده می شود. در واقع از این فیلد در پیاده سازی ساختار منطقی فایل در محیط فیزیکی استفاده می شود . ساختار منطقی فایل ، همان دیدی است که کاربر نسبت به فایل دارد. یعنی یک مجموعه ای از رکوردها که با نظم خاصی و بر اساس صفت خاصی به طور مرتب ، پشت سر هم ذخیره شده اند ولی در هنگام ذخیره سازی رکوردها ، لزوما دارای همجواری فیزیکی نستند. لذا ما از این فیلد ، جهت برقراری ارتباط منطقی بین رکوردهای که از نظر فیزیکی همجوار نیستند استفاده می کنیم.

فصل سوم

تکنیک های بلاک بندی

بلاک بندی :

قرار دادن چندین رکورد را در قالب بزرگتر جهت عملیات خواندن و نوشتن از رسانه ذخیره سازی ، بلاک بندی می گویند. در واقع بلاک مجموعه ای است از تعدادی رکوردها با طول ثابت.

بلاک کمترین داده ای است که در عملیات خواندن و نوشتن به یا از دیسک منقل می شود. بین هر دو بلاک فضای هرزی وجود دارد که به آن IBG (Inter Block Gap) گفته می شود . ظرفیت بلاک را با B نشان می دهیم.

ضریب بلاک بندی (blocking factor) :

به تعداد رکوردهای موجود در بلاک بندی گفته می شود و آن را با Bf نمایش می دهیم.

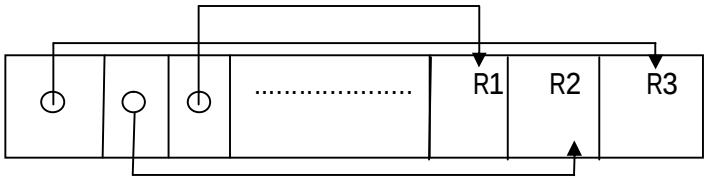
$$bf = [\frac{B}{R}]$$

B: اندازه بلاک

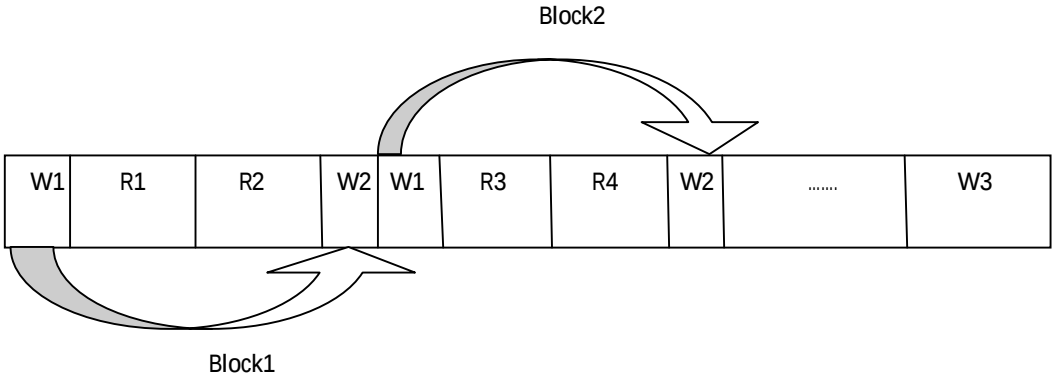
R: اندازه رکورد

تکنیک های انتهای رکوردهای متغییر

1) درج نشانگر در پایان رکورد (در آخر هر رکورد علامتی قرار می دهیم که نشانگر پایان آن می باشد.)
2) درج طول در بخش پیوندی (در ابتدای هر رکورد با طول هر رکورد مشخص می شود.)
3) جدول مکان نما (آدرس هر رکورد نسبت به ابتدای بلاک در جدولی ذخیره می شود.)



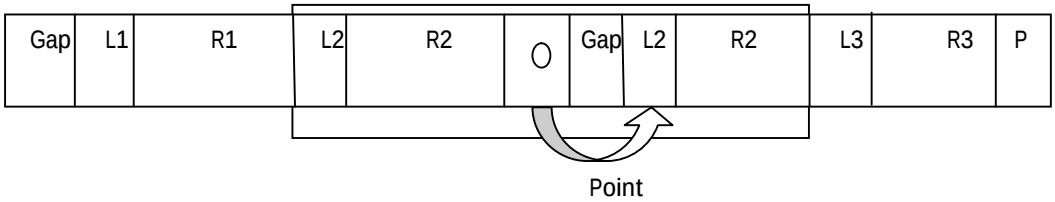
1) رکوردها با طول ثابت و یک پاره
   - Ø W1 : فضای هرز مربوط به Gap
   - Ø W2: فضای هرز ناشی از نگنجیدن آخرین رکورد (به علت ثابت بودن طول رکورد)
   - Ø W3 : فضای هرز ناشی از نگنجیدن آخرین بلاک در شیار



ضریب بلاک بندی ⟸ $bf = [\frac{B}{R}]$

2) بلاک بندی رکوردها ی با طول متغییر و دوپاره

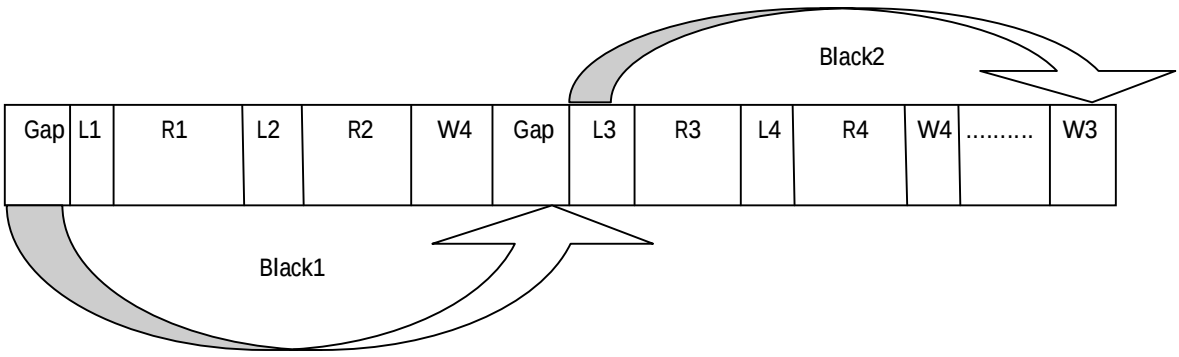
ü در ابتدای هر رکورد طول رکوردها ذخیره می شود.



ضریب بلاک بندی ⟸ $bf = [\frac{B-p}{R+p}]$

$$bf = [\frac{\text{اندازه بلاک} - point}{\text{طول رکورد} + \text{اندازه رکورد}}]$$

3)تکنیک بلاک بندی رکوردهای با طول متغییر و یکپاره



در این روش چون جای R3 در بلاک 1 وجود ندارد ، R3 به بلاک دوم رفته در نتیجه فضای هرزی به نام W4 به وجود می آید.

به طور متوسط W4 می تواند صفر یا به اندازه ی یک رکورد باشد.

ضریب بلاک بندی ⟸ $bf = [\frac{B-w4}{R+p}]$

مقایسه سه روش :

در روش اول پیاده سازی و مدیریت راحت تر است ولی در صورتی که طول رکوردها عوض نشود به همین دلیل روش اول فاقد انعطاف پذیری می باشد. اما در روش دوم و سوم نسبت به انعطاف پذیری بیشتری وجود دارد اما نرم افزار پیچیده تری نیاز می باشد.

روش سوم از نظر حافظه مقرون به صرفه تر می باشد ولی به نرم افزار پیچیده تری نیاز دارد ، در تکنیک های اول و دوم مشکلی که مطرح است این است که حداکثر طول رکورد به طول بلاک محدود می شود یعنی در مرحله ایجاد فایل باید حداکثر طول رکورد را برابر طول بلاک در نظر بگیریم در حالی که تکنیک دوم این محدودیت را ندارد و از این نظر انعطاف پذیر تر است. از نظر زمانی به علت این که روش سوم حافظه هرز بیشتری دارد طول فایل بیشتر از روش دوم می باشد. برای خواندن تمام فایل و یا پردازش آن زمان بیشتری صرف می کنیم. به عبارتی از روش سوم بیشتر از روش دوم برای خواندن فایل استفاده می کنیم.

مزایا و معایب بلاک بندی

مزایا :

1-کاهش دفعات وردی / خروجی و در نتیجه کاهش زمان برنامه

2-صرفه جویی در مصرف حافظه دیسک به علت کم شدن میزان Gap ها چرا که در صورت استفاده نکردن از بلاک بندی رکوردها دارای Gap بوده که سبب افزایش طول فایل و در نتیجه افزایش زمان پردازش خواهد شد.

3-مجموعه ی Gap های ما بسیار کم تر خواهد بود.

معایب:

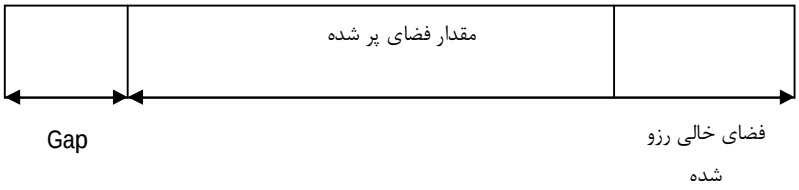
1-مصرف حافظه ی اصلی به علت نیاز به Buffering

2-کارنرم افزاری بیش تر برای بلاک بندی و بلاک گشایی (یک نرم افزار باید بتواند چندتا رکورد بلاک کند و یا آن را از حالت بلاک در بیاورد.)

فصل چهارم

چگالی لود اولیه و لوکالیتی

چگالی لود اولیه (loading Density)



چگالی لود اولیه ⟸ $\frac{ld}{B} < 1$ قسمت پر بلاک (LD) تقسیم بر کل بلاک .

مزایای در نظر گرفتن حافظه رزو در بلاک

1- این امر موجب افزایش لوکالیتی فایل می شود چرا که از پراکندگی نشست رکوردها روی دیسک تا حدودی جلوگیری می کند، این امر میزان پراکندگی در زمان دستیابی تصادفی و نیز زمان پردازش فایل را بهبود می‌بخشد.

2- موجب تسهیل در انجام برخی از عملیات بر روی فایل خواهد شد. مثلا در هنگام درج رکوردی که طول آن در اثر عملیات بهنگام سازی افزایش یافته دیگر نیاز به حذف منطقی از بلاک فعلی و درج در بلاک دیگر نخواهد بود و می توان با یک شیفت درون بلاکی رکورد تغییر یافته را در جای قبلی خود باز نویسی کرد.

معایب :

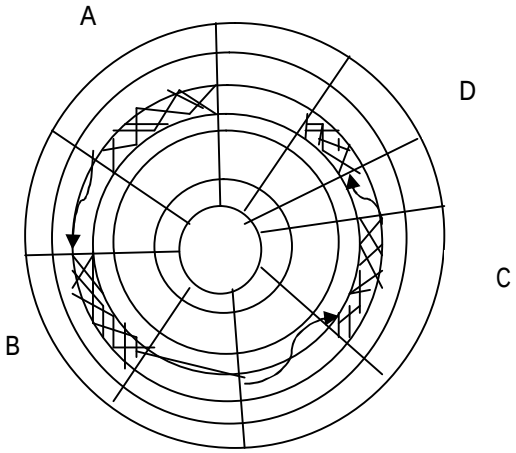
1- این حافظه در واقع نوعی حافظه هرز محسوب می شود که موجب افزایش اندازه حجم فایل خواهد شد و در نتیجه زمان خواندن کل فایل افزایش می یابد.

2- در صورتی که توزیع رکوردها یکنواخت نباشد ، موجب باقی ماندن حافظه هرز در انتهای بعضی از بلاک ها خواهد شد.

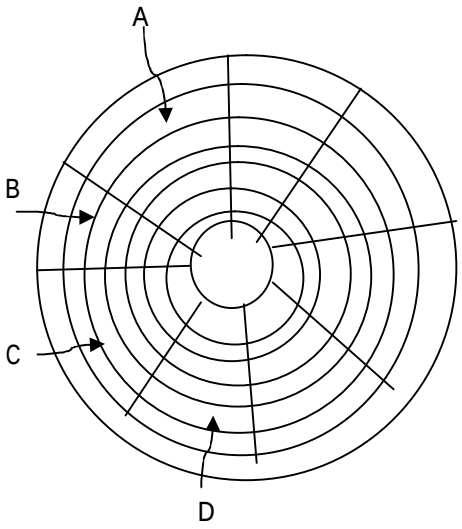
- هرچه چگالی لود اولیه کمتر باشد تعداد بلاک ها بیشتر و حجم فایل بیشتر می شود.

لوکالیتی (locality): میزان همسایگی (نزدیکی) فیزیکی رکورد منطقی بعدی به رکورد فعلی. به عبارتی شاخصی است که نشان می دهد رکوردهای منطقا هم جوار تا چه اندازه فیزیکی اً هم جوار اند.

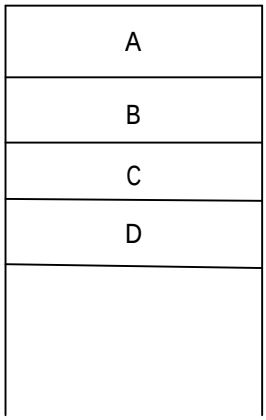
هر چه لوکالیتی رکوردها قوی تر باشد زمان پردازش سریال آن کمتر خواهد بود. چرا که دیگر در هنگام خواندن ، زمان کمتری برای انتقال هد دیسک به جلو و عقب برای خواندن رکورد بعدی صرف خواهد شد.



نا همجواري فيزيكي ركوردها



همجواري فيزيكي ركوردها



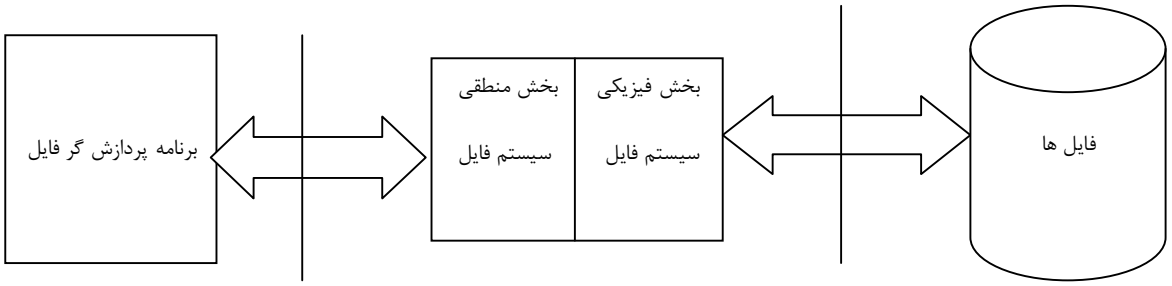
فايل منطقي و هم جواري ركوردها

سطوح آدرس دهی

الف) در سطح برنامه پردازش گر فایل

ب) در سطح سیستم فایل منطقی

ج) در سطح سیستم فیزیکی



محیط فیزیکی ذخیره سازی

سیستم فایل

سطح کاربردی

- بخش فیزیکی سیستم فایل با فایل در ارتباط است و بخش منطقی سیتم فایل با کاربر در ارتباط است.

آدرس دهی در سطح برنامه کاربردی (کاربر)

سه سطح آدرس دهی داریم:

1- نسبی 2-محتوایی 3- نمادی

در آدرس دهی نسبی کاربر فایل ، فایل را به صورت یک ساختار خطی مشاهده می کند که در آن هر رکورد دارای شماره است با شروع از صفر برای اولین رکورد فایل ، در این روش کاربر آدرس رکورد RRA (Relative Record Addres) (آدرس نسبی رکورد) را به عنوان آرگومان جستجو می دهد. در آدرس دهی محتوایی کاربر یک مقدار صفت خاصه را به عنوان پارامتر جستجو می کند. در آدرس دهی نمادی کاربر رکورد مورد نظرش را به کمک یک اسم سمبلیک مشخص می‌کند مثال برای آدرس دهی محتوایی : رکوردی را پیدا کن که شماره شناسنامه 4357 داشته باشد.

آدرس دهی در سطح سیستم فایل

بخش منطقی سیستم فایل کل فضای ذخیره سازی را به شکل آرایه ای از بلاک ها می بیند آدرس یا شماره بلاک را با RBA(Relative Block Address) نشان می دهند که آدرس نسبی بلاک اول برابر صفر در نظر گرفته می شود.

آدرس دهی نسبی بلاک حاوی رکورد آی ام نسبت به بلاک اول فایل ⟸ شماره بلاك آي ام

$$\left\lfloor \frac{(i-1)*R}{B} \right\rfloor$$

این آدرس نسبت به ابتدای فایل است. مثلا ادرس شماره 5 یعنی از ابتدای فایل رکورد پنجم

RBA مطلق : اگر RBA_R نسبی را بعلاوه RBA_B ابتدای فایل کنیم آدرس مطلق بدست می آید.

RBA_R ⟸ $RBA_B + \left[\frac{(i-1)*R}{B}\right]$

RBA_B: آدرس نسبی بلاک اول فایل (شماره اولین بلاک ابتدای فایل)

RBA_R : آدرس نسبی حاوی رکورد مورد نظر

آدرس دهی فیزیکی : یعنی شماره سیلندر، شماره استوانه و... چند است؟

فصل پنجم

ارزیابی پارامترهای دیسک و نوار

ظرفیت واقعی نوار

ظرفیت اسمی بیشتر از ظرفیت واقعی است زیرا به علت Gap ها یک سری حافظه را از دست می دهیم.

درصد استفاده واقعی نوار ⟸ $\frac{B}{B+G}$*100

ظرفیت اسمی نوار ⟸ L*D

ظرفیت واقعی نوار ⟸ $\frac{B}{B+G}$*L*D

ظرفیت واقعی دیسک:

نحوه فرمت بندی شیار در ارزیابی ظرفیت واقعی دیسک تاثیر بسزایی دارد. برای ارزیابی ظرفیت واقعی باید تمام حافظه های هرز (گپ) را با توجه به نحوه فرمت بندی شیار و تکنیک های بلاک بندی در سطح یک شیار از ظرفیت اسمی کم نمود. اندازه یک بلاک از جمله پارامترهای مهم در میزان استفاده واقعی از دیسک می باشد . هر چه طول بلاک کمتر در نظر گرفته شود ،تعداد بلاک ها افزایش پیدا می کند و در نتیجه تعداد گپ ها ی ایجاد شده در یک شیار افزایش خواهد یافت و از طرف دیگر اگر طول بلاک را بزرگ در نظر بگیریم باعث مشکل پارگی و همچنین باعث بروز حافظه هرز ناشی از نگنجیدن آخرین بلاک در شیار می شود.

تعداد سکتور در بلاک ⟸ $N = \left[\frac{R}{Ls}\right]$

LS : طول سکتور

درصد استفاده واقعی از دیسک ⟸ $E = \frac{R*Bf}{Ls*N}$*100

نرخ انتقال واقعی در دیسک (دستیابی تصادفی)

عوامل موثر در نرخ انتقال واقعی عبارتند:

1-سرعت انتقال بلاک به حافظه (مدت زمانی که طول می کشد یک بلاک خوانده شود Btt ($\frac{B}{t}$)که t نرخ انتقال اسمی بر حسب بایت در ثانیه است.

2-زمان استوانه جوییs

3-زمان درنگ دورانی r

S+r+btt برابر است با زمان خواندن یک بلاک به صورت دستیابی تصادفی یک بلاک.

نرخ انتقال واقعی ⟸ $\frac{B}{S+r+btt}$

بافر (Buffer)

ناحیه ای است از حافظه اصلی که جهت هماهنگی در عملیات ورودی/خروجی و عملیات سی پی یو بکار می رود. بافر ناحیه ای است که حداقل یک بلاک را بتواند در خود جای دهد که در هر بار عملیات خواندن و نوشتن از دیسک حداقل یک بلاک در بافر گذاشته و یا خوانده می شود. در صورتی که فایل را بلاک بندی کرده باشیم معمولا برای انجام عملیات ورودی و خروجی بر روی فایل ها از دو بافر ورودی و خروجی استفاده می شود.

- قسمتی از Ram است که جداگانه قرار می گیرد برای این که بتواند بین دستگاههای ورودی خروجی و سی پی یو یک هماهنگی سرعتی داشته باشد(یه قسمت از Ram را در نظر می گیریم که اطلاعات قبل از این که به سی پی یو بروند در آنجا کپی شوند.)

دلیل استفاده از بافر : چون سرعت دستگاههای ورودی/خروجی خیلی کمتر از سرعت سی پی یو است این عدم هماهنگی توسط بافر بتواند کنترل شود.

انواع بافر

1- **بافر نرم افزاری** : ناحیه ای است از حافظه اصلی که توسط سیستم عامل در اختیار برنامه ها قرار می گیرد.
2- **بافر سخت افزاری** : بافر موجود در دستگاههایی مانند کارت خوان ، چاپگر و .... استفاده می شود.این بافرها معمولا حجم زیادی از اطلاعات را می توانند ذخیره کنند این نوع بافر با سرعت دستگاههای ذخیره سازی پر شده و پس از آن که بافر تکمیل شد محتوای آن را با سرعت انتقال کانال به کامپیوتر و از آن جا به بافر نرم افزاری منتقل می شوند بافرهای سخت افزاری خود به چند دسته تقسیم می شوند.

الف) بافرینگ استاندارد (single buffering) : ما یک بافر داریم.

ب) بافرینگ مضاعف (double buffering): یعنی ما دو بافر داریم که یکی کمکی است. برای این که کارایی خودش را داشته باشد چیزی به نام شرط مضاعف داریم در غیر این صورت هیچ کارایی برای ما نخواهد داشت.

شرط مضاعف ⟸ cb<btt

Cb : زمان خواندن یک بلاک

ج) بافرینگ چندگانه (multiple buffering): یعنی ما چندتا بافر داریم.

فصل ششم

فایل پـــــایل

فایل پایل (pile)

فایلی است که در آن هیچ گونه نظم و ترتیبی وجود ندارد تنها بر اساس زمان ایجاد می شود. رکوردها در این نوع فایل بر اساس هیچ فیلدهایی مرتب نیستند در واقع در این نوع ساختار که ساده ترین ساختار جهت پیاده سازی می باشد ، رکوردها دارای طول متغییر هستند. تعداد فیلدها و همچنین مکان آن ها نیز در نمونه های مختلف رکوردها متفاوت می باشد. برای ساختن این فایل رکوردها بخش بندی نشده و هیچ استراتژی مشخصی برای دستیابی به رکوردها وجود ندارد.

کاربرد و موارد استفاده فایل پایل

- در محیط های عملیاتی که داده ها اساسا نظم پذیر نباشند، وجود نظم درجه ایمنی فایل را کاهش می دهد.
- در محیط هایی که فقط عملیات خواندن از ابتدا به انتهای فایل نیاز است.

متوسط اندازه رکورد(رکورد با طول متغییر)

زمان واکشی (TF)

واکشی رکورد دلخواه (fetch) : واکشی یک رکورد یک عمل محتوایی است که در آن یکی یا تعداد بیشتری از صفات خاصه رکورد به عنوان آرگومان جستجو مد نظر قرار داده می شود که لازمه این امر جستجو در فایل ، دستیابی به بلاک حاوی رکورد مورد نظر و خواندن آن می باشد که رسیدن به بلاک مورد نظر بستگی به ساختار فایل دارد. به دلیل بی نظم بودن رکوردها ممکن است که رکورد در اولین بلاک های فایل را با بلاک فرض کنیم که هرکدام B بایت حافظه اشغال می کند و تعاد کل رکوردها برابر N در نظر بگیریم آنگاه زمان واکشی TF از فرمول های زیر بدست می آید:

زمان واکشی برای بلاک $\Longleftarrow$ $TF = \frac{1}{2} b \frac{B}{tr}$

زمان واکشی برای رکورد $\Longleftarrow$ $TF = \frac{1}{2} n \frac{R}{tr}$

زمان بازیابی رکورد بعدی (TN)

واکشی رکورد بعدی (Get Next) : بدست آوردن رکورد بعدی عملی است که وابسته به ساختار فایل می باشد در حالی که واکشی یک رکورد (fetch) عملی است محتوایی. موقعیت رکورد بعدی نسبت به رکورد فعلی می تواند یکی از سه حالت زیر باشد.

1) با رکورد فعلی ارتباطی ندارد یعنی باید بر روی رکورد بعدی دوباره عمل جستجو انجام شود.
2) آدرس رکورد بعدی از رکورد فعلی بدست می آید یعنی از رکورد فعلی به بعدی اشاره گر وجود دارد.
3) رکورد بعدی همجوار فیزیکی با رکورد فعلی است.

در فایل پایل حالت اول را داریم پس $\Longleftarrow$ TN= TF

زمان درج رکورد (TI)

با توجه به ساختار بی نظم فایل پایل برای درج کردن یک رکورد در این فایل باید به انتهای فایل رفته و عمل درج انجام گیرد ، بنابراین ابتدا باید بلاک آخر فایل خوانده شده و بعد از درج رکورد در بلاک موجود در بافر بلاک را بازنویسی کنیم. برای رفتن به انتهای فایل و خواندن بلاک انتهایی زمان S+r+btt صرف می شود و بازنویسی بلاک را نیز TRW در نظر می‌گیریم. بنابر این

Ti=s+r+btt+Trw(read/write) + Trw=2r ⟹ Ti=s+3r+btt

زمان بهنگام سازی (Tu)

رکورد را به عنوان یک رکورد جدید در انتهای فایل درج کرده و آن را از جای قبلی اش حذف می کنیم.

Tu=Tf+Trw+Tr=Tf+2r+Tr

زمان حذف یک رکورد (TD):

برای حذف یک رکورد ابتدا باید آن را واکشی کرد پس علامت حذف منطقی را در آن قرار داده و دوباره رکورد را بازنویسی می کنیم چون در این حالت اندازه رکورد تغییری نمی یابد. بنابراین همان بهنگام سازی درجا خواهد بود. در نتیجه:

TD=Tf+Trw=Tf+2r

سازمان دهی مجدد (Ty)

سازمان دهی مجدد فایل (Restracturing): هر فایل پس از لود اولیه به علت انجام عملیات بازیابی یا ذخیره سازی دستخوش تغییراتی می شود که موجب کاهش کارایی اولیه آن خواهد شد.

دلایل سازمان دهی مجدد فایل عبارتند از:

- احیا نظم ساختار اولیه فایل
- از بین بردن حافظه های هرز موجود
- اصلاح استراتژی دستیابی (در فایل های شاخص دار)

Ty= (n+i) $\frac{R}{c'}$ + $(n+1-d)\frac{R}{c'}$

(n+i) $\frac{R}{c'}$ ⟸ خواندن کل فایل ، فایل اولیه به همراه رکوردهای اضافه شده جدید.

$(n+1-d)\frac{R}{c'}$ بازنویسی فایل بدون در نظر گرفتن رکوردهای حذف شده منطقی و همراه با رکوردهای درج شده بعد از لود اولیه.

در فرمول فوق مقدار کل رکوردها در لود اولیه برابر n فرض شده که از این رکوردها d تا رکورد ، حذف منطقی و تعداد i رکورد جدید نسبت به حالت اولیه در انتهای فایل درج شده اند.

فصل هفتم

فایل ترتیبی (sequential)

فایل ترتیبی (sequential)

معرفی فایل: بر اساس یک فیلد به نام کلید اصلی مرتب است(کلید اصلی ، اجزا و مقادیر آن غیر تکراری می باشد.) ویژگی کلید اصلی یکتایی مقدار آن است. این ساختار نسبت به ساختار فایل پایل دارای **مزایایی** است که عبارتند از:

- صرفه جویی در مصرف حافظه به خاطر عدم ذخیره سازی اسم صفت در رکورد
- ساده تر بودن قالب رکورد(بعلت این که طول رکورد ثابت است) ، به نحوی که رکورد ذخیره شده عملا نگاشتی از آنچه در برنامه پردازشگر است می باشد.
- نرم افزار ساده تر برای مدیریت و پردازش فایل
- وجود یک استراتژی دستیابی، که در این نوع ساختار شیوه دسترسی ترتیبی است(رکوردها بر اساس کلید اصلی مرتب می شوند.)
- پردازش سریال رکوردها سریع تر و راحت تر انجام می پذیرد.

معایب

- مصرف حافظه بیشتر به خاطر شکل ثابت و مکانی رکوردها (ممکن است در بعضی نمونه ها بعضی از فیلدها مقدار دهی نشوند.)
- وجود پدیده عدم تقارن (Asymmetry): زیرا فقط دسترسی اساس فیلد یا فیلد های کلید اصلی انجام می شوند و سایر فیلدها نقشی ندارند و در واقع استراتژی دستیابی متکی به کلید اصلی است.
- کاهش انعطاف پذیری ساختار

فایل تراکنش T.L.F

جهت بالا بردن سرعت عملیات و پردازش فایل عمل درج در فایل اصلی انجام نمی شود بلکه در یک فایل کمکی به نام ثبت تراکنش ها انجام می پذیرد که در آن رکوردهای اضافه شونده ی جدیدی درج خواهند شد. چرا که اگر قرار باشد رکورد در جای اصلی خود در فایل اولیه ذخیره گردد مجبوریم تعدادی عمل شیفت انجام دهیم که بسیار زمان بر است.مخصوصا وقتی که حجم فایل زیاد باشد ، بنابراین رکوردهای جدید را در انتهای فایل tlf ذخیره می کنیم (دقت داشته باشید که فایل tlf مرتب نمی باشد.) سپس بعد از هر چند وقت فایل اصلی را سازمان دهی مجدد کرده و آنگاه است که رکوردهای فایل tlf در جای اصلی خود قرار می گیرد.

اندازه رکورد: باید مقادیر را ذخیره کنیم.

زمان واکشی رکورد (بر اساس فیلد غیر کلید) دنبال رکوردی بگردیم که فیلد مورد نظر غیر کلید باشد(مثلا شماره شناسنامه که در فایل دانشجو رکوردها بر اساس شماره دانشجو مرتب اند نه شماره شناسنامه)

- جستجوی آن همانند فایل پایل جستجوی خطی است ، برای پیاده کردن رکورد به طور متوسط باید نصف فایل را بگردیم.

زمان واکشی رکورد (بر اساس فیلد کلید اصلی) : اگر رکوردهای ما بر اساس کلید اصلی مرتب شده باشند و ما بر اساس کلید اصلی جستجو رو اعمل کنیم از فرمول زیر بدست می آید:

زمان واکشی ⟸ $tfbinary = log2b(s + r + btt + cb) + tftlf$

$s + r + btt$: زمان خواندن یک بلاک(به فرض این که ما بلاک را پیدا کردیم و می خواهیم آنرا بخوانیم)

cb :زمان پردازش یک بلاک در بافر که می توان به علت کم بودن از آن صرف نظر کرد.

$log2b$: تعداد دفعات مراجعه به فایل برای خواندن بلاک

$tftlf$: ممکن است بلاک مورد نظر در فایل اصلی وجود نداشته باشد پس در فایل تی ال اف می باشد.

زمان خواندن رکورد بعدی ⟸ $TN = \frac{R}{Tr}$

رکورد بعدی معمولا بلافاصله بعد از رکورد فعلی قرار دارد.

- امکان دارد که در بلاک بعدی قرار داشته باشد با این توضیح احتمال این که رکورد بعدی در همان بلاک خوانده شده که در بافر موجود است قرار داشته باشد برابر $1 - \frac{1}{bf}$ نیز رکورد فعلی آخرین رکورد بلاک است که برای دستیابی به رکورد بعدی نیاز داریم.

در صورتی که رکورد بعدی در فایل tlf باشد به علت عدم ارتباط منطقی بین فایل اصلی و فایل tlf مجبور به یک جستجوی خطی در فایل tlf خواهیم بود که این جستجو امکان پذیر نیست. مگر آن که کلید رکورد بعدی را داشته باشیم.

زمان درج رکورد (فایل های کوچک)

در هنگامی که فایل اصلی کوچک باشد ، رکورد را در جای منطقی اش درج می کنیم.

در حالت اول رکورد باید در سر جای خود درج شود و سپس کلید رکوردهای بعد از آن یک واحد به سمت انتهای فایل شیفت داده شوند ، چون دقیقا مشخص نیست که رکورد در کجا باید درج شود،به طور متوسط نصف رکوردهای فایل شیف داده می شوند بنابراین خواهیم داشت: ⟸ $Tr=Tf+\frac{1}{2}b(btt+TRw)$

- برای شیفت دادن به هر بلاک باید اول آن را بخوانیم سپس آن را بنویسیم.

$\frac{1}{2}b$: به طور متوسط باید نصف بلاک شیفت پیدا کند،اگر بخواهیم یک رکورد را در یک بلاک درج کنیم اگر به ابتدای بلاک اشاره کنیم باید تمام بلاک ها را یک شیفت به سمت راست انجام بدیم ، اگر به انتهای یک بلاک اشاره کنیم هیچ شیفتی انجام نمی شود.

Tf : زمان پیدا کردن محل منطقی رکورد جدید جهت درج.

در حالت دوم یعنی حالتی که حجم فایل زیاد می باشد به علت زمان گیر بودن عمل شیفت ،عمل درج در انتهای فایل tlf انجام می پذیرد تا بعدا در سازمان دهی مجدد فایل به سر جای اصلی خود باز گردد.(مشابه فایل پایل است.)

Tf= s+r+btt+Trw = s+3r+btt

زمان بهنگام سازی Tu:

اول این که بهنگام سازی بر روی فیلد یا فیلدهایی غیر از کلید اصلی انجام می پذیرد. بنابراین بعد از انجام عمل بهنگام سازی رکورد در همان جای قبلی اش باز نویسی می شود چرا که کلید اصلی تغییر نکرده ، سپس مکانی منطقی فایل تغییر ی نمی کند بنابراین :

Tu=Tf+Trw=Tf+2r

ولی در حالت دوم بهنگام سازی بر روی کلید اصلی انجام می شود بنابراین دیگر در هنگام بازنویسی نمی توانیم رکورد را در سر جای قبلی اش بازنویسی کنیم چرا که ترتیب رکوردها برهم خواهد خورد ، بنابراین ابتدا رکورد را خوانده و به صورت منطقی حذف می کنیم و بعد از آن رکورد جدید را با کلید تغییر یافته در انتهای فایل اضافه خواهیم کرد.

اگر بهنگام سازی روی فایل اصلی باشد دقیقا مشابه زمان فایل پایل است.

خواندن متمایل فایل(Tx):

زمان خواندن فایل بصورت پی در پی $\Longleftarrow$ $\mathbf{Txseq} = (n+i)\frac{R}{t'}$

n : تعداد رکورد در فایل اصلی(لود اولیه)

i : تعداد رکورد در فایل tlf

- اگر بخواهیم فایل را پی در پی بخوانیم باید کل فایل خوانده شود.

اگر بخواهیم فایل را به صورت **سریال** بخوانیم:

1- باید فایل tlf را مرتب کنیم. (فایل اصلی مرتب است و نیاز به مرتب کردن ندارد.)
2- کل فایل را به صورت پی در پی بخوانیم.

Txser=Tsort(i)+Txseq

زمان سازمان دهی مجدد(Tr)

زمان سازمان دهی مجدد $\Longleftarrow$ $\mathbf{Tr} = Tsort(i) \mid n\frac{R}{t'} \mid i\frac{R}{t'} \mid (\mathbf{n} \mid \mathbf{i} \quad \mathbf{d})\frac{R}{t'}$

$Tsort(i)$: زمان مرتب سازی فایل tlf با i رکورد

$n\frac{R}{t'}$: زمان خواندن فایل اصلی با n رکورد اولیه

$i\frac{R}{t'}$: زمان خواندن فایل tlf با i رکورد

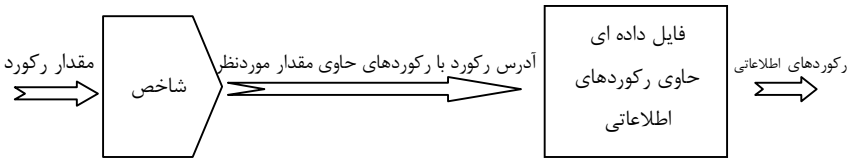
$(\mathbf{n}+\mathbf{i}-\mathbf{d})\frac{R}{t'}$: زمان باز نویسی دوباره رکوردها به همراه حذف رکورد حذف شده منطقی

فصل هشتم

فایل شاخص دار و چند شاخصی

معرفی شاخص

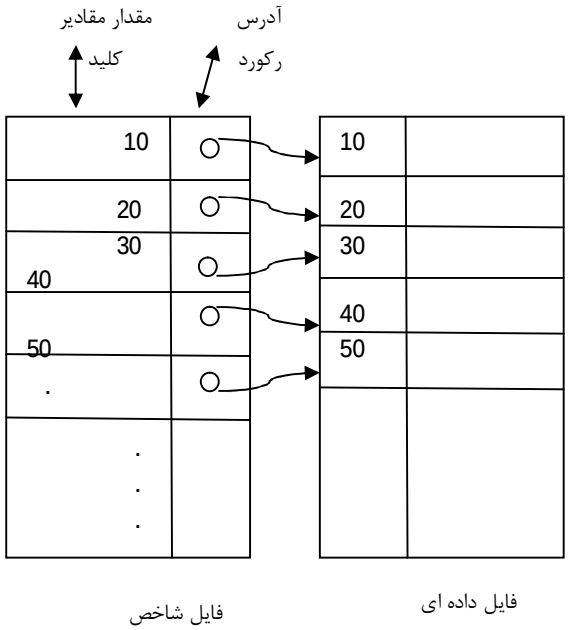
شاخص ساختمان داده ای است که مقدار کلیدی از رکورد را به عنوان ورودی می پذیرد و رکوردهایی با آن مقدار کلید (در صورت وجود) را سریعا پیدا می کند.



- هر رکورد یک فیلد دارد و آن را به عنوان کلید اصلی که شاخص نام دارد در نظر می گیرد.
- اطلاعات در شاخص کاملا مرتب است.
- هرسطر شاخص از دو قسمت تشکیل شده است.

آدرس(P)	مقدار(V)

یک فایل شاخص و فایل داده ای (شکل الف)



موارد استفاده شاخص

1- پردازش سریال فایل
2- اگر عمل واکشی رکورد از طریق مقدار کلیدها زیاد شود مثلا شماره دانشجویی را می توان یک شاخص در نظر گرفت.

بیشتر در محیط هایی که نیاز یه پردازش سریال فایل بر روی یک فیلد (کلید اصلی) مطرح باشد مورد استفاده قرار می گیرد. همچنین در سیتم هایی که عمل واکشی رکوردها از طریق مقدار کلیدها انجام می شود.

انواع شاخص

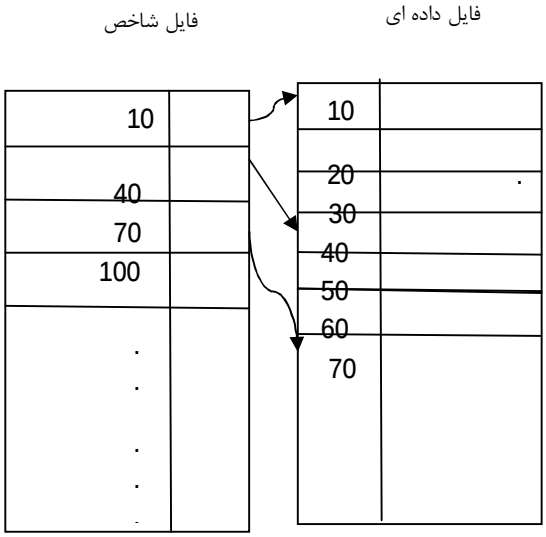
لنگر گاه(An-chor pint) : جایی که شاخص به آن اشاره می کند.

نقطه ای از فایل داده ای که از مدخل بخش شاخص به آن اشاره گر (نشانگر) داریم.

2 نوع شاخص داریم.

شاخص متراکم مانند شکل الف و **شاخص غیر متراکم** مانند شکل زیر.

هر سطر شاخص به یک گروه در فایل داده ای اشاره می کند.



شاخص اصلی و شاخص های ثانویه:

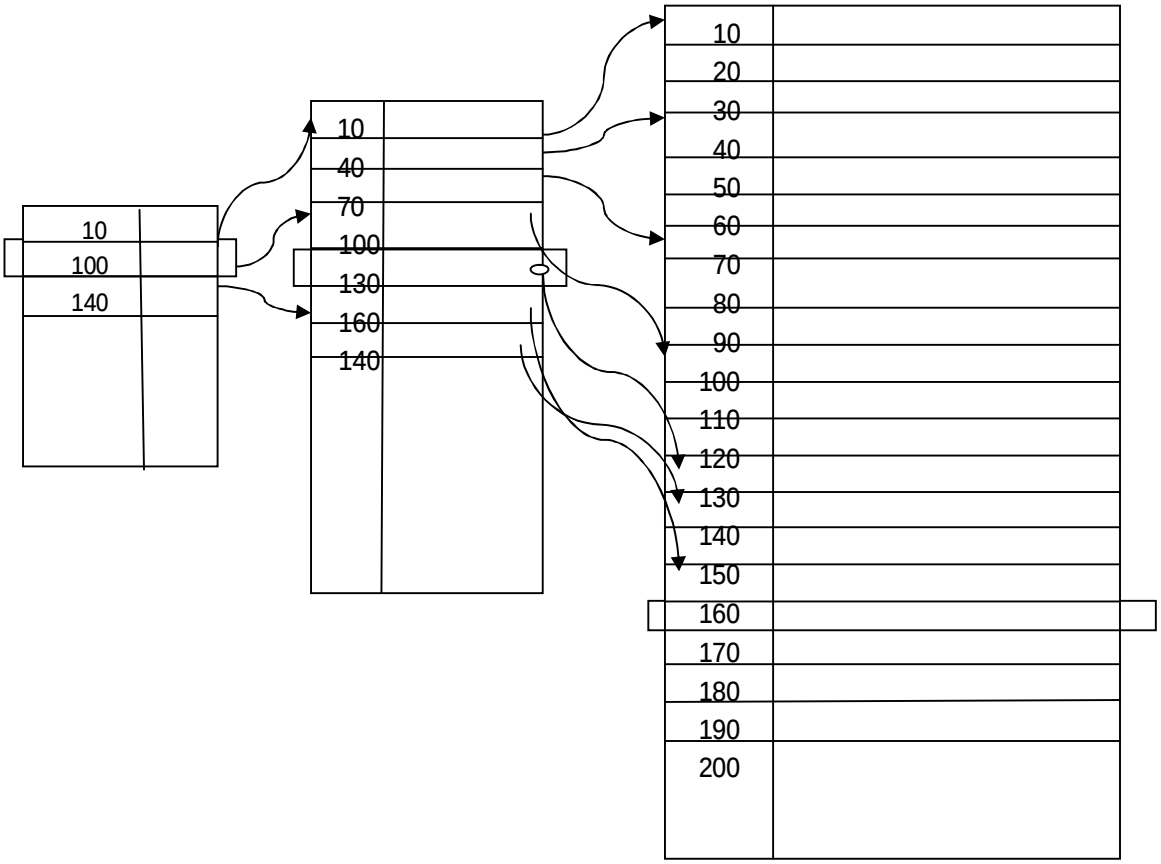
اگر فیلد مورد نظر کلید اصلی باشد شاخص اصلی و اگر کلید اصلی نباشد شاخص ثانویه است.

در صورتی که شاخص را بر روی کلید اصلی بنا کنیم. شاخص را شاخص اصلی گوییم اما هر فیلد دیگری نیز می تواند به عنوان فیلد مدخل و روی شاخص انتخاب شود که در این حالت شاخص های ثانویه خواهیم داشت.

شاخص چند سطحی

اگر فایل داده ای بسیار بزرگ باشد شاخص ما یک سطحی کاربرد ندارد و یک شاخص چند سطحی تولید می شود.

اگر فایل معمولی باشد تعداد رکوردها خیلی بالاست و جستجو طولانی می شود.فرض کنیم در شکل زیر دنبال فیلد 170 هستیم:



مراحل از سمت چپ به راست طی می شوند. این یک فایل شاخص سطح اول است.

ظرفیت نشانه روی شاخص

شاخص چند تا سطر دارد ⟸ $y = \left[\frac{\text{اندازه بلاک شاخص}}{\text{طول مدخل شاخص}}\right] = \left[\frac{B}{V+p}\right]$

پارامترهای شاخص چند سطحی

B1*(v+p) : سایز فایل شاخص در سطح اول

$B1=\frac{n}{Bf}$: تعداد مدخل های سطح اول

ü اگر شاخص چند سطحی بود شاخص روال فرمول ها همین است فقط b1 می شود bn

تعداد سطوح شاخص ⟸ $X=[\log_y b1]$

ناحیه سر ریز:

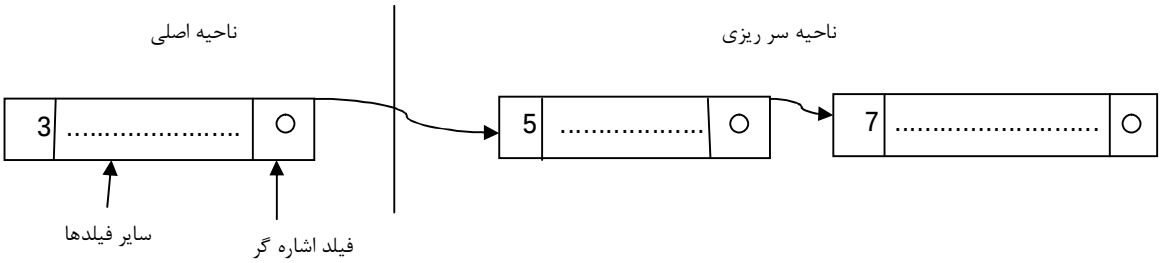
قسمتی جدا از شاخصه که ما برای درج بعضی رکوردها نه همه رکوردها مجبوریم به این ناحیه مراجعه کنیم، برای اضافه کردن رکوردهای جدید ، باید یک ناحیه سرریزی داشته باشیم.

- در نظر گرفتن یک فایل جداگانه برای ناحیه سرریزی : این روش به دلیل زمانگیر بودن ارتباط بین فایل اصلی و فایل سر ریزی چندان مناسب نمی باشد.
- در نظر گرفتن فضایی از بلاک در لود اولیه برای رکوردهای جدید: این روش هر چند که لوکالیتی را بهتر خواهد کرد ولی امکان توزیع نا مناسب داده ها و در نتیجه پر شدن سریع بعضی از این حافظه و خالی بودن بقیه وجود دارد.
در صورتی که بتوانیم محدوده داده های سیستم را حدس بزنیم می توانیم تا حدودی از به وجودآمدن این مشکل جلوگیری کنیم.
- انتخاب فضایی از فایل اصلی برای ناحیه سرریزی : که مناسب ترین روش می باشد که خود این روش به دو صورت قابل پیاده سازی است:

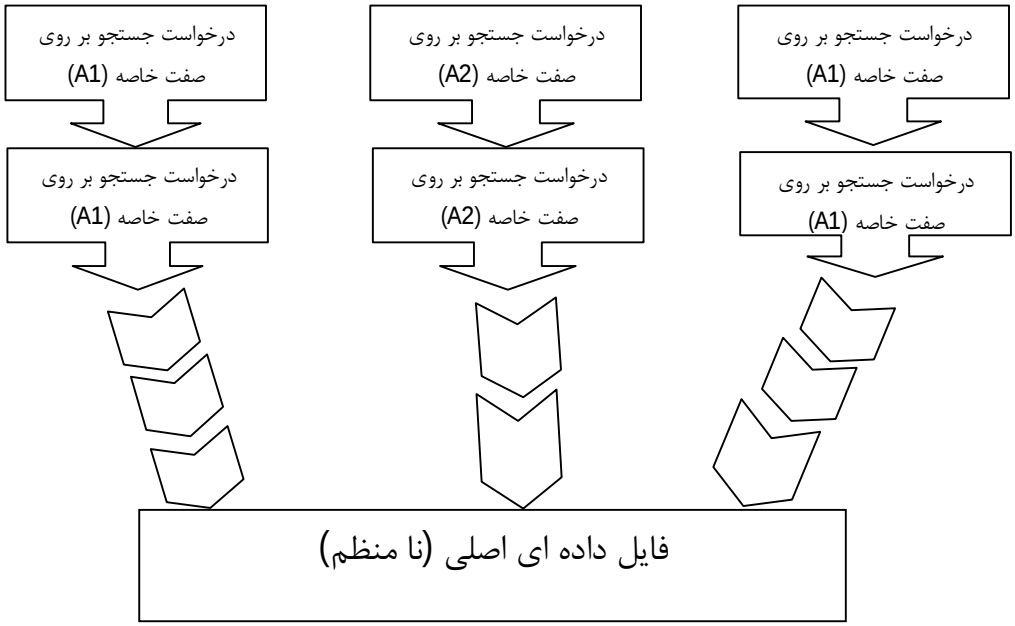
1) اختصاص ناحیه ای از شیارهای انتهایی هر استوانه به ناحیه سرریزی که باعث قرار گرفتن رکوردهای جدید مربوط به هر استوانه در همان استوانه می شود و باعث کاهش زمان استوانه جویی نیز خواهد شد.
2) اختصاص استوانه هایی در انتهای فایل برای ناحیه سرریزی که موجب کاهش لوکالیتی و افزایش زمان دستیابی خواهد شد.

روش های درج در سرریز:

1) **درج در اولین بلاک جا دار در ناحیه سرریزی** : در این روش رکورد در اولین بلاک جادار در ناحیه سرریزی اضافه می شود و از رکوردی که منطقا (در مرتب سازی منطقی) قبل از آن قرار دارد ، اشاره گری به رکورد جدید ایجاد می شود. در این تکنیک برای هر رکورد ناحیه اصلی و ناحیه سرریزی فیلد اشاره گر در نظر می گیریم (هر چند که امکان "وجود نداشتن" آن وجود دارد.)



2) **درج با جابجایی** : در این روش ، رکورد جدید در بلاک مربوطه در ناحیه اصلی در محلی که باید منطقا درج شود ، قرار گیرد و رکوردهای بعدی همان بلاک ، به سمت انتهای بلاک شیفت داده می شوند و آخرین رکورد بلاک به اولین بلاک جادار در ناحیه سر ریزی منتقل می شود. در این روش نیز زنجیره رکوردهای سر ریزی هم ایجادمی شود ولی تفاوت اصلی آن این است که در این روش بای هر بلاک از ناحیه اصلی ، یک اشاره گر به ناحیه سرریزی وجود دارد و نه برای هر رکورد یک اشاره گر به ناحیه سر ریز(در روش قبل)


درخواست جستجو بر روی
صفت خاصه (A1)
درخواست جستجو بر روی
صفت خاصه (A2)
درخواست جستجو بر روی
صفت خاصه (A1)
درخواست جستجو بر روی
صفت خاصه (A1)
درخواست جستجو بر روی
صفت خاصه (A2)
درخواست جستجو بر روی
صفت خاصه (A1)
فایل داده ای اصلی (نا منظم)

فصل نهم
تکنیک های بهبود آدرس دهی

بهبود کارایی سیستم فایل

دسترسی تصادفی ⟸ s+r+btt

- استفاده از بافر برای افزایش سرعت پردازش فایل ها(در فصل های قبل گفته شد)
- سازماندهی داده ها با استفاده از سیلندرها (استوانه ها) جهت کاهش زمان پیگرد درنگ دورانی
- استفاده از الگوریتم های مناسب جهت حرکت هد خواندن/ نوشتن برای ترتیب انتخاب درخواست ها و کاهش زمان پیگرد.
- توزیع داده ها بین چند دیسک کوچکتر به جای یک دیسک بزرگ ، در این روش با تعداد هدهای بیشتر برای خواندن ، به تعداد بیشتری از بلوک ها در واحد زمان دستیابی خواهیم داشت و زمان پیگرد کاهش می یابد (استفاده از سیستم RAid)
- استفاده از حافظه نهان دیسک
- استفاده از تکنیک درهم یا تداخل بلاک ها (Interleawing) جهت کاهش درنگ دورانی
- تغییر نقطه آغاز شیارها (track stayging) جهت کاهش درنگ دورانی

سازمان دهی داده ها با استفاده از سیلندرها (استوانه ها)

در این روش سعی می کنیم تمام داده ها را در یک استوانه جا بدهیم ، با اینکار فقط یکبار زمان استوانه جویی داریم و برای بقیه داده ها صفر است. اگر روی سیلندر یا استوانه جا نشد باید فایل رو روی استوانه های هم جوار قرار بدهیم. وقتی این کار را انجام دادیم زمان استوانه جویی آن کوتاه است فقط کافی است از یک سیلندر به سیلندر دیگر حرکت کند.

روش انجام کار این است اگر ما چند تا پک داریم تمام اطلاعات را روی سیلندرهای مشابه قرار دهیم یعنی مثلا یک فایل رو روی تمام سیلندرهای شماره هشت قرار دهیم، با این کار ما فقط یکبار سیلندرهای شماره 8 رو پیدا می کنیم و تمام اطلاعات ما روی همون سیلندرهای شماره هشت قرار دارد. در این حالت دیسک پک از چندین استوانه تشکیل شده است.

چون زمان پیگرد تقریبا نصف میانگین زمان دستیابی به بلوکها را شامل می شود ، در بسیاری از کاربردها بهتر است داده هایی که با احتمال زیاد با هم دستیابی می شوند در یک سیلندر قرار داده شوند. اگر فضای کافی وجود نداشته باشد (حجم بلوک ها بیش از ظرفیت یک سیلندر می باشد.) می توان از چند سیلندر همجوار استفاده کرد.

در واقع اگر تمام بلوک های موجود در یک شیار یا یک سیلندر را به طور ترتیبی بخوانیم ،می توانیم از تمام زمان های پیگرد و درنگ دورانی (به جز زمان اولین پیگرد برای انتقال به سیلندر و اولین درنگ دورانی جهت انتقال بر روی اولین بلاک در سیلندر) صرف می شود.

این روش در مواردی که بلوک های داده روی دیسک به ترتیب خوانده و نوشته می شوند به طوری که از قبل قابل پیش بینی باشند و فقط یک فرآیند در هر زمان از دیسک استفاده کند ،بسیار مفید است. اما در مواردی که چندین فرآیند به طور موازی اجرا می شوند و از یک دیسک به صورت اشتراکی استفاده می نمایند ، مناسب نیست.

استفاده از الگوریتم مناسب

فرض کنیم برای اطلاعات ما چندین درخواست متفاوت وجود دارد ما باید از الگوریتم مناسب استفاده کنیم.

الگوریتمFiFo

اساس کار این الگوریتم ورود زودتراست یعنی هرکی زودتر بیاد زودتر سرویس می شود.

مثال: فرض کنید هد خواندن/نوشتن بر روی شیار 80 قرار گرفته باشد و درخواست های زیر را ا ز چپ به راست برای دسترسی به شیارها داشته باشیم ، متوسط زمان حرکت به ازای هر درخواست استفاده از الگوریتم FiFo بدست آورید:

30-55-95-70-150-120-180-25= درخواست ها

80 -30-55-95-70-150-120-180-25=خروجی

تمام شیارهای استوانه های ما از 0 تا 200 هستند.

در این حالت اگر از شیار 80 تا 30 برود 50 استوانه باید رد شود ، اگر از شیار30 به 55 برود باید 25 شیار بگذراند، بنابراین تعداد استوانه هایی که باید رد کنیم زیاد می شوند.

الگوریتم sstf (short remain time)

در این حالت ما کاری به ترتیب درخواست ها نداریم ما به سمت شیاری می رویم که نزدیکتره. مثلا شیار ما 80 پس 70 از همه نزدیک تره در این حالت زمان حرکت بین شیارها خیلی کوتاه است پس خروجی مثال قبل می شود:

80-70-55-30-25-95-120-150-180

الگوریتم scan(up-down)

مثل آسانسور عمل می کنه ، فرض کنیم کمترین شماره ما 0 و بیشترین 200 است وقتی ما 80 هستیم دو حالت داریم یا باید بالا باشیم یا پایین. حال فرض کنیم به سمت بالا گفته از بالا به پایین حرکت کنیم:

80-95-120-150-180-200-70-55-35-25-0

الگوریتم cscan

الگوریتم scan از 180 تا 200 فضای هرز بود یا کسی نبود ، از 25 تا 0 هم همین طور پس بی جهت این دو مسیر را طی می کردیم اما در این الگوریتم 0 و 200 را حذف می کنیم. زمان استوانه جویی در این الگوریتم کاهش پیدا می کند:

80-95-120-150-180-70-55-30-25

حافظه نهان دیسک

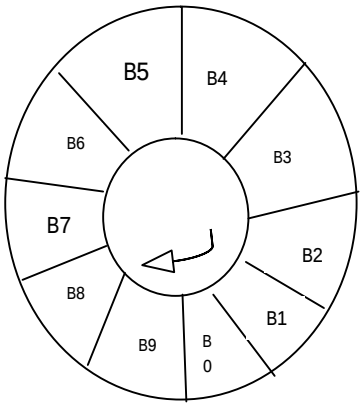
روش آخر با استفاده از حافظه نهان بوده است. این حافظه یک حافظه خیلی کوچک با حجم کم است. اما سرعت دسترسی آن بسیار بیشتر از ram است. مثلا اگر کسی بگوید رکورد شماره 5 ، رکورد 0 تا 10 در آن قرار می گیرد چون حدس می زند کاربر بعد از انجام کار یا با رکورد 4 یا 6 کار داشته باشد. اگر این حدس درست باشد دیگر کاربر نیازی ندارد به حافظه اصلی برود.

حافظه نهان (cash memeory) به حافظه ای گفته می شود که کوچکتر و سریعتر از حافظه اصلی است و بین حافظه اصلی و پردازنده قرار می گیرد ،این نوع حافظه با استفاده از اصل محلی بودن از میانگین زمان دستیابی به حافظه می کاهد چنین قاعده ای را می توان برای حافظه دیسک نیز در نظر گرفت.

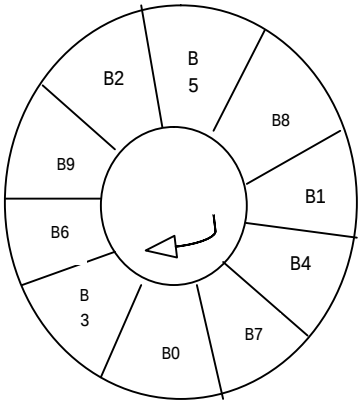
حافظه نهان دیسک حاوی نسخه ای از چند قطاع دیسک است وقتی درخواستی برای داده خاصی صورت می گیرد ، بررسی می شود که آیا آن داده در حافظه نهان دیسک وجود دارد یا خیرد ، اگر وجود دارد که درخواست از طریق حافظه نهان برآورد می شود و گر نه درخواست به دیسک ارجاع داده می شود تا داده خوانده شده و به حافظه نهان دیسک

منتقل شود. به دلیل محلی بودن ارجاعات وقتی بلوکی از داده ها به حافظه نهان واکشی می شود. احتمالا درخواست های بعدی نیز در همان بلوک داده ای منتقل شده می باشد.

درهم چینی Interleaving



شکل 1

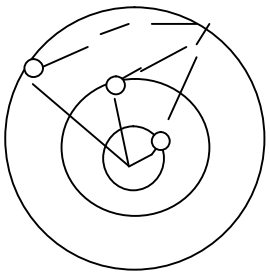


شکل 2

در شکل 1 اگر بلوک 0 را بخوانیم مدت زمانی طول می کشد تا توسط سی پی یو پردازش بشه به همین دلیل سی پی یو تا به حدی بلوک 1 را هم می خونه ، حالا اگر بخواهیم بلوک1 رو بخونیم نمی تونیم چون مقداری از بلوک1 گذشته و ما مجبوریم یک دور دیگر صبر کنیم تا بلاک 1 را بخوانیم یعنی در هر دور یک بلاک رو می خونیم.
شکل 2 درهم: وقتی بلوک0 رو می خونیم اول بلوک7 رو می خواند سپس می رسد به بلوک1 به همین منوال ادامه پیدا می کند در این حالت با درهم چینی خیلی راحت می توانی اطلاعات رو پشت سر هم بدون این که منتظر بشیم هر بلاک یک دور بزنه این کار رو انجام می دهیم.

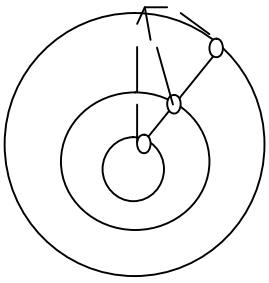
تغییر نقطه آغازین شیارها

نقطه آغاز شیارها



شکل 1

نقطه آغاز شیارها



شکل 2

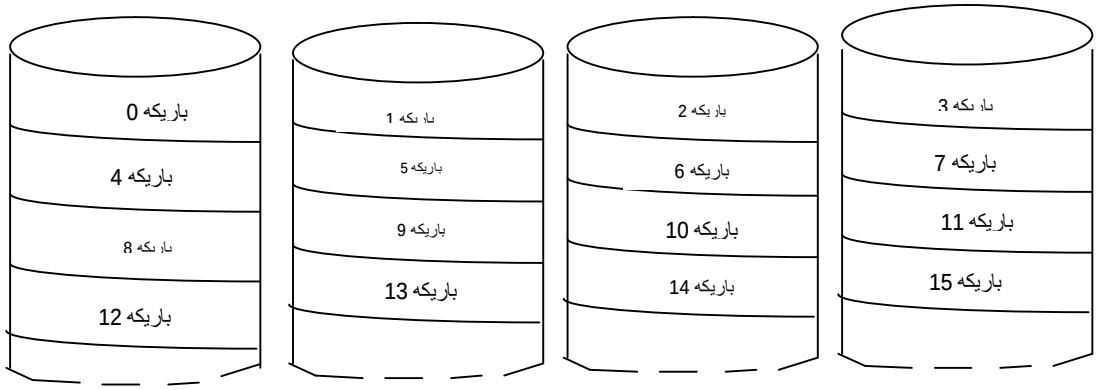
اگر ما چندین شیار داشته باشیم بهتر است نقطه ی آغاز آن ها را روی یک خط همانند شکل 2 نگیریم.

با استفاده از یک زاویه این کار را انجام بدهیم یعنی یقطه ی آغازین یک شیار یک زاویه داشته باشد دلیل این است که زمان درنگ دورانی در شکل 2 بالا می رود و در شکل 1 پایین می رود. در حالت دوم تا بخواد یک دور بزند ما یه ذره از شیار دو را گذشتیم.

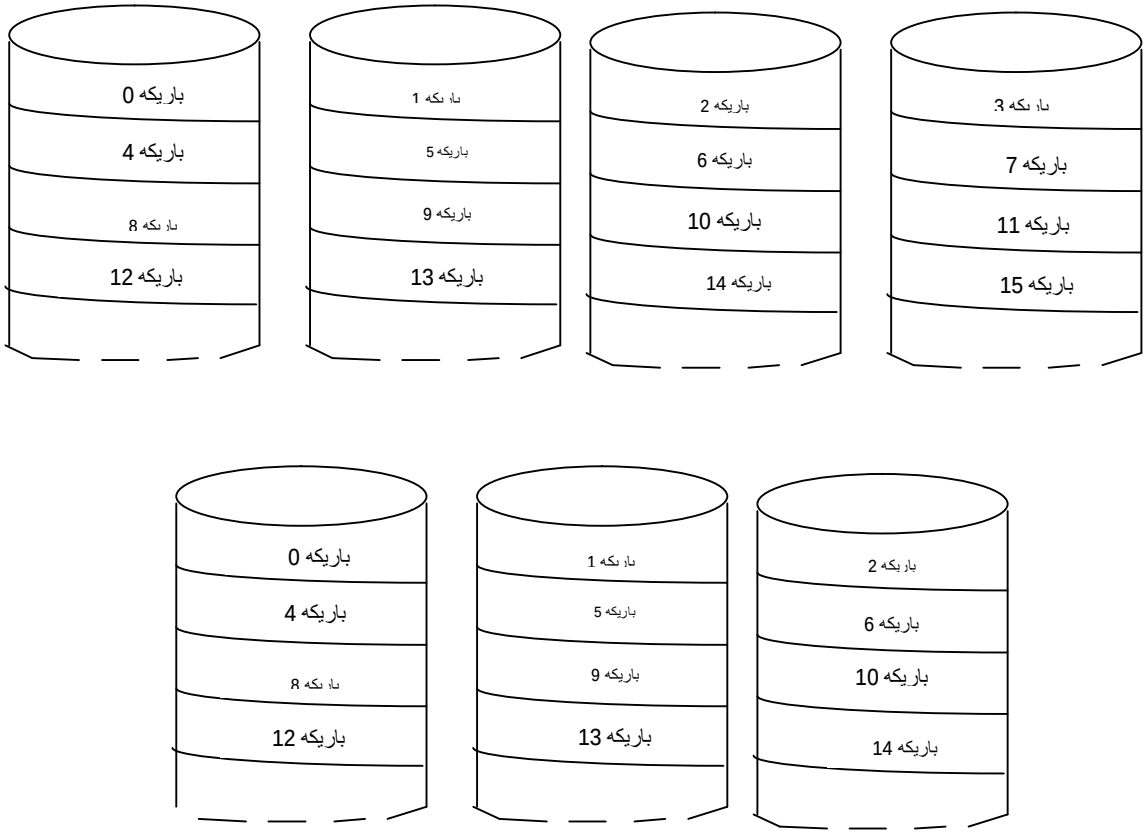
تکنیک RAID

اطلاعات را روی چندین دیسک قرار می دهیم و به طور همزمان می خوانیم یعنی ما یک فایل داریم یه قسمتی روی یه دیسک پک قسمت دیگه اش روی دیسک پک دیگه(به طور مساوی) این RAID سطح صفر است و هیچ گونه تکراری ندارد، در این حالت با استفاده از تکنیک RAID و سخت افزارها ما سریعتر می تونیم اطلاعات رو بخونیم.

سطح صفر



در سطح یک علاوه بر 4 نسخه سطح صفر کپی هم بگیرید چون خطر از دست رفتن اطلاعات یا هم زمانی بیشتری را بخواهیم انجام دهیم. تکنیک Raid سعی می کند با استفاده از موازی سازی کاری که ما انجام می دهیم رو سریعتر انجام دهد. یعنی ما یک ورودی/خروجی یا یک وسیله برای ورودی/خروجی نداشته باشیم ، چندین وسیله برای ورودی /خروجی داشته باشیم. به طور همزمان ما بتونیم از روی چندین دیسک بخوانیم.


باریکه 0
باریکه 4
باریکه 8
باریکه 12
باریکه 1
باریکه 5
باریکه 9
باریکه 13
باریکه 2
باریکه 6
باریکه 10
باریکه 14
باریکه 3
باریکه 7
باریکه 11
باریکه 15
باریکه 0
باریکه 4
باریکه 8
باریکه 12
باریکه 1
باریکه 5
باریکه 9
باریکه 13
باریکه 2
باریکه 6
باریکه 10
باریکه 14

فصل دهم

فایل های مستقیم (Direct Access File)

فایل هایی که دسترسی مستقیم دارند.

روش تولید آدرس

هر رکورد یک کلید دارد مثلا هر دانشجو یک شماره دانشجویی دارد و یک تابع وجود دارد وقتی ما شماره ی دانشجویی را به تابع می دهیم آدرس صندلی که باید شما بشینید را می گوید. اما عکس آن امکان پذیر نیست یعنی اگر آدرس را بدهیم کلید رو نمی تونه پیدا کنه چون یک طرفه است. به این روش دسترسی مستقیم گفته می شود.

مثال:

Add = Key MoD 13

Key	Address
4	4
24	11
130	0

اگر شماره 4 وارد شد ، 4 تقسیم بر 13 باقیمانده اش می شود4 پس باید روی صندلی 4 بشیند.

در این روش تابع سریعا آدرس را پیدا می کند.

اما مشکل به وجود آمده این است که خیلی از فضاها خالی اند یا هیچ کس روی آن ها نشسته نیست. مشکل اساسی دیگر تصادف می باشد مثلا اگر در مثال قبل 141 بیاید باقی مانده ی آن 11 می شود ، پس هم 24 باید روی صندلی 11 بشیند و هم 141 که این کار غیر ممکن است.

تصادف: دو رکورد متفاوت آدرس های یکسان براشون تولید شده

راه حل های تصادف

1- برای 141 یک تابع دیگر بنویسیم شاید تابعی ایجاد کند که با آدرس های قبلی تصادف نداشته باشد.
2- نشاندن در قسمت رکوردهای اضافه و تولید زنجیر

در رکوردهای خالی یا اضافه یک زنجیر تولید کنیم ، مثلا 11 که نشسته رکورد 333 اضافه بشه و باید جای 11 بشینه ما اون رو در جای رکورد خالی قرار می دهیم و با زنجیر وصل می کنیم.

اگر کسی با 333 کار داشته باشه میاد 11 ، 11 بهش می گه زنجیر رو بگیر تا برسی به 333 که در جای 541 نشسته و اگر 541 بیاد می فرستیمش یه جای دیگر.

این زنجیرها دسترسی رو زیاد می کند یعنی در این مثال 3 بار ورودی/خروجی وجود دارد تا بتواد 541 را پیدا کند.

سوال : کدام روش بهتر است:

روش اول : 1 2 3 4 5 6 7 8 9 1 2

روش دوم :1 2 3 4 5 6 7 8 9 1 1

در روش اول عدد یک و دو دوم با یک و دو اولی تصادف دارند پس مجموع تصادف ها 2 می باشد.

در روش دوم دو تا یک دومی با یک اولی تصادف دارند پس مجموع تصادف ها 2 می باشد.

برای اعداد 1 تا 9 : همه ی آن ها هر کدام یک بار ورودی /خروجی نیاز است و برای 1 و 2 آخری هر کدام دوبار جمعا می شود 13 بار ورودی و خروجی.

برای اعداد 1 تا 9 : هر کدام یک بار ورودی و خروجی ،برای یک دومی 2 بار و برای 1 آخری 3 بار ورودی/خروجی نیاز است، جمعا می شود 15 بار. پس طول زنجیر در روش دوم بیشتر است.

محدوده آدرس دهی

یعنی ما چه فضای آدرسی را داریم برای مثال اگر تابع ما بصورت زیر باشد یعنی تعداد خانه های ما 23 تا است از 0 تا22.

Add=Key mod 23

تابع زیر مفروض است:

Add=Random(0 to 1)*1000

پس تعداد خانه های ما می تواند از 0 تا 1000 باشد یعنی 1001 خانه می پذیرد(تولید کند).

و اگر تابع به این صورت باشد:

Add=key Mod 13

*

Key mod 25 ⟹ B*25=325

یکی از ویژگی های تابع این است که باید محدوده ی آدرس متناسب با تعداد رکوردها باشد.

فصل یازدهم

روش تخصیص حافظه

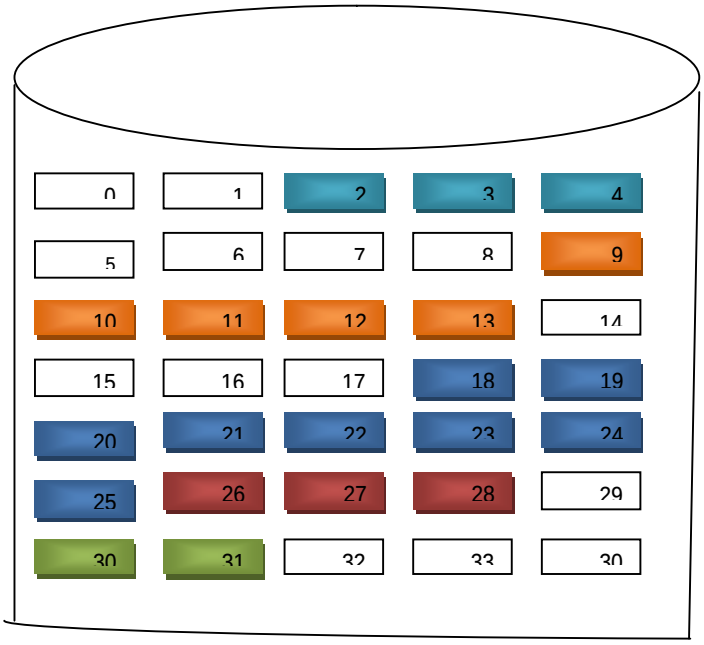
فرض کنیم حافظه ما مثل بلوک های یک ساختمان است ، یک سری از این واحدها پر هست یعنی حافظه اشغال شده و یک سری از این واحد ها خالی هستد یعنی حافظه ی ما آزاد است. در صورتی که برنامه ی جدیدی وارد این بلوک ها شود ما باید یه جای مناسب برای آن ها پیدا کنیم که در ذیل اشاره می گردد.

الف) تخصیص پیوسته:

اگر یک فایل با 5 بلاک داشته باشیم ، ما پنج بلاک پشت سرهم را پیدا کرده و به اون تخصیص بدهیم.

این روش پیوسته است یعنی فایل به صورت پیوسته و پشت سر هم ذخیره می شود. در این روش چیزی که ما می توانیم پیدا کنیم این است که دستیابی به کل فایل سریع است.

ایراد این روش : فرض کنیم کل هارد مانند شکل زیر است ما یه خونه ی 6 تایی می خواهیم ، جمعا 6 تا فضای آزاد شاید هم بیشتر داریم اما این فضاها (6 تا) پشت سر هم نیستند و چون ما 6 خونه ی پشت سرهم آزاد پیدا نمی کنیم این هارد ما جایی برای فایل جدید ندارد. به شکل زیر توجه کنید.

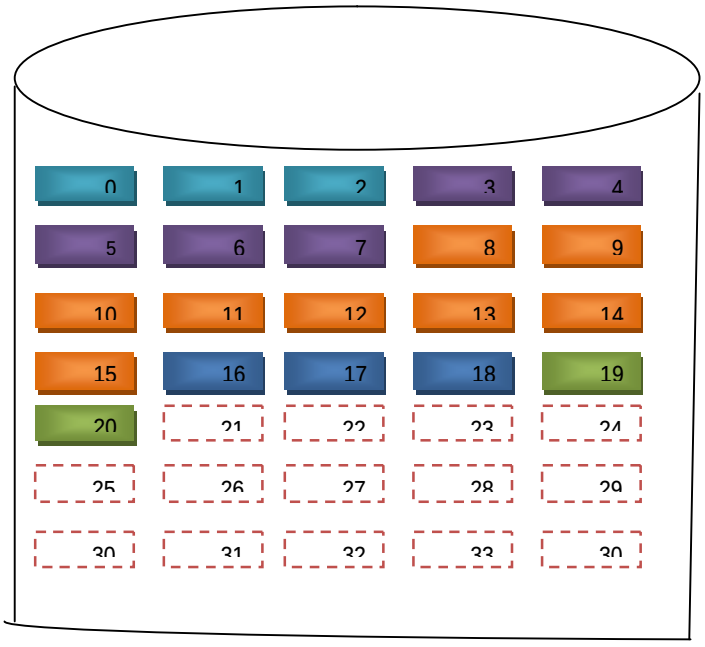


نام فایل	بلوک شروع	طول
فایل A	2	3
فایل B	9	5
فایل C	18	8
فایل D	26	3
فایل E	30	2

مثلا فایل A از بلوک 2 شروع می شود و طول آن 3 بلوک است یعنی 3بلوک اشغال می کند.

ب)تخصیص پیوسته با فشرده سازی

مانند حالت قبل است فقط فشرده شده است. یعنی

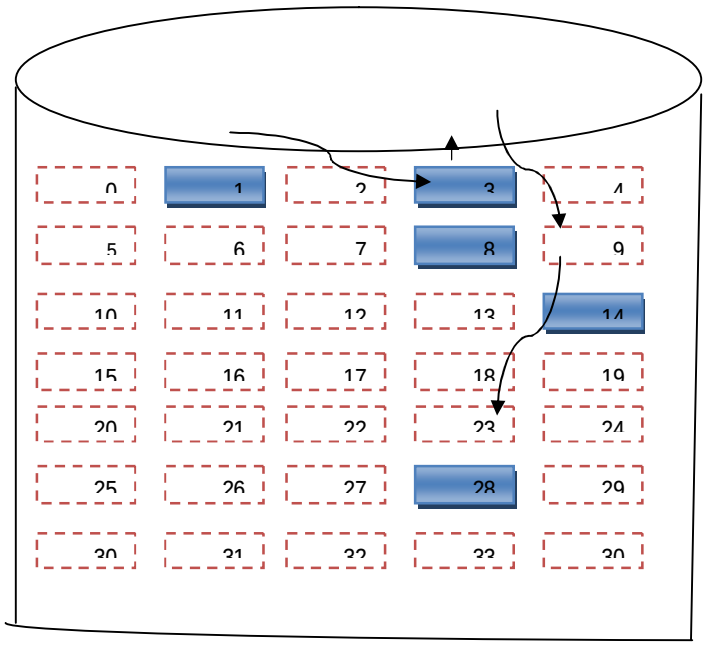


نام فایل	بلوک شروع	طول
فایل A	0	3
فایل B	3	5
فایل C	8	8
فایل D	16	3
فایل E	19	2

ج) تخصیص نا پیوسته و زنجیره ای:

یعنی فایل به بلوک هایی تقسیم می شوند که این بلوک ها پشت سر هم نیستند.

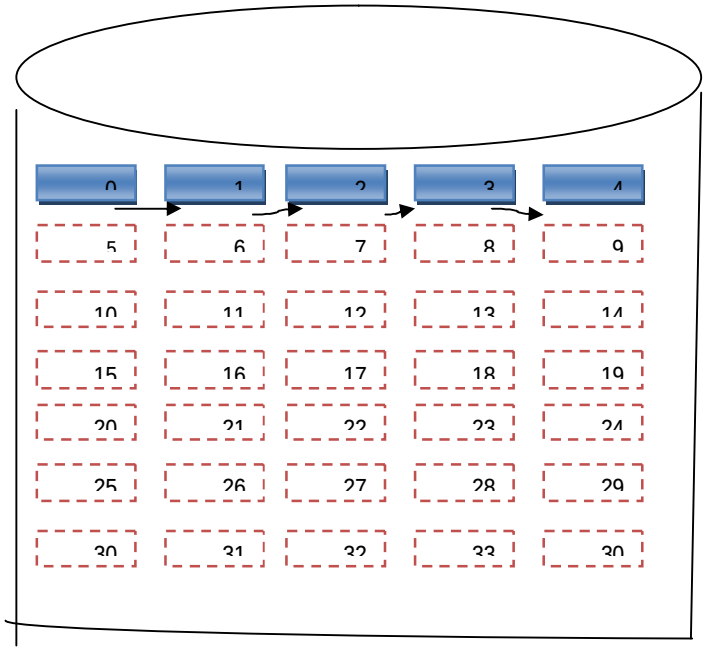
مزیت این روش : نیاز به پیدا کردن 5 تا بلوک پشت سر هم نداریم فقط باید اتصال و زنجیر را داشته باشیم. یکی از ایرادهای این روش زمان بازیابی است. یعنی ما اگر بخواهیم یک فایل با طول 5 را بخونیم باید 5 بار ورودی/ خروجی داشته باشیم. باید بخونیم ببینیم فایل کجای هارد هست. این عمل زمان دستیابی را بسیار ناپیوسته است.



نام فایل	بلوک شروع	طول
فایل B	1	5

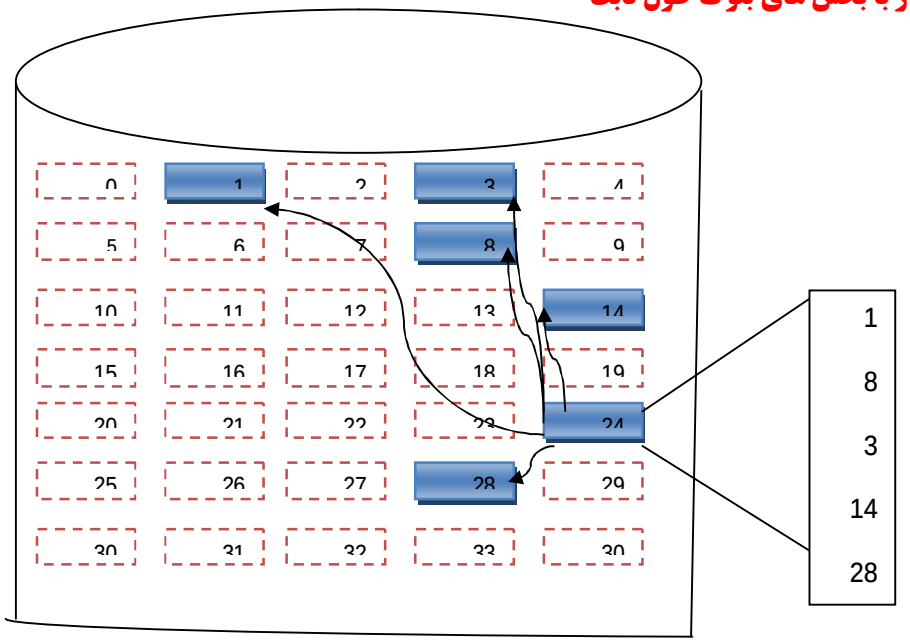
مثلا فایل B از بلوک یک شروع شده ، زنجیر شده به بلوک 8 (یعنی 8 باید خوانده شود) سپس به 3 و همین طور تا آخر پس عمل ورودی / خروجی زیاد می شود.

د) تخصیص ناپیوسته زنجیری



نام فایل	بلوک شروع	طول
فایل B	0	5

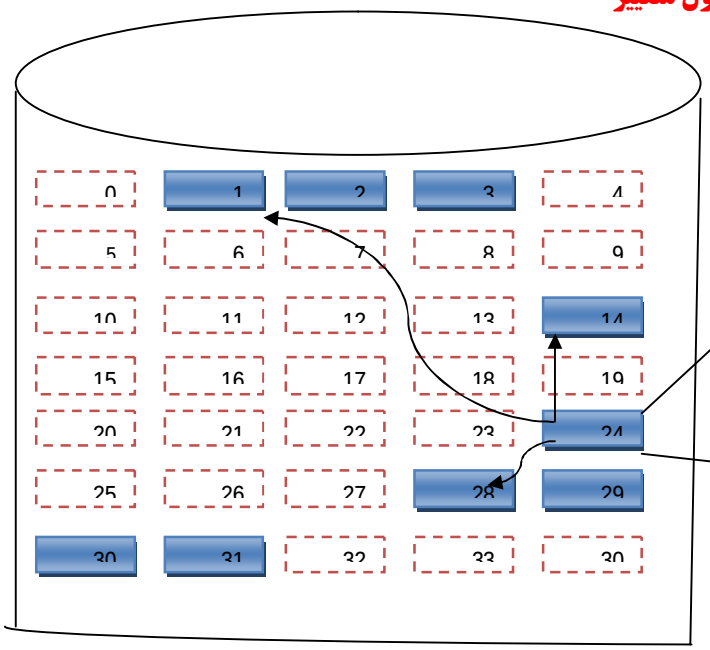
ه) تخصیص شاخص دار با بخش های بلوک طول ثابت



نام فایل	بلوک شاخص
فایل B	24

در این مثال فایل B بلوک 24 شاخص آن است.بلوک 24 هیچ اطلاعاتی از فایل اصلی ندارد ، یعنی خودش جز فایل حساب نمی شود یعنی به ترتیب بلوک های 1 8 3 14 28 را بخوانیم. دیگر نیاز به طی کردن (مسیر) زنجیر نیست یعنی اگر بلوک چهارم را بخواند که 14 است پس سریع به 14 می رود.(نیازی به طی کردن زنجیرها نیست چون زنجیرها همه از یک جا خارج شده اند .) مزیت این روش نسبت به زنجیر که سریعا واکشی انجام می گیرد و نیازی نیست زنجیرها پشت سر هم طی بشن اما اگر بخواهیم کل فایل را بازیابی کنیم باز همین مشکل را داریم. یکی از ایرادهای این روش این است که یک یا چند بلوک شاخص به فایل اضافه می شود.

د) تخصیص شاخص دار با بخش بلوک طول متغیر



بلوک شاخص	طول
1	3
28	4

نام فایل	بلوک شاخص
فایلB	24

در این حالت طول بلوک ها متغییر است یعنی بلوک شماره 1 به تعداد(طول) 3 بلوک، یا بلوک 28 با طول 4 بلوک پشت سر هم اجزای فایل B هستند.

ترکیبی از حالت پیوسته و زنجیری است(از این حالت ما می گوییم پیوسته که خیلی بلوک ها پشت سر هم قرار دارند و از این حالت می گوییم زنجیری چون همه از یه حالت شاخص برخودار می شود.)

مدیریت فضاهای آزاد دیسک(جدول بیتی) :

دیسک ما یکسری فضا دارد ، مثلا ما هزاران بلوک داریم، ما چطور باید بفهمیم کجای هاردمان پر یا خالی است ، سه روش وجود دارد:

- **روش اول (جدول بیتی)**: یک سری بیت است متشکل از 0 و 1 که 0 یعنی بلوک خالی و 1 یعنی بلوک پر است.
مزیت : حجم کم، مثلا اگر یک میلیارد بلوک داشته باشیم فقط یک میلیارد بیت برای مدیریت فضا نیاز داریم.
- پیدا کردن فضای خالی : یک رابطه بین فضاهای خالی وجود دارد بنابراین ما سریعا می توانیم فضای خالی را پیدا کنیم.
- وجود زنجیر : چون زمان مصرفی بالا می رود.

و دیگری اگر ما بخواهیم 3 تا بلوک خالی پشت سر هم پیدا کنیم نمی شود چون بعد از 1بلوک 8 و بعد از 8 بلوک 3 و این که ما نمی دانیم بعد از 1 بلوک های 2و3 خالی هستند یا نه؟ توانایی آگاه کردن ما از تعداد بلوک های آزاد در فایل را ندارد.

روش دوم (شاخص بندی):

مانند شکل اول با این تفاوت که فرض کنیم بلوک های سفید پرو غیر سفید خالی اند.

مزیت نسبت به زنجیره ای : سریعا ما می توانیم دسترسی پیدا کنیم و نیاز به طی کردن زنجیر نداریم.

- نیاز نبودن به طی کردن زنجیرها
- وجود یک استراتژی مثل همان شاخص برای این که به ما نشون بده از هر بلاک چند بلاک پشت سر هم خالی است.

فصل دوازدهم

درخت B

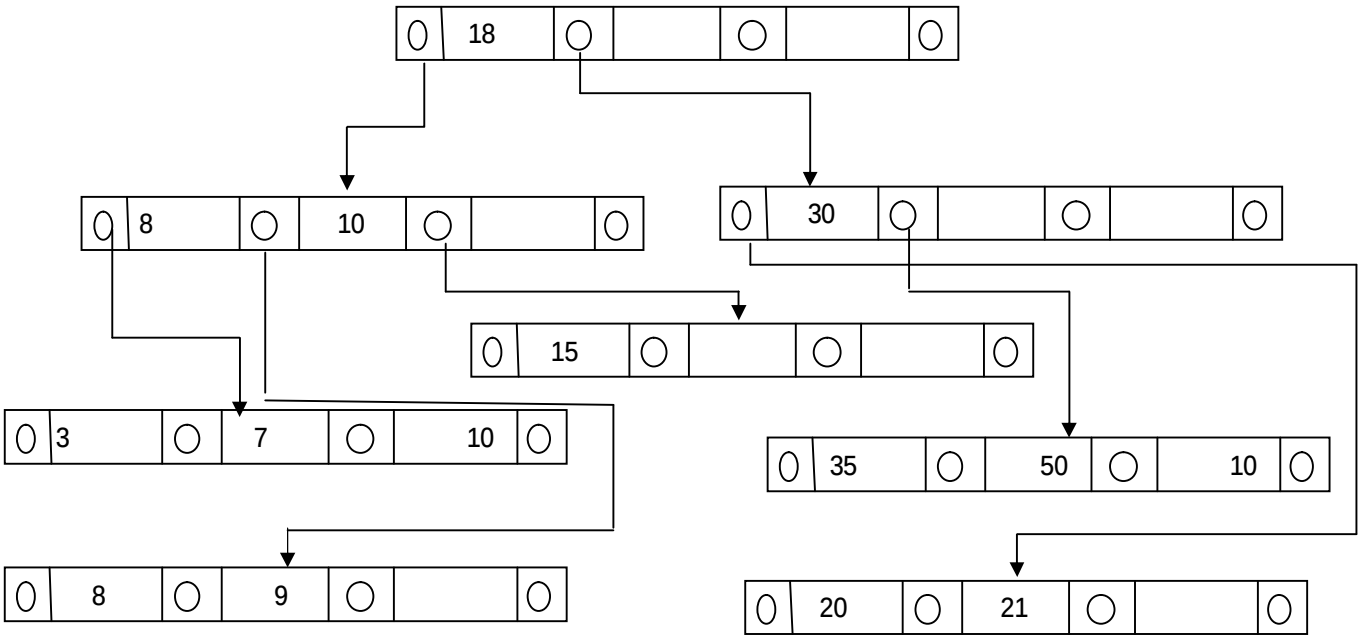
در مثال هایی که ما مطرح می کنیم همیشه حرف از یک مرتبه است که در اینجا مرتبه ما 5 می باشد. یعنی هر گره حداکثر می تواند5 خانه داشته باشد یا 5 عنصری باشد و حداقل می تواند نصفش (5 تقسیم بر 2 می شود 2) باید داشته باشد که به آن درخت مرتبه 5 گفته می شود.

گره : بلوک های مستطیل شکل گره نام دارند.

نکات زیر را به خاطر بسپارید:

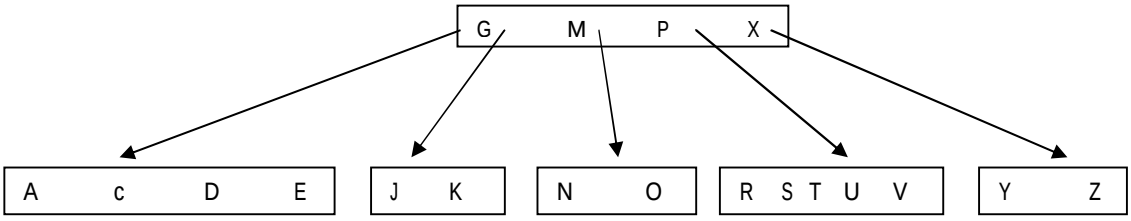
- گره ریشه حداقل دارای 2 فرزند باشد
- تمام گره ها غیر از ریشه حداقل دارای نصف مرتبه فرزند باشند.
- تمام گره های برگ در سطح مشابه و یکسانی قرار دارند.
- در سطح برگ ها ، تمام کلیدها بصورت مرتب قرار دارند.

در زیر گره ها سه عنصری هستند.



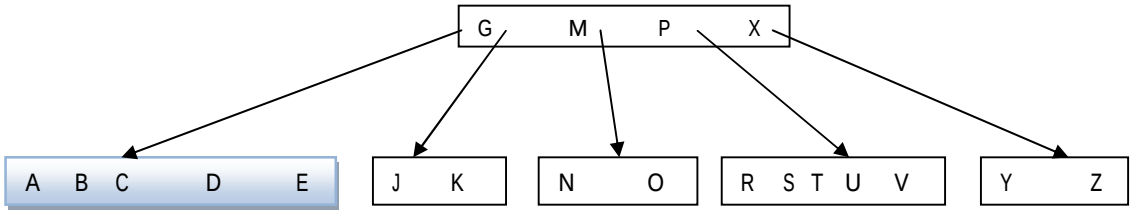
درج (مرتبه 5)

ساختار درخت 5 مرتبه ای



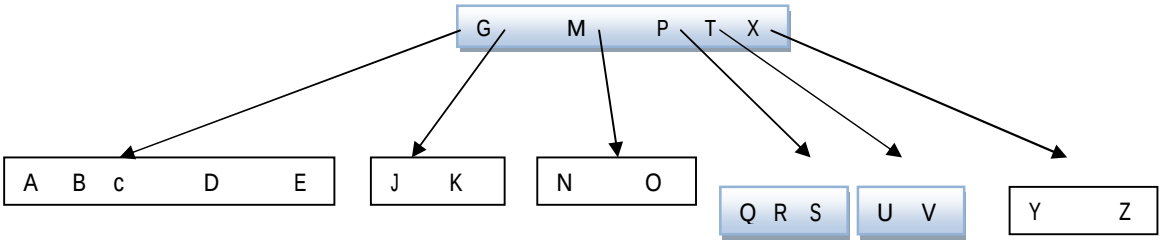
اضافه کردن کلید B

- باید مواظب باشیم تعداد خانه ها از 5 بیشتر و از 2 کمتر نشود.

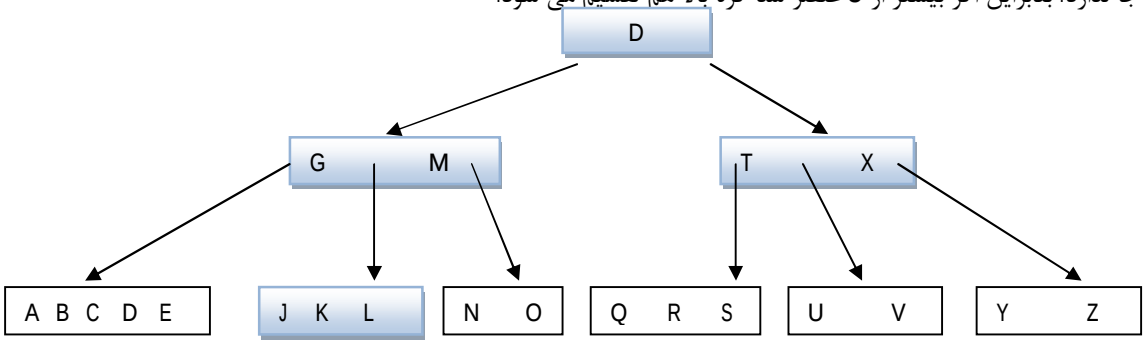


اضافه کردن کلید Q

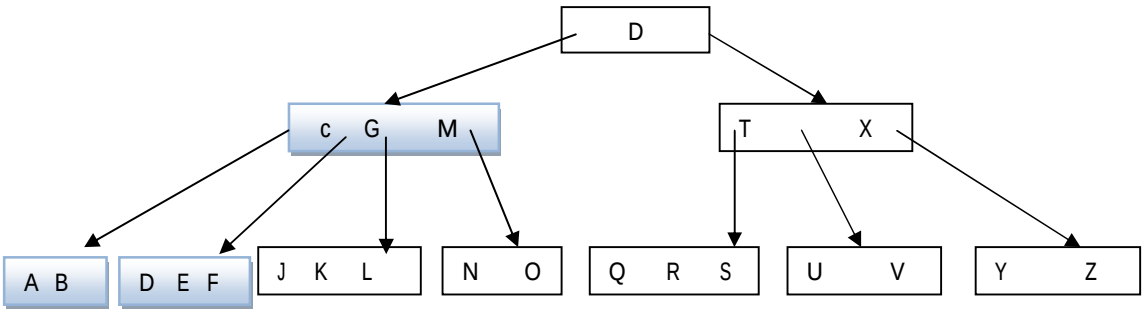
وقتی کلید Q اضافه شود 6 عنصر به وجود می آید. عنصر وسطی به بالا و بقیه عناصر را به 2 گره تقسیم می کنیم.



در نمودار قبل گره بالا 4 عنصر بود یکی بهش اضافه شد تبدیل به 5 عنصری شد ، چون درخت ما 5 مرتبه ای است بیشتر از این جا ندارد. بنابراین اگر بیشتر از 5 عنصر شد گره بالا هم تقسیم می شود.



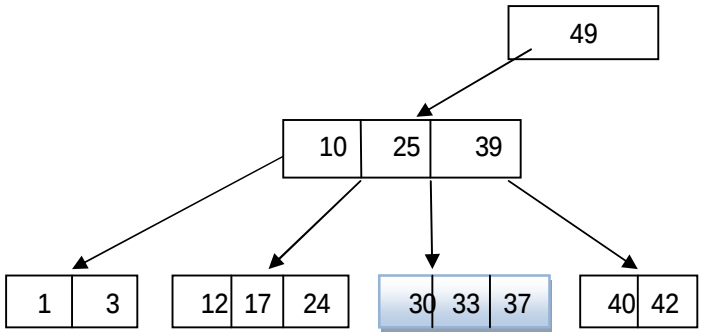
اضافه کردن کلید F



کلید F بعد از کلید E قرار دارد پس عنصر وسط C را به بالا می فرستیم و مابقی را به دو گره تقسیم می کنیم.

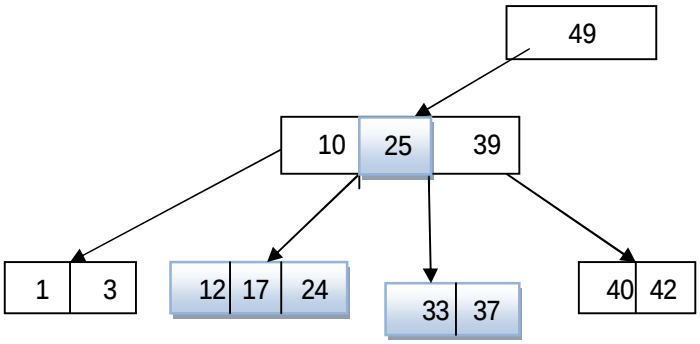
حذف (1)

در حالت عادی حذف اگر تعداد عناصر کمتر از 2 تا و بیشتر از 5 تا نباشد ایرادی ندارد مانند شکل زیر.

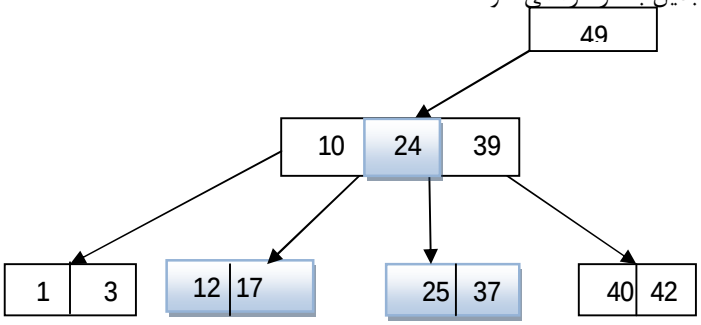


در اینجا به عنوان مثال اگر خانه 30 را حذف کنیم اتفاق خاصی نمی افتد زیرا با حذف 30 دو خانه ی دیگر (حداقل عناصر) وجود خواهند داشت.

حذف (2)

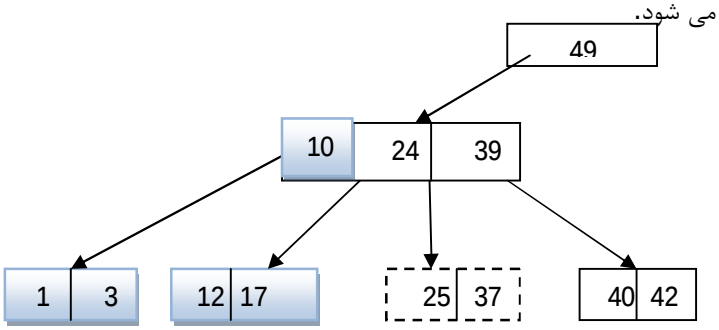


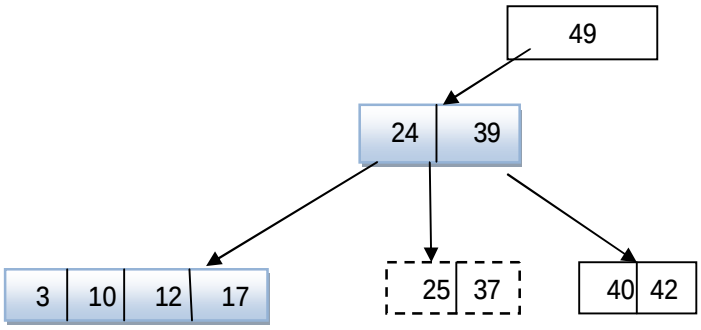
در این مثال اگر 33 حذف شود عنصر ما از حداقل کمتر می شود بنابراین ادغام می شود. پس 12 17 24 25 37 را یک بلاک در نظر می گیریم ، عنصر وسطی 24 است و ما بقی تبدیل به دو گره می شود.



حذف (3)

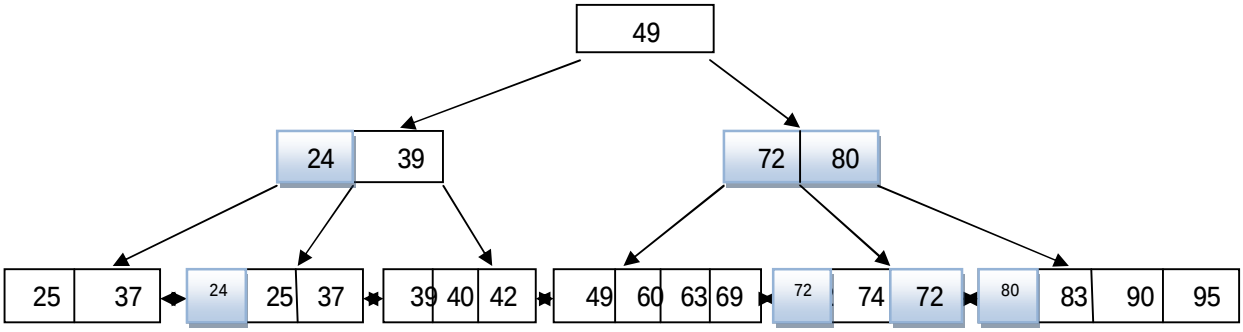
ممکن است پیچیده تر هم شود مثلا ما 1 را حذف کنیم ، 3 10 12 17 همگی 4 عنصراند و سپس باید 10 را هم از بلا بشکنیم به پایین. یعنی شکل نهایی ما به صورت زیر می شود.





درخت B^+ tree

در این حالت تمام عناصر برگ هستند یعنی وقتی 24 را در گره های میانی داریم در برگ هم داریم.

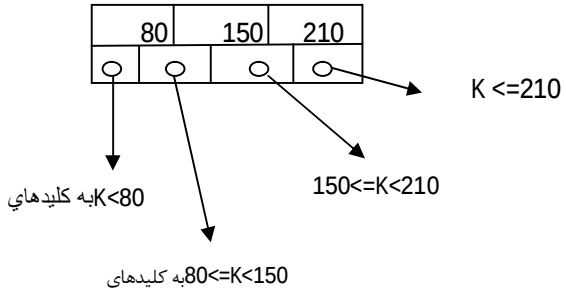


گره های برگ ، گره های اصلی ما هستند ، گره های وسطی گره هایی هستند برای دستیابی سریع.

هر بلاک به بلاک بعدی راه دارد. سرعت دستیابی درخت بیشتر است و بیشتر استفاده می شود.

ساختار داخلی گره ها

ساختار گره های داخلی درخت B^+ tree :



ساختار گره های برگ درخت B^+ به صورت زیر است: