اهداف اصلی هر سیستم فایل عبارتند از:

۱) سرعت عملیاتی(در بازیابی و ذخیره سازی)
۲) صرفه جویی در حافظه

برای رسیدن به این دو هدف اصلی، تلاش بر این است که در طراحی سیستمهای ذخیره و بازیابی ضابطه های اساسی زیر در نظر گرفته شوند.

۱) حداقل بودن میزان افزونگی برای کاهش میزان حافظه مصرفی و کاهش هزینه بهنگام سازی.
۲) دستیابی سریع (برای داشتن سرعت مطلوب در بازیابی و ذخیره سازی).
۳) سهولت در عملیات بهنگام سازی(تا اطلاعات با کمترین هزینه به هنگام سازی شوند).
٤) سهولت نگاهداری سیستم .
٥) قابلیت اطمینان بالا .

در ادامه دو ساختار مبنایی فایل ها را مورد بررسی قرار می دهیم:

۱) فایل با ساختار بر هم بی نظم Pile .
۲) فایل با ساختار ترتیبی Sequential.

تعریف فایل متراکم:

فایل متراکم به فایلی گفته می شود که تمام مقادیر همه صفات خاصه تمام رکوردهایش مشخص باشد .

تعریف فایل غیر متراکم:

فایل غیر متراکم فایلی است که برخی از مقادیر بعضی از صفات خاصه، در برخی از رکوردها، موجود نباشد.

تعریف افزونگی:

فایلی را افزونگی می گوییم که مقادیر بعضی از صفات خاصه اش بیش از یکبار در محیط فیزیکی ذخیره سازی شده باشد.

این تکرار ذخیره سازی یا افزونگی دو حالت دارد:

۱) افزونگی طبیعی ۲) افزونگی تکنیکی

افزونگی طبیعی:

در افزونگی طبیعی صفت خاصه چنان است که یک مقدار مشخص از آن در تعدادی از نمونه رکوردها وجود دارد.

مثلاً در فایل ثبت نام دانشجویان، شماره یک درس مشخص، در رکورد تمام دانشجویانی که در آن درس را اخذ کنند، ذخیره می شود. این افزونگی را می توان با تکنیک هایی کاهش داد.

برای کاهش مصرف حافظه در حالت وجود افزونگی طبیعی، در اساس دو تدبیر متصور است:

۱) طراحی فایل با ساختار کاراکتر(مثلاً ساختار چند حلقه ای).

۲) استفاده از یک تکنیک فشرده سازی.

تکنیک ماتریس بیتی:

این تکنیک هنگامی کاربرد دارد که اولاً فقره اطلاع تکرار شونده(صفت خاصه چند مقداری) داشته باشیم.
ثانیاً: مقادیر صفت خاصه، از مجموعه ای محدود برگرفته شده باشند، در این صورت هم طول رکوردها متغییر و هم افزونگی طبیعی تشدید می شود.
مثال :فرض می کنیم فایلی حاوی اطلاعات در مورد دانشجو – درس را می خوانیم و ذخیره می کنیم؟
جواب:

روش اول: بدون استفاده از ماتریس بیتی :برای هر نمونه دانشجو، رکوردی با طول متغییر در نظر می گیریم.

	Student Number	Course Number
R1	54381	177,179,184,185,187
R2	54407	177,178,181,183,187,191

روش دوم:استفاده از ماتریس بیتی:

Student number	Courses number															
	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
54381	0	1	0	1	0	0	0	0	1	1	0	1	0	0	0	0
54407	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0	1
54408	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
54503	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0
54504	0	0	1	0	0	0	0	1	0	1	0	0	1	0	0	1

ذخیره‌سازی رکوردهای فایل دانشجو ـ درس به کمک ماتریس بیتی

افزونگی تکنیکی:

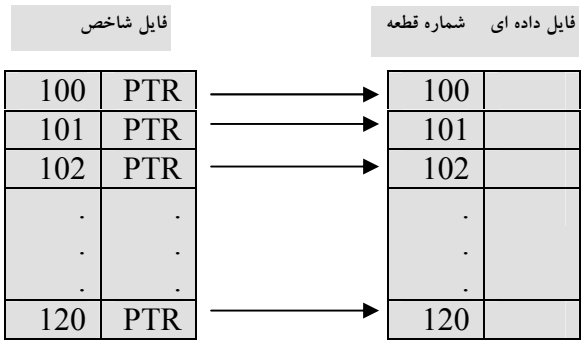
عبارتست از تکرار بعضی یا تمام مقادیر یک(یا چند)صفت خاصه در محیط فیزیکی ذخیره سازی(خود فایل داده ای یا فایل کمکی آن)، به خاطر ایجاد یک شیوه دستیابی کاراکتر برای فایل.

مثال:

به عنوان مثالی از این نوع افزونگی می توان از شاخص بندی نام برد وقتی که یک فایل روی یک صفت خاصه شاخص ایجاد می کنیم، مقادیر آن صفت خاصه(بعضی یا تمام) در محیط ذخیره سازی تکرار می شوند.

این مثال را در زیر با شکل نشان می دهیم

.

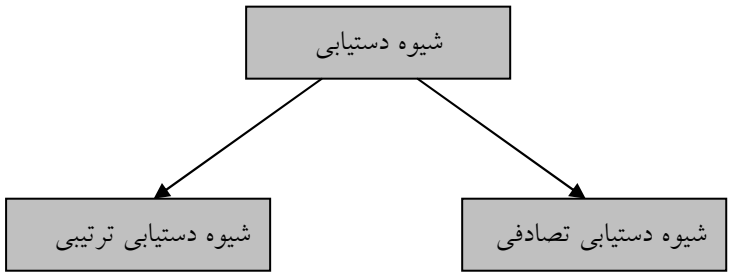


شرح اصول عملیات ششگانه:

۱) واکشی رکورد دلخواه:

این عمل اساساً عملی محتوایی است، یعنی رکوردی باید واکشی شود که مقدار یکی از صفات خاصه اش به عنوان نشانوند جستجو داده شده است.

لازمه این عمل جستجو کردن در فایل، دستیابی به بلاک حاوی رکورد مورد نظر و خواندن آن است.



روشهای تنظیم درخواست واکشی:

این روشها بیشتر در محیطهای DMSیا DBMSبه کار می روند .در محیط FSمعمولاً روش اول از روشهای زیر مورد استفاده قرار می گیرد.

۱) درخواست ساده Single Request
۲) درخواست طیفی Range Request
۳) درخواست محاسباتی Functional Request
٤) درخواست بولی Boolean Request
٥) درخواست مرکب Composite Request

درخواست ساده:

درخواستی که در آن به یک نشانوند جستجو داده می شود و جواب آن در صورت وجود، یک رکورد است .

مثلاً مشخصات دانشجو به شماره Xرا بدهید.

درخواست طیفی:

درخواستی که در آن طیفی از مقادیر کلید اصلی داده شود .مثلاً مشخات دانشجویان از شماره Xتا شماره Y.

درخواست محاسباتی:

درخواستی است که لازمه پاسخ دادن به آن، انجام محاسبه توسط سیستم است، مانند: درخواست بازیابی معدل.

درخواست بولی:

درخواستی است که پاسخ به آن با انجام عملیات بولی و با استفاده از عملگرهای AND ، OR و XOR به دست می آید.

مثلاً اسامی دانشجویانی که قدشان X و وزنشان Y باشد را بدهید .

درخواست مرکب :

درخواستی است که در آن مقدار چند صفت خاصه داده شود و حالت خاصی از پرس و جوی بولی تلقی می گردد.

۲) بازیابی رکورد بعدی:

در بحث لوکالیتی دیدیم که رکورد بعدی منطقی، رکوردی است که بر اساس یک نظم خاص مورد نظر پردازشگر فایل، بغد از رکورد فعلی باید بازیابی شود .

این نوع بعدی، همان بعدی مقداری است و رکوردی است که مقدار نشانوند جستجوی آن، مقدار بعدی مقدار نشانوند جستجوی رکورد فعلی در نظم صعودی مقادیر نشانوند جستجو است.

به عنوان مثال: فایل دانشجو – درس را در نظر می گیریم. اگر این فایل بر اساس نظم صعودی مقادیر شماره دانشجو مرتب شده باشد و دانشجوی شماره X را واکشی کرده باشیم، رکورد بعدی، رکورد دانشجویی است که مقدار شماره دانشجویی اش بلافاصله بزرگتر از مقدار X باشد.

بطور کلی موقعیت رکورد بعدی نسبت به رکورد فعلی به سه صورت زیر می باشد:

۱) رکورد بعدی همجوار فیزیکی رکورد فعلی است.
۲) رکورد فعلی به رکورد بعدی نشانه رو دارد(مستقیم یا غیر مستقیم).
۳) هیچ ارتباطی بین رکورد فعلی و رکورد بعدی وجود ندارد.

۳) بهنگام سازی از طریق درج:

منظور از این عمل درج یک رکورد جدید، بعد از لود اولیه فایل، در فایل است و ما از این پس فقط آنرا عمل

درج می نامیم.

نکته اساسی در عمل درج اینست که بایستی رکورد وارد بلاک شود. این بلاک باید یافته و خوانده شود اما اینکه این بلاک، کدام بلاک باشد، بستگی به ساختار فایل دارد که به طور کلی به دو حالت کلی تقسیم می شوند .

۱) رکورد باید در یک بلاک خاصی درج شود .به این بلاک اصطلاحاً نقطه منطقی درج می گویند.

۲) بلاک خاصی مطرح نیست، رکورد در هر جای فایل می تواند درج شود. در این حالت معمولاً رکـورد در آخرین بلاک فایل وارد می شود و اصطلاحاً می گوییم رکورد را در انتهای فایل الحاق می کنیم و نـشانگر پایان فایل تغییر می کند.

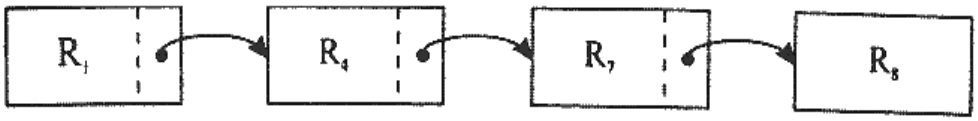
اصول عملیات در عمل درج چنین است:

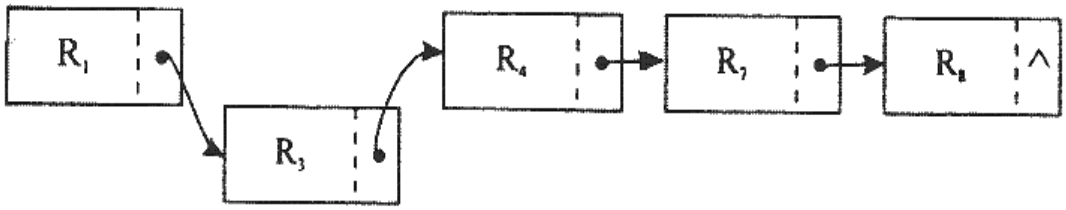
۱) یافتن و خواندن بلاکی که رکورد آن درج شود.

۲) جا دادن رکورد در بلاک که اینک در بافر است.

۳) باز نویسی بلاک

٤) در بعضی از ساختارها، عملیات دیگری هم لازم است که به آنها عملیات پس از درج می گوییم.





٤) بهنگام سازی از طریق تغییر محتوای رکورد:

این عمل یعنی تغییر مقدار یا بیش از یک صفت خاصه در یک رکورد مشخص. نکتـه مهـم اینـست کـه رکـورد بهنگام در آمدنی باید واکشی شود. از این پس این عمل را عمل بهنگام سازی می نامیم

رکورد بهنگام آمده، ممکن است در مکان قبلی اش باز نوشته شود و یا اساساً در مکانی جدید نوشته شود .بـرای حالت اول بهنگام سازی در جا In Place رخ می دهد و در حالت دوم بهنگام سازی برون از جا Out Place رخ می دهد.

بطور کلی اصول عملیات بهنگام سازی یک رکورد چنین است:

۱) واکشی رکورد بهنگام در آمدنی.
۲) کار در بافر (ایجاد نسخه جدی).
۳) بازنویسی نسخه جدید (در جای قبلی)، در بهنگام سازی درجا.
٤) بازنویسی نسخه قدیم با نشانگر((حذف شده)) در بهنگام سازی برون از جا و درج نسخه جدید در جای دیگر.
۵) در برخی از ساختارها انجام عملیات پس از بهنگام سازی به منظور تنظیم ارتباط ساختاری بین رکورد با رکوردهای دیگر فایل. زیرا عمل بهنگام سازی نیز، مثل عمل درج محیط فیزیکی ذخیره سازی را تغییر می دهد .

۵) خواندن تمام فایل:

در خواندن فایل عملیاتهای زیر اجرا می شود.

۱) پیرو درخواست کاربر، مثل لیست گیری ها، انجام یک پردازش خاص روی تمام رکوردها.
۲) ایجاد نسخه ای دیگر از فایل(پیرو درخواست کاربر یا بنابر نیاز سیستمی).
۳) در سازماندهی مدد.
٤) در ایجاد یک استراتژی برای فایل.

خواندن تمام فایل به دو صورت ممکن است انجام شود.

۱) به صورت پی در پی، یعنی بلاک به بلاک از آغاز فایل تا انتهای آن.
۲) به صورت سریال یعنی بر اساس نظم صعودی مقادیر یکی از صفات خاصه.

٦) سازماندهی مجدد:

هر فایلی پس از لود اولیه دوره حیاتی دارد که طی آن عملیاتی را اعم از بازیابی یا ذخیره سازی تحمل می کند.در اثر تغییراتی که در فایل ایجاد می شود ممکن است به تدریج فایل کارایی اولیه اش را در دست بدهد و لذا باید آنرا سازماندهی مجدد کرد.

دلایل کاهش تدریجی کارایی فایل به طور کلی عبارتند از:

۱) از بین رفتن نظم (یا وضع)ساختاری آغازین.
۲) بروز فضای هرز در فایل.
۳) بروز وضعیت نامطلوب در استراتژی دستیابی (در فایلهای شاخص دار).

با توجه به دلایل فوق می توان دلایل سازماندهی مجدد فایل را بصورت زیر بیان کرد:

۱) احیاء نظم یا وضع ساختاری آغازین.

۲) بازستانی فضای هرز.

۳) اصلاح استراتژی دستیابی.

ریز عملیاتهای لازم برای سازماندهی مجدد:

۱) خواندن تمام فایل (پی در پی، سریال).

۲) بلاک بندی مجدد رکوردها ضمن خارج کردن رکوردهای حذف شدنی.

۳) بازنویسی رکوردهای فعال.

٤) بازسازی ساختار مربوط به استراتژی دستیابی (در صورت وجود).

زمان بازنویسی بلاک:

اگر $C_B << 2r$ می توان بلاک را در همان دور جاری دیسک بازنویسی کرد.

$$T_{RW} = 2r - b_{tt} + b_{tt} = 2r$$

R/W زمان انتظار برای رسیدن مجدد آغاز بلاک به زیر نوک

زمان انتقال از بافر به رسانه(نوشتن فیزیکی)

در بحث آتی، غالباً این فرض بر نهاده می شود.اما شرط $C_B << 2r$ از دقت کافی برخوردار نیست می توان این شرط را به گونه ای دقیقتر بیان کرد به شرح زیر:

رابطه $T_{RW} = 2r$ هنگامی صادق است که زمان لازم برای پردازش بلاک یعنی C_B چنان باشد که داشته باشیم:

$$C_B < 2r \frac{(T_F - 1)(B + G) + G}{T_F (B + G)}$$

اگر در رابطه فوق T_F تعداد بلاکها در شیار باشد و $T_F > 1$ آنگاه:

$$C_B < 2r \frac{T_F - 1}{T_F}$$

اگر عملیات در بافر به موقع صورت نگیرد سیستم یک دور دیگر را از دست می دهد.

$$T_{RW} = 4r$$

اگر $C_B >> 2r$

$$T_{RW} = C_B + T_F + b_{tt}$$

فایل با ساختار پایل:

این فایل ساختاری فاقد هر گونه نظم است، یعنی رکوردها بر اساس مقادیر هیچ صفت خاصه ای مرتب نیستند.
در بهترین حالت، نظم بین رکوردها، نظمی است زمانی، مانند اینکه رکوردها بر روی یکدیگر پشته شده باشند.در این ساختار رکوردها قالب غیر ثابت مکانی و طول متغییر دارند.

موارد استفاده فایلهای پایل:

*در محیطهایی که در آنها، داده ها نظم پذیر نباشند و پیش پردازشی روی داده ها انجام نشده باشد و فایل اساساً برای بایگانی ایجاد شود.
*در محیطهای با داده های استراتژیک وقتی که ایمنی داده ها مورد نظر باشد، بی نظمی می تواند ایمنی فایل را افزایش دهد. البته باز هم فایل باید برای بایگانی باشد.
*مبنایی است برای مطالعه و درک بهتر ساختارهای دیگر و نیز طراحی ساختار کاراکتر.

متوسط اندازه رکورد:

*فایل در لود اولیه دارای n رکورد می باشد.
*کل تعداد صفات خاصه در نظر گرفته شده و در محیط عملیاتی را a می نامیم.
*متوسط حافظه لازم برای ذخیره سازی اسم صفت خاصه را، A بایت در نظر می گیریم.
*متوسط حافظه لازم برای ذخیره سازی مقدار صفت خاصه را Vبایت فرض می کنیم.
با این مفروضات و با توجه به قالب رکورد می توان نوشت:

$$R = a'(A + V + 2)$$

که در آن یک بایت برای علامت انتساب و یک بایت برای علامت جداساز منظور شده است.

واکشی رکورد:

نشانوند جستجو درخواست به صورت $A_i = V$ داده می شود.

عملیات لازم:

خواندن بلاک حاوی رکورد مورد نظر با جستجوی خطی می باشد .رکورد ممکن است در اولین بلاک فایل باشد و یا مثلاً در آخرین بلاک فایل باشد. بنابراین به طور متوسط نصف بلاکهای فایل باید خوانده و وارسی شود.
اگر تعداد بلاکهای فایل b و اندازه بلاک B بایت باشد زمان واکشی از این رابطه به دست می آید.

$$T_F = \frac{1}{2} n \frac{R}{t'}$$

$$T_F = \frac{1}{2} b \frac{B}{t'}$$

در این ساختار، زمان واکشی، زیاد و سیستم ناکار است، برای تسریع در واکشی، می توان از تکنیک دسته بندی درخواستها کرد و به آنها به صورت یکجا پاسخ داد. اگر Lدرخواست واکشی را دسته بندی کنیم، در این صورت خواهیم داشت:

$$T_F(L) = 2T_F, L >> 1$$

و به طور متوسط، زمان لازم برای واکشی یک رکورد: $\frac{2}{L}T_F$ خواهد بود.

$$\frac{1}{2}T_F + \frac{1}{4}T_F + ...$$
$$T_F(L) = T_F + T_F = 2T_F$$
$$T_F(1) = \frac{2}{L}T_F$$

بازیابی رکورد بعدی:

رکورد بعدی، در این ساختار مفهومی ندارد، زیرا هیچگونه ارتباط ساختاری بین رکورد فعلی و بعدی آن برقرار نیست.

$$T_N = T_F$$

عمل درج:

چون فایل فاقد هرگونه نظم است، لذا رکورد جدید به انتهای فایل الحاق می شود و برای این منظور:

۱) خواندن آخرین بلاک فایل که سیستم آدرس آنرا دارد.

۲) کار در بافر (که زمانش را در ارزیابی دخالت نمی دهیم): انتقال رکورد از ناحیه کاری برنامه کاربر به بلاک که در بافر است.

۳) بازنویسی بلاک

لازم است:

$$T_I = S + r + b_{tt} + T_{RW}$$
$$T_{RW} = 2r$$
$$T_I = S + 3r + b_{tt}$$

عمل بهنگام سازی:

در این ساختار عمل بهنگام سازی در حالت کلی، به صورت برون از جا انجام می شود.

عملیات لازم:

۱) واکشی رکوردی که می خواهیم به هنگام کنیم.

۲) ضبط نشانگر ((حذف شده))در بخش قدیم.
۳) ایجاد نسخه جدید.
۴) بازنویسی نسخه قدیم.
۵) درج نسخه جدید در انتهای فایل.

پس داریم:

$$T_U = T_F + T_{RW} + T_I$$

T_F:واکشی رکوردی که می خواهیم به هنگام کنیم.
T_{RW} : بازنویسی همین رکورد با نشانگر((حذف شده)).
T_I :درج نسخه جدید.

عمل حذف حالت خاصی است از بهنگام سازی که در آن درج نسخه جدید انجام نمی شود و زمان آن برابر است با:

$$T_{U_{delet}} = T_F + T_{RW}$$

خواندن تمام فایل:

به سادگی می توان دریافت که:

Seqپی در پی $T_{X_{seq}} = 2T_F$

در خواندن تمام فایل بصورت سریال، ابتدا باید مرتب شود و سپس بصورت پی در پی خوانده شود.

$$T_{X_{ser}} = T_{sort(n)} + T_{X_{seq}}$$

سازماندهی مجدد:

زمان سازماندهی مجدد بصورت زیر خواهد بود:

$$T_Y = (n+o)\frac{R}{t'} + (n+o-d)\frac{R}{t'}$$

که در آن:

زمان خواندن کل فایل: $(n+o)\frac{R}{t'}$

زمان بازنویسی کل فایل: $(n+o-d)\frac{R}{t'}$

$\frac{R}{t'}$:زمان لازم برای نوشتن یک رکورد می باشد.

فایل ترتیبی:

در این ساختار طول رکوردها ثابت بوده و مکان و طول هر فیلد در رکورد ثابت و مشخص است. بنابراین در این ساختار دیگر نیازی نیست که نام فیلد در کنار مقدار ذخیره شود. همچنین در این ساختار در لود اولیه، تمام رکوردها بر مبنای یک فیلد (یا ترکیبی از چند فیلد) مرتب شده می باشد.

مثال ۱:

فایل زیر نمونه ای از فایل ترتیبی است که رکوردها بر اساس صفت خاصه (فیلد)فامیلی به صورت صعودی مرتب شده اند:

شماره رکورد	نام	فامیلی	معدل	سال تولد
1	علی	امیری	16	53
2	حسن	جعفری	17	55
3	امیر	رازی	15/5	54
4	جواد	سعیدی	14	52
5	سامان	مولایی	18/5	53

طول و ساختار هر رکورد عموماً در هدر فایل ذخیره می گردد.

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

در این ساختار(فایل ترتیبی)تقارن وجود ندارد چرا که تنها بر اساس یک فیلد مرتب شده است.

تذکر ۱:

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، عمل درج در فایل اصلی انجام نمی گیرد، بلکه در یک فایل کمکی به نام فایل ثبت تراکنشها یا TLF(Transaction Log File)صورت می گیرد.

تذکر ۲:

مکان ثبت تراکنشها هم می تواند به صورت یک فایل مجزا (TLF)در نظر گرفته شود و هم می تواند ناحیه ای در انتهای فایل اصلی باشد. به این امکان ثبت تراکنشها، ناحیه سر ریزی (Over Flow Area) نیز می گویند.

تذکر ۳:

فایل ترتیبی اغلب در مواردی استفاده می شود که طول رکوردها ثابت بوده و معمولاً واکشی سریع رکوردها به صورت تک تک مورد نیاز نباشد. هنگامی که پردازش سریالی رکوردها مورد نظر باشد، ساختار ترتیبی به مراتب بهتر از پایل است. در بسیاری از سیستم های تجاری که رکوردها به صورت دسته ای (Bach) پردازش می شوند، از ساختار ترتیبی می توان استفاده کرد.

متوسط اندازه رکورد در فایل ترتیبی:

$$R = av$$
$$S_{file} = n.av$$

در این ساختار پس از ود اولیه در عمل در فایل ثابت تراکنش ها انجام می شود. لذا برای ارزیابی دقیقتر در لود اولیه باید ظرفیت آن فایل را نیز دخالت دهیم.

$$S_{file} = (n + o).av$$

در این فرمول فرض می کنیم که فایل تراکنش ها ظرفیت Oرکورد را دارد.

واکشی رکورد:

برای واکشی رکوردها دو حالت زیر را در نظر می گیریم:

الف) نشانوند جستجو غیر از صفت خاصه نظم باشد.

ب) نشانوند جستجو همان صفت خاصه نظم (کلید)باشد.

حالت الف) در این حالت، فایل عملاً تبدیل می شود به حالت خاصی از پایل و با جستجوی خطی خواهیم داشت:

$$T_F = \frac{1}{2} n . \frac{R}{t'}$$

و اگر فرض کنیم TLFبه تعداد o رکورد باشد و در لحظه واکشی o' رکورد در ان وجود داشته باشد $(O' \le O)$ارزیابی دقیقتر به صورت زیر می باشد.

$$T_F = \frac{1}{2}(n + o')\frac{R}{t'}$$

حالت ب) برای واکشی رکورد باید جستجوی خارجی انجام داد و زمان بستگی به الگوریتم جستجو، یکی از الگوریتم های رایج برای جستجو، الگوریتم جستجوی دودویی است که در اینجا باید در یک محیط منظم خارجی اجرا شود.

جستجوی دودویی:

جستجوی دودویی، در یک محیط منظم خارجی باید در دو سطح انجام می شود. در سطح اول، جستجویی در فایل داریم تا بلاک مورد نظر پیدا شود، برای این کار طبعاً باید بلاکها خوانده شوند.

در سطح دوم برای هر بلاک که به بافر آورده می شود، یک جستجوی دودویی درون بلاکی داریم. هر دو جستجو در ارزیابی زمان دخالت دارند.

$$T_{F_{binarysearch}} = \log_2(n\frac{R}{B})(s + r + b_{tt} + C_B) + T_{F_O}$$

,

$$T_{F_O} = \frac{1}{2}O'\frac{R}{t'}$$

یاد آوری می شود که T.L.Fدارای نظم زمانی است. ممکن است رکورد در فایل اصلی نباشد، یعنی رکورد، درجی باشد، لذا دخالت دادن T_{F_O} لازم است. چون واحد جستجو در سطح خارجی بلاک است، $\log_2(n\frac{R}{B})$، دفعات مراجعه به فایل خواهد بود و در هر بار مراجعه، یک بلاک (بلاک میانی) به طور مستقیم خوانده می شود.

جستجو با پرش بلاکی:

این روش در اساس همان روش جستجوی خطی است. با این تفاوت که در جستجوی درون بلاک، کلید رکورد مورد نظر با کلید آخرین کورد هر بلاک مقایسه می شود، چنانچه این کلید از کلید رکورد مورد نظر بزرگتر باشد، محتوای بلاک خوانده شده مثلاً به صورت خطی بررسی می شود. در غیر این صورت سیستم با پرش از بلاک، بلاک بعدی را می خواند و به طول متوسط، نصف بلاکها باید خوانده شوند.

محاسبه نشان می دهد که مناسبترین اندازه برای بلاک(مقدار بهینه B_f)برابر است با $\sqrt{n}$و $B_f = \sqrt{n}$.

اگر Nتعداد مقایسه لازم برای یافتن رکورد باشد می توان نوشت:

$$N = \frac{b}{2} + \frac{b_f}{2}$$

$$N = \frac{\frac{n}{B_F}}{2} + \frac{B_F}{2}$$

$$\frac{d(N)}{d(B_F)} = 0 \Rightarrow N' = \frac{-n}{(B_F)^2} + 1 = 0 \Rightarrow B_F = \sqrt{n}$$

جستجو با تخمین و کاوش:

در این روش ابتدا آدرس تقریبی تخمین زده می شود و سپس از این آدرس جستجوی خطی انجام می گیرد تا رکورد مورد نظر پیدا شود. .محاسبه لازم واکشی رکورد در این روش آسان نیست، ولی بطور کلی می توان نوشت:

$$T_F = s + r + b_{tt} + k(\frac{B}{t'}), k > 1$$

k: تعداد بلاکهایی است که باید خوانده شوند تا رکورد یافته شود.

بازیابی رکورد بعدی در فایلهای ترتیبی:

در این ساختار، با خواندن هر بلاک، B_F رکورد به دست می آید که هر یک بعدی رکورد قبلی است.
بنابراین می توان نوشت:

$$T_N = \frac{\frac{B}{t'}}{B_F} = \frac{R}{t'}$$

البته ممکن است رکورد بعدی در بلاک بلافاصله بعدی باشد. احتمال این حالت $\frac{1}{B_F}$ است.

فایلهای ترتیبی:

عمل درج:

رکورد درج دنی را نمی توان به انتهای فایل الحاق کرد. زیرا باید در نقطه خاصی درج شود و این کار در فایلهای بزرگ زمانگیر است.
در عمل رکورد در TLF درج می شود تا در سازماندهی مجدد، فایل بر اساس نظم مورد نظر بازآرایی شود.

برای ارزیابی زمان درج، دو حالت را در نظر می گیریم:

حالت اول: درج در فایل کوچک

عملیات لازم برای این حالت عبارتند از:

۱) یافتن نقطه منطقی درج.
۲) درج رکورد در بلاک مورد نظر (کار در بافر).
۳) شیفت دادن بلاکها از نقطه منطقی درج به سمت .EOF

و خواهیم داشت:

$$T_I = T_F + \frac{1}{2}b(\frac{B}{t'} + T_{RW})$$

T_F :یافتن نقطه منطقی درج

$\frac{B}{t'} + T_{RW}$:زمان شیفت یک بلاک

حالت دوم: درج در حالت کلی:

رکورد درج شدنی در آخرین بلاک فایل ثبت تراکنش ها درج می شود و زمان این عمل همان زمان درج در

پایل است. به علاوه زمان دیگری نیز باید دخالت داده شود که $\frac{T_Y}{O}$ است.

زمان T_Y بین O رکورد سرشکن می شود.

عمل بهنگام سازی فایل ترتیبی:

عملیات لازم:

۱) واکشی رکوردی که باید به هنگام شود.

۲) عمل بهنگام سازی در بافر (ایجاد نسخه جدید)

۳) درج رکورد بهنگام درآمده، همراه با یک رکورد کوچک ضمیمه شده به آن که این رکورد کوچک، تاریخ بهنگام سازی و نشانگر ((حذف شده)) وجود دارد

$$T_U = T_F + T_I$$

خواندن تمام فایل(فایلهای ترتیبی):

خواندن پی در پی به روش معمول انجام می شود و زمان خواندن بطور پی در پی برابر است با:

$$T_{X_{seq}} = (n + o')\frac{R}{t'}$$

برای خواندن سریال، فایل تراکنشها باید مرتب شود. زمان چنین برآورد می شود:

$$T_{X_{ser}} = T_{sort}(o') + (n + o')\frac{R}{t'}$$

سازماندهی مجدد(فایلهای ترتیبی):

برای سازماندهی مجدد عملیات زیر انجام می شود:

۱) مرتب کردن فایل تراکنش(تا با فایل اصلی همتوالی شود).

۲) خواندن فایل اصلی.

۳) خواندن فایل تراکنش.

۴) بلاک بندی مجدد رکوردها ضمن خارج کردن رکوردهای حذف شدنی.

۵) بازنویسی کل فایل.

$$T_Y = T_{sort}(o) + n.\frac{R}{t'} + o.\frac{R}{t'} + (n + o - d)\frac{R}{t'}$$

تمرین ۱:

اگر تعداد دروس یک دانشکده 20درس باشد و هر دانشجو حداکثر درس را اخذ کند و فیلد شماره دانشجویی 2بایت باشد، فایل اطلاعاتی دانشجو، درس در روش ماتریس بیتی برای 350دانشجو چند بایت خواهد بود ؟ (بدون احتساب فیلدها)

پاسخ ۱:

برای20 درس به 20 بیت یعنی 3 بایت فضا نیاز داریم:

$$350\times(3\times2)=1750Bbyte$$

تمرین۲:

فایل پایلی با 100000رکورد 400بایتی داریم مرتباً به ازای هر 3 رکورد اضافه شده جدید یک رکورد حذف می شود تا هنگامی مه تعداد رکوردهای فعال (Active)به 150000 برسد. اگر بایت B=24000 و $b'_{tt}=./84ms$ باشد زمان سازماندهی مجدد ان چند ثانیه خواهد بود ؟

پاسخ۲:

$$100000-x+3x=150000 \Rightarrow x=25000$$

X: تعداد باری که یک رکورد حذف شده است.

پس هنگامی که تعداد رکوردهای فعال به 150000 می رسد ، تعداد 25000 رکورد علامت حذف خورده اند و تعداد کل رکوردها برابر 175000 می باشد. $B_F=\frac{2400}{400}=6$

تعداد بلاکهای فایل قبل از سازماندهی مجدد: $b=\frac{175000}{6}$

$$T_Y=(b+\frac{n}{B_F})\times b'_{tt}$$

$$T_Y=(\frac{175000}{6}+\frac{150000}{6})\times0.8=45500ms=45.5\sec$$

تمرین ۳:

فایل پایلی شامل 450بلاک بوده و هر بلاک برابر 2400بایت است. اگر نرخ انتقال 3000بایت در ثانیه باشد . زمان خواندن کل فایل به صورت ترتیبی ( ) Sequential چند ثانیه خواهد بود ؟

پاسخ۳:

$$T_{X_{seq}}=2T_F=2\times\frac{1}{2}b\frac{B}{t'}=450\times\frac{2400}{3000}=360s$$

تمرین ٤:

فایل پایلی شامل 450بلاک بوده و هر بلاک برابر 2400بایت است.اگر نرخ انتقال 3000بایت در ثانیه باشد، زمان بدست آوردن رکورد بعدی (T_N)چند دقیقه خواهد بود ؟

پاسخ ٤:

$$T_N = T_F = \frac{1}{2} b \times \frac{B}{t'} = \frac{1}{2} 450 \times \frac{2400}{3000} = 180s$$
$$T_N = 180s = 3\min$$

تمرین ٥:

فایل ترتیبی با *100000*رکورد *400*بایتی داریم اگر *b=2400*بایت و $b_{tt} = .8ms$ و *r=8/3 ms* و s=16msباشد، زمان واکشی 10رکورد با جستجوی باینری چند میلی ثانیه خواهد بود ؟

پاسخ٥:

$$b = \frac{100000 \times 400}{2400} \approx 16667$$
$$T_F = \left[(\log_2 b - 1)\right] \times (s + r + b_{tt}) = 13 \times (16 + 8.3 + 8.0) = 326ms$$

زمان واکشی 10 رکورد: $10 \times 326 = 3260ms$

تمرین ٦:

فایلی ترتیبی با 16667بلاک داریم. می خواهیم 10000رکورد مختلف را واکشی کنیم. با روش جستجوی باینری این عمل چند ثانیه طول می کشد؟

$$S = 16ms, r = 8/3ms, b_{tt} = .8ms$$

پاسخ٦:

$$10000T_F = 10000\left[\log_2 b - 1\right] \times (s + r + b_{tt}) = 10000 \times 13 \times (16 + 8.3 + 0.8) = 3260000ms = 3260s$$

تمرین ۷:

یک فایل ترتیبی شامل 900رکورد است. هر بلاک شامل چند رکورد باشد تا جستجوی بلاکی با سرعت خوبی انجام گیرد؟

پاسخ۷:

$$B_F = \sqrt{n}$$
$$B_F = \sqrt{900} = 30$$

تمرین ۸:

یک فایل ترتیبی شامل 1500رکورد 300بایتی می باشد. اگر نرخ انتقال اطلاعات 2500کیلو بایت در ثانیه باشد، زمان واکشی یک رکورد بر مبنای فیلد غیر کلید چند میلی ثانیه خواهد بود؟(هر کیلو 1000بایت است)

پاسخ۸:

$$T_F = \frac{1}{2} n \frac{R}{t'} = \frac{1500 \times 300}{2 \times 2500 \times 1000} = 0.09s = 90ms$$

تعریف شاخص:

در انتهای کتابها شاخص(index)وجود دارد بع این معنا که لیست کلمات و اصطلاحات مهم (کلیدها)کتاب به ترتیب حروف الفبا، به همراه شماره صفحاتی که در آن کلمات استفاده شده (آدرس فیلدها)آورده می شود. پس شاخصها بر مبنای کلیدها و آدرس فیلدها ساخته می شوند. شاخص باعث بالا رفتن سرعت دستیابی می گردد.

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

فایل ترتیبی

	معدل	نام پدر	نام	شماره دانشجوئی
۱	۱۷	سعید	علی	۳۹۲۵
۲	۱۹	مجید	حسن	٤٧١٣
۳	١٦	شاهین	امیر	٥٤١٧
٤	۱۵	سهیل	جواد	٧٣٥٤
			⋮	

فایل ایندکس معدل

شماره رکورد	معدل
٤	۱۵
۳	١٦
۱	۱۷
۲	۱۹

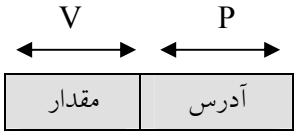
فایل ایندکس اولیه

شماره رکورد	شماره دانشجویی
۱	۳۹۲۵
۲	٤٧١٣
۳	٥٤١٧
٤	٧٣٥٤

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

با توجه به مطالب فوق می توان گفت فایل شاخص مجموعه ای از تعدادی مدخل (entry) به فرم کلی زیر است:

:



تذکر: به فایل داده ای اصلی، فایل شاخص بندی شده (Indexed File)و به فایل ایندکس کنار آن فایل شاخص (Index File) می گویند.

شاخص متراکم و غیر متراکم:

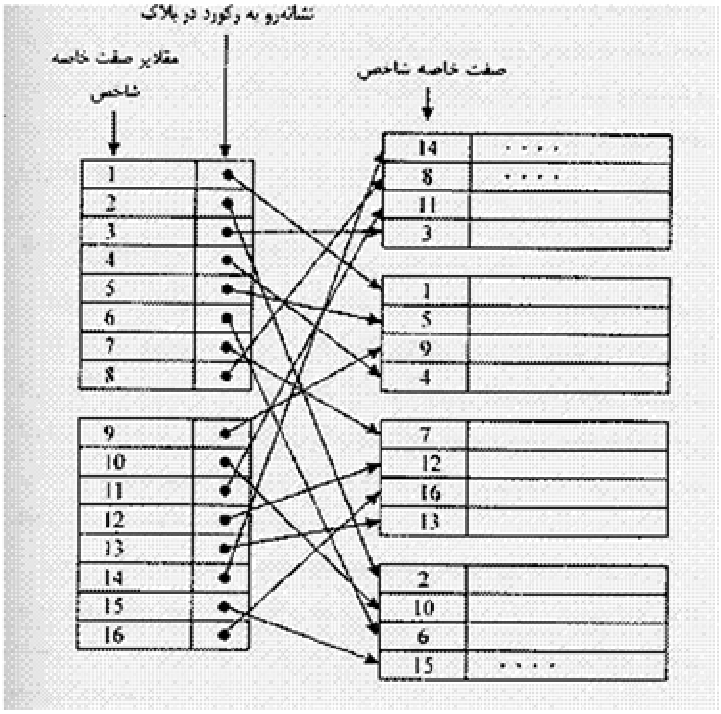
به هر نقطه از فایل داده ای که از مدخل شاخص به آن نشانگر وجود دارد را لنگرگاه یه Anchor Pointمی

گویند.

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

فایل داده ای و فایل شاخص می توانند بلاک بندی شده باشند یا نشده باشند. در حالت بلاک بندی شده اغلب اندازه بلاک شاخص و بلاک فایل داده ای یکسان است.

مثال: نمونه ای که در مثال قبل ذکر شد از نوع شاخص متراکم بود.شکل زیر نمونه ای از شاخص نا متراکم می باشد.



در شاخص نا متراکم مقدار موجود داده هر مدخل، می تواند کوچکترین یا بزرگترین مقدار در هر گروه باشد. در مثال فوق کوچکترین مقدار در هر گروه را در شاخص قرار داده ایم.

اگر فایل ضاخص، بلاک بندی شده باشد اغلب اندازه بلاک آن را با بلاک فایل داده ای یکسان می گیریم. تعداد مدخلهای یک بلاک شاخص را ظرفیت نشانه روی یا Index Fanout گفته و آن را با yنمایش می دهیم بدیهی است که داریم:

$$y = \left\lfloor \frac{B}{V + P} \right\rfloor$$

تمرین ۱:

اگر طول بلاک 200بایت اندازه صفت خاصه شاخصV برابر 14بایت و اندازه اشاره گر شخص Pبرابر 6بایت باشد، ظرفیت نشانه روی هر بلاک شاخص چقدر است ؟

پاسخ ۱:

$$y = \left\lfloor \frac{2000}{14+16} \right\rfloor = \left\lfloor \frac{2000}{20} \right\rfloor = 100$$

یعنی هر بلاک دارای 100 سطر یا مدخل است و هر مدخل اشاره گری به رکورد (یا گروهی از رکوردها) در فایل داده ای اصلی می باشد

تمرین۲:

فایلی با مشخصات زیر را در نظر می گیریم مطلوب است طراحی شاخص غیر متراکم برای این فایل ؟

$$p = 6byte, V = 14byte, B = 2000byte, R = 200byte, n = 10^6, e_i = b_{i-1}$$

پاسخ ۲:

$$e_i = b_i - 1$$

$$b_i = \left\lceil \frac{e_i}{y} \right\rceil$$

$$S_{Ii} = B.b_i$$

$$B_F = \frac{2000}{200} = 10$$

طول مدخل یک شاخص $V + P = 14 + 6 = 20$

$$y = \left\lceil \frac{2000}{20} \right\rceil = 100$$

$$e_1 = b_0 = b = \frac{10^6}{10} = 10^5$$

$$b_1 = \left\lceil \frac{10^5}{100} \right\rceil = 1000....., S_{I_1} = 2000 * 1000 = 2MB$$

$$e_2 = b_1 = 1000....., b_2 = \left\lceil \frac{1000}{10} \right\rceil = 100....., S_{I_2} = 2000 * 10 = 20KB$$

$$e_3 = b_2 = 10....., S_{I_3} = 10 * 2000 = 20000byte$$

$$X = 3$$

$$S_I = \sum_{i=1}^{X-1} S_{I_i} = 2020000byte$$

---

e_i:تعداد مدخلهای شاخص در سطح i ام

b_i: تعداد بلاکهای شاخص در سطح iام

S_{I_i}: حافظه لازم برای سطح iام

S_I:میزان حافظه لازم(دیسک) برای شاخص

ظرفیت نشانه روی روی بلاک شاخص:

Entry های شاخص در بلاکهایی جای داده می شوند .تعداد entryهای یک بلاک شاخص را ظرفیت نشانه روی آن بلاک می نامیم و آنرا با yنمایش می دهیم.

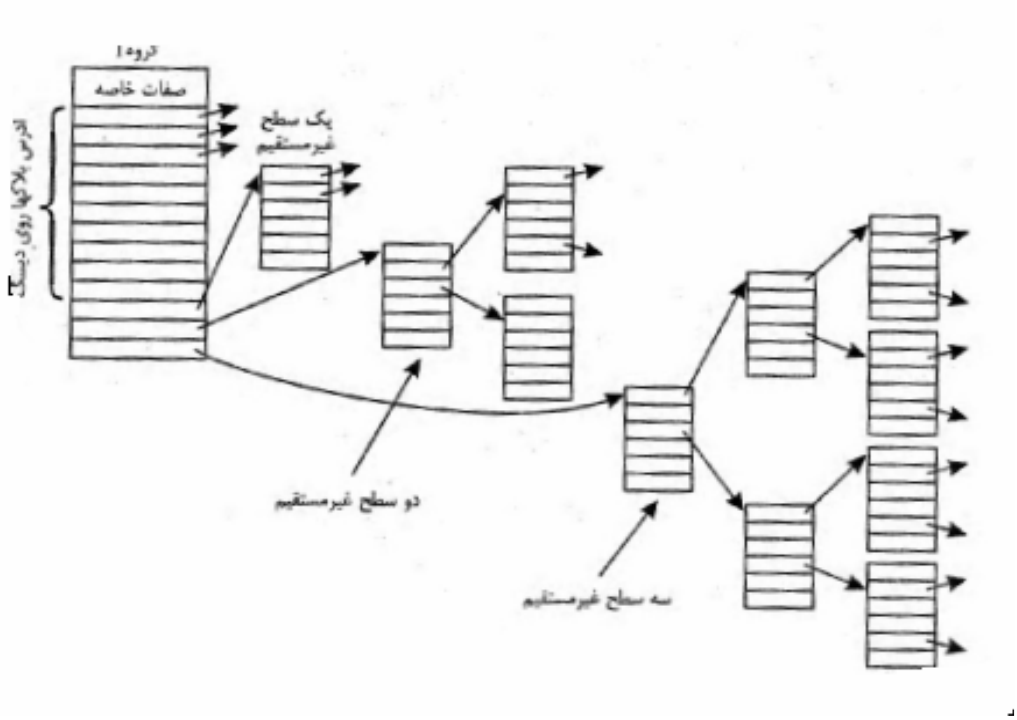
$$y = \left\lfloor \frac{B}{V + P} \right\rfloor$$

$V + P$:طول شاخص

شاخص چند سطحی:

وقتی که تعداد مدخلهای شاخص زیاد باشد، جستجو در شاخص برای یافتن مدخل شاخص مورد نظر، زمانبر می شود. برای سرعت بخشیدن به جستجو در شاخص آنرا در چند سطح ایجاد می کنند.

تعداد سطوح شاخص را عمق شاخص می نامند و با Xنمایش می دهند. در حالت X=1، شاخص را خطی می گویند.



محاسبه ژرفای شاخص:

اگر تعداد مدخلهای سطح شاخص را e_1 بنامیم. با توجه به مفهوم y می توان نوشت:

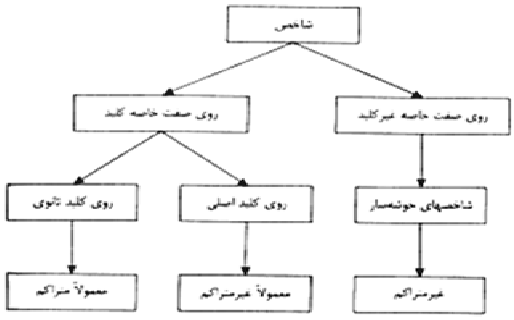
$$y^{x-1} \le e_1 \le y^x$$

و ژرفای شاخص برابر است با:

$$X = \lceil \log_y e_1 \rceil$$

شاخص چد سطحی، ساختاری درختی دارد و معمولاً بر اساس $B-TREE$ یا B^+-TREE ساخته می شود.

جمع بندی انواع شاخص:



ساختار ترتیبی شاخص دار:

این ساختار برای تسریع واکشی تک رکورد از یک فایل ترتیبی طراحی و ایجاد می شود.

اجزاء تشکیل دهنده آن عبارتند از:

۱) فایل ترتیبی که به آن اصطلاحاً ناحیه اصلی می گویند.

۲) ناحیه سرریزی برای انجام عملیات ذخیره سازی پس از لود اولیه.

۳) اشاره گرها.

۴) مجموعه شاخص.

نحوه انجام عملیات در این ساختار:

الف)خواندن پی در پی: سیستم ناحیه اصلی و ناحیه سرریزی را بلاک به بلاک می خواند و چون رکوردهای ناحیه سرریزی مرتب نیستند، پس در این نحوه خواندن، رکوردها به طور سریال خوانده نمی شوند.

ب)خواندن فایل به صورت سریال: بلاکهای ناحیه اصلی روی کلید خوانده می شوند تا به بلاک یا رکوردی برسیم که دارای یک نشانه رو به ناحیه سرریزی است.

ج) واکشی رکورد از روی کلید: برای واکشی رکورد، باید در فایل شاخص جستجو کرد و با یافتن مدخل مربوطه

در فایل شاخص قسمتی از ناحیه اصلی و در صورت لزوم، زنجیره سرریزی های آن قسمت باید خوانده شود تا رکورد مورد نظر در صورت جود، واکشی شود.

ساختار ترتیبی شاخص دار:

این ساختار از تعدادی فایل ترتیبی و تعدادی فایل شاخص تشکیل یافته است .فایل ترتیبی که به آن ناحیه اصلی نیز گفته می شود، بر اساس فیلدی مرتب شده می باشند .در این ساختار به یک ناحیه سرریزی (Over Flow) نیاز است تا عملیات ذخیره سازی پس از بار شدن اولیه فایل اصلی، در ان ناحیه صورت پذیرد. در این ساختار از شاخص برای سرعت بخشیدن به عملیات واکشی استفاده می شود.

مسأله اصلی در این ساختار، مشکل سرریزی است.برای مکان رکوردهای سرریزی انتخابهای زیر وجود دارند.

۱) می توان در انتهای هر بلاک در لود اولیه فضایی را به صورت رزرو کنار گذاشت.

۲) می توان مشابه فایل تراکنش در ساختار ترتیبی، ناحیه سرریزی را در یک فایل جداگانه در نظر گرفت. ولی این روش نیز سرعت کمی داشته و همچنین لوکالیتی رکوردهای سرریزی در آن ضعیف می باشد.

۳) بهترین را ان است که در همان فایل داده یی، فضایی را جهت سرریزی در نظر بگیریم جهت تخصیص فضا در این حالت دو روش وجود دارد:

الف) اختصاص استوانه هایی در انتهای فایل جهت ناحیه سرریزی، ولی از آنجا که با این روش لوکالیتی رکوردهای سرریزی ضعیف شده و متوسط زمان استوانه جویی زیاد می شود، این تکنیک مناسب نیست.

ب) اختصاص شیارهای انتهایی هر استوانه به عنوان ناحیه سرریزی همان استوانه. این روش مناسبی است که زمان استوانه جویی را نیز کاهش می دهد.

روشهای درج سرریزی در ساختار ترتیبی شاخص دار:

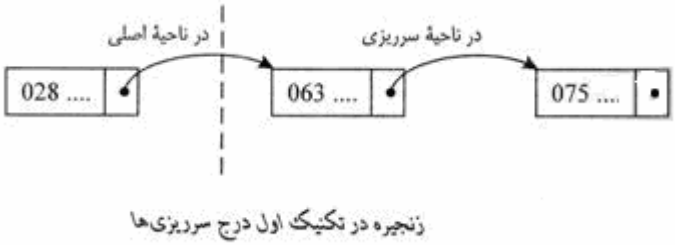
درج در اولین بلاک جا دار در ناحیه سرریزی:

در این روش رکورد جدید به سادگی در اولین فضای آزاد در ناحیه سرریزی قرار داده می شود و از رکورد منطقی قبل از آن، اشاره گری به سمت آن ایجاد می گردد تا ترتیب زنجیره رکوردها حفظ شود.

در این روش در انتهای هر رکورد یک اشاره گر وجود دارد که مقدار آن می تواند NULLباشد مبدأ زنجیره هایی که بدین ترتیب پدید می آید، در ناحیه اصلی بوده و ادامه آنها در ناحیه سرریزی قرار دارد.

مثال:

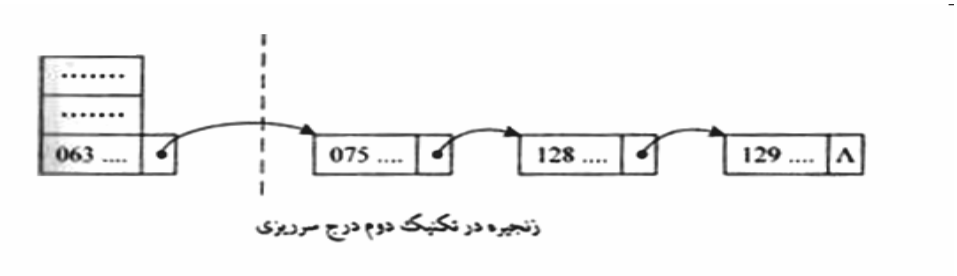
شکل زیر یک فایل ترتیبی دانشجویان را با ناحیه سرریزی آن نشان می دهد ابتدا ناحیه سرریزی خالی بوده است سپس به ترتیب از چپ به راست رکوردهای زیر وارد سیستم شده اند:



زنجیره در تکنیک اول درج سرریزی‌ها

درج به روش Push Through :

در این روش درج به گونه ای انجام می شود که سریالیتی رکوردها در ناحیه اصلی و سرریزی حفظ گـردد .رکـورد جدید ممکن است نهایتاً در ناحیه اصلی یا ناحیه سرریزی قرار بگیرد



زنجیره در تکنیک دوم درج سرریزی

ویژگی های ساختار ترتیبی شاخص دار:

در این ساختار اغلب سطح اول ساخص نا متراکم بوده و بلاکهای شاخص در یک استوانه رار گرفته اند و ناحیـه سرریزی هر استوانه در همان استوانه قرار دارد.

سه عیب اصلی این ساختار عبارتند از:

۱) ایستا بودن.
۲) درج رکوردها در ناحیه سرریزی و طولانی بودن زنجیره ها.
۳) عدم تقارن.

متوسط اندازه رکورد(ساختار شاخص دار)

برای محاسبه اندازه رکورد باید عوامل زیر را در نظر گرفت:
۱) حافظه لازم برای یک رکورد از ناحیه اصلی.
۲) حافظه مصرف شده برای ناحیه سرریزی به ازاء یک رکورد از ناحیه اصلی.
۳) حافظه مصرف شده برای شاخص به ازاء یک رکورد از ناحیه اصلی

$$R = R_{data} + R_{over} + R_{index}$$

$$R_{data} = av + \frac{p}{B_F}$$

$$R_{over} = \frac{o}{n+o}.R', R' = av + p$$

$$R_{index} = \frac{\text{(کل حافظه مصرف شده برای شاخص)}}{n+o} = \frac{S_I}{n+o}$$

$$R = \frac{n(av + \frac{p}{B_F}) + o(av + p) + S_I}{n+o}$$

۱) واکشی رکورد:

برای واکشی یک رکورد دلخواه، عملیات زیر باید صورت پذیرد:

۱) بررسی سرشاخص.

۲) جستجو در سطوح شاخص تا رسیدن به مدخل مربوطه در سطح اول.

۳) خواندن بلاکی از ناحیه اصلی که آدرس آن از مدخل مربوطه در سطح اول شاخص بدست می آید.

۴) به احتمالی، رفتن به ناحیه سرریزی و جستجو در زنجیره سرریزیها.

حال این زمان را در دو بخش زیر مورد بررسی قرار می دهیم.

۱) $T_{F_{main}}$:زمان واکشی رکورد از ناحیه اصلی

۲) $T_{F_{over}}$ زمان یافتن رکورد از ناحیه سرریزی.

$$T_F = T_{F_{main}} + T_{F_{over}}$$

$$T_{F_{main}} = C_B + (x-1)(s + r + b_{tt}) + s + r + b_{tt}$$

C_B :زمان بررسی شاخص

$(x-1)(s + r + b_{tt})$:زمان خواندن بلاکهای شاخص در x-1 سطح

$(s + r + b_{tt})$:خواندن بلاک در ناحیه اصلی

حال اگر تعداد رکوردهای زنجیره یرریزی، L_C باشد می توان نوشت:

$$T_{F_{over}} = pro.(s + r + b_{tt}) + pro.\frac{L_C - 1}{2}(s + r + b_{tt})$$

Proعبارتست از احتمال اینکه رکورد مورد نظر در ناحیه سرریزی باشد.

$$pro = \frac{o'}{n+o'}$$

o':تعداد رکوردهای ناحیه سرریزی در لحظه واکشی

۲) بازیابی رکورد بعدی در ساختار ترتیبی شاخص دار:

اگر رکورد بعدی در بلاکی از ناحیه اصلی باشد با زمان $\frac{1}{B_F} b_{tt}$ به دست می آید .احتمال اینکه رکورد بعدی در ناحیه سرریزی باشد $\frac{o'}{n+o'}$ است پس می توان نوشت:

$$T_N = \frac{1}{B_F} b_{tt} + \frac{o'}{n+o'}(r + b_{tt})$$

۳) عمل درج در ساختار ترتیبی شاخص دار:

برای درج یک رکورد جدید، عملیات زیر باید انجام شود:

۱) یافتن بلاکی که رکورد باید در آن درج شود.

۲) وارد کردن رکورد در این بلاک ضمن خارج کردن آخرین رکورد بلاک و قرار دادن آن در بافر کمکی و ساختن فیلد PTRبه رکورد جابجا شده.

۳) بازنویسی این بلاک.

٤) خواندن بلاکی از ناحیه سرریزی.

۵) وارد کردن رکورد خارج شده از بلاک اصلی، در این بلاک.

٦) بازنویسی همین بلاک.

$$T_I = T_F + T_{RW} + r + b_{tt} + T_{RW}$$
$$T_I = T_F + 5r + b_{tt}$$

٤) عمل بهنگام سازی در ساختار ترتیبی شاخص دار:

عملیات لازم:

۱) واکشی رکوردی که می خواهیم بهنگام کنیم.

۲) ایجاد نسخه جدید(در بافر)

۳) بازنویسی نسخه جدید.

$$T_{U_{inplace}} = T_F + T_{RW}$$
$$T_{U_{inplace}} = T_F + 2r$$

بهنگام سازی برون از جا:

$$T_{U_{outplace}} = T_F + T_{RW} + T_I$$
$$T_{U_{outplace}} = 2T_F + 7r + b_{tt}$$

۵) خواندن تمام فایل ساختار ترتیبی شاخص دار:

۱) در حالت سریال.

اولین رکورد واکشی می شود و بقیه رکوردها طی یک سلسله عملیات بازیابی رکورد بعدی، خوانده می شوند.

$$T_{X_{ser}} = T_F + (n + o' - 1)T_N$$

اگر تکنیک درج را، درج با جابجایی فرض کنیم، زمان خواندن سریال را به طرز دیگری هم مـی تـوان ارزیـابی کرد. در این تکنیک، رکوردهای اولین بلاک ناحیه اصلی همیشه مرتب است .پس سیستم می تواند رکوردهای این بلاک را بخواند و سپس بقیه رکوردها را طی یک سلسله عملیات بازیابی بعدی به دست آورد در اینصورت خواهیم داشت:

$$T_{X_{ser}} = s + r + b_{tt} + (n + o' - B_F)T_N$$

۲) در حالت پی در پی:

$$T_{X_{seq}} = (n + o')\frac{R}{t'}$$

٦) سازماندهی مجدد در فایلهای ترتیبی شاخص دار:

در سازماندهی مجدد عملیات زیر انجام می شود:

۱) خواندن سریال فایل

۲) .بلاک بندی رکوردها ضمن حذف رکوردهای حذف شدنی.

۳) بازنویسی نسخه جدید فایل.

٤) بازسازی ساختار شاخص که بلاکهایش به تدریج در بافر ساخته می شوند.

$\frac{S_I}{t'}$:زمان بازنویسی بلاکهای شاخص

$$T_Y = T_{X_{ser}} + (n + o - d)\frac{R}{t'} + \frac{S_I}{t'}$$

در پایان معایب عمده ساختار ترتیبی شاخص دار به صورت زیر می باشد:

۱) عدم تقارن

۲) ایستا بودن شاخص.

۳) مسئله درج سرریزی ها.

فایل مستقیم:

این ساختار، ساختاری است جدای از ساختارهای قبلی معرفی شده برای ایجاد فایل در لود اولیه، یکی از صفات خاصه رکورد به عنوان کلید در نظر گرفته می شود.مقادیر این کلید به سیستم فایل داده می شود و سیستم پردازشی روی کلید انجام می دهد .حاصل پردازش، آدرسی است که رکورد باید در آن جای گیرد. این آدرس به آدرس طبیعی Natural Address یا حفره طبیعی Slot Natural موسوم می باشد.

در این ساختار فایل دارای یک فضای آدرسی است با m آدرس از 1تا m یا از صفر تا m-1 هر آدرس مربوط به یک حفره است و هر حفره، مکان ذخیره سازی یک رکورد می باشد .در این فضای آدرسی باید nرکورد درج شوند $m \geq n$ است .به ضریب $\frac{n}{m}$ فاکتور لود می گوییم .بنابراین پردازشی که در این نوعفایل انجام می شود تبدیل یک کلید به یک آدرس می باشد، که با استفاده از تابع مبدل Mapping Functionصورت می گیرد.

از بین این تبدیلات ممکن، فقط تعداد $\frac{m!}{(m-n)!}$ تبدیل یک به یک است، یعنی در هر حفره، فقط یک رکورد جای می گیرد و احتمال اینکه چنین نباشد زیاد است.

مثال:

می خواهیم 25رکورد را در فایلی ذخیره کنیم. کلید رکوردها نام افراد است. مکان در نظر گرفته شده برای فایل شامل 1000حفره است. یعنی n=25 و m=1000می باشد .جهت در هم سازی، کدهای اسکی دو حرف اول هر اسم را انتخاب کرده و آنها را در هم ضرب می کنیم سپس سه رقم سمت راست حالضرب را به عنوان آدرس در نظر می گیریم. مثلاً

کلید	کد اسکی دو حرف اول		حاصلضرب	آدرس خانگی
BAHRAM	66	65	66*65=4290	290
AMIN	65	77	65*77=5005	005
ROYA	82	79	82*79=6478	478

مشکل برخورد یا تصادف:

اگر دو رکورد پس از اینکه بوسیله توابع مبدل به آدرس حفره تبدیل شدند مقدار یکسانی داشته باشد در این صورت دو رکورد می بایست در یک حفره آدرسی قرار گیرند که یک برخورد یا تادف ایجاد می کند.

جهت کم کردن تصادف (برخوردها)می توان از روشهای زیر استفاده نمود:

۱) پراکنده کردن رکوردها:

توابع در هم ساز باید رکوردها را به صورت اتفاقی و تصادفی بین آدرسها توزیع کنند .بدیهی است که احتمال قرارگیری کلیدها در خانه های مشخصی بیشتر از سایر خانه ها می باشد .پراکنده کردن رکوردها این احتمال را کاهش می دهد.

۲) استفاده از حافظه اضافی:

اگر تعداد کمی رکورد را بخواهیم در بین تعداد زیادی آدرس قرار دهیم تابع در هم ساز ساده تر بوده و احتمال برخورد نیز کمتر می شود.

۳) ذخیره بیش از یک رکورد در یک آدرس (تکنیک باکت بندی):

می توان فایل را به گونه ای پیاده سازی کرد که هر آدرس آن بتواند چند رکورد را در خود ذخیره سازد.مثلاً می توان هر رکورد فیزیکی فایل را به اندازه یک سکتور 512 بایتی در نظر گرفت که هر سکتور یک آدرس یکتا دارد.

انواع توابع در هم ساز:

در یک تقسیم بندی کلی توابع در هم ساز را می توان به دو دسته تقسیم کرد:

۱) توابع دارای قطعیت(Deterministic):

که در آنها کلید $K_i \neq K_j$ باشند، آنگاه آدرسهای $a_i \neq a_j$ می باشد .به عبارت دیگر در این توابع هیچگاه برخوردی رخ نمی دهد ولی همانطور که گفتیم محاسبه این توابع بسیار پیچیده بوده و در عمل کاربردی ندارد. گاهی اوقات به چنین توزیعی، توزیع یکنواخت می گویند.

۲) توابع احتمالی(Probabilistic)که به دو بخش تقسیم می شوند:

الف) توابع منظم که در آنها اگر $K_i < K_j$ باشد آنگاه اگر برخوردی اتفاق نیفتد $a_i < a_j$ خواهد بود. این نوع توابع برای فایلهای کوچک قابل استفاده بوده و برای فایلهای بزرگ استفاده نمی شوند.

ب) توابع در هم سازکه رکوردها را به صورت تصادفی و نا منظم در فضای آدرس طبیعی پخش می کنند، در کاربردهای واقعی از این توابع استفاده می شود.

توابع در هم ساز معروف عبارتند از:

۱) مجذور کلید و گرفتن عدد میانی (Mid Square):

در این روش که به نام میانه مجذور نیز معروف است، ابتدا عدد به توان 2 رسیده سپس تعداد ارقام لازم از وسط عدد انتخاب می شود.

مثال:

فرض کنیم می خواهیم آدرسهایی بین 0تا 99 ایجاد گردد .اگر کلید، عدد 453 باشد، مجذور آن 205209 شده و دو رقم وسطی آن 52 می باشد پس کلید 453 تبدیل به 52 می شود.

$$453 \rightarrow 205209$$

مثال:اگر m=6000 و k=6793 باشد:

$$K^2 = (6793)^2 = 46144849$$

آدرس طبیعی: $1448 \times 0.6 = 868$

۲) **تبدیل مبنا(Radix Conversion):**

در این روش کلید به مبنای دیگری برده می شود تا آدرس طبیعی را تولید کند.

مثال:

اگر کلید در مبنای 10عدد 453باشد آنگاه در مبنای 11عدد 382می شود و می توان آن را به عنوان آدرس در نظر گرفت. به طور کلی روش تبدیل مبنا معتبرتر از روش میانه مجذور است ولی در عین حال تکنیک میانه مجذور برای بعضی کلیدها نتایج خوبی را می دهد.

۳) **تقسیم کردن (Division):**

در این روش کلید عددی، بر عددی تقسیم شده و باقی مانده به عنوان آدرس در نظر گرفته می شود.

مثال:

اگر m=2000و کلید برابر 48248باشد آنگاه آدرس معادل را می توان با روش زیر بدست آورد:

A=48248 mod 19937=8374

19937بزرگترین عدد اول به 20000می باشد.

٤) **تا کردن (Folding)**

در این تکنیک با انتخاب مکانهای مناسب، عددکلید را مانند کاغذ تا زده و سپس اعداد حاصله را با هم جمع می کنند.

مثال:

$$\begin{array}{r} 23 \mid 678 \mid 407 \\ \\ 404 \\ +\ 328 \\ \hline 1032 \end{array}$$

۵) **برش دادن و جمع کردن (Chopping And Adding):**

در این تکنیک کلید به بخشهایی تجزیه شده و سپس اعداد حاصل با هم جمع می شوند .پس از انجام جمع ممکن است تعدادی از ارقام سمت راست حاصل جمع انتخاب شوند و یا اینکه باقی مانده حاصل بر عددی محاسبه شود.

مثال:

اگر کلید K=37964862باشد، می توان کلید را به قسمتهای 3تایی تقسیم کرد:

آدرس=037+964+862=1863

٦) انتقال دادن(Shifting):

در یک تکنیک یکبار ارقام سمت راستی و یکبار ارقام سمت چپی را به سمت داخل انتقال داده و اعـداد بدسـت آمده را با هم جمع می کنیم.

مثال:

K=23450736

23|4507|36 => 2345 + 0736 = 3081

منابع:

[۱] روحانی رانکوهی-محمد تقی-سیستم و ساختار فایلها-۱۳۸٤

[۲] مقسمی-حمیدرضا-ذخیره و بازیابی اطلاعات-۱۳۸۵