

به نام ایزد دانا

جزوه درسی ذخیره و بازیابی اطلاعات

(ویژه داوطلبان آزمون کاردانی به کارشناسی – نرم افزار)

حافظه: هر دستگاهی که قادر به ذخیره سازی و بازیابی اطلاعات باشد.

انواع حافظه: اصلی و جانبی.

سیستم های فایل لینک: ذخیره سازی اطلاعات در محیط برون ماشین و بررسی مکانیزم دستیابی و بازیابی آن ها.

علت به کارگیری و استفاده از حافظه های جانبی این است که حافظه های درون ماشین گران اند و ظرفیت محدودی دارند.

چگالی ذخیره سازی اطلاعات به تعداد شیار در واحد طول بستگی دارد.

از نظر تعداد شیار ها دو نوع نوار وجود دارد: ۷ شیاره و ۹ شیاره

تعریف ۱ـ گپ فضای بلا استفاده بین دو گروه کاراکتر یا بلوک یا رکوردِ ضبط شده می باشد.

تعریف ۲ـ گپ از نظر ذخیره سازی اصطلاحاً حافظه Waste می باشد.

پارامترهای اساسی نوار: سرعت، چگالی، نرخ انتقال.

برای این که هد خواندن و نوشتن بتواند داده‌ای را حس کند باید پس از توقف به سرعتی مطلوب و یکنواخت موسوم به سرعت حس برسد که برای این کار فضای خالی گپ مورد نیاز است. همچنین برای رسیدن سرعت حس تا توقف کامل نیز فضای خالی گپ لازم می باشد.

به چند سکتور پشت سر هم یک کلاستر می گویند.

دیسک ها با هد ثابت سریع تر و گران تر از دیسک ها با هد متحرک می باشد.

طبله رسانه‌ای منطقاً معادل دیسک با نوک ثابت متشکل از یک استوانه با یک یا چند هد خواندن و نوشتن است و در قدیم به عنوان حافظه اصلی استفاده می شد.

حافظه کش حافظه ایست مابین CPU و RAM و جزء حافظه اصلی می باشد.

زمان پیگرد زمان لازم جهت انتقال هد به سیلندر است و متوسط این زمان را با حرف s نشان می دهند. این زمان حدود ۲ تا ۱۰ میلی ثانیه است.

زمان درنگ دورانی: پس از آن که هد به سیلندر مورد نظر رسید زمانی برای چرخش دیسک لازم است تا سکتور مورد نظر در زیر هد قرار بگیرد که به آن زمان درنگ دورانی می گویند. متوسط این زمان را با حرف R نشان می دهند که نصف زمان لازم جهت یک دور چرخیدن دیسک می باشد.

نوع موجودیت به فرد، شیء، پدیده یا مفهومی که می خواهیم در رابطه با آن اطلاعات داشته باشیم گفته می شود.

محیط عملیاتی: به محیطی که در رابطه با آن می خواهیم یک سری داده ها را در آن ذخیره، بازیابی یا پردازش کنیم گویند. مثلاً محیط عملیاتی دانشگاه از موجودیت های دانشجو، استاد، درس، کارمند، کلاس و ... تشکیل یافته است. انواع موجودیت ها توسط صفحات خاص مربوط به هر یک از سایر موجودیت ها متمایز می گردد. مثلاً موجودیت استاد میتواند صفحات خاصه‌ی مدرک، نام، آدرس، سابقه تدریس و ... را داشته باشد.

فیلد: مکان ذخیره شدن یک واحد معنادار یا یک فقره اطلاعات را فیلد گویند که کوچکترین واحد اطلاعات در فایل است.

اطلاع: هر صفت خاصه از دو مؤلفه تشکیل یافته است. یکی اسم صفت خاصه و دیگری مقدار صفت خاصه. به مجموع این دو مؤلفه اطلاع گفته می شود. اطلاع توسط انسان یا ماشین تولید، ذخیره، بازیابی و پردازش می شود. مثلاً نام خانوادگی صفت خاصه است و مثلاً احمدی مقدار صفت خاصه است.

رکورد: مجموعه‌ای از فیلدها تشکیل رکورد را می دهند و مجموعه‌ای از رکوردها فایل را تشکیل می دهند.

ساختارهای فیلد:

برای مشخص ساختن فیلدها در طول رکوردها راه حل‌های مختلف زیر وجود دارد:

۱ـ قرار دادن فیلدها در طول های از قبل تعیین شده.

۱۲ بایت	۲۰ بایت	۱۴ بایت
نام	فامیل	مدرک
علی	حسینی	لیسانس

یک ایراد این روش این است که برای رساندن فیلدها به طول معین می بایست از فاصله خالی استفاده شود و فضای خالی باعث بزرگ شدن اندازه فایل و اتلاف حافظه دیسک می گردد.

۲ـ قرار دادن طول فیلد در ابتدای هر فیلد.

03Ali 06Javadi 06Doctor 07Physics

۳ـ استفاده از یک کاراکتر ویژه به عنوان حد فاصل در انتهای هر فیلد.

Ali, Javadi, Doctor, Physics

۴ـ بکار بردن نام هر فیلد در مقابل مقدار هر فیلد. به عبارت دیگر استفاده از یک عبارت کلیدی برای شناسایی هر فیلد.

Name=Ali, Family=Javadi, City=Tehran

مزیت این ساختار آن است که فیلد خودتوصیف بوده و فیلدها می توانند جابجا شوند؛ همچنین مقادیر بعضی از فیلدها در صورت عدم وجود ذخیره نمی گردد. ایراد این روش اتلاف حافظه ایست که در اثر ذخیره ی نام فیلدها با آن مواجه می شویم.

ساختار رکوردها:

بعضی از روش های سازمان دهی رکوردها به صورت زیر می باشد:

۱ـ رکوردهایی با طول ثابت: در این روش طول همه رکوردهای فایل با هم برابر می باشد و این روش متداول ترین سازمان دهی رکوردهاست. ثابت بودن طول رکورد الزاماً به منظور ثابت بودن طول فیلدهای تشکیل دهنده آن نیست.

۲ـ تعیین طول رکوردها بر حسب تعداد فیلدهای آن: در این روش هر رکورد از n فیلد تشکیل یافته است و n برای کل فایل ثابت است. مثلاً اگر n = 4 آنگاه فایل می تواند به صورت زیر باشد:

Ali, Javadi, Doctor, Physics, Mohammad, Husseini, Doctor, Computer

۳ـ ذخیره طول رکورد در اول هر رکورد: در این روش در فیلدی در ابتدای هر رکورد طول آن ذخیره می شود. این روش اغلب برای کار با رکوردهای با طول متغیر بکار می رود.

۴ـ استفاده از اندیس برای آدرس های هر رکورد نسبت به اول فایل.

0	26	....

۵ـ ذخیره یک علامت ویژه فاصل در انتهای هر رکورد.

از یک نظر می توان گفت دو ساختار کلی جهت پیاده سازی رکوردها وجود دارد:

الف) رکورد با قالب ثابت و مکانی که تعداد، مکان و طول فیلدها در نمونه های مختلف ثابت بوده و تعریف این ساختار از قبل مشخص شده است.

نام	فامیل	رشته
علی	کریمی	برق
حسین	محمودی	فیزیک

ب) رکورد با قالب غیر ثابت و غیر مکانی که در هر فیلد، اسم فیلد به همراه مقدار آن ذخیره می شود.

نام=علی، فامیل=کریمی، رشته=برق
نام=حسین، رشته=فیزیک

طول یک رکورد بنا به دلایل زیر ممکن است متغیر شود:

الف) طول بعضی فیلدها مثل آدرس ممکن است متغیر باشد.

ب) تعداد فیلدهای نمونه های یک رکورد (موجودیت) ممکن است متغیر باشد. مثلاً موجودیت استاد ممکن است به دو دسته ی " رسمی با حقوق ثابت " و " حق التدریس " تقسیم گردد.

نوع رسمی: نام استاد، مدرک، رشته، حقوق ماهانه

نوع حق التدریس: نام استاد، مدرک، رشته، تعداد ساعات تدریس، حق الزحمه هر ساعت

ج) ممکن است در رکورد، فیلد (فقره اطلاع) تکرار شونده داشته باشیم.

نام	مدرک	دانشکده‌ای که تدریس می کند
اکبری	دکترا	برق، کامپیوتر
حسینی	دکترا	ریاضی، برق، کامپیوتر

از سه دیدگاه می توان به رکورد نگاه کرد:

الف) رکورد در سطح انتزاعی که رکورد را مستقل از جنبه های نمایشی آن و بصورت کلی نگاه می کنیم.

ب) رکورد در سطح منطقی که رکورد را از دیدگاه برنامه نویس مشخص می سازد و Sort شده است.

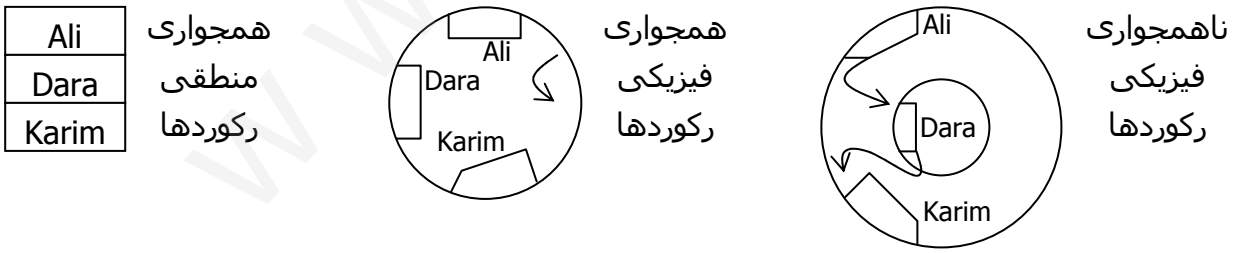
ج) رکورد ذخیره شده یا رکورد در سطح فیزیکی که رکورد را به صورتی که در محیط ذخیره سازی مثل دیسک قرار می‌گیرد معنی می سازد و ممکن است به آن اطلاع بیشتری اضافه شود و یا ساختار آن قدری تغییر کند و معمولاً رکورد ذخیره شده دارای دو بخش مجزای داده‌ای و کنترلی می باشد و به بخش کنترلی، بخش پیشوندی، بخش غیرداده‌ای یا Meta Section نیز گفته می شود. بخش کنترلی اغلب توسط سیستم فایل استفاده شده و از دید برنامه مخفی است.

اغلب در بخش کنترلی اطلاعات زیر ذخیره می شود: ۱ـ طول رکورد ۲ـ نوع رکورد ۳ـ اشاره گرها ۴ـ پرچم های عملیاتی و حفاظتی ۵ـ اطلاعاتی خاص، ویژه بعضی ساختارها

۱ـ طول رکورد: هنگامی که طول رکوردها متغیر باشد در بخش کنترلی طول آن رکورد ذخیره میشود و رکوردهای با طول ثابت به این اطلاع نیازی ندارند.

۲ـ نوع رکورد: ممکن است در یک فایل اطلاعات دو یا چند رکورد ذخیره شود (فایل چند نوعی) ممکن است در یک فایل هم اطلاعات اساتید و هم اطلاعات دانشجویان ذخیره گردد در اینجا نوع هر رکورد باید در ابتدای آن مشخص گردد و فایلی را که فقط یک نوع رکورد دارد، فایل تک نوعی می گویند.

۳ـ اشاره گرها: مثلاً پردازشگر فایل ممکن است رکوردهای اساتید را به ترتیب حروف الفبا مشاهده و پردازش کند؛ ولی این رکوردها که منطقاً مجاور یکدیگرند هنگام ذخیره شدن بر روی دیسک الزاماً به همان ترتیب نخواهند بود. در این حال با استفاده از اشاره گر ها ارتباط منطقی بین رکوردها پیاده سازی می گردد.



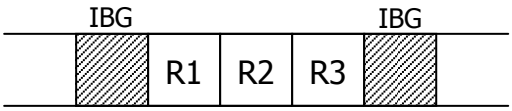
۴ـ پرچم (Flag): این پرچم ها برای نشان دادن عملیاتی که قرار است روی رکورد انجام بگیرد و یا نشان دادن عملیاتی که روی آن رکورد انجام شده بکار می روند. مثلاً در بسیاری از سیستم ها حذف به دو صورت منطقی و فیزیکی صورت می گیرد؛ بدین معنا که هنگام صدور فرمان حذف (جهت بالا بردن سرعت عملیات) تنها در ابتدای آن رکورد پرچمی "|" می شود (بدون حذف واقعی) و در این حالت مثلاً هنگام نمایش رکوردها آن‌هایی که علامت حذف خورده‌اند نشان داده نمی شوند و سپس در فرصتی مناسب این اطلاعات بطور فیزیکی حذف می شوند. همچنین در محیط های اشتراکی نیاز به پرچم های کنترلی است که نحوه دستیابی افراد را به رکوردها معین می سازد. مثلاً اگر پرچم Read Only برای کاربری فعال شود آنگاه آن کاربر نمی تواند رکورد را تغییر دهد.

۵ـ اطلاعاتی خاص، ویژه بعضی ساختارها: در ساختارهای مختلف فایل جهت پیاده سازی آن ها گاهی اوقات لازم است اطلاعات خاصی همراه رکوردها ذخیره گردد.

کلید (Key): صفت خاصه یا ترکیبی از چند صفت خاصه را که در تمام نمونه های مختلف یک موجودیت، مقدار یکتایی را داشته باشد، کلید می گویند. مثلاً در فایل اطلاعات دانشجویان فیلد شماره دانشجویی کلید است، زیرا هر شماره دانشجویی فقط یک دانشجو را معرفی می کند (شماره دانشجویی تکراری نیست).

فایل: مجموعه‌ای از نمونه های مختلف یک یا چند رکورد با ساختار مشخص است. فایل نیز دارای دو ساختار منطقی و فیزیکی است. فایل ها بر روی حافظه جانبی ذخیره شده و محتویات آن ها ماندگار است. اغلب فایل ها به قدری بزرگند که نمی توان آن ها را جهت پردازش بطور کامل به حافظه اصلی آورد. همچنین در حالت کلی فایل ها بصورت اشتراکی توسط چند کاربر استفاده می شوند.

بلاک بندی (Blocking): بلاک واحد رد و بدل کردن اطلاعات بین حافظه جانبی و حافظه اصلی توسط سیستم فایل است. البته در یک عمل I/O ممکن است چندین بلوک یکباره خوانده یا نوشته شوند. از نظر برنامه پردازشگر، فایل مجموعه‌ای از رکوردها با ساختار مشخص است ولی از نظر سیستم فایل، یک فایل از تعدادی بلاک تشکیل یافته است. نمایش ساده یک بلوک:



به تعداد رکوردهای موجود در هر بلاک ضریب بلاک بندی گفته می شود و آن را با BF (مخفف Blocking Factor) نمایش می دهیم. مابین بلاک ها یک فضای بلا استفاده (GAP) وجود دارد که باعث هدر رفتن فضای ذخیره سازی می گردد.

تعداد رکوردها را با n، تعداد بلوک ها را با b، اندازه هر بلوک را با B و اندازه ی هر رکورد را با R نمایش می دهیم. بنابراین داریم: $BF = \frac{B}{R}$

بلاک بندی در نوار توسط کاربر انجام گرفته و اندازه آن می تواند تغییر کند.

«GAP» بین بلوک ها در نوار جهت رسیدن سرعت هد به سرعت حس و یا توقف هد مورد نیاز است.

بلاک بندی در دیسک: بلاک در دیسک می تواند یک سکتور یا ترکیبی از چند سکتور سخت افزاری، یک شیار یا بخشی از یک شیار باشد. یک بلوک را نمی توان بین دو یا چند شیار تقسیم کرد.

شیارهای دیسک را می توان بر حسب سکتورها یا بر حسب بلوک ها تقسیم بندی کرد، تقسیم بندی بلوکی توسط کاربر و یا سیستم عامل انجام می پذیرد. بلوک واحد رد و بدل اطلاعات بین دیسک و حافظه است و بلوک ها می توانند طول ثابت یا متغیری داشته باشند که بستگی به نیاز طراح فایل و قابلیت های سیستم عامل دارد.

بلوک ها را مشابه سکتورها می توان رکوردهای فیزیکی در نظر گرفت. بلوک ها طوری سازماندهی می شوند که تعداد ثابتی از رکوردهای منطقی را نگهداری می کنند.

تکنیک های بلاک بندی:

۱ـ بلاک بندی رکوردها با طول ثابت و یکپاره

۲ـ بلاک بندی رکوردها با طول متغیر و یکپاره

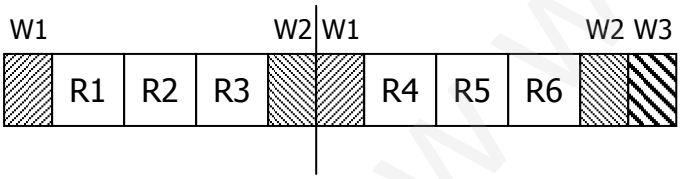
۳ـ بلاک بندی رکوردها با طول متغیر و دوپاره

در هر کدام از این سه مورد که بررسی کنیم باید دو مسئله را مد نظر داشته باشیم:

۱ـ فاکتور بلاک بندی؛ یعنی تعداد رکورد موجود در هر بلاک.

۲ـ حافظه هرز

روش اول:



W1 = G = گپ بین بلاک ها

W2 = حافظه هرز ناشی از جا نگرفتن رکورد اضافی در بلاک

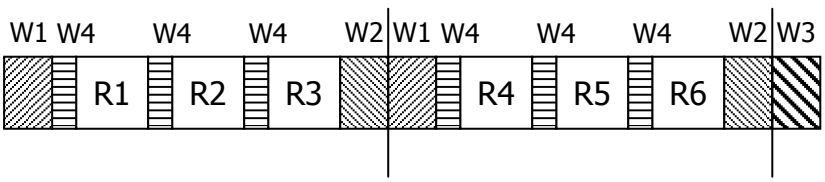
W3 = حافظه هرز ناشی از جا نگرفتن بلاک سوم در شیار

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

حافظه هرز به ازاء هر بلاک $WB = W1 + W2 + \frac{W3}{TF}$

حافظه هرز به ازاء هر رکورد $WR = \frac{WB}{BF}$

روش دوم:

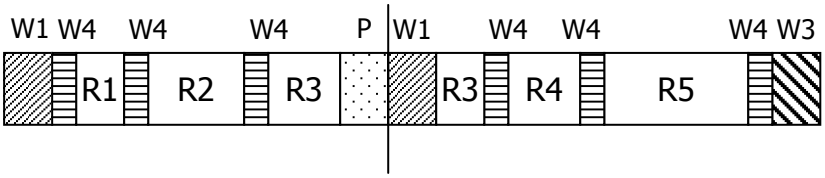


P = W4 = فضای هرز فیلدی که طول رکورد نوشته می شود

R = طول متوسط رکوردها

$$WR = P + \frac{1}{BF} \times \left(G + \frac{R}{2} + \frac{W3}{TF} \right)$$

روش سوم:



$$BF = \frac{B - P}{R + W4}$$

$$WB = W1 + P + BF \times W4 + \frac{W3}{TF}$$

$$WR = P + \frac{1}{BF} \times \left(G + P + \frac{W3}{TF} \right)$$

معیار سنجش یک نرم افزار: پیچیدگی خود نرم افزار، میزان حافظه‌ای که اشغال می کند، سرعت، کارایی بالا.

زمان حیات یک فایل: از وقتی که یک فایل ایجاد می شود تا وقتی که پاک می گردد.

تغییر طول فایل به دو علت می باشد: ۱ـ تعداد رکوردهای فایل عوض شود ۲ـ طول رکوردهای فایل عوض شود.

مزایای بلاک بندی: کاهش دفعات خواندن و نوشتن، کاهش حافظه هرز ناشی از وجود گپ بین رکوردها.

معایب بلاک بندی: کار نرم افزاری بیشتر، مصرف حافظه ی اصلی بیشتر، بالا رفتن احتمال خطا در اطلاعات به علت افزایش میزان اطلاعات انتقالی در یک عمل ورودی و خروجی (هر چه حجم اطلاعات بالا رود حجم خطا هم بالا میرود)

باکت بندی: مجموعه‌ای از تعدادی بلاک با حداقل طول یک بلاک می باشد. مزایا و معایب باکت بندی همان مزایا و معایب بلاک بندی است.

اگر بلاک بندی داشته باشیم واحد خواندن می شود بلاک و اگر باکت بندی داشته باشیم واحد خواندن و نوشتن می شود باکت و هر وقت با کل فایل کار داشتیم سیستم عامل مطرح است و هر وقت با رکورد کار داشتیم نرم افزار مطرح می شود.

اگر رکوردی دوپاره گردد در روش سوم بلاک بندی داشتیم، دوباره باید به سراغ دیسک برویم؛ یعنی دوبار خواندن و نوشتن صورت می گیرد ولی در روش باکت بندی این مشکل را نداریم.

چگالی Load اولیه در فایل ها:

تعریف Load اولیه در فایل ها: ایجاد اولیه ی یک فایل و اطلاعات اولیه ای که در فایل قرار می گیرد و لحظه Load اولیه فایل یک مقداری از بلاک ها را خالی می گذارند برای اضافه کردن اطلاعات بعدی. اگر پشت سر هم رکوردها را بنویسیم و یکدفعه فضای خالی بگذاریم، اگر رکوردی را اضافه بنماییم، همه رکوردها باید یک شیفت به جلو بخورند و این نرم‌افزاری است و بافری که تعیین می کنیم مقداری از آن را خالی می گذاریم و به سراغ بعدی می رویم.

$$\text{چگالی Load اولیه} = \frac{\text{مقدار فضای پر شده}}{\text{اندازه بلاک}} = \frac{LD}{B} < 1$$

مزایای چگالی Load اولیه: ۱ـ حافظه ی Locality (همسایگی) بیشتر در فایل (در اثر ناحیه ی رزرو شده) همسایگی و همجواری و نزدیکی بیشتر، نظم منطقی فایل با نظم فیزیکی آن. ۲ـ تسهیل عملیات روی فایل. مثلاً درج رکورد یا جستجوی رکورد.

معایب چگالی Load اولیه: افزایش حافظه ی هرز در Load اولیه (در اثر وجود ناحیه ی رزرو) و سبب افزایش طول خطی فایل می شود و از این رهگذر خواندن تمام فایل زمان گیرتر می شود.

همسایگی یا Locality: میزان نزدیکی رکورد منطقاً بعدی در محیط فیزیکی به رکورد فعلی.

هرچه میزان همسایگی کم شود، زمان دسترسی به رکورد بعدی زیاد می شود.

سطوح نشانی دهی به فایل (آدرس دهی به فایل): بسته به این که از چه دیدگاهی به فایل نگاه می کنیم، میزان آدرس دهی فرق می کند و بحث بر روی سیستم فایل است.

سیستم فایل از دو قسمت تشکیل شده است: ۱ـ بخش منطقی ۲ـ بخش فیزیکی

بخش منطقی سیستم فایل: وظیفه اش اجرای درخواست های کاربر است و یک سری عملیاتی را که کاربر احتیاج دارد، انجام می دهد و معمولاً عملیات عبارتند از: بازکردن فایل، خواندن فایل (رکورد)، و نوشتن فایل (رکورد)، بستن فایل و جستجوی یک رکورد.

بخش فیزیکی سیستم فایل: وظیفه اش دریافت دستورات از بخش منطقی و تبدیل آن ها به فرامین کنترل کننده ی رسانه ی ذخیره سازی است.

سطوح برخورد با فایل: ۱ـ سطح برخورد با کاربر ۲ـ سطح برخورد بخش منطقی سیستم فایل ۳ـ سطح برخورد بخش فیزیکی سیستم فایل.

نشانی دهی کاربر (پردازشگر فایل): منظور آدرس دهی به یک رکورد است؛ یعنی مشخص کردن یک رکورد خاص.

نوع نشانی دهی کاربر سه قسمت است: ۱ـ نشانی دهی محتوایی: از نظر مشخص کردن محتوای یکی از فیلدها، و معمولاً این فیلد، کلید آن رکورد است. ۲ـ نشانی دهی نسبی در یک فایل: یک شماره به رکورد می دهیم و وقتی کاربر رکورد مشخصی را صدا می کند آن شماره را می دهد. ۳ـ نشانی دهی نمادی یا

سمبلی: برای بعضی رکوردها اسم سمبلیک می گذاریم. مثلاً برای دانشجویانی که معدلشان بالاست واژه ی First یعنی رتبه ی اول و Second یعنی دانشجویان رتبه ی دوم به عنوان سمبلیک که صدا کردن آن رکورد از طریق این اسم می باشد.

آدرس دهی فیزیکی شامل این موارد می شود: ۱ـ شماره رسانه ۲ـ شماره استوانه ۳ـ شماره شیار ۴ـ شماره بلاک

دسترسی اطلاعات در یک فایل، اجرای درخواست کاربر توسط سیستم فایل: ۱ـ بازکردن فایل ۲ـ خواندن یک رکورد ۳ـ نوشتن یک رکورد ۴ـ بستن فایل

منظور از بازکردن یک فایل: ابتدا فایل را روی رسانه پیدا کنیم و بعد مشخصات فایل از روی دیسک باید خوانده شود و سپس بررسی حق دستیابی کاربر

تخصیص بافر به فایل: وقتی که می خواهیم با فایلی کار کنیم، خواندن و نوشتن از طریق بافر انجام می گیرد.

مشخصات فایل و بررسی حق دستیابی کاربر را راهنمای فایل می گویند.

راهنمای فایل: ۱ـ اسم فایل ۲ـ اسم صاحب فایل ۳ـ تعداد رکوردهای منطقی در فایل ۴ـ طول رکوردها (رکورد ثابت) ۵ـ آدرس اولین رکورد ۶ـ اندازه ی فایل ۷ـ تاریخ ایجاد فایل ۸ـ حقوق دستیابی به فایل (Password) ۹ـ تاریخ آخرین تغییرات در فایل

بافرها: بخشی از حافظه ی اصلی است که واسطه می شود بین بخش های داخلی کامپیوتر (CPU)، ورودی و خروجی ها.

علت استفاده از بافرها: بالا بردن سرعت و هماهنگی کردن بین سرعت CPU و لوازم جانبی.

هر فایلی که باز می کنیم سیستم عامل یک بافر برای آن فایل در نظر می گیرد و بر ای خواندن و نوشتن دو بافر در نظر می گیریم و این بافرها را سیستم عامل در ناحیه جمع می کند و آن را ناحیه ی بافرها (Buffer Pool) می گویند.

روش های ایجاد بافر: به سه روش ممکن است یک بافر ایجاد شود: ۱ـ به طور خودکار توسط سیستم عامل ۲ـ توسط برنامه نویس و با درخواست از سیستم عامل ۳ـ توسط برنامه نویس (سیستم فایل با استفاده از امکانات زبان)

روش های دستیابی به بافرها: ۱ـ روش انتقالی ۲ـ روش مکان یابی

در روش انتقالی سیستم فایل یا سیستم عامل اطلاعات را می خواند و داخل بافر می گذارد و یک حافظه ی کاری برای خودش دارد. اطلاعات مورد نظر از بافر منتقل می شود به حافظه ی کاری و بعداً در آن جا پردازش می شود و نتیجه بر می گردد.

در روش مکان یابی آدرس اطلاعات منتقل می شود به برنامه و برنامه می آید مستقیماً با اطلاعات داخل بافر کار می کند.

انواع بافر از نظر محل قرار گیری: ۱ـ بافرهای سخت افزاری: آن بافری هستند که داخل سخت افزار قرار می گیرند و همچنین در دستگاه های جانبی. ۲ـ بافرهای نرم افزاری: آن بافری است که در حافظه ی اصلی قرار می گیرد و از طریق نرم افزار (برنامه ها) کنترل می گردد.

مدیریت بافرها: در سیستم چند کاربره این واحد حتماً باید باشد که مدیریت روی اندازه ی بافرها ، تعدادشان و اندازه و زمان ایجاد و حذف و نوع بافر و ... کنترل این ها را مدیریت بافر گویند. در سیستم های تک کاربره هم اگر چند فایل بخواهیم داشته باشیم، مدیریت بافرها بسیار مهم است.

انواع بافرها از نظر ساختمانی (بحث در مورد بافرهای نرم افزاری است و بافرهای سخت افزاری قابل تغییر نیستند): ۱ـ بافرینگ ساده ۲ـ بافرینگ مضاعف ۳ـ بافرینگ چندگانه: الف) صف معمولی ب) صف حلقوی (بافرینگ چرخشی)

۱ـ بافرینگ ساده تکی است و بافرینگ مضاعف دوتایی می باشد. بافرینگ ساده مدیریتش ساده است ولی سرعت عملیات را پایین می آورد و زمان انتظار برای CPU دارد.

۲ـ بافرینگ مضاعف مدیریتش پیچیده تر است ولی سرعت بالایی دارد.

۳ـ بافرینگ چندگانه: چند بافر را در نظر می گیریم و در دو حالت یکی به صورت صف و دیگری به صورت حلقوی.

نرخ انتقال اطلاعات: میزان اطلاعاتی که در یک ثانیه می شود منتقل گردد و واحدش بایت در ثانیه می باشد.

سرعت واقعی انتقال اطلاعات← زمان درنگ دورانی (r) + زمان استوانه جویی (s) = زمان دسترسی تصادفی

زمان انتقال یک رکورد (Rtt):

$$Rtt = \frac{\text{طول رکورد } R}{\text{زمان انتقال بایت در یک ثانیه } t}$$

زمان انتقال یک بلاک (Btt):

$$Btt = \frac{\text{طول بلاک } B}{\text{نرخ انتقال اطلاعات } t}$$

$$\text{سرعت انتقال واقعی یک بلاک} = \frac{\text{اندازه بلاک } B}{\text{زمان انتقال بلاک } Btt + \text{زمان دستیابی } str}$$

هدف از طراحی سیستم های فایل دو مسئله است: ۱ـ دست یابی سریع به اطلاعات ۲ـ استفاده ی مفید از حافظه که سعی می کنیم این دو پارامتر را برای سرعت بیشتر تقویت کنیم.

ضابطه های اساسی در سیستم های فایل (طراحی فایل ها): ۱ـ حداقل بودن افزونگی: در سیستم فایل تا آن جا که امکان دارد باید از اطلاعات تکراری کاست. ۲ـ دست یابی سریع به اطلاعات: در فایل های خیلی بزرگ روش های معمولی جست و جو روش کندی می باشد و باید تکنیک هایی به کار برد که مستقیم به آن نقطه ی دلخواه برسیم. ۳ـ سهولت عملیات به هنگام سازی: منظور این است که فایل را از نظر ایجاد تغییرات در آن، وضعیت آخرین تغییرات را در آن ثبت کنیم یا تغییرات در فایل را به هنگام سازی نماییم. ۴ـ سهولت نگهداری سیستم: سیستم در حین کار ممکن است یک سری تغییرات داشته باشد که بتواند به راحتی این تغییرات را اعمال نماید. مثلاً ورژن ها که تغییر می کنند، تغییرات به راحتی صورت گیرند. ۵ـ ضریب اطمینان بالا: یک سیستم مثل سیستم فایل که با اطلاعات سر و کار دارد باید ضریب اطمینان بالایی داشته باشد. مثلاً سیستم در موقع قطع برق حداقل از بین رفتن اطلاعات را داشته باشد. یا مثلاً در یک سیستم بیمارستان نباید اطلاعات غلط بدهیم؛ چرا که با جان انسان ها سر و کار داریم.

ملاک های ارزیابی فایل ها: ۱ـ متوسط اندازه ی رکورد که علاوه بر اطلاعات رکورد اطلاعات سیستمی در رکورد را نیز داریم. ۲ـ زمان لازم بر ای واکشی TF: گرفتن اطلاعات از جایی ۳ـ زمان لازم بر ای به دست آوردن رکورد بعدی TN: زمانی که لازم است تا رکورد بعدی را پیدا کنیم. چون به لحاظ فیزیکی رکوردها پشت سر هم

نیستند. ۴ـ زمان لازم برای به هنگام سازی از طریق درج یک رکورد TI: یعنی یک رکورد را ببریم سر جایش و به ترتیب قرار دهیم. ۵ـ زمان لازم بر ای به هنگام سازی از طریق تغییر یک رکورد TU: مثلاً فیلدهای یکی از رکوردها را عوض کنیم. مثلاً دانشجویی که درسش را حذف کرده؛ و تغییراتی در رکورد بدهیم. ۶ـ زمان لازم برای خواندن همه ی فایل TX (خواندن سری): مثلاً مسئول یک دانشگاه بخواهد لیستی از تمام دانشجویان داشته باشد، چقدر زمان لازم است؟ ۷ـ زمان لازم برای سازمان دهی مجدد TY: دوباره سازمان دهی کردن: مثلاً یک فایل را حذف کنیم و فایل دیگری را بنویسیم. لازم است دوباره آن را مجدداً بازسازی و سازمان دهی کنیم که چقدر زمان می برد.

عملیاتی که در یک فایل می توان انجام داد و سیستم فایل باید امکانات زیر را بدهد (عملیات بخش منطقی) و این عملیات از نظر User مطرح می باشد: ۱ـ دسترسی به رکورد مورد نظر (واکشی) TF ۲ـ به دست آوردن رکورد بعدی TN ۳ـ درج رکورد TI ۴ـ تغییر رکورد فعلی TU ۵ـ خواند فایل TX ۶ـ سازمان دهی مجدد TY و این شش عمل را می توان در یک سیستم فایل انجام داد.

در بخش فیزیکی سه عمل کلّاً انجام می شود (این ها از نظر هد و دیسک مطرح اند):

۱ـ یافتن یا Seek ۲ـ خواندن یا Read ۳ـ نوشتن یا Write

شرح ضوابط مربوط به فایل ها: ۱ـ متوسط اندازه ی رکورد: مثلاً رکوردی که برای اطلاعات یک دانشجو در نظر می گیریم غیر از این ساختارها یک سری اطلاعات سیستمی اضافه می شود و اطلاعات سیستمی که اضافه کنیم باید بدانیم که چقدر و چه مقدار می باشد و هرچه کمتر باشد رکورد کمتر می شود و کل فایل کمتر می شود.

در سطح رکوردها فایل ها را داریم که عبارتند از: الف) متراکم: فایلی بدون فیلد خالی. ب) فایل های پراکنده: فایلی با تعدادی فیلد خالی. مثلاً برای دانشجویان ده درس در نظر گرفته ایم. دانشجویی که سه درس را گرفته باشد بقیه ی فیلدها خالی می مانند و اندازه ی فایل بزرگ می شود چون که رکوردهایی وجود دارند که فیلدی از آن خالی است. ج) فایل هایی دارای افزونگی.

افزونگی یعنی چه؟ فایلی که مقادیر بعضی از صفات خاصه اش چند بار تکرار شده باشند (در رکوردهای مختلف) دارای افزونگی می باشد. مثلاً که درس می گیرد صفات خاصه ی درس تکرار می شود و عنوان درس نیز چندین بار تکرار می گردد و یا نام استاد چند بار تکرار می شود که به این افزونگی می گویند. افزونگی دو نوع است: الف) طبیعی (به خاطر شرایط موجود) در محیط عملیاتی وجود دارد مثلاً شماره ی درس. و گاهی به خاطر مسائل تکنیکی که سیستم فایل می خواهد مجبوریم یک سری اطلاعات تکراری را نگه داریم. ب) تکنیکی: به خاطر ایجاد یک استراتژی دستیابی خاص برای فایل تکرار بعضی یا تمام مقادیر یک یا چند صفت خاصه در محیط فیزیکی ذخیره سازی.

برای این که یک رکورد را پیدا کنیم چند روش Sort وجود دارد (روش های جست و جو در فایل یا استراتژی واکشی یک رکورد): ۱ـ دستیابی با جست و جوی خطی ۲ـ دستیابی با جست و جوی بلاکی ۳ـ دستیابی با جست و جوی باینری ۴ـ دستیابی با جست و جو به کمک شاخص در محیط ترتیبی ۵ـ دستیابی با جست و جو به کمک شاخص در محیط غیر ترتیبی ۶ـ دستیابی با به دست آوردن آدرس از روی کلید (دستیابی مستقیم) یا از روی شماره ی نسبی رکورد در فایل ۷ـ دستیابی با استراتژی ترکیبی

هر چه پایین برویم سرعت زیاد می شود پیچیدگی نیز زیاد می شود و اتلاف حافظه نیز داریم. چون سرعت زیاد می شود به خاطر مکانیزم هایی که به کار می بریم.

۱ـ دستیابی با جست و جوی خطی: از اول فایل شروع می کنیم و رکوردها را یکی یکی می خوانیم تا برسیم به رکورد مورد نظر که این کندترین روش و ساده ترین روش می باشد ۲ـ دستیابی با جست و جوی بلاکی: فایل ها بلاک بندی شده باشند و بلاک ها نظمی نیز داشته باشند. اگر به بلاکی رسیدیم که در آن به نقطه ی مورد نظر رسیدیم خیلی خوب و سرعتش بهتر است چرا که مجموع رکوردها را با هم می خوانیم ۳ـ دستیابی با جست و جوی باینری: ابتدا باید رکوردها مرتب شده باشند و با نصف کردن متوالی به رکورد مورد نظر می رسیم. ۴ـ دستیابی با جست و جو به کمک شاخص در محیط ترتیبی: فایل ترتیبی و Sort شده است. علاوه بر آن یکسری اشاره گر جدا داریم که آن ها ما را هدایت می کنند به نقطه ی مورد نظر و شاخص کمک می کند که فایل را در یک محدوده ی کوچک بررسی نماییم. ۵ـ دستیابی با جست و جو به کمک شاخص در محیط غیر ترتیبی: در محیط غیر ترتیبی چون نظمی ندارند در یک محدوده نمی شود بررسی شود. ۶ـ دستیابی با به دست آوردن آدرس از روی کلید: با فایل هایی که به صورت تصادفی هستند سر و کار داریم و همان فایل های ترتیبی هستند با چگالی بالا. سرعت بالاست منتها اتلاف حافظه به همراه دارد.م ۷ـ روشی از ترکیب شش روش بالا: از دو یا چند استراتژی بالا استفاده می کنیم تا به رکورد مورد نظر برسیم.

دلایل کاهش کارایی فایل: ۱ـ از بین رفتن نظم ساختاری اولیه ۲ـ بروز حافظه های هرز در فایل ۳ـ بروز وضعیت نامطلوب در استراتژی دست یابی (شاخص دار)

عملیاتی که باید برای سازمان دهی مجدد انجام گیرد: ۱ـ خواندن تمام فایل ۲ـ خارج کردن رکوردهای حذفی ۳ـ فرمت بندی مجدد بلاک ها ۴ـ بازنویسی رکوردهای فعال ۵ـ بازسازی ساختار مربوط به استراتژی دست یابی (تصحیح فایلی که در آن آدرس ها نگهداری می شوند)

زمان بازنویسی بلاک ها btt + str

خواندن فایل ها به دو روش است: ۱ـ ترتیب منطقی رکوردها ۲ـ ترتیب فیزیکی رکوردها

به دست آوردن رکورد بعدی (Get Next): این عمل را یک عمل ساختاری می گویند و از محتوای یک فیلد استفاده می کنیم و رکورد بعدی از رکورد فعلی به دست می آید. رکورد فعلی محتوایی و رکورد بعدی ساختاری است.

به هنگام سازی از طریق تغییر محتوای رکورد: وقتی می خواهیم در فایلی در یک رکوردش تغییر ایجاد نماییم و نیازی به اضافه کردن رکورد جدید نداشته باشیم، دو حالت پیش می آید: ۱ـ یا با تغییر طول رکورد فایل تغییر نکند؛ این را به هنگام سازی درجا یا In Place می گویند. مثلا فقط اسم را عوض کنیم. ۲ـ اگر مثلا شماره ی دانشجویی را عوض کنیم، ساختار فایل تغییر می کند چون Sort شماره ها به هم می خورد. اگر رکورد تغییر طول بدهد با تغییر طول یک فیلد؛ در این حالت به هنگام سازی را برون از جا می گویند که مجبوریم رکورد را از جا برداریم و جای دیگری قرار دهیم.

عمل Delete کردن در فایل یک عمل مستقل نیست و شکل خاصی از Update کردن می باشد.

پایان ذخیره و بازیابی۱

***

موقعیت رکورد بعدی به رکورد فعلی می تواند یکی از سه حالت زیر باشد:

۱ـ هیچ ارتباطی بین آن ها وجود نداشته باشد.

۲ـ همجوار فیزیکی باشند.

۳ـ از رکورد فعلی به رکورد بعدی اشاره گری وجود داشته باشد.

زمان واکشی در یک فایل پایل برابر است با: برای بلاک $TF = \frac{1}{2} b \frac{B}{t'}$ و برای رکورد $TF = \frac{1}{2} n \frac{R}{t'}$

در ساختار فایل پایل TN=TF ، چون ارتباط ساختاری بین رکورد فعلی و بعدی وجود ندارد.

عملیاتی که در بافر انجام می شود را در ارزیابی دخالت نمی دهیم چون زمان آن بسیار کم است.

حذف حالتی خاص از به هنگام سازی است؛ پس TUD=TF+TRW .

فایل پایل را نمی توان به صورت سریال خواند. چون بازیابی رکورد بعدی عملی نمی باشد ولی زمان خواندن پی در پی فایل برابر 2TF می باشد.

چه عواملی در ارزیابی متوسط اندازه ی رکورد دخیل اند:

۱ـ بخش داده ای و غیر داده ای رکورد ۲ـ متراکم یا غیر متراکم بودن فایل ۳ـ وجود یا عدم وجود افزونگی در فایل

فایلی متراکم است که همه ی مقادیر صفات خاصه ی همه ی رکوردهایش مشخص باشد.

فایلی غیر متراکم است که بعضی از مقادیر بعضی از صفات خاصه ی بعضی از رکوردهایش موجود نباشد.

فایل را دارای افزونگی گوییم که مقادیر بعضی از صفات خاصه اش بیش از یک بار در محیط فیزیکی ذخیره شده باشند.

اگر N تعداد عناصر مجموعه ای باشد که مقادیر صفات خاصه ی مورد نظر از آن گرفته شده است، برای ذخیره سازی تمام این صفات به N بیت حافظه نیاز داریم.

واکشی یک رکورد دلخواه عملی است محتوایی و واکشی رکورد بعدی عملی است ساختاری.

عمل درج رکورد، مجموعه ای از عملیات لازم دارد که حجم آن بستگی به نوع ساختار فایل دارد.

عمل درج و به هنگام سازی، هر دو، محیط ذخیره سازی را تغییر می دهند.

حالاتی را که نیاز به خواندن تمام فایل می باشد، نام ببرید:

۱ـ سازمان دهی مجدد فایل ۲ـ ایجاد نسخه ای دیگر از فایل ۳ـ ایجاد یک استراتژی دستیابی

چه زمانی احتیاج به خواند تمام فایل نمی باشد؟ به هنگام سازی

موارد استفاده ی ساختار فایل پایل را بنویسید:

۱ـ در محیط هایی که در آن ها داده ها نظم پذیر نمی باشند ۲ـ بایگانی اطلاعات

به چه دلیلی فایل پایل سازمان دهی مجدد می شود؟ خارج کردن حافظه ی هرز

در ایجاد نسخه ی دیگری از فایل نیاز به سازمان دهی مجدد نمی باشد.

در یک فایل پایل عملیات لازم برای انجام عمل درج به ترتیب برابر است با:

۱ـ خواندن آخرین بلاک فایل که سیستم آدرس آن را دارد ۲ـ انتقال رکورد از ناحیه ی کاربری برنامه به بلاک ۳ـ بازنویسی بلاک

در فایل پایل بین رکورد بعدی و فعلی هیچ گونه ارتباط ساختاری وجود ندارد و واکشی رکورد بعدی به یک عمل واکشی نیاز دارد.

***

تمرینات ذخیره و بازیابی

۱ـ زمان خواندن کل فایل پایلی به صورت ترتیبی با مشخصات زیر چند ثانیه خواهد بود؟

تعداد بلاک=۱۰۰

اندازه هر بلاک=۲۰۰۰ بایت

نرخ انتقال =۴۰۰ بایت در ثانیه

زمان خواندن ترتیبی فایل پایل برابر با 2TF

$$TF\frac{1}{2}\times 100\times\frac{2000}{4000}=25s$$

$$T_{x\,seq}=2TF=5s$$

۲ـ فایل دانشجو ـ درس برای ۱۷ دانشجو در تکنیک ماتریس بیتی چند بایت اشغال خواهد کرد؟

در تکنیک ماتریس بیتی برای ۱۷ درس به ۱۷ بیت نیاز داریم یعنی ۳ بایت. همچنین فیلد دانشجو نیز ۲ بایت در بر می گیرد.

تعداد درس=۱۷

حداکثر تعداد دروسی که یک دانشجو می تواند ثبت نام کند=۶

بایت ۵۰=۱۰×(۲+۳)

۳ـ زمان بدست آوردن رکورد بعدی در فایل پایلی با مشخصات زیر چند دقیقه می باشد؟

تعداد بلاک ها=۳۶۰

اندازه ی هر بلاک=۲۰۰۰بایت

نرخ انتقال=۳۰۰۰ بایت در ثانیه

$$T_A=\frac{1}{2}\times 360\times\frac{2000}{3000}=120s$$

$$T_A=2\,\text{دقیقه}$$

۴ـ فایل پایلی با ۱۰ رکورد ۴ بایتی مفروض است. به ازاء هر ۲ رکورد درج شده یک رکورد حذف شده داریم و این عمل ادامه می یابد تا تعداد رکوردهای فعال برابر ۱۳ شوند. زمان واکشی یک رکورد قبل از سازمان دهی مجدد چند میلی ثانیه خواهد بود؟

تعداد رکوردهای فایل وقتی از ۱۰ به ۱۵ می رسد که ۵ بار عمل گفته شده انجام شود (به ازاء هر دو درج یک حذف).

تعداد اضافه شده $10 + 2x - x = 15 \Rightarrow x = 5$

$$B = 8$$

$$\frac{B}{t'} = 0.5ms$$

در حالتی که تعداد رکوردهای فعال ۱۵ می باشد ۵ رکورد با علامت حذف شده موجود است و تعداد کل رکوردها ۲۰ می باشد.

$$T_F = \frac{b}{2} \times \frac{B}{t'} = \frac{10}{2} \times 0.5 = 2.5 \, m/s$$

$$\text{تعداد کل بایت‌ها} = \frac{\text{تعداد رکوردها} \times \text{اندازه رکورد}}{\text{ظرفیت‌هر بلاک}} = \frac{20 \times 4}{8} = 10 \text{ بلاک}$$

زمان واکشی رکورد بعد از سازماندهی مجدد چند میلی ثانیه است؟

بعد از سازماندهی مجدد رکوردها با علامت حذف شده خارج می شوند. بنابراین تنها 10 رکورد فعال در فایل وجود دارد.

$$\text{طول بلاک} = b = \frac{n \times R}{B} = \frac{10 \times 4}{8} = 5$$

یعنی بعد از سازماندهی مجدد TF کاهش می یابد. $T_F = \frac{b}{2} \cdot \frac{B}{t'} = \frac{5}{2} \times 0.5 = 1.25ms$

کد درس / شماره دانشجویی	۱۷۶	۱۷۷	۱۷۸	۱۷۹	...	صفر به معنی عدم انتخاب و یک به معنی انتخاب
۵۴۲۸۱	۱	۰	۱	۱		
۵۴۲۸۲	۰	۰	۱	۰		
۵۴۲۸۳	۱	۱	۰	۰		
۵۴۲۸۴	۰	۰	۱	۰		
...						

۵ـ در یک فایل پایل $T_W = 2ms$ و تعداد دور دیسک ۳۰۰۰ دور در دقیقه است. زمان حذف چند میلی ثانیه می باشد؟

$$TD = ?$$

$$TD = T_F + T_{RW}$$

$$T_F = T_N$$

$$\text{یک دور دیسک} \quad 2r = \frac{60 \times 1000}{3000} = 20ms$$

$$TD = 2 + 20 = 22ms \text{ میلي ثانیه}$$

۶ـ اگر تعداد رکوردهای یک فایل 10^6 باشد، مناسب ترین اندازه برا یBF چند باید در نظر گرفته شود تا تعداد رکوردهای بررسی شونده در جست و جوی پرش بلاکی به حداقل برسد؟

$$BF = \sqrt{n} = \sqrt{10^6} = 10^3 = 1000$$

۷ـ در فایل ترتیبی با مشخصات زیر تعداد دفعات مراجعه به فایل کدام است؟

n=20480

BF=20

$$BF = \frac{B}{R} \Rightarrow \frac{1}{BF} = \frac{R}{B}$$

$$\log_2 \frac{nR}{b} = \log_2 \frac{n}{BF} = \log_2 \frac{20480}{20} = \log_2 1024 = \log_2 2^{10} = 10 \log_2 2 = 10$$

۸ـ زمان واکشی یک رکورد از فایل ترتیبی با مشخصات زیر چند میلی ثانیه می باشد؟

$t' = 2000\,kb/s$

$R = 200B$

$n = 100$

$$TF = \frac{n}{2} \times \frac{R}{t'} = \frac{1000}{2} \times \frac{200}{2000 \times 2^{10}} = 0.04s = 40ms$$

۹ـ یک فایل ترتیبی داریم که روی دیسک ذخیره شده است. زمان خواند رکوردهای این فایل به ترتیب معکوس به صورت ترتیبی کدام است؟

$n = 400$

$R = 100B$

$t' = 250000\,B/s$

$$T_{x\,ser} = \frac{nR}{t'} = \frac{400 \times 100}{250000} = 0.16 = 160ms$$

۱۰ـ در یک فایل ترتیبی اگر تعداد صفات خاصه ۴ و متوسط حافظه ی لازم برای ذخیره سازی مقدار صفت خاصه ۱۰ بایت و تعداد کل رکوردها ۲۰ و مقدار $2\,kb/s$ انتقال وجود داشته باشد، مطلوب است TF در صورتی که نشانه گر جست و جو غیر از صفت خاصه منظم باشد (یعنی کلید اصلی نباشد).

متوسط اندازه ی رکورد در ساختار ترتیبی برابر است با $R = a.V = 4 \times 10 = 40$

$$TF = \frac{n}{2} \times \frac{R}{t'} = \frac{20}{2} \times \frac{40}{2 \times 1024} = 0.195s = 195ms$$

موارد استفاده ی فایل ترتیبی:

۱ـ در کاربردهای تجاری ۲ـ تغییر طول رکورد مطرح نباشد ۳ـ در ایجاد بعضی از ساختارها لازم است ابتدا فایل به صورت ترتیبی ایجاد گردد.

در فایل ترتیبی تمام نمونه ی رکوردها قالب از قبل طراحی شده ای دارند و رکوردها دارای قالب ثابت مکانی می باشند و در Load اولیه تمام نمونه ی رکوردها بر اساس مقادیر یکی از صفات خاصه ی موجود در فایل منظم می باشند و این نظم با همجواری فیزیکی رکوردها پیاده سازی می گردد.

مزایای فایل ترتیبی نسبت به پایل: ساده تر بودن قالب رکوردها و تسهیل در پردازش سریال رکوردها و وجود یک استراتژی دستیابی می باشد.

معایب فایل ترتیبی نسبت به فایل پایل: کاهش انعطاف پذیری ساختار در عملیات تغییر دهنده مثل درج، حذف و به هنگام سازی ـ پدیده ی عدم تقارن ـ کاهش انعطاف پذیری ساختار از نظر طول رکورد.

عملیات تغییر دهنده (تراکنش) در فایل ترتیبی در چه فایلی صورت می گیرد؟ فایل ثبت تراکنش TLF یا سر ریزی

بعد از چه مدتی فایل TLF را در فایل اصلی ادغام می کنیم؟ مدتی که طراح تعیین می کند یا در یک پریود زمانی ثابت

متوسط اندازه ی رکورد و ظرفیت کل فایل در ساختار ترتیبی چه می باشد؟ R=a.V ظرفیت کل فایل (n+o)a.V

تعداد دفعات مراجعه به فایل ترتیبی در روش جست و جوی باینری چه می باشد؟ $\log_2 \frac{nR}{B}$

بازیابی رکورد بعدی در یک فایل ترتیبی هنگامی امکان پذیر نمی باشد که رکورد بعدی در فایل ثبت تراکنش یا TLF باشد.

بین رکورد در ناحیه ی اصلی و رکورد در TLF هیچ گونه ارتباط ساختاری وجود ندارد. مثلاً از طریق اشاره گر ها. بنابراین اگر رکورد بعدی در TLF باشد به دست آوردن آن ممکن نیست. مگر این که آدرس آن را داشته باشیم که در این حالت به یک عمل واکشی در TLF تبدیل می گردد.

***

نکات آخرین جلسه

تکنیک ماتریس بیتی تکنیکی است برای فشرده سازی وقتی که صفت خاصه ی چند مقداری وجود داشته باشد و در شرایطی که هم طول رکوردها متغیر و هم افزونگی طبیعی تشدید می شود، تکنیک ماتریس بیتی یکی از روش های کاهش این افزونگی است و اگر n تعداد عناصر مجموعه ای باشد که مقادیر صفت خاصه ی مورد نظر از آن گرفته شده است برای ذخیره سازی تمام این صفات به n بیت حافظه نیاز است.

وقتی که برای یک فایل روی یک صفت خاصه شاخص ایجاد می کنیم، در واقع یک افزونگی تکنیکی بوجود آورده ایم.

در ساختار ترتیبی نمی توان طول رکورد را تغییر داد.

زمان بازیابی رکورد بعدی در فایل ترتیبی کدام است؟

فایل عدم تقارن: فایل هایی که بر اساس یک فیلد مشخص می شوند لزومی ندارد که بر اساس یک کلید یا فیلد دیگر مرتب شوند. مثلا اگر بر اساس شماره ی دانشجویی مرتب شود، اسم ها به هم می خورد و بر عکس.

فایل تقارن: فایل تقارن فایلی است که اگر مثلا بر اساس یک فیلد آن را Sort کنیم فیلدهای دیگر هم به هم نخورند.

شاخص اصلی: وقتی که صفت خاصه ی شاخص کلید اصلی باشد. وقتی که صفت خاصه ی شاخص کلید ثانویه باشد.

کلید ثانویه کلیدی است غیر از کلید اصلی که خاصیت کلید بودن را دارد.

در شاخص متراکم لزومی بر مرتب بودن فایل داده ای نمی باشد اما در شاخص غیر متراکم باید فایل داده ای مرتب باشد برای این که بتوان رکوردها را گروه بندی کرد.

شاخص نرم افزاری و سخت افزاری داریم.

شاخص نرم افزاری: گروه در شاخص غیر متراکم بلاک یا باکت می باشد.

شاخص سخت افزاری: گروه در شاخص غیر متراکم شیار استوانه یا در حالتی که فایل روی چند دیسک ذخیره شده باشد می تواند خود دیسک باشد.

لنگرگاه چیست؟ نقطه ای از فایل داده ای است که مدخل شاخص به آن اشاره می کند و اگر لنگرگاه رکوردی باشد شاخص را متراکم می گویند.

اجزای تشکیل دهنده ی ترتیبی ساختار شاخص دار عبارتند از: ۱ـ فایل ترتیبی یا ناحیه ی اصلی ۲ـ ناحیه ی سر ریزی ۳ـ نشانه رو ها

شاخص در ساختار ترتیبی شاخص دار در چه حالتی بازسازی می شود؟ در سازمان دهی مجدد.

زمان خواندن کل فایل در ساختار ترتیبی شاخص دار به صورت پی در پی از چه رابطه ای به دست می آید؟

$$(n+o)\frac{R}{t'}$$

معایب ساختار ترتیبی شاخص دار را نام ببرید. عدم تقارن ـ مسئله ی درج سر ریزی ها ـ ایستا بودن شاخص.

واکشی تک رکوردها با استفاده از شاخص سریع تر انجام می گیرد.

در شاخص متراکم واکشی تک رکوردها سریع تر از شاخص غیر متراکم انجام می گیرد.

برای انجام عمل درج یک رکورد جدید در روش Push Trough چه اعمالی انجام می گیرد؟ خواندن بلاکی که باید رکورد در آن درج شود ـ بازنویسی بلاک ـ واکشی رکورد منطقا قبلی و تنظیم اشاره گر ـ بازنویسی همین رکورد.

***

مطالب تکثیر شده روی ۱۲ صفحه

ص۱

فایل با ساختار پایل یا برهم:

این فایل ساده ترین ساختار را داشته و رکوردهای آن بر اساس هیچ فیلدی مرتب نیستند. طول رکوردها متغیر بوده و تعداد فیلدها و مکان آن ها در رکورد، در نمونه های مختلف ممکن است متفاوت باشد بنابراین در کنار مقدار هر فیلد نام آن نیز نوشته می شود.

در بهترین حالت، نظم بین رکوردها، نظمی است زمانی (ترتیبی Entry sequential) انگار رکوردها بر یکدیگر پشته شده اند.

شماره رکورد	
۱	۴۲=سن ، برق = رشته ، لیسانس = مدرک ، حسینی = نام
۲	فوق لیسانس = مدرک ، احمدی = نام ، ۳۸= سن
...	

فرم کلی ساختار هر رکورد به شکل زیر است: A1=V1,A2=V2A3=V3,... که A1 نام فیلد (اسم صفت خاصه)، و V1 مقدار آن فیلد (مقدار صفت خاصه) می باشد.

متوسط اندازه ی رکورد: فایل در Load اولیه n رکورد دارد.

کل تعداد فیلدها را a در نظر می گیریم و متوسط تعداد فیلدها را در یک رکورد با 'a نشان می دهیم.

متوسط حافظه ی لازم برای هر فیلد را A بایت در نظر می گیریم و R=a'(A+V+2)

متوسط حافظه ی لازم برای هر مقدار را V بایت در نظر می گیریم؛ پس داریم: برای علامت مساوی یک بایت، و برای جداسازی نیز یک بایت در نظر می گیریم.

پس برای کل فایل میزان حافظه n برابر می شود. عامل منفی لزوم تکرار اسم فیلدها در نمونه های مختلف رکوردها می باشد.

زمان واکشی در رکورد TF:

از کل ساختمان فایلی داریم، بهترین حالت حداقل یک رکورد را بخواند و بدترین حالت آن است که کل رکوردها را بخواند. پس زمان واکشی برابر با زمان خواندن نصف فایل ها است به طور متوسط .

برای حالت بلاک بندی: $TF = \frac{1}{2} b \frac{B}{t'}$ برای حالت بدون بلاک بندی: $TF = \frac{1}{2} n \frac{R}{t'}$

$t' b/s$ =سرعت انتقال خواندن انبوه و b=تعداد بلاک ها و B=طول بلاک ها و n=تعداد رکوردها

$\frac{B}{t'}$ این قدر بایت بر ثانیه خوانده می شود ← طول فایل برحسب بایت = nR = bB

زمان به دست آوردن رکورد بعدی TN:

چون در این سازمان هیچ نظمی نداریم معلوم نیست که رکورد بعدی کجاست و یک بار باید همه را بخواند یعنی TN=TF

زمان درج رکورد TI:

از آن جا که فایل پایل هیچ گونه نظمی ندارد، رکورد جدید همواره در انتهای فایل اضافه (Append) می شود.

۱ـ خواندن آخرین بلاک فایل از دیسک به بافر s + r+ btt

۲ـ اضافه کردن رکورد به بلاک خوانده شده. از آن جا که این عمل در حافظه صورت گرفته و بسیار سریع است، زمان آن را در محاسبات خود وارد نمی کنیم.

۳ـ بازنویسی بلاک مذکور

پس زمان درج در فایل پایل برابر است با: $T_I = s + r + btt + T_{RW}$ و از آن جا که زمان عمل درج رکورد در بلاک موجود در حافظه اغلب کم تر از زمان یک دور زدن دیسک است پس: $T_I = (s + r + btt) + 2r$

***

ص۲

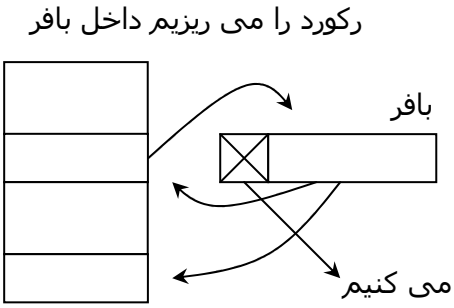
زمان به هنگام سازی از طریق تغییر مقادیر فیلدها (Update):

به هنگام سازی درجا و برون از جا داشتیم و در این سیستم های فایل (پایل) مجبوریم از برون از جا استفاده کنیم.

۱ـ رکورد مورد نظر را واکشی می کنیم ۲ـ این رکورد را برای حذف علامت گذاری کنیم ۳ـ رکورد حذف شدنی را در جای خود بنویسیم ۴ـ رکورد را اصلاح می کنیم ۵ـ رکورد جدید را به فایل اضافه می کنیم. (موارد ۱و ۲ و ۳، Delete یا نتیجه ی Update شده را می بریم ته فایل می نویسیم به خاطر سادگی کار یا سر جایش آن را قرار می دهیم با علامت گذاری)

موارد ۲و ۴ در بافر صورت می گیرد

۴ـ ایجاد دسته جدید است.



زمان Update $Tu = TF + T_{WR} + T_I$

محاسبه ی TD: از روی همین، عمل Delete کردن را می نویسیم و آن یک Update کردن درجا است.

حذف یک رکورد اینقدر زمان می گیرد: $T_{uD} = TF + T_{WR}$

زمان خواندن کل فایل Tx:

خواندن کل فایل به علت های زیادی خوانده می شود و خواندن فایل دو روش دارد.

۱ـ یا بر اساس ترتیب منطقی یا سریال Serial $T_{x\,seq} = 2TF$

۲ـ یا بر اساس ترتیب فیزیکی یا ترتیبی Sequential $T_{x\,ser} = nTF$ که زمان کمتری می برد

ممکن است موقعی که سریال می خوانیم مثلا اگر بر اساس ترتیب دانشجویی بخوانیم ممکن است در یک فیلد نداشته باشیم چرا که رکوردها نظمی ندارند و آن هایی را که نداریم ته فایل قرار می گیرند.

اگر بر اساس Sort کردن حساب کنیم و فایل Sort شده باشد، قبل از خواندن.

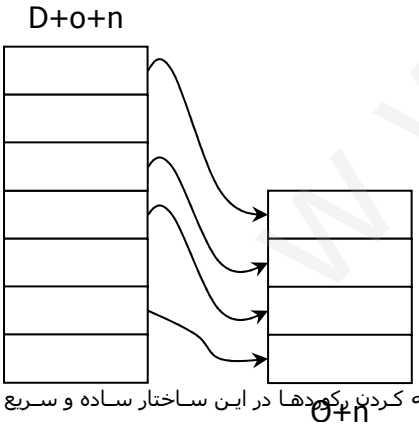
تذکر: Sort یعنی طوری رکوردها را عوض کنیم که ترتیب منطقی و فیزیکی یکی نشوند.

زمان سازمان دهی مجدد TY: $T_{x\,ser} = T_{Sort} + T_{x\,seq}$

برای این که یک نظم به فایل بدهیم و حافظه های هرز را از بین ببریم و فایل را کوچک کنیم.

حال فرض کنیم D رکورد حذف شدنی داشته باشیم و ۵ رکورد منتقله و n تعداد رکوردهای اولیه در فایل.

در فایل جدید o+n رکورد داریم. n تعداد رکوردهای اولیه در فایل و o تعداد رکوردهای درج شده.



$$T_y = \underbrace{(n+o)\frac{R}{t'}}_{\text{زمان خواندن رکوردها}} + \underbrace{(n+o-D)\frac{R}{t'}}_{\text{زمان نوشتن رکوردها}}$$

ویژگی ها و کاربرد ها:

در هنگامی که رکوردهای کمی حذف می شوند استفاده از این فایل از نظر مصرف حافظه مؤثر است. اضافه کردن رکوردها در این ساختار ساده و سریع است.خواندن پی در پی این فایل ساده و سریع است. ولی این فایل در عملیات واکشی یک رکورد و به دست آوردن رکورد بعدی کند است. ترتیبی خواندن رکوردهای این فایل بسیار کند می باشد. فقط هنگامی که میخواهیم فایل را از ابتدا تا انتها بخوانیم (بدون هیچ ترتیب خاص) ساختار پایل از بقیه ی ساختارها مناسب تر است.

حذف رکوردهای تکراری در فایل پایل زمان بر است.هنگامی که اندازه ی فایل کوچک باشد، چگونگی ساختار فایل اثر زیادی در سرعت اجرای عملیات ندارد.

***

ص۳

فایل ترتیبی Sequential:

در این ساختار طول رکوردها ثابت بوده و مکان و طول هر فیلد در رکورد ثابت و مشخص است. همچنین در این ساختار در Load اولیه تمام رکوردها بر مبنای یک فیلد (یا ترکیبی از چند فیلد) مرتب شده می باشند. معمولا رکوردها بر مبنای کلید اصلی مرتب شده هستند. بعضی مواقع به هر رکورد یک شماره ی یکتا داده می شود که به آن کلید خارجی می گویند.

شماره رکورد	نام	فامیلی	معدل	سال تولد
۱	علی	احمدی	۱۷	۵۴
۲	حسین	جهانی	۱۸	۵۵
۳	احمد	سعیدی	۱۵	۵۳
۴	رضا	معتمد	۲۰	۵۴

رکوردها بر اساس فیلد (صفت خاصه) به صورت صعودی مرتب شده اند.

طول و ساختار هر رکورد عموما در هر فایل ذخیره می گردد.

در فایل ترتیبی، نام فیلد ها در هر رکورد ذخیره نمی شود و مصرف حافظه ی آن کم تر از فایل پایل است.

پردازش سریالی رکوردها با سرعت و سادگی بیشتری نسبت به فایل پایل انجام می گیرد.

گاهی اوقات به فایل ترتیبی، فایل ترتیبی کلیدی (key) یا فایل ترتیبی مرتب شده Sorted Sequential نیز گفته می شود در مقابل به فایلی که بر اساس فیلدی مرتب نشده باشد، فایل ترتیبی زمانی یا Unordered Sequential می گویند.

یک ویژگی مهم فایل ترتیبی آن است که جهت بالا بردن سرعت، عملیات درج در فایل اصلی انجام نمی گیرد بلکه در یک فایل کمکی به نام فایل ثبت تراکنش ها یا TLF (Transaction Log File) صورت می گیرد. یعنی مثلا رکوردهای جدید به سادگی و با سرعت در انتهای فایل TLF ذخیره شوند بر اساس ترتیب زمانی ورود و بدون انجام مرتب سازی و بر اساس فیلد اصلی؛ سپس در یک دوره ی متناوب (مثلاً در آخر هر روز) در هنگام سازماندهی مجدد، محتویات فایل TLF (که نا مرتب است) خوانده شده و اطلاعات آن با فایل اصلی ادغام می شود. بدین ترتیب پس از سازماندهی مجدد فایل TLF خالی شده و کلیه ی اطلاعات در فایل اصلی به صورت مرتب شده موجود خواهد بود. مکان ثبت تراکنش ها هم می تواند به صورت یک فایل مجزا (TLF) در نظر گرفته شود و هم می تواند ناحیه ای در فایل اصلی باشد. به این مکان ثبت تراکنش ها، ناحیه سر ریزی یا Overflow Area نیز می گویند.

فایل ترتیبی اغلب در مواردی استفاده می شود که طول رکوردها ثابت بوده و معمولا واکشی سریع رکوردها به صورت تک تک مورد نیاز نباشد.

هنگامی که پردازش سریالی رکوردها مورد نظر باشد، ساختار ترتیبی بهتر از پایل است. در بسیاری از سیستم های تجاری که رکوردها به صورت دسته ای (Batch) پردازش می شوند، از ساختار ترتیبی استفاده می شود.

متوسط اندازه رکورد: $R=a.V$ a تعداد فیلد

کل فایل: $S=n.a.V$ V طول فیلد

کل فایل با تراکنش: $S=(n+o).a.V$ n تعداد رکوردها

***

ص۴

محاسبه ی TF در فایل ترتیبی: چقدر زمان طول می کشد به یک رکورد دسترسی پیدا کنیم.

الف) واکشی در فایل ترتیبی اگر جستجو بر مبنای فیلدی غیر کلید باشد: فایل ترتیبی مشابه یک فایل پایل بوده و لذا مجبوریم که از جست و جوی خطی استفاده کنیم. $T_F = \frac{1}{2}(n+o)\frac{R}{t'}$

n تعداد رکوردها در داخل فایل اصلی و o تعداد رکوردها در ناحیه ی سر ریزی و t' نرخ انتقال انبوه است.

ب) اگر جست و جو بر مبنای فیلد کلید یعنی فیلدی که فایل بر اساس آن مرتب شده است، باشد آنگاه می توان با روش جست و جوی باینری که بسیار سریعتر از جست و جوی خطی است، رکورد دلخواه را واکشی کرد.

در جست و جوی باینری ابتدا بلاک وسطی فایل به بافر آورده می شود سپس با وارسی کلید اولین و آخرین رکورد موجود در آن بلاک، مشخص می شود که رکورد مورد جست و جو در بلاک هست یا نه. اگر رکورد در بلاک مذکور نباشد بلاک بعدی خوانده می شود و اگر رکورد در بلاک باشد، با یک جست و جوی باینری درون بلاکی پیدا می شود. فرض کنید این زمان های بررسی در بافر برابر C_B باشد بنابراین اگر رکورد مذکور در قسمت اصلی فایل مرتب شده باشد زمان متوسط T_F برابر است با:

$$T_F = [(\log_2 b)-1]\times(S+r+b_{tt}+C_B)$$

C_B زمان پردازش بلاک است و زمان بسیار کمی است و می شود از آن صرفنظر کرد.

b_{tt} زمان خواند یک بلاک. تعداد دفعات مراجعه به دیسک برای b بلاک از مرتبه ی $\log_2 b$ بیشتر و مقدار متوسط این تعداد مراجعات برابر $[(\log_2 b)-1]$ می باشد.

ولی اگر رکورد در ناحیه ی سر ریزی (فایل تراکنش) باشد پس از بررسی کل فایل اصلی با جست و جوی دودویی که زمان $\log_2 b\times(S+r+b_{tt})$ را می برد باید به سراغ ناحیه ی سر ریزی رفته و آن را به صورت خطی جست و جو کنیم. پس اگر o تعداد رکوردها در ناحیه ی سر ریزی و b تعداد رکوردها در ناحیه ی اصلی (مرتب شده) باشد آنگاه:

$$T_F = \log_2 b \times (S + r + b_{tt}) + r + S + \frac{1}{2} o \frac{R}{t'}$$

مثال: فایل ترتیبی داریم شامل ۴۰۰۰۰۰ رکورد می باشد. اگر $B_F = 24$ باشد، قبل از اضافه شدن رکوردی به ناحیه ی سر ریزی، زمان متوسط واکشی چه میزان خواهد بود؟

S=16 ms, r=8.3 ms, btt=18 ms

$$b = \frac{\text{تعداد رکوردها } n}{\text{تعداد رکوردها در بلاک } B_F} = \frac{400000}{24} = 16667$$

$$T_F = \underbrace{[(\log_2 b) - 1]}_{\text{درنگ دوراني}} \times \left(\overbrace{S + r}^{\text{استوانه جويي}} + \underbrace{b_{tt}}_{\text{زمان خواند يک بلاک}} \right)$$

$$T_F = 13 \times (16 + 8.3 + 0.8) = 32\,s \quad \text{ثانيه} \quad \text{ميلي}$$

و اگر فایل به صورت پایل می بود: $(b'_{tt} = 0.84\,ms)$

$$T_F = \frac{1}{1} b \times b'_{tt} = \frac{1}{2} \times 16667 \times 0.84 = 7000\,\text{ثانيه} \quad \text{ميلي} = 7\,\text{ثانيه}$$

***

ص۵

جست و جو با پرش بلاکی (Skipped Block Search)

این روش در واقع بهبود یافته ی روش جست و جوی خطی بوده و هنگامی که آرگومان مورد جست و جو بر مبنای فیلد کلید باشد قابل استفاده است. فرض کنید فایل بر اساس کلید به صورت صعودی مرتب شده باشد. در این حال اولین بلاک را خوانده و کلید مورد جست و جو را با کلید آخرین رکورد موجود در بلاک مقایسه می کنیم. اگر کلید مورد جست و جو بزرگ تر باشد، پس حتما در آن بلاک نیست و بنابراین بلاک بعدی را می خوانیم. بدین ترتیب دیگر نیازی نیست که بقیه ی رکوردهای موجود در آن بلاک را برسی کنیم. به همین ترتیب بلاک ها را پشت سر هم خوانده و تنها کلید آخرین رکورد هر بلاک را وارسی می کنیم تا هنگامی که کلید مورد جست و جو از کلید آخرین رکورد بلاکی کوچک تر باشد. در این حالت رکورد مورد جست و جو در آن بلاک بوده و فقط کافی است آن بلاک را با روش خطی یا باینری جست و جو کنیم.

بهترین اندازه ی بلاک یا به عبارتی دیگر مقدار بهینه ی B_F برحسب تعداد رکوردها (n) برای آن که تعداد مقایسه ها در روش جست و جوی بلاکی حداقل باشد برابر است با:

$$B_F = \sqrt{n}$$

زمان دستیابی به رکورد بعدی در فایل ترتیبی TN:

احتمال دارد رکورد بعدی در همان بلاکی باشد که اخیرا خوانده شده است. در این حالت بدست آوردن رکورد بعدی (TN) رجوع به دیسک را نیاز ندارد و زمان آن را تقریباً صفر می گیریم. مثلا اگر $B_F = 5$ باشد به احتمال $\frac{1}{5}$ ، رکورد بعدی در بلاک بعدی است. چرا که به احتمال $\frac{1}{5}$ رکورد جاری خوانده شده آخرین رکورد بلاک است.

زمان متوسطی که برای هر کدام صرف می شود $T_N = \dfrac{\overbrace{S + r + b_{tt}}^{\text{زماني که براي خواندن بلاک است}}}{\underbrace{B_F \to \frac{B}{R}}_{\text{تعداد رکوردها در بلاک}}}$

اگر فرض کنیم رکوردها در جدول T.L.F باشد، با کلید نمی شود رکورد بعدی را در این جدول حدس زد و پیدا کرد؛ بنابراین پیدا کردن رکورد بعدی بی معنی است.

محاسبه TI در فایل ترتیبی:

اگر فایل کوچک باشد می توان عمل درج را در همان فایل اصلی مرتب شده انجام داد. در این حالت ابتدا می بایست در زمان T_F مکان درج آن رکورد را پیدا کنیم سپس رکوردهای زیر آن را به سمت پایین شیفت دهیم. به طور متوسط نصف بلاک های فایل می بایست به سمت پایین شیفت داده شوند. پس در این حالت داریم:

$$T_I = T_F + \frac{1}{2} b (b_{tt} + T_{RW})$$

T_F : یافتن نقطه ی منطقی درج ، $b_{tt} + T_{RW}$: زمان شیفت یک بلاک

با فرض کوچک بودن فایل، در حد یک استوانه، دیگر زمان S را در ارزیابی زمان شیفت هر بلاک دخالت ندادیم و زمان r را نیز به دلیل این که عملیات بلاک به بلاک به طور پی در پی انجام می گیرد در محاسبه ئارد نمی کنیم.

***

درج فایل های بزرگ:

برای هر یک درج، اگر همه ی رکوردها را شیفت بدهیم وقت گیر است و از روش T.L.F استفاده می کنیم و اگر رکوردی بخواهیم درج بشود می بریم ته فایل

$$T_I = b_{tt} + S + r + T_{RW} + \frac{T_y}{o}$$

$$T_I = \underbrace{\frac{3r + S + b_{tt}}{\text{فایل}}}_{\text{نوشتن در ته}} + \underbrace{\frac{\overbrace{T_y}^{\text{زمان سازماندهي مجدد}}}{o}}_{\text{تعداد رکوردها در T.L.F}}$$

زمان به هنگام سازی تغییر دهنده:

Tu (Update) , TD (Delete)

برای حذف یک رکورد از روش حذف منطقی به صورت درجا استفاده می شود. رکورد مورد نظر در زمان TF خوانده شده و علامت حذف در ابتدای آن در بافر نوشته شده و در گردش بعدی دیسک (2r) در سر جای اولیه اش رو نویسی می گردد. پس داریم: $T_D = T_F + 2r$

عملیات اصلاح دو وضعیت متفاوت دارد. اگر اصلاح بر روی فیلد غیر کلید باشد می توان عملیات به هنگام سازی را به صورت درجا انجام داد یعنی:

برای اصلاح بر روی فیلد غیر کلید $T_u = T_F + 2r$

اگر عملیات بر روی فیلد کلید باشد می بایست به صورت برون از جا صورت بگیرد چرا که ترتیب رکوردها به هم خواهد خورد. در این حال رکورد مورد نظر را حذف منطقی کرده (در زمان TD) و سپس رکورد اصلاح شده ی جدید را در ناحیه ی سر ریزی درج می کنیم (در زمان TI) پس داریم:

برای اصلاح بر روی فیلد کلید $T_u = T_D + T_I$

در عملیات اصلاح در فایل ترتیبی طول رکورد تغییر نمی کند.

اگر رکورد مورد اصلاح در ناحیه ی سر ریزی باشد دیگر فرقی نمی کند که فیلد کلید آن اصلاح می شود و یا فیلد غیر کلید آن. در هر دو حالت عملیات اصلاح به صورت در جا بوده و TF+2r زمان می برد.

برای خواندن کل فایل دو روش داریم:

الف ـ سریال (پی در پی) بدون نظم ، ب ـ ترتیبی که بر اساس نظم منطقی است.

الف) $T_{x\,Ser} = (n+o)\frac{R}{t'}$ ، R طول رکورد ، t' زمان انتقال انبوه یک بلاک ، $\frac{R}{t'}$ زمان انتقال یک رکورد.

ب) $T_{x\,Seq} = T_y + (n+o+d)\frac{R}{t'}$ ، d تعداد رکوردهایی که حذف می شوند و علامت گذاری می شوند

زمان سازماندهی مجدد TY: $T_y = \underbrace{T_{Sort(o)}}_{1} + \underbrace{n\frac{R}{t'}}_{2} + \underbrace{o\frac{R}{t'}}_{3} + T_{Merg}\underbrace{(n+o-d)}_{4} \times \frac{R}{t'}$

۱: زمان Sort به اندازه ی o رکورد

۲: خواندن فایل اصلی

۳: خواندن فایل T.L.F

۴: ادغام دو فایل و نوشتن آن ها به صورت یک فایل جدید (Merg).

***

فایل با ساختار ترتیبی شاخص دار Indexed Sequential:

شاخص ها بر مبنای کلیدها و آدرس فیلدها ساخته می شوند. شاخص باعث بالا رفتن سرعت دستیابی می گردد و تکنیک شاخص بندی در اکثر نرم افزارهای امروزی استفاده می شود و جزو شیوه های دستیابی تصادفی به حساب می آید و شاخص هایی که در این جا بررسی می شود، از نوع شاخص ساده هستند.

مثال: فایل ترتیبی دانشجویان در زیر برحسب شماره دانشجویی مرتب شده می باشد که در کنار این فایل ترتیبی یک فایل ایندکس (شاخص) برحسب کلید اصلی (شماره دانشجویی) و یک فایل ایندکس برحسب معدل ترسیم شده است.

فایل ترتیبی

شماره رکورد	معدل	نام پدر	نام	شماره دانشجویی

فایل ایندکس معدل

شماره رکورد	معدل

فایل ایندکس اولیه

شماره رکورد	شماره دانشجویی

۱	۱۷	سعید	علی	۳۹۲۵
۲	۱۹	مجید	حسن	۴۷۱۳
۳	۱۶	شاهین	امیر	۵۴۱۷
۴	۱۵	سهیل	جواد	۷۳۵۴
...	...	...	...	...

۴	۱۵
۳	۱۶
۲	۱۷
۱	۱۹
...	...

۱	۳۹۲۵
۲	۴۷۱۳
۳	۵۴۱۷
۴	۷۳۵۴
...	...

حال اگر مثلا بخواهیم مشخصات دانشجویی با معدل ۱۹ را ببینیم کافی است ابتدا در فایل کوچک ایندکس معدل، با روش باینری ستون سمت چپی را به دنبال عدد ۱۹ جست و جو کنیم بدین ترتیب متوجه خواهیم شد که مشخصات این دانشجو در سطر ۲ فایل ترتیبی اصلی قرار دارد لذا به سرعت بر سر رکورد ۲ فایل اصلی رفته و اطلاعات مورد نیاز را می خوانیم. بدون این فایل کمکی شاخص مجبور بودیم با جست و جوی خطی در فایل اصلی آن را پیدا کنیم که کاری زمان گیر بود.

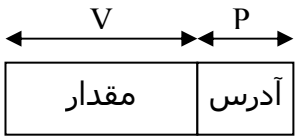
در مثال ساده ی فوق تعداد سطرهای فایل شاخص برابر سطرهای فایل اصلی می باشد ولی تعداد ستون ها آن تنها ۲ فیلد است. بدین دلیل فایل شاخص به مراتب کوچک تر از فایل اصلی بوده و جست و جو در آن سریع تر صورت می گیرد. حتی در صورتی که فایل ایندکس خیلی کوچک باشد می توان آن را در حافظه ی اصلی نگهداری کرد و بدین ترتیب سرعت جست و جو افزایش بسیار زیادی می یابد.

اگر در فایل ایندکس صفت خاصه ی شاخص، کلید اصلی باشد به آن شاخص اولیه یا اصلی می گویند (Primary Index). و در صورتی که فایل ایندکس بر اساس فیلدی غیر از کلید اصلی ساخته شود به آن شاخص ثانویه گویند (Secondary Index).

***

ص۸

پس فایل شاخص مجموعه ای از تعدادی مدخل (Entry) می باشد که به فرم کلی زیر:



فیلد آدرس به طول P بایت حاوی یک نشانه گر به یک یا گروهی از رکوردهاست. در فایل داده ای اصلی فیلد مقدار به طول V بایت شامل صفت خاصه ای یا ترکیبی از صفات خاصه است که ایندکس بر اساس آن ساخته شده است بنابراین طول هر رکورد فایل شاخص برابر V+P بایت است. به هر نقطه از فایل داده ای اصلی که از مدخل شاخص به آن نشانه گر وجود دارد را لنگرگاه یا Anchor Point گویند.

اگر هر مدخل فایل شاخص به یک رکورد اشاره کند، شاخص را متراکم (Dense Index) گویند و اگر به گروهی از رکوردها مثلا یک بلاک اشاره کند، شاخص را غیر متراکم (Non Dense Index) گویند.

در شاخص غیر متراکم فایل اصلی داده ای باید بر اساس فیلد متناظر شاخص مرتب شده باشد تا رکوردها را بتوان دسته بندی کرد ولی در شاخص متراکم لزومی نیست که فایل داده ای از قبل مرتب باشد. فایل داده‌ای و فایل شاخص می توانند بلاک بندی شده باشند یا نشده باشند. در حالت بلاک بندی شده اغلب اندازه ی بلاک فایل شاخص و بلاک فایل داده ای یکسان است. در شاخص نا متراکم مقدار موجود در فیلد داده هر مدخل میتواند کوچک ترین یا بزرگ ترین مقدار در هر گروه باشد.

تعریف ظرفیت نشانه روی شاخص (Index fdnout):

فایل شاخص نیز مثل فایل داده ای بلاک بندی شده است. تعداد مدخل های یک بلاک شاخص را ظرفیت نشانه روی آن می گویند. در واقع همان فاکتور بلاک بندی است برای بلاک شاخص و با پارامتر y آن را نمایش می دهند.

$$y = \left\lfloor \frac{\text{اندازه بلاک شاخص}}{\text{طول مدخل شاخص}} \right\rfloor \rightarrow y = \left\lfloor \frac{B}{V+P} \right\rfloor$$

مثال: اگر طول بلاک ۲۰۰۰ بایت، اندازه ی صفت خاصه ی شاخص V برابر ۱۴ بایت و اندازه ی اشاره گر شاخص P برابر ۶ بایت باشد، ظرفیت نشانه روی هر بلاک شاخص چقدر است؟

$$y = \left\lfloor \frac{2000}{14+6} = 100 \right\rfloor$$

یعنی هر بلاک شاخص دارای ۱۰۰ سطر یا مدخل است و هر مدخل اشاره گری به رکورد یا گروهی از رکوردها در فایل داده ای اصلی می باشد و با توجه به مفروضات زیر ساختار شاخص برای این فایل چیست؟

$n = 10^6$, R=200 بایت , B=2000 بایت

در فایل اصلی ۱۰ رکورد جای میگرد

$$\frac{B}{R} = BF = \frac{2000}{200} = 10$$

$$b = \frac{n}{BF} = \frac{10^6}{10} = 10^5$$

حافظه ی مصرفی برای سطح اول شاخص

$$SI_1 = 10^5 \times 20$$

تعداد بلاک های سطح اول $b_1 = \frac{10^5}{100} = 1000$

حافظه ی مصرفی زیاد است لذا سطح دوم شاخص را ایجاد می کنیم

برای نگه داری در حافظه ی اصلی زیاد است $SI_2 = 1000 \times 20 = 20000$

تعداد بلاک ها در سطح دوم $b_2 = \frac{1000}{100} = 10$

پس ساختار شاخص را در سه سطح ایجاد می کنیم $SI_3 = 10 \times 20 = 200$

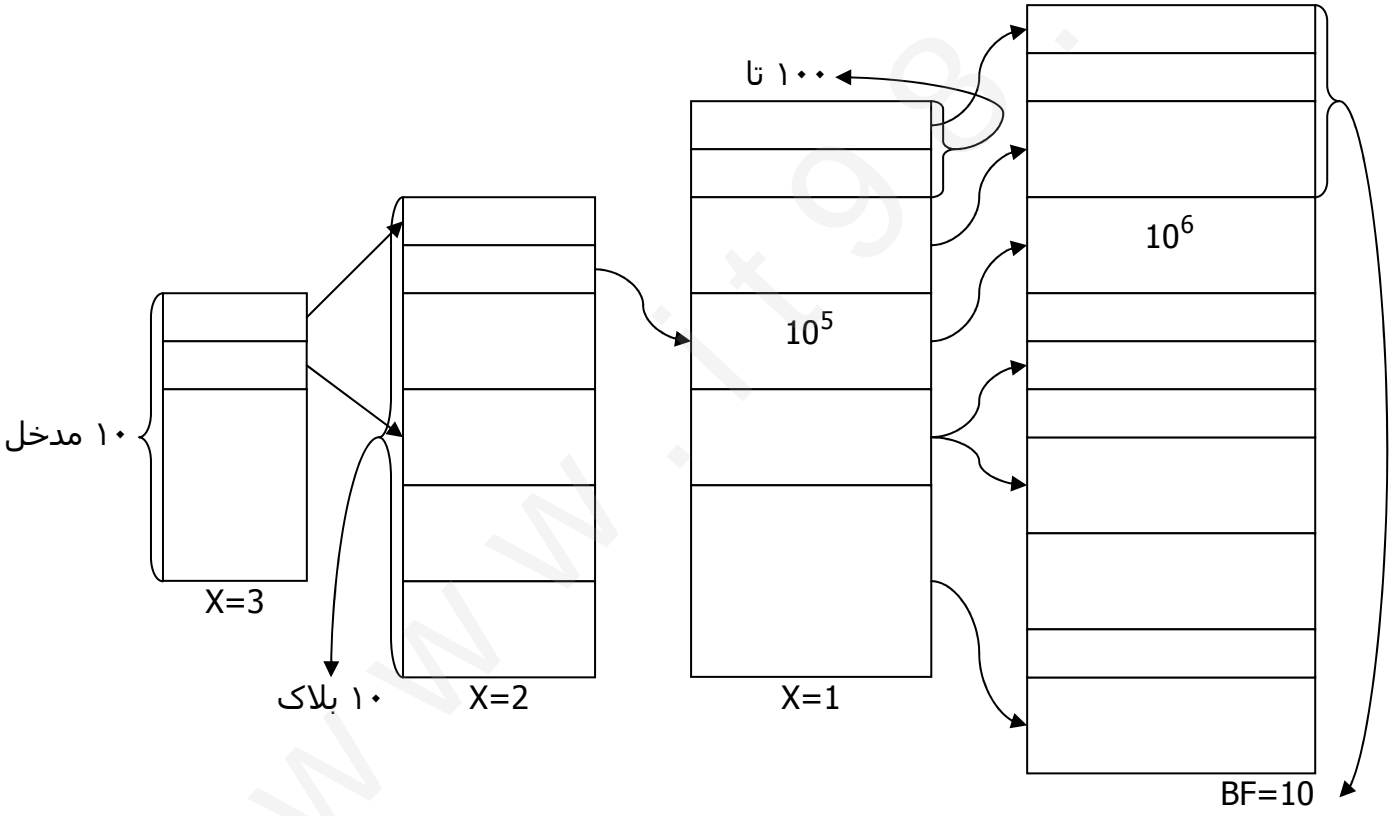
بلاک های سطح سوم $b_3 = \frac{10}{100} = 0.1$

و تعداد سطوح شاخص از رابطه ی زیر به دست می آید:

$$x = \left\lceil \log_y \left(\frac{n}{BF} \right) \right\rceil = \lceil \log_y b \rceil = \lceil \log_{100} 10^5 \rceil = 3$$

هرچه تعداد سطوح بیشتر باشد دفعات دستیابی برای واکشی رکورد بیشتر خواهد بود.

اینقدر ادامه می دهیم تا حداقل به یک بلاک برسیم و این کار را موقعی انجام می دهیم که فایل دارد ساخته می شود و شاخص هایش را می سازیم.



***

ص۹

بررسی مسئله ی سر ریزی: به این چند سؤال ابتدا باید پاسخ داد.

۱ـ فضای لازم برای درج رکورد سر ریزی چگونه انتخاب می شود؟

۲ـ فضای انتخاب شده چگونه در محیط فیزیکی (دیسک) به فایل تخصیص می یابد؟

۳ـ عمل درج با چه تکنیکی انجام می شود؟

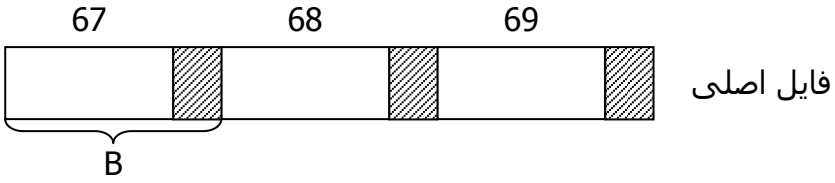
برای حالت ۱ سه راه حل پیشنهاد شده:

الف: در نظر گرفتن جا در هر بلاک

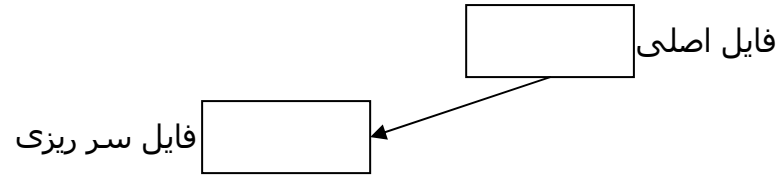
ب: در نظر گرفتن فایل جداگانه

ج: در نظر گرفتن جا در همان فایل اصلی

الف) موقعی که فایل را می سازیم در هر بلاک فایل اصلی یک مقداری فضای خالی در نظر می گیریم. هر چند که به نظر می رسد از نظر قوی بودن لوکالیتی رکوردها، راه خوبی باشد ولی عیب اش این است که پیش بینی چه مقدار جا را نداریم.



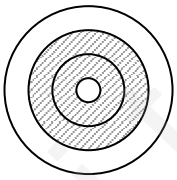
ب) در روش ب یک فایل اصلی را در نظر می گیریم و یک فایل سر ریزی که سر ریزی ها را داخل آن بریزیم و اشاره گر داریم از فایل اصلی به فایل سر ریزی که این مشکل است از یک فایل به فایل دیگر اشاره کردن و زمان گیر است و تلف زمانی دارد.



ج) بهترین روش و رایج ترین روش است. در همان فایل اصلی جایی برای سر ریزی را در نظر می گیریم و لی باید ببینیم کجای فایل اصلی و دو روش وجود دارد.

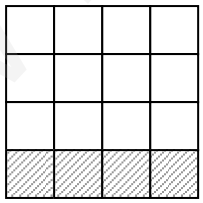
۲ـ تقسیم بندی سخت افزاری است:
- الف: استوانه های جداگانه در نظر بگیریم
- ب: شیارهایی از هر استوانه برای سر ریزی همان استوانه در نظر گرفته شود (در انتهای هر استوانه)

الف) این راه حل مناسب نیست زیرا سبب می شود که لوکالیتی رکوردهای سر ریزی ضعیف شود و در نتیجه متوسط زمان استوانه جویی افزایش یابد.

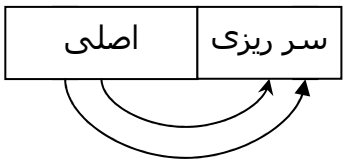


ب) این راه حل متوسط زمان استوانه جویی را کاهش می دهد زیرا رکوردهای سر ریزی هر استوانه در همان استوانه جای دارند. البته وقتی که ناحیه ی سر ریزی یک استوانه پر شود ناحیه ی دیگری برای درج سر ریزی ها باید ایجاد کرد (ناحیه سر ریزی اولیه و ثانویه) و یا این که فایل را سازمان دهی مجدد کرد.

استوانه ها



و به صورت نرم افزاری بهترین کار این است که ناحیه ی سر ریزی را بگذاریم در ناحیه ی فایل اصلی و لوکالیتی اش کم تر است.



۳ـ عمل درج در ناحیه ی سر ریزی با چه تکنیکی انجام می شود؟
- الف: درج در اولین بلاک جادار در ناحیه سر ریزی
- ب: درج با Push Trough

الف) در این تکنیک، رکورد جدید مستقیما وارد بلاکی از ناحیه ی سر ریزی می شود و در اولین مکان آزاد جای می گیرد (در اولین بلاک جادار). سپس از رکورد منطقا پیشین به رکورد درج شده نشانه رو ایجاد می شود و به ترتیب زنجیره ی رکوردهای سر ریزی پدید می آید. در این تکنیک برای هر رکورد از ناحیه ی اصلی و ناحیه ی سر ریزی یک نشانه رو وجود دارد. البته ممکن است در بعضی از رکوردها محتوای این فیلد Null باشد.

***

ص۱۰

در فیلد نشانه رو آخرین رکورد هر زنجیره، Null داریم به معنای این که پایان زنجیره است.

ممکن است از یک بلاک چند اشاره گر به ناحیه ی سر ریزی داشته باشیم.

روش درج ساده است ولی جست و جو وقت گیر می باشد.

ب) در روش ب درج مشکل است ولی جست و جو ساده تر است و اساس بر این دو نکته ی زیر می باشد:

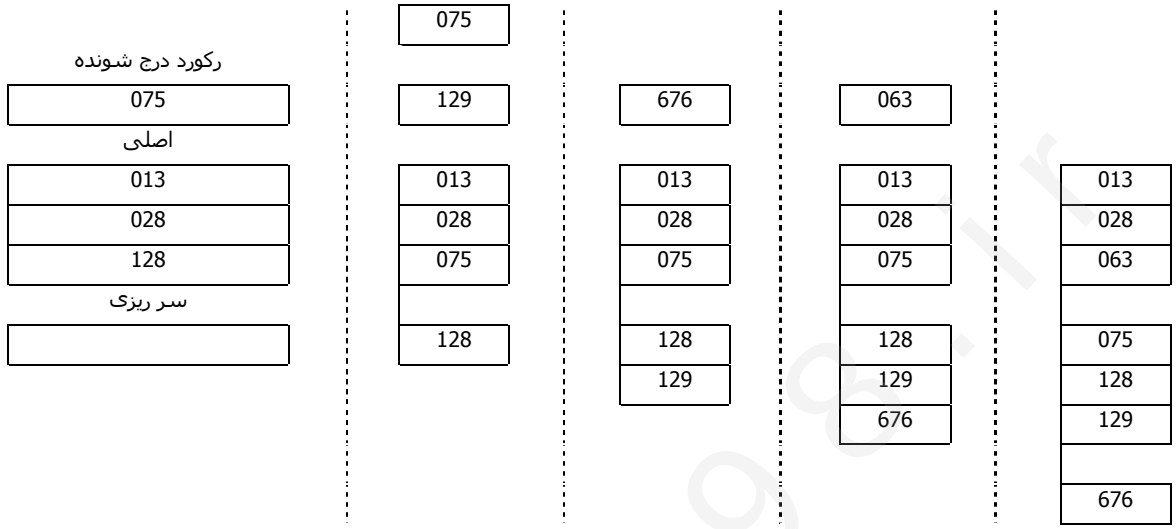
۱ـ نظم درون بلاکی همیشه وجود داشته باشد، حتی برای ناحیه ی سر ریزی.

۲ـ از هر بلاک ناحیه ی اصلی تنها یک اشاره گر به ناحیه سر ریزی اشاره کند.

وقتی یک رکورد درج می شود باید محل منطقی اش را پیدا کنیم در داخل بلاک و رکورد درج شده را در آن محل قرار دهیم و رکوردهای از آن به بعد را یک رکورد شیفت بدهیم جلو و ته فایل یک بلاک اضافه پیدا می کنیم و آن را می آوریم در ناحیه ی سر ریزی و کار تا آن جا پیدا می کند که دیگر شیفت کردن لازم نداشته باشیم.

در ناحیه ی اصلی داریم در لحظه ی شروع کار:

حال می خواهیم یک سری رکورد درج کنیم و رکورد درج شونده است مثلا



موارد استفاده ی ساختار:

این ساختار در محیط هایی به کار می رود که در آن ها نیاز به پردازش سریال فایل روی یکی از صفات خاصه کلید (مطرح) بوده. به علاوه واکشی تک رکوردها از طریق مقدار کلید آن ها عمل رایجی باشد، در اغلب سیستم های تجاری ـ مدیریتی، این ساختار مورد استفاده قرار می گیرد.

***

ص۱۱

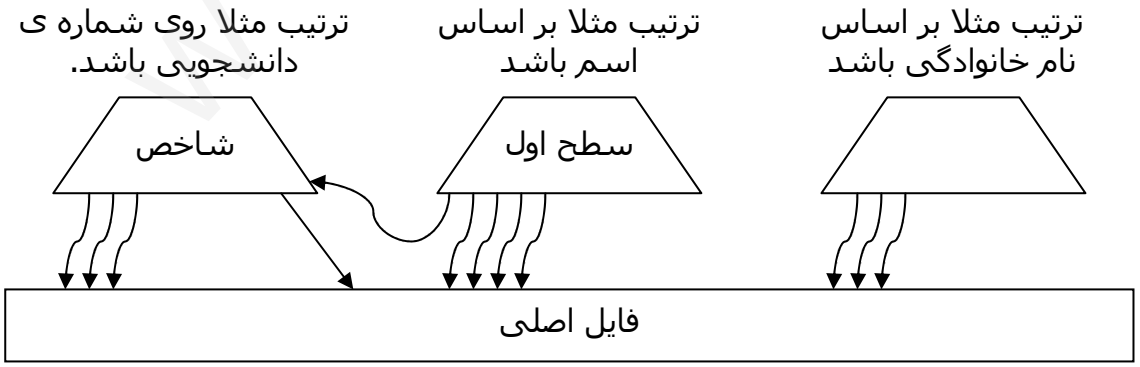
ساختار فایل چند شاخصی Multi Indexed:

اهمیت این ساختار به خاطر این است که ۷۰% الی ۸۰% ساختار بانک های اطلاعاتی با این ساختار است.

در فایل های ترتیبی یک فایل اصلی داشتیم که مرتب بود و یک فایل جانبی داشتیم برای درج کردن و مشکل این بود که هیچ ارتباط منطقی بین آن ها نبود و برای رفع این اشکال فایل ترتیبی شاخص دار داشتیم و از یک سیستم شاخص استفاده می کردیم که کمک می کرد به دستیابی سریع.

سه ایراد عمده داشت: ۱ـ ایستا بودن شاخصها بود. ۲ـ عدم تقارن بود. ۳ـ مسئله سر ریزی بود.

و برای رفع این معایب: در فایل های چند شاخصی روش کار به این صورت است که یک فایل داریم به اسم فایل غیر اصلی و غیر ترتیبی است و حتی می تواند یک فایل پایل باشد.



فایل اصلی، غیر ترتیبی یا پایل میتواند باشد

شاخص متراکم، شاخص به همه ی رکوردها داریم.

شاخص دینامیک است، شاخص همراه تغییر رکوردها تغییر می کند.

این ساختار چنان است که پدیده ی عدم تقارن در آن وجود ندارد. زیرا روی تعدادی، حتی تمام صفات خاصه می توان شاخص داشت و مسئله ی رکوردهای سر ریزی به صورتی که در ساختار سوم مطرح بود در این جا وجود ندارد. یعنی درج رکوردهای جدید آسان تر و پویاتر است و بالاخره خود ساختار شاخص وضعیتی پویا دارد و هم روند با تغییرات فایل داده ای، قابل تنظیم و به هنگام در آوردن است. اگر a تعداد صفات خاصه در فایل باشد، حداکثر a فایل شاخص می توان داشت.

از آن جا که در یک رکورد به طور متوسط، 'a تا صفت خاصه وجود دارد، لذا به یک رکورد 'a ساختار شاخص ناظر است. پس این ساختار در اساس از نظر فایل داده ای همان پایل است اما مجهز به یک سری استراتژی دستیابی قوی، پویا و سریع.

کاربر می تواند هر تعداد از صفات خاصه ای که در فایل دارد، درخواست ایجاد شاخص کند و برای واکشی سریع تک رکوردها، الزامی ندارد که حتما از کلید اصلی به عنوان آرگومان جست و جو استفاده نماید.

ضابطه ی انتخاب صفات خاصه ی شاخص: لزومی ندارد که روی تمام صفات خاصه، شاخص ایجاد نمود، می توان بین صفات خاصه قائل به اولویت شد و آن صفاتی را برگزید که در بیشترین درخواست ها به عنوان آرگومان جست و جو به کار برده می شوند.

***

ص۱۲

اجزاء ساختار ترتیبی شاخص دار عبارتند از: ناحیه اصلی، ناحیه سر ریزی، مجموعه شاخص ها.

در فایل چند شاخصی داریم: هرچه تعداد صفات خاصه شاخص بیشتر باشد، عمل بازیابی کارآتر است و هرچه تعداد صفات خاصه شاخص بیشتر باشد، عدم تقارن کمتر است و ساختار فایل داده ای اصلی می تواند پایل باشد و ایجاد شاخص روی ترکیبات مختلف صفات خاصه امکان پذیر است.

فایل وارون، فایل است که روی تمام فیلدهای آن شاخص داشته باشیم.

در فایل چند شاخصی می توان بین صفات خاصه اولویت قائل شد و برای ایجاد شاخص آن فیلدهایی را انتخاب کرد که در بیشتری پرس و جو ها به کار برده می شوند.

تعداد مدخل های شاخص در سطح اول برای شاخص های مختلف ممکن است یکسان نباشد.

فایل های شاخص تأمین کننده ی استراتژی دستیابی برای فایل داده ای هستند.

در فایل داده ای مقدار فیلدی می تواند Null یا ناشناخته باشد.

شاخص گذاری جزو نوع دستیابی تصادفی است.

فایلی شامل اطلاعات ثبت نام دروس دانشجویان با ۲۰۰۰ رکورد داریم. هر دانشجو به طور متوسط ۴/۵ درس اخذ کرده است. اگر روی فیلد درس اخذ شده (که تکرار شونده است) شاخص ایجاد کنیم تعداد مدخل های این شاخص چقدر خواهد بود؟

2000×4.5=9000

تفاوت بین شاخص اولیه و ثانویه آن است که در شاخص اولیه کلید تکراری وجود ندارد ولی در شاخص ثانویه ممکن است کلید تکراری داشته باشیم.

اگر در فایلی که شاخص اولیه و ثانویه دارد بر اثر اصلاح، کلید اولیه تغییر کند، چه تنظیم دیگری را باید انجام دهیم؟

ممکن است لازم باشد تا شاخص اولیه و تمام شاخص های ثانویه تغییر کنند.

بر روی فایل های ترتیبی و پایل و فایل پایل مرتب شده می توان شاخص ایجاد کرد.