**تعريف حافظه**: هر دستگاهي كه قادر به نگهداري اطلاعات باشد و در مواقع مورد نياز بتوان به آن اطلاعات دسترسي پيدا كرد را حافظه مي­گويند.

**خصوصيات حافظه**

1. قابليت خواندن و نوشتن
2. نشانه پذيري
3. دستيابي پذيري
4. زمان دستيابي
5. نرخ انتقال

**زمان دستيابي**: به زماني كه بين لحظه خواندن و نوشتن داده مي­شود، در لحظه­اي كه حافظه مورد نظر مورد دستيابي قرار مي­گيرد.

**نرخ انتقال**: به كميتي از اطلاعاتي كه در واحد زمان از حافظ قابل انتقال است را كه واحد آن $\frac{b}{s}$ است، گفته مي­شود.

**چگالي نوار**: تعداد بيت­هاي قابل ضبط در هز اينچ نوار را چگالي نوار مي­گويند و واحد آن bpi است.

**گپ** : فضايي است بلا استفاده بين دو گروه كاراكتر ضبط شده و اگر بين دو ركورد باشد آن را گپ بين ركوردها گويند.

وجود گپ براي متوقف كردن نوار و يا حركت دوباره آن لازم است . زيرا براي آنكه نوك خواندن و نوشتن بتواند دادهاي ذخيره شده را "حس" كند، بايد كه نوار پس از توقف، به سرعتي مطلوب و يكنواخت موسوم به سرعت حس برسد، و در اثناء اين مدت تكه­اي از نوار زير نوك R/W مي­گذرد، ضمن اينكه از لحظه شروع كاهش سرعت حس تا توقف نيز تكه­اي از نوار از زير نوك رد مي­شود . اين دو تكه نوار همان گپ را تشكيل مي­دهند و چون اين قسمت از نوار در حالت توقف- حركت با سرعت كمتر از سرعت حس طي مي­شود و در نتيجه بلااستفاده ("هرز") است. X0= v0 t IBG= v0t0

**خصوصیات یک حافظه ایده آل**

1.سرعت بسیار زیاد

2.حجم بسیار زیاد

3.قیمت بسیار پایین

**پارامترهاي نوار**

 1- ظرفيتي 2- زماني

**پارمترهاي ظرفيتي** : چگالي (D)، طول نوار (L)

\* براي بدست آوردن ظرفيت اسمي نوار از فرمول ذيل استفاده مي­كنيم:

SN = ظرفيت اسمي SN= L $×$D

مثال) حجم فايلي بر روي نوار به چگالي bpi 1600 برابر است با 800.000 بايت . طول اين فايل بر روي نوار چقدر است؟

L=→ $\frac{800,000}{1600} =$500inch

- اگر نواري با سرعت inch/s 60 و طول گپ بين بلاك inch 15/0 داشته باشيم زمان حركت –توقف چند ثانيه است؟

IBN = V0T0→ T0=→ $\frac{60}{0/15}=400t$

**پارامترهاي زماني**

1- سرعت لغزش نوار به واحد اينچ در ثانيه

2- نرخ انتقال به واحد بايت در ثانيه (يا مضاربي از بايت در ثانيه)

3- زمان حركت - توقف به واحد ميلي ثانيه . اين زمان در واقع توقف نوار و حركت وباره آن تا رسيدن با سرعت حس و زمان طي كردن گپ در حالت توقف نوار و حركت دوباره آن تا رسيدن به سرعت حس است.

مثال ) 100.000 ركورد داريم كه هر ركورد 80 بيت چگالي آن bpi1600 است ، طول فايل چقدر است؟

5000L = $\frac{8000,000}{1600}$ *= = L100*,000 = 8,000,000 SN80$×$

**طریقه ذخیره اطلاعات در دیسک**

1.دایره ای متحد المرکز

2.حلزونی

**تقسيمات ديسك**

 1- استوانه 2- شيار 3- سكتور

**استوانه** : تمام شيارها با شعاع يكسان تشكيل يك استوانه را مي­دهند.

**شيار**: محل ضبط بيت هاي اطلاعات در هر رويه، هر شيار از تعدادي سكتور تشكيل شده است.

**سكتور**: تقسيماتي است از شيار با اندازه مساوي، هر شيار از تعدادي سكتور تشكيل شده است.

**پارامترهاي ديسك:** 1- پارامترهاي ظرفيت 2- پارامترهاي زماني

**پارامترهاي ظرفيتي** : 1- اندازه سكتور 2- تعدا سكتور در شيار 3- تعداد رويه در استوانه 4- تعدا شيار در رويه

**پارامترهاي زماني**

1- زمان استوانه جويي: زماني است كه سپري مي­شود تا نوك خواندن و نوشتن به استوانه موردنظر برسد و آن را با s نمايش مي­دهيم.

2- سرعت گردش ديسك : به واحد دور در دقيقه گفته مي­شود، يعني زمان يك دور گردش بر روي ديسك كه واحد آن ميلي ثانيه است و آن را با 2r نمايش مي­دهيم.

3- زمان انتقال دوران(درنگ دوران): مدت زماني است كه سپري مي­شود تا آغاز داده مورد نظر در اثر دوران ديسك، به زير نوك R/Wبرسد. اين زمان را با r نمايش مي­دهيم.

4- نرخ انتقال : به تعداد بيتي كه در يك ثانيه قابل انتقال است نرخ انتقال مي­گويند.

T1= زمان توقف =T0 زمان بدون توقف T = t1 – t0

مثال ) در ديسكي با سرعت 6000 دور در دقيقه، متوسط زمان درنگ دوراني برابر است با؟

\*زمان درنگ دوران نصف زمان یک دور است.

V = $\frac{1}{2}×\frac{60,000}{6,000}$ = 5ms

|  |  |
| --- | --- |
| 6000 | 60000 |
| 0.5 | v |

**مفاهيم مقدماتي:**

**فيلد:** مكان ذخيره­سازي يك واحد معنايي داده و نامدار(يك فقره اطلاع) را فيلد مي­گويند.

**ركورد:**

مفهوم ركورد را از سه ديدگاه (در سه سطح) مورد بررسي قرار مي­دهيم. اين سه ديدگاه عبارتند از:

1. ركورد در سطح انتزاعي
2. ركورد در سطح برنامه كاربر
3. ركورددر محيط ذخيره سازي

**ركورد در سطح انتزاعي:** مجموعه اطلاعاتي است كه در مورد هر يك از نمونه­هاي متمايز يك يا بيش از يك نوع موجوديت(هست - شيئ) از يك خرد جهان واقع مي­خواهيم در اختيار داشته باشيم.

اطلاعات: صفات – مقدار

**مثال:**

 **یک رکورد:**

|  |  |  |
| --- | --- | --- |
| مقدار |  | صفات |
| محسن | = | نام |
| بنی اسد | = | نام خانوادگی |
| محمد رضا | = | نام پدر |

مجموعه ای از رکوردها سیستم فایل نام دارد.

نمونه متمايز: هر موجوديت در محيطي كه مشخصات آن تك باشد، مثلاً هر ركوردي يك نمونه است. مثلاً در ايران مثل كد ملي

نمونه های مختلف موجودیت ها توسط صفات خاصه آنها از یکدیگر متمایز می شوند.

**موجودیت:** هر آن چیزی که در محیط عملیاتی می خواهیم از آن اطلاعات داشته باشیم .

مانند:دانشجو،کارمند در محیط دانشگاه (محیط عملیاتی دانشگاه)

نوع موجوديت: هر چيزي كه مي­خواهيم در موردش "اطلاع" داشته باشيم.مثلاً دانشجو، درس و...

يك خرد جهان واقع: محيطي مانند دانشگاه

**مثال:**

**ركورد از ديد برنامه ساز**

مجموعه­اياست نامدار از تعدادي فيلد و داراي ساختار مشخص (مبتني بر طرح خاص).

براي نمايش ركورد، نياز به ساختار داريم و ساختار خود بايد طرح مشخصي داشته باشد. طرح ركورد نشان دهنده نوع ركورد است. دو طرح براي ساختار ركورد وجود دارد:

1- طرح با قالب ثابت مكانی

2- طرح با قالب غير ثابت مكانی

**مثال طرح با قالب ثابت مکانی:**

طول رکوردها فیکس است.

در این طرح چون طول رکوردها فیکس است،ما مجبوریم که حداکثر طول را در نظر بگیریم که حافظه هدر میرود.

در طرح ثابت مکانی جستجو راحت تر است.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| آدرس | رشته | نام خانوادگی | نام | شماره دانشجویی |
| بندرعباس | ادبیات فارسی | سلیمانیان | حمیدرضا | 8794783722 |
| ------ | ریاضی محض | هدایت فر | بهناز | 870508482 |

**مثال برای طرح با قالب غیر ثابت مکانی:**

در این قالب null value امکان پذیر است،در این قالب اطلاعات می تواند در هم وارد شود.

زمانی که اطلاعات null value زیادی داریم،این طرح کاربرد بیشتری دارد،چون حافظه کمتری اشغال می کند.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ; | رحمت ا... | = |  | نام | ; | عابد | = | نام خانوادگی | ; | 8794783722 | = | شماره دانشجویی |

این طرح انعطاف پذیرتر است و در آن طول ثابت نیست.

**ركورد ذخيره شده (در محيط ذخيره­سازي)**

در اين سطح، ركورد، علاوه بر داده­هايي كه دارد، معمولاً بخش ديگري هم داشته باشد كه به آن بخش غير داده­اي مي­گوييم:

|  |  |
| --- | --- |
| بخش داده­اي | بخش غير داده­اي |

بخش غير داده­اي حاوي اطلاعاتي است كه سيستم فايل براي پردازش ركورد به آنها نياز دارد.

**كليد ركورد**

كليد ركورد در واقع شناسه يك نوع موجوديت است و به كمك هر مقدار آن يك نمونه از موجوديت از هر نمونه ديگر متمايز مي­گردد.

**بلاك­بندي**

**تعريف بلاك­** : بلاك قالبي است با ساختار مشخص و شامل تعدادي ركورد، جاي دادن چندين ركورد در اين قالب بزرگتر را بلاك بندي گوييم.

تعداد ركوردهاي درون بلاك را ضريب بلاك­بندي مي­گوييم و با Bf) ) نشان مي­دهيم.

برخي رابطه­هاي مهم بين تعداد ركورد (n)، طول ركورد(R)، طول بلاك(B)< فاكتور بلاك­بندي (Bf) و تعداد بلاك(b) عبارتند از :

b= $\left⌈\frac{n}{B\_{f}}\right⌉$ ، b= $\left⌈\frac{nR}{B}\right⌉$ ، Bf=$\left⌊\frac{B}{R}\right⌋$

مثال) فرض مي­كنيم در فايلي 1.000.000 ركورد با طول هر يك 250بايت موجود باشد. اگر طول بلاك 2000بايت باشد، فاكتور بلاك­بندي و تعداد بلاك به صورت زير محاسبه مي­شوند:

n = 1000000

R= 250

B=2000

Bf= ?

BF= $\left⌊\frac{B}{R}\right⌋$ = $\left⌊\frac{2000}{250}\right⌋$= 8

**b =** $\left⌈\frac{n}{B\_{f}}\right⌉$ **=** $\left⌈\frac{1000000}{8}\right⌉$ **= 125000**

**b=** $\left⌈\frac{nR}{B}\right⌉=\left⌈\frac{1000000\*250}{2000}\right⌉=125000$

 IBG

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ... |  | R4 | R3 | R2 | R1 |  | ... |

**نمايش ساده شده بلاك**

بلاك كمترين مقدار داده است كه در يك عمل ورودي/خروجي توسط سيستم فايل بين بيرون و درون ماشين مبادله مي­شود و به بيان ديگر واحد عمليات خواندن/ نوشتن است.

**تكنيك­هاي تعيين محدوده ركورد در بلاك**

در اساس براي ركوردهاي با طول متغيير، سه تكنيك وجود دارد:

* درج نشانگر پايان ركورد
* در طول در بخش غير داده­اي
* ايجاد جدول مكان­نما



نشانگر پايان ركورد در انتهاي ركورد درج مي­گردد.

|  |  |
| --- | --- |
| R1$∎$ R2 $∎$ . . . . . |  |

براي هر ركورد، يك فيلد طول ايجاد مي­شود.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| . . . . | R3 | L3 | R2 | L2 | R1 | L1 |  |

آدرس نسبي پايان ركورد در بلاك نسبت به آغاز بلاك، در جدول درج مي­شود.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **.** | **.** | **.** | R1 … R2 … R3 … . . . |  |



 **جدول مكان نما**

**‹ تكنيك­هاي تعيين محدوده ركورد ›**

**معايب بلاك­بندي**

1. كار نرم افزاري بیشتر براي بلاك­بندي و بلاك­گشايي كه عكس عمل بلاك­بندي است و طي آن ركوردهاي منطقي در اختيار كاربر قرار داده مي­شود.
2. مصرف بيشتر حافظه اصلي به خاطر لزوم بافرينگ.
3. بالا رفتن احتمال اشتباه در مبادله اطلاعات به خاطر افزايش مقدار داده­اي كه منتقل مي­شود.

**مزایای بلاک بندی**

1.کاهش دفعات ورودی و خروجی و در نتیجه کاهش زمان اجرای برنامه ی فایل پرداز

2.صرفه جویی در مصرف حافظه از طریق کاهش گپ ها

**خوشه** : تعدادي بلاك همجوار ياسكتور همجوار، خوشه گفته مي­شود. تعدد بلاك خوشه را اندازه خوشه مي­نامند.

**باکت:** به مجموعه ای از بلاک ها گویند.

**گسترش**: مجموعه­ای از شيارها درون يك استوانه و يا تعدادي استوانه همجوار را گسترش مي­گويند.

**چگالي لود اوليه:**يعني چند درصد از بلاك در لحظه اول پر شود.(یعنی صد درصد بلاک ها پر نشود و مقداری برای رزرو باقی بماند)

**مزايا و معايب:**

1. ايجاد ناحيه رزرو سبب مي­شود لوكاليتي ركوردهاي فايل بهتر حفظ شود، زيرا از پراكندگي نشست ركورها روي رسانه ذخيره­سازي تا حدي جلوگيري مي­گردد.(مزایا)
2. ايجاد ناحيه رزرو، انجام بعضي عمليات روي فايل را تسهيل و تسريع مي­كند. مثلاً براي درج ركوردي كه طول آن در اثر بهنگام سازي تغيير كرده باشد و يا درج ركورد جديد وقتي كه الزام به درج آن در نقطه منطقي درج وجود داشته باشد، امكان انجام شيفت درون بلاكي را فراهم سازد.(مزایا)
3. وجود ناحيه رزرو، نوعي حافظه هرز است و سبب افزايش اندازه فايل مي­شود و از اين رهگذر، خواندن تمام فايل زمان­گيرتر مي­شود.(معایب)
4. اگر توزيع درج ركوردها در بلاك­ها يكنواخت نباشد، بلاك­ها سبكبار مي­شوند و حافظه هرز در انتهاي بعضي بلاك­ها باقي مي­ماند.(معایب)

**موضعي بودن ركوردها (لوكاليتي)**

**تعريف :** ميزان همسايگي (نزديكي) فيزيكي ركوردهاي منطقاً همجوار را لوكاليتي ركوردها مي­گويد.(هر چه همجواری بیشتر باشد می گوئیم لوکالیتی قوی تر است)

**درجات لوكاليتي:**

ميزان نزديكي فيزيكي (خوشه­وار فيزيكي) ركوردهاي منطقاً همجوار، درجاتي دارد و البته مطلوب اين است كه همجواري فيزيكي پياده­سازي شود(هنگامي كه پردازش سريال فايل مورد نظر باشد)، هرچه لوكاليتي ركوردها قويتر باشد، زمان پردازش سريال آنها كمتر خواهد شد.

**آشنايي با بافر و بافرينگ**

**بافر:** ناحیه ای است در حافظه اصلی که جهت ایجاد هماهنگی بین سرعت وسایل ورودی و خروجی(I/O) و CPU برای بالا بردن سرعت کلی سیستم استفاده می شود

.ممکن است توسط برنامه ساز (با ایجاد ماکرو) یا سیستم عامل ایجاد شود.

براي خواندن اطلاعات را از روي سي­دي، چون سرعت كم است براي جبران سرعت بايد از حافظه­اي استفاده شود و براي اين كار از بافر استفاده مي­كنيم .

**انواع بافرينگ**

* بافرينگ ساده
* بافرينگ مضاعف
* بافرينگ چند گانه

**بافرينگ ساده**

در اين نوع بافرينگ، يك بافر در اختيار برنامه فايل پرداز قرار داده مي­شود. در بافرينگ ساده طبعاً زمان انتظار واحد پردازش مركزي و اجراي افزايش مي­يابد.

**بافرينگ مضاعف**

با دو بافر، مي­توان در اثناء خواندن يك بلاك و انتقال آن به يك بافر، محتواي بافر ديگر را كه پر است پردازش كرد.

در اين پردازش فايل­ها بصورت پي­در پي و انبوه (بعني تعداد زيادي بلاك خوانده مي­شوند)، حتماً لازم است دو بافر در اختيار داشته باشيم، وگرنه عمليات نه سريع خواهد بود و نه كارا.

بديهي است زماني را كه واحد پردازش مركزي(CPU) براي پردازش محتواي يك بافر مصرف مي­كند كمتر از زماني باشد كه پردازنده ورودي/خروجي و كنترل كننده ديسك براي انتقال بلاك به يك بافر لازم دارند، يعني داشته باشيم:

 شرط کار آییCB<btt یا CB$\leq \frac{B+G}{t}$ يا CR$\leq \frac{R+W\_{R}}{T}$

پارامترها عبارتند از :

B: طول بلاك

G: طول گپ

t: نرخ انتقال رسانه

btt: زمان انتقال يك بلاك = $\frac{B}{t}$

CB: زمان براي پردازش محتواي يك بافر (يك بلاك)

CR: زمان براي پردازش يك ركورد

t': نرخ انتقال واقعي

اگر شرط زماني فوق­الذكر برقرار نباشد، بافرينگ مضاعف كارايي خود را از دست مي­دهد و نرخ انتقال واقعي كاهش مي­يابد.

**مثال از فصل:**

 1-اگر n کوچک تعداد کل رکوردها و R طول رکورد و B طول بلاک باشد.آن گاه کدام گزینه معادل تعداد بلاک در یک فایل است؟

الف)$ \left⌈\frac{nR}{B}\right⌉$ ب)$ \left⌈\frac{nB}{R}\right⌉ $ ج)$ \left⌈\frac{B}{nR}\right⌉$ د)$ \left⌈\frac{R}{nB}\right⌉$

**ارزيابي ظرفيت واقعي**

با داشتن ظرفيت اسمي، مي­توان شرفيت واقعي را ارزيابي كرد. عامل اصلي كه نقش كاهنده در ميزان استفاده واقعي از ظرفيت ذخيره­سازي نوار دارد، همان گپ بين بلاك­ها است.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | G$$\leftrightarrow $$ |  |
|  |  | Data |  |  |

بلاك و گپ بين بلاك­ها

با توجه به شكل فوق مشخص است كه به ازای B بايت داده، بايد B+G بايت از ظرفيت نوار مصرف شود. لذا درصد استفاده واقعي از نوار : 100$×\frac{B}{B+G}$ است.

ظرفيت اسمي نوار L.D است كه در آن L طول نوار به فوت و D چگالي نوار به بايت در فوت مي­باشد. پس ظرفيت واقعينوار برابر است با :

SE= $\frac{B}{B+G}$ . L . D

مثال) در نواري با طول 600 اينچ و چگالي bpi800 و با بلاك 200 بايتي و گپ بين بلاكي 50 بايت ، ظرفيت اسمي و واقعي اين نوار عبارتند از:

ظرفيت اسمي = SN = L.D = 600 $×$800 = 480,000 بايت

ظرفيت واقعي = SE = $\frac{B}{B+G}$ . L . D = $\frac{200}{200+50}×$ 600 $×$ 800 = 384,000 بايت

**خواندن بلاکها**

**1.اسلوب بلاکی:**  بعد از خواندن یک بلاک می ایستیم.

**2.اسلوب جریانی:** بعد از خواندن N بلاک مایستیم.( این خواندن زمان کمتری می برد)

**نرخ انتقال واقعي**

نرخ انتقال واقعي در ديسك به عواملي چند بستگي دارد (غير از s , r , t , G , B) عوامل مهمتر عبارتند از :

* نوع بافرينگ
* شيوه دستيابي به بلاك
* زمان پردازش بلاك cB
* طرز ذخيره­سازي بلاك­ها روي شيار
* طرز پردازش ركوردها
* طرز دستيابي برنامه به ركورد مورد نظر

**دستيابي مستقيم به بلاك**

 هرگاه سيستم آدرس بلاك حاوي ركورد را به نحوي داشته باشد، مي­تواند آنرا مستقيم بخواند، زمان خواندن مستقيم يك بلاك s + r + btt است. بنابراين مي­توان t(D)، نرخ انتقال در اين حالت را چنين محاسبه كرد:

 بايت= B زمان = s+r+btt

 t(D) 1

t(D) = $\frac{B}{s+r+b\_{tt}}$

البته مي­توان به جاي B نوشت : B-WB ، تانرخ انتقال واقعي باز هم دقيق­تر محاسبه شود:

t(D) = $\frac{B-W\_{B}}{s+r+b\_{tt}}$

**پردازش ركورد**

منظور از پردازش ركوردي اين است، هر بار كه سيستم بلاكي را مي­خواند، برنامه فايل پرداز فقط يك ركورد را پردازش مي­كند، اما در پردازش بلاكي، برنامه همه ركوردهاي بلاك را پردازش مي­كند.

اگر تعداد ركوردهاي فايل را n فرض كنيم ، در حالت پردازش ركوردي به طور تصادفي داريم:

TPfile= n(s+r+btt)+ n.cR

كه در آن cR زمان پردازش يك ركورد و TPfileزمان پردازش كل فايل است.

در حالت پردازش بلاكي مي­توان نوشت:

TPfile= b(s+r+btt)+ b.cR

**بافرينگ ساده و درهم خواني (شرط کارآیی داریم)**

 در اين حالت ، محدوديت خواندن بلاك­ها به ترتيب نشست آنها روي شيار را در نظر نمي­گيريم.

اگر btt≥cB باشد، در اين صورت بلاك­هاي شيار در دو دور خوانده مي­شوند و داريم :

Tf.B 2\*2r

X 1



X = $\frac{b\*B}{4r}$

مشخص است كه اگرشرط cB>btt باشد(شرط کارآیی نداشته باشیم) بسته به مقدارcB تعداد كمتري بلاك در يك دوره خوانده و پردازش مي­شود.

**بافرینگ ساده و مرتب خوانی (شرط کارآیی داریم)**

B 2r+btt

X 1

X= $\frac{B}{2r+btt}$

**بافرينگ مضاعف و عدم وجود شرط كارايي (مرتب خواني)**

شرط كارايي بافرينگ چنين است :

CB$\leq \frac{B+G}{t}$

پس در حالت عدم وجود شرط كارايي داريم :

CB$>\frac{B+G}{t}$

در اين وضعيت سيستم ابتدا دو بلاك را در دو بافر مي­خواند و پردازش مي­كند. با توجه به عدم وجود شرط كارايي، وقتي كه در همين دور ديسك، آغاز بلاك سوم به زير نوك خواندن/نوشتن برسد، بافر اول هنوز اشغال است و لذا سيستم نمي­تواند بلاك سوم را بخواند. بدين ترتيب براي خواندن دو بلاك بعدي، يك دور ديسك انتظار دوران لازم است (البته با فرض cB<r) و داريم :

2B 2r+2btt

X 1

X= $\frac{T\_{f}.B}{2r}$

**بافرينگ مضاعف و وجود شرط كارايي (درهم خوانی)**

شرط كارايي بافرينگ

CB$\leq btt$

|  |  |
| --- | --- |
| b \*B | 2r |
| x | 1 |

x= $\frac{b\*B}{2r}$

**تکنیک های کاهش زمان درنگ دوران(r):**

1.تداخل بلاکها

2.تغییر مکان نقطه آغاز شیارها

3.پراکنده خوانی

**تکنیک های کاهش زمان استوانه جویی(s):**

1.استفاده از دیسک های با بازوی ثابت (سخت افزاری)

2.توزیع فایل روی چند دیسک(یعنی فایل روی استوانه ها پخش شود)

3.استفاده از الگوریتم مناسب برای حرکت دادن بازوی دیسک (تکنولوژی RAID به جای استفاده از یک دیسک با ظرفیت بالا از چند دیسک باظرفیت پایین تر و به صورت آرایه استفاده می نماییم)

**نمونه سوالات فصل سوم**

1. چگالي نواري bpi1800 و طول گپ آن 6/0 اينچ مي­باشد. اگر اندازه بلاك­ها 1400 بايت باشد، درصد استفاده واقعي از نوار چند درصد است؟

$\frac{B}{B+G}$ = $\frac{1400}{1080+1400}×$ 100

1. فایلی با 10 هزار رکورد و bf=10 و چگالی لود اولیه %60 مفروض است .تعداد بلاکهای اشغال شده توسط فایل کدام است؟

n = 10000

b f=10

چگالی لود اولیه=60 درصد

$\frac{60}{100}$ = $\frac{x}{10}$

X=6

tf =b=1666.66

به بالا گرد می شود و نتیجه 1667 است .

**ضوابط ارزيابي كارايي**

اين ضوابط عبارتند از :

1. اندازه ركورد (متوسط حافظه لازم براي ذخيره­سازي يك ركورد) : R
2. زمان لازم براي واكشي يك ركورد دلخواه از فايل: TF(fetch)
3. زمان لازم براي بازيابي ركورد بعدي : TN(next)
4. زمان لازم براي بهسازي از طريق درج يك ركورد: T1(insert)
5. زمان لازم براي بهسازي از طريق ايجاد تغيير در يك ركورد: TU(update)

**1.درجا :**در فایل خودش برچسب delete می خورد و بعدا بهینه می شود.

 **2. برون از جا :** برچسب delete خورده و در فایلی دیگر نوشته می شود.( مثلا: در فایل ثبت تراکنش) و بعدا بر می گرداند و در فایل خودش بهینه سازی می کند

1. زمان لازم براي خواندن تمام فايل: TX

**1.سریال :** از اول تا آخر به ترتیب خوانده می شود.(serial)

**2. پی درپی:** همه خوانده می شود و ترتیب هم مهم نیست.(sequential)

1. زمان لازم براي سازماندهي مجدد فايل: TY

با توجه به ضوابط گفته شده، اساساً شش نوع عمل روي فايل­ها، توسط سيستم فايل انجام مي­شود:

الف- واكشي ركورد

ب- بازيابي ركورد بعدي

ج- درج ركورد جديد (بهسازي از طريق درج)

د- تغيير در يك ركورد موجود (بهسازي)

هـ : خواندن تمام فايل

و- سازماندهي مجدد

**متوسط اندازه ركورد**

در اين ارزيابي تنها در نظر گرفتن بخش داده­اي ركورد كافي نيست . زيرا به طوري كه ديده شد، در بخش پياده­سازي ركوردها در يك ساختار مشخص، فيلد يا فيلدهاي ديگري نيز وجود دارند(بخش غير داده­اي). به علاوه عوامل ديگري را هم بايد در محاسبه متوسط اندازه ركورد در نظر گرفت. به عنوان مثال بايد ديد آيا فايل متراكم است يا غير متراكم. آيا افزونگي وجود دارد؟

**فايل متراكم**: به فايلي گفته مي­شود كه تمام مقادير همه صفات خاصه تمام ركوردهایش مشخص باشند. (فايلي است كه همه مشخصات مثلاً يك فرد بصورت كامل داريم و تمام اطلاعات آن مفيد است و فضاي هرز زيادي ندارد)

**فايل غير متراكم**: فايلي است كه برخي از مقادير بعضي از صفات خاصه ، در برخي از ركوردها، موجود نباشند. (فايلي است كه مشخصات مثلاً يك فرد كامل نيست و فضاي استفاده نشده زيادي دارد و تمام اطلاعات آن مفيد نيست فقط يكسري از اطلاعات آن مفيد است، بقيه بصورت فضاي هرز است.)

**تعريف افزونگي**:

بعضي از مقادیر صفات خاصه­اش بيش از يكبار در محيط فيزيكي، ذخيره سازي شده باشد. اين تكرار ذخيره سازي يا افزونگي دو حالت دارد : افزونگي طبيعي و افزونگي تكنيكي

**افزونگي طبيعي**: صفت خاصه چنان است كه يك مقدار مشخص از آن در تعدادي از نمونه ركوردها وجود دارد. مثلاً در فايل ثبت نام دانشجويان، شماره يك درس مشخص، در ركوردها وجود دارد. مثلاً در فايل ثبت نام دانشجويان، شماره يك درس مشخص، در ركورد تمام دانشجوياني كه آن درس را اخذ كنند، ذخيره مي­شود.

**افزونگي تكنيكي** : تكرار بعضي يا تمام مقادير يك (يا چند) صفت خاصه در محيط فيزيكي ذخيره­سازي (خود فايل داده­اي يا فايل كمكي آن) به خاطر ايجاد يك شيوه دستيابي كاراتر براي فايل. مثل فهرست مطالب. اين فايل قابل حذف است و بستگي به نياز خودمان است كه بگذاريم ، يا نه.

**واكشي ركورد دلخواه**

اين عمل اساساً عملي محتوايي است، يعني ركوردي بايد واكشي شود كه مقدار يكي از صفات خاصه­اش (گاه بش از يك صفت خاصه) به عنوان نشانوند جستجو داده شده است (محتواي يك يا بيش از يك فيلد). لازمه اين عمل جستجو كردن در فايل، دستيابي به بلاك حاوي ركورد مورد نظر و خواندن آن است.

براي دستيابي به بلاك حاوي ركورد، در اساس دو دسته شيوه دستيابي وجود دارد:

**1.تصادفی**(مستقیم)

**2.ترتیبی**(غیر مستقیم)

**شيوه دستيابي**

شيوه دستيابي ترتيبي(غير مستقيم)

$$\frac{B}{t'}$$

شيوه دستيابي تصادفي (مستقيم)

S+r+btt

در اين روش بايد به طريقي از فايل­هاي قبلي مورد نظر بگذريم تا برسيم به فايل دلخواه.

در اين روش نيازي به خواندن اطلاعات قبلي فايل خواسته شده نيست، بلكه مستقيم سراغ فايل مورد نظر مي­رود.

**بازيابي ركورد بعدي**

در اين روش، جستجو بر اساس نظم ركوردها انجام مي­گيرد.

**درج ركورد**

منظور از اين عمل عمل، درج يك ركورد جديد ، بعد از لود اوليه فايل، در فايل است آنرا عمل درج مي­ناميم. اين عمل جزء عمل­هاي تغيير دهنده محيط فيزيكي ذخيره سازي است.

نكته اساسي اين است كه ركورد بايد در بلاكي وارد شود. اين بلاك بايد يافته و خوانده شود. اما اينكه اين بلاك، كدام بلاك باشد، بستگي به ساختار فايل دارد. به طور كلي دو حالت وجود دارد:

1. **درجا** : ركورد مورد نظر در فايل در جايي كه نظم ركوردها حفظ شوند، ذخيره مي­شود.
2. **برون­جا** : چون در اين قسمت نظم داريم و از طرفي جايي براي ركورد جديد، درجايي كه نظم حفظ شده، نداريم، اين ركورد در انتهاي فايل درج مي­شود.

**خواندن تمام فايل**

در اين عمل، تمام بلاك­هاي فايل بايد خوانده شود. وجود حافظه­هاي هرزي كه به هر دليل پديد مي­آيند، سبب افزايش زمان اين عمل مي­شود. خواندن تمام فايل به دو صورت ممكن است انجام شود:

* **Sequsentional**: به صورت پي­درپي، يعني بلاك به بلاك از آغاز فايل تاانتهاي آن
* **Serial**:به­صورت سريال يعني براساس نظم صعودي مقاديريكي ازصفات خاصه، معمولاًكليد.

**سازماندهي مجدد**

هر فايلي، پس از لود اوليه، دوره حياتي دارد كه در طي آن عملياتي را اعم از بازيابي يا ذخيره­سازي تحمل مي­كند. عمليات ذخيره­سازي پس از لود اوليه، تغيير دهنده محيط فيزيكي است. در اثر تغييراتي كه در فايل ايجاد مي­شود، ممكن است به تدريج فايل كارايي اوليه­اش را از دست بدهد و لذا بايد آنرا سازماندهي مجدد كرد. دلايل كاهش تدريجي كارايي فايل به طور كلي عبارتند از:

1. از بين رفتن نظم (يا وضع) ساختاري آغازين
2. بروز فضاي هرز در فايل
3. بروز وضعيت نامطلوب در استراتژي دستيابي .

با توجه به اين دلايل، مي­توان دلايل سازماندهي مجدد را به صورت زير بيان داشت:

1. احياء نظم يا وضع ساختاري آغازين
2. بازستاني فضاي هرز
3. اصلاح استراتژي دستيابي

**زمان بازنويسي بلاك**

ديديم كه در عمليات درج، بهنگام­سازي و سازماندهي مجدد، خواندن تعدادي بلاك انجام عملياتي در آنها، و بالاخره بازنويسي آنها لازم است. زمان عمل نوشتن بلاك را، كه همان انتقال محتواي بافر به ديسك است، در يك ارزيابي كلي، مي­توان مساوي زمان عمل خواندن بلاك در نظر گرفت، يعني : s+r+btt و يا در حالت نوشتن انبوه، $\frac{B}{t'}$ و اگر اين زمان را به ازاء يك ركورد بخواهيم بيان كنيم، خواهيم داشت $\frac{R}{t'}$. اما با توجه به اينكه بازنويسي يك بلاك، در اكثر عمليات، در مكان سابقش صورت مي­گيرد، تحت شرايطي مي­توان زمان آن را برابر با 2r در نظر گرفت.

**شرح ساختار مبنايي**

در اين قسمت دو روش وجود دارد:

* ساختار پايل (برهم)
* ساختار ترتيبي (دنباله­اي)

**فايل با ساختار پايل**

**معرفي ساختار**

اين فايل ساختاري دارد فاقد هرگونه نظم، يعني ركوردها بر اساس مقادير هيچ صفت خاصه­اي مرتب نيستند. در بهترين حالت، نظم بين ركوردها، نظمي است زماني، انگار ركوردها بر يكديگر ‹پشته› شده باشند. اين ساختار فاقد هر گونه استراتژي دستيابي كمكي است. نمايش منطقي اين ساختار به صورت ذيل است.



**شكل منطقي فايل پايل**

ركوردها قالب غير ثابت مكان و طول متغير دارند. تعداد صفات خاصه و نيز مكان فيلدهاي مربوط به صفات خاصه، در نمونه­هاي مختلف ركوردها، متفاوت است. ساختار ركورد در اين فايل به صورت ذيل است:

A 1 = V1 , A2 = V2 , …..

**ساختار ركورد در فايل پايل**

A1 : اسم صفت خاصه V1 : مقدار صفت خاصه

به عنوان مثال در يك فايل حاوي اطلاعاتي در مورد دانشجويان، ركورد يك دانشجو مي­تواند به صورت ذيل باشد:

... = شماره، ... = وزن، ... =قد، ... = اسم

و ركورد دانشجوي ديگر بصورت ذيل:

... = اسم ،... =قد، ... = شماره

**دو نمونه از ركورد دانشجو**

**ارزيابي كارايي**

**متوسط اندازه ركورد**

مفروضات :

* فايل در لود اوليه، n ركورد دارد.
* كل تعداد صفات در نظر گرفته شده در محيط عملياتي را a مي­ناميم.
* متوسط تعداد صفات خاصه در يك ركورد را با a' نمايش مي­دهيم (متوسط تعداد فقره اطلاع­ها در يك ركورد).
* متوسط حافظه لازم براي ذخيره­سازي اسم صفت خاصه را، A بايت در نظر مي­گيريم.
* متوسط حافظه لازم براي ذخيره­سازي مقدار صفت خاصه را، V بايت در نظر مي­گيريم.

با اين مفروضات و با توجه به قالب ركورد مي­توان نوشت:

R = a' (A + V + 2)

كه درآن يك بايت براي علامت انتساب و يك بايت براي علامت جداساز منظور شده است.

**واكشي ركورد**

نشانوند جستجو در درخواست به صورت Ai = Vداده مي­شود (يعني اسم يك صفت خاصه و مقدارش).

عمليات لازم:

خواندن بلاك حاوي ركورد مورد نظر با جستجوي خطي. اما ركورد مورد نظر ممكن است در اولين بلاك فايل باشد يا مثلاً در آخرين (و يا در هر بلاك ديگر). بنابراين به طور متوسط نصف بلاك­هاي فايل بايد خوانده و وارسي شود. اگر تعداد بلاك­هاي فايل b و اندازه بلاك B بايت باشد، زمان واكشي از اين رابطه به دست مي­آيد:

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

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

يا

چون تعدادي بلاك بايد خوانده شوند، لذا نرخ انتقال انبوه در نظر گرفته شده است.

**بازيابي ركورد بعدي**

ركورد بعدي، در اين ساختار مفهومي ندارد، زيرا هيچگونه ارتباط ساختاري بين ركورد فعلي و بعدي آن برقرار نيست. اگر كاربر خود نشانوند جستجوي ركورد بعدي را بدهد، اين عمل تبديل به واكشي تك ركورد مي­شود و داريم :

TN = TF

كه بسيار ناكار است.

**عمل درج**

عمليات لازم:

چون فايل فاقد هرگونه نظم است . لذا ركورد جديد به انتهاي قايل الحاق مي­شود و براي اين منظور كارهاي زير بايد انجام شود:

1. خواندن آخرين بلاك فايل كه سيستم آدرس آنرا دارد.
2. كار در بافر (كه زمانش را در ارزيابي دخالت نمي­دهيم): انتقال ركورد از ناحيه كاري برنامه كاربر به بلاك كه در بافر است.
3. بازنويسي بلاك

TI= s + r + TRW

كه در آن TRW زمان بازنويسي بلاك است و مي­تون آنرا با 2r برابر گرفت.

TI= s+r+ TRW

سيستم در اين عمل كارا است.

**عمل بهنگام­سازي**

در اين ساختار عمل بهنگام­سازي در حالت كلي، به صورت برون از جا انجام مي­شود.

عمليات لازم:

1. واكشي ركورد بهنگام درآمدني
2. ضبط نشانگر ‹حذف شده› در بخش پيشوندي نسخه قديم
3. ايجاد نسخه جديد
4. بازنويسي نسخه قديم (كه اينك نشانگر "حذف شده" دارد)
5. درج نسخه جديد در انتهاي فايل

پس داريم:

TU = TF+ TRW+TI

كه در آن :

TF : واكشي ركورد بهنگام درآمدني

TRW : بازنويسي همين ركورد با نشانگر ‹حذف شده›

TI : درج نسخه جديد

عمل حذف حالت خاصي است از بهنگام­سازي كه در آن درج نسخه جديد انجام نمي­شود، و زمان آن برابر است با:

TUdelete= TF+ TRW

**خواندن تمام فايل**

با سادگي مي­توان دريافت كه :

TXseq= 2TF

Seq :پي در پي

زيرا سيستم بايد تمام بلاك­ها را بخواند.

**سازماندهي مجدد**

اين فايل از نظر احياء وضع ساختاري آغازين، نياز به سازماندهي مجدد ندارد، ولي بخاطر نحوه عمل بهنگام­سازي، و همچنين در اثر عمل حذف، ممكن است جاي جاي، فضاي هرز پديد آيد، و اين حالت عملاً در تمام زمان­ها تأثير مي­گذارد. لذا بايد بطور متناوب سازماندهي مجدد شود. اگر فرض كنيم تعداد ركوردهاي درج شده در يك دوره از حيات فايل (يعني از لود اوليه تا درست قبل از سازماندهي مجدد) o باشد و d ركورد حذف شده باشد( نشانگر ‹حذف شده› خورده است) زمان سازماندهي مجدد چنين خواهد بود:

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

كه در آن:

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

(n + o -d )$\frac{R}{t'}$ : زمان بازنويسي كل فايل

توجه داريم كه در ارزيابي زمان بازنويسي كل فايل، زمان لازم براي نوشتن يك ركورد را $\frac{R}{t'}$ در نظر مي­گيريم، يعني همان زمان انتقال يك ركورد در خواندن انبوه فايل.

**فايل با ساختار ترتيبي**

**معرفي ساختار**

فايل ترتيبي بر دو نوع است:

* فايل ترتيبي كليدي
* فايل ترتيبي زماني

فايل ترتيبي زماني : ركوردها به ترتيب ورود به سيستم ذخيره مي­شوند و نوع خاصي از فايل پايل است كه در آن ركوردها معمولاً قالب ثابت مكان دارند.

اين فايل نسبت به فايل پايل دو بهبود ساختاري دارد:

1. در لود اوليه، تمام نمونه ركوردها بر اساس مقادير يكي از صفات خاصه منظم هستند و اين نظم با همجواري فيزيكي ركوردها پياده سازي مي­شود.
2. ساختار ركوردها، طرح ثابت مكان دارند و به همين خاطر، فضاي هرز زيادي داريم ولي در ذخيره سازي فايل نيازي نيست كه اسم صفت خاصه هر بار ذخيره شود و همان مقدار كافيست.

|  |  |  |  |
| --- | --- | --- | --- |
| Vn | . . . . . . . . .  | V2 | V1 |

ساختار ركورد در فايل ترتيبي

اين دو بهبود ساختاري مزايا و معايبي (نسبت به فايل پايل) در بر دارد.

**مزايا:**

1. صرفه­جويي در مصرف حافظه به خاطر عدم ذخيره­سازي اسم صفت خاصه در نمونه ركوردها.
2. ساده­تر بودن قالب ركورد، به نحوي كه ركورد ذخيره شده عملاً نگاشتي است از آنچه كه در برنامه پردازشگر اعلان مي­شود.
3. نرم­افزار ساده­تر براي ايجاد، مديريت و پردازش فايل
4. وجود يك استراتژي دستيابي: چون منظم است از جستجو ي دودويي مي­توان استفاد كرد.
5. پردازش سريال ركوردها تسريع و تسهيل مي­شود، به ويژه اگر فايل كاملاً ترتيبي باشد.

**معايب**

1. مصرف حافظه بيشتر به خاطر در نظر گرفتن فيلد براي اطلاع نهست(قالب ثابت مكان).
2. وجود پديده عدم تقارن
3. كاهش انعطاف­پذيري

تمام عمليات تغيير دهنده، اصطلاحاً موسوم به تراكنش، در فايل جداگانه­اي به نام فايل ثبت تراكنش­ها (T.L.F ) انجام مي­شود.

T.L.F : بدين معناست كه مي­خواهيم تراكنشي ايجاد نمائيم ولي چون جا نداريم آنرا آخر قرار مي­دهيم و بعد در سازماندهي در جاي مناسب قرار مي­دهيم.

**ارزيابي كارايي**

**متوسط اندازه ركورد**

با توجه به ساختار ركورد مي­توان نوشت:

R = a . v

و داريم :

Sfile = n . a . v

عمليات در اين فايل پس از لود اوليه، در عمل در فايل ثبت تراكنش­ها انجام مي­شود، لذا براي ارزيابي دقيق­تر در لود اوليه بايد ظرفيت آن فايل را نيز دخالت داد:

Sfile = (n + o) . a . v

فرض كرده­ايم كه فايل تراكنش­ها ظرفيت o ركورد را دارد.

**واكشي ركورد**

دو حالت در نظر مي­گيريم:

الف) نشانوند جستجو غير از صفت خاصه نظم باشد( يعني صفت جستجو غير از نظم)

ب) نشانوند جستجوي همان صفت خاصه نظم (كليد)باشد.

حالت الف: در اين حالت؛ فايل عملاً تبديل مي­شود به حالت خاصي از پايل و با جستجوي خطي خواهيم داشت:

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

و اگر فرض كنيم ظرفيت T.L.F به تعداد o ركورد باشد و در لحظه واكشي،o' ركورد در آن وجود داشته باشد (o'$\leq $o و o' به تدريج به سمت o ميل مي­كند) ارزيابي دقيق­تر چنين خواهد بود:

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

مي­بينيم كه در اين حالت ، زمان بالاست و فايل كارايي لازم را ندارد.

**جستجوي دودويي**

جستجوي دودويي، در يك محيط منظم خارجي بايد در دو سطح اانجام گيرد. در سطح اول، جستجوي در فايل داريم تا بلاك مورد نظر پيدا شود(واحد جستجو در اين سطح، بلاك است). براي اين كار طبعاً بايد بلاك­ها خوانده شوند. در سطح دوم براي هر بلاك كه به بافر آورده مي­شود، يك جستجوي دودويي درون بلاكي داريم. اين هر دو جستجو در ارزيابي زمان دخالت دارند.

TFbinery search = logb2 (s + r + btt + cB)+ TF o

و

TFo= $\frac{1}{2}$ o' $\frac{R}{t'}$

يادآوري مي­شود كه T.L.F داراي نظم زماني است . ممكن است ركورد در فايل اصلي نباشد، يعني ركورد، درجي باشد، لذا دخالت دادن TFo لازم است.

**بازيابي ركورد بعدي**

در اين ساختار با خواندن هر بلاك، Bf ركورد به دست مي­آيد كه هر يك بعدي ركورد قبلي است.

بنابراين مي­توان نوشت:

TN = $\frac{{B}/{t'}}{B\_{f}}$ = $\frac{R}{t'}$

**عمل درج**

ركورد درج شدني را نمي­توان به انتهاي فايل اصلي الحاق كرد. زيرا بايد در نقطه خاصي درج شود و اين كار در فايل­هاي بزرگ زمانگير است.

براي ارزيابي زمان درج، دو حالت را در نظر مي­گيريم:

* **حالت اول : درج در فايل كوچك**

مي­توان در فايل كوچك، نقطه درج را يافت و ركورد را درآن نقطه درج كرد. در اين صورت بقيه ركوردها بايد به سمت پايان فايل (EOF) شيفت داده شوند. چون معلوم نيست كه نقطه درج كجاست، دقيقاً مشخص نيست چه تعداد بلاك بايد شيفت داده شود. به طور متوسط، نصف بلاك­هاي فايل شيفت داده مي­شوند. پس عمليات لازم در اين حالت عبارتست از:

1. يافتن نقطه منطقي درج
2. درج ركوردها در بلاك مورد نظر (كار در بافر)
3. شيفت دادن بلاك­ها از نقطه منطقي درج به سمت EOF .

و خواهيم داشت:

TI = TF +$ \frac{1}{2}$ b( $\frac{B}{t'}$ +TRW)+2r

TF: يافتن نقطه منطقي درج

$\frac{B}{t'}$ + TRW*: زمان شيفت يك بلاك*

* ***حالت دوم : درج در حالت كلي***

*ركورد درج­شدني در آخرين بلاك فايل ثبت تراكنش­ها درج مي­شود و زمان اين عمل همان زمان درج در پايان است و به علاوه زماني ديگر نيز بايد دخالت داده شود كه* $\frac{T\_{Y}}{o}$ *است.*

TI = s+ 3r+btt+$\frac{T\_{Y}}{O}$

**عمل بهنگام­سازي**

در ركورد كوچك، تاريخ بهنگام­سازي و نشانگر ‹حذف شده› وجود دارد. معنايش اين است كه نسخه قديمي ركورد، در فايل اصلي، در سازماندهي مجدد، حذف شدني است. طبعاً واكشي­هاي بعدي ركورد بهنگام درآمده بايد از فايل تراكنش­ها صورت گيرد و زمان در يك برآورد كلي برابر است با:

TU$≈$TF + TI +2r

TI: زمان درج

TF: زمان واكشي ركورد بهنگام درآمدني

**عمل delete:**

در این عمل رکورد حذف می شود.

TUo$≈$TF +2r

**خواندن تمام فايل**

1. خواندن پي در پي به روش معمول انجام مي­شود و زمان خواندن به طور پي در پي برابر است با :

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

فرمول ساده تر شده:

TXSeq= $\frac{nR}{t'}=\frac{nR}{t'}$

1. براي خواندن سريال(براساس صفت منظم)، فايل تراكنش­ها بايد مرتب شود. زمان چنين برآورد مي­شود:

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

**سازماندهي مجدد**

براي سازماندهي مجدد اين فايل، بايد عمليات ذيل را انجام داد:

1. مرتب كردن فايل ثبت تراكنش (تا با فايل اصلي همتوالي شود)
2. خواندن فايل اصلي
3. خواندن فايل تراكنش
4. بلاك­بندي مجدد ركوردها (ادغام آنها طبق نظم) ضمن خارج كردن ركوردهاي ‹حذف شدني›
5. بازنويسي كلي فايل

زمان برابر است با:

TY =TSort(o)+ (n + o ). $\frac{R}{t'} $+(n + o – d ) $\frac{R}{t'}$

فرمول ساده تر شده:

TY =$\frac{nR}{t'}$+(n – d ) $\frac{R}{t'}$

**ویژگی ها و کاربردهای فایل ترتیبی:**

1.برای فایل های static ای (درج و حذف در آن کم است) که مرتب بودن در آنها مهم است ،مفید می باشد. مثل: راهنمای تلفن

2.هنگامیکه رکوردهای مجزا و زیادی واکشی می شوند،ساختار ترتیبی چندان مناسب نیست.

(ساختارهای شاخص دار برای این کار بسیار مناسب تر است)

3.در اینگونه ساختارها،لیست گیری به ترتیب رکوردها و یافتن رکوردهای بعدی سریع است.اما در صورتی که درج زیادی نیاز داشته باشیم عملا این ساختار به سمت pile میل می کند و کارآیی خودش را از دست می دهد.

**نمونه سوالات فصل چهارم**

1.در یک فایل ترتیبی اگر طول بلاگ 2100 بایت و نرخ انتقال 30 b/s و تعداد دور در دقیقه 2000 باشد.زمان شیفت بلاگ را بدست بیاورید.

 btt +2r=$\frac{b}{t'}$+2r

|  |  |
| --- | --- |
| 2000 | 60 |
| 1 | v |

$\frac{2100}{30}$+30\*10-3=70.03

**2.**در یک فایل ترتیبی در صورتی که ضریب بلاک بندی bf=2 و تعداد رکوردها مساوی 16 باشد.حداکثر تعداد دفعات مراجعه به فایل در عمل جستجو را به دست آورید.

Bf=2 , n=16

 8 = b =$ \frac{n}{Bf}$

 Log2 b=3

3. با استفاده اطلاعات داده شده، فرمول­هاي فايل ترتيبي(همه كارها بر اساس نظم) را بدست آوريد.

t' = 1 r =$ \frac{1}{2}$ a = 15 s = 1 V = 10 btt=10 R=1

B= 10 d=1 n= 1000 o'=10 Tsort=20

اندازه ركورد R= a.v$\rightarrow $15 $×$10 = 150

صفت جستجو، صفت نظمTFbinery search= logb2 (s + r + btt + cB)+ TF o$\rightarrow $'log1002(11/5) + 5 = 80/5

درج درجا TI = TF+ $\frac{1}{2}$ + b (btt + TRW) $\rightarrow $80.5 + 50(11) = 630.5

درج برون جا TI = s +3r +btt$ \rightarrow 12.5$

بهنگام سازي (تغيير) TU $≈$ TF + TI$ \rightarrow $ برون از جا))80.5+1+12.5=94 $\rightarrow $ (درجا)80.5+1+630.5=712

بهنگام سازي (حذف) TU $≈$ TF + TRW$ \rightarrow $80.5 +1=81.5

سازماندهي مجددTY =$\frac{nR}{t'}$+ TSORT+(n+o – d )$\frac{R}{t'} \rightarrow $ 1010 +20+ 1009 = 2039

**زمان خواندن فایل پی در پی** (n +0)$ \frac{R}{t^{'}} $=1010

**زمان خواندن فایل سریال** (n +0)$ \frac{R}{t^{'}} $+ TSORT =1030

Tf1= logb2 (s + r + btt )=log2 100(11.5)=80.5

Tf2=$\frac{1}{2}\*b\*$btt=$\frac{1}{2}\*100\*10=500$

**شاخص**

شاخص يا فايل ساختارهاي شاخص مجموعه­اي است از تعدادي مدخل(Entry) هريك به صورت زير:



|  |  |
| --- | --- |
| آدرس | مقدار |

در حالت كلي، در فيلد مقدار، مقدار هر صفت خاصه­اي (ساده و مركب) مي­تواند قرار گيرد و در حالت خاص مقدار كليد اصلي در اين فيلد گذاشته مي­شود.

**لنگرگاه (نقطه اتكاء)**

نقطه­اي كه از فايل داده­اي كه مدخل شاخص به آن نشانه مي­رود. لنگرگاه يا نقطه اتكاء شاخص ناميده مي­شود.

اگر لنگرگاه، ركورد باشد، شاخص را متراكم و اگر لنگرگاه، گروهي از ركوردها(معمولاً يك بلاك) باشد، شاخص را نا متراكم مي­گويند.

|  |  |  |  |
| --- | --- | --- | --- |
| معدل | نام پدر | نام | شماره دانشجویی |
| 17 | عبدا.. | مصطفی | 3925 |
| 18 |  علی  | مرتضی | 4012 |
| 16 | رضا | حسین | 4076 |

**شاخص متراکم:**

|  |
| --- |
| ش .د |
| 3925 |
| 4012 |
| 4076 |

**شاخص نامتراکم:**

|  |
| --- |
| 1 |
| 32 |
| 40 |

|  |  |
| --- | --- |
| 1 |  |
| 4 |
| 9 |
| 32 |  |
| 35 |
| 37 |
| 40 |  |
| 45 |

از شاخص برای واکشی سریع تک رکورد استفاده می شود و در عملیاتی مثل خواندن تمام فایل (به صورت پی در پی)کاربردی ندارد.

**ظرفیت نشانه روی:** تعداد مدخل های شاخص در یک بلاک (همان Bf در بلوک)



|  |  |
| --- | --- |
| آدرس | مقدار |

 y = $\left⌊\frac{B}{V+P}\right⌋$

- مثالي از طراحي شاخص نرم­افزاري

فايلي يا مشخصات ذيل را در نظر بگيريد. مطلوبست طراحي شاخص غيرمتراكم براي اين فايل.

P=6بايت V= 14بايت B= 2000بايت R= 200 بايت n = 106

حل: بر مي­نهيم: ei : تعداد مدخل­هاي شاخص در سطح i ام .

Bi : تعداد بلاك­هاي شاخص در سطح i ام.

SI i : حافظه لازم براي سطح iام شاخص.

داريم:

ei = bi-1

bi = $\left⌈\frac{e\_{i}}{y}\right⌉$

Si = B.bi

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

V + P = 14 + 6 = 20 طول يك مدخل شاخص

Y = $\left⌊\frac{2000}{20}\right⌋$ = 100

تعداد بلاکها برای فایل اصلی و تعداد شاخص های سطح اول = $\frac{10^{6}\*200}{2000}$ = 105

فضای کل برای شاخص سطح اول = 20\*105=2\*106 بایت

تعداد بلاکها برای شاخص سطح اول و تعداد شاخصهای سطح دوم = $\left⌈\frac{2\*10^{6}}{2\*10^{3}}\right⌉$ = 1000

فضای کل برای شاخص سطح دوم = 20\*1000 = 20,000 بايت

تعداد بلاکها برای شاخص سطح دوم و تعداد شاخص های سطح سوم = $\left⌈\frac{20000}{2000}\right⌉$ = 10

فضای کل برای شاخص سطح سوم = 10$×$20 = 200بايت

تعداد بلاکها برای شاخص سطح سوم = $\left⌈\frac{200}{2000}\right⌉$=1

بنابراين ساختارشاخص در سه سطح ايجاد مي­شود :

x = 3

 فضای کل برای همه شاخصها$S\_{I=}\sum\_{i=1}^{x}S\_{i}=2020000$بايت

 ميزان حافظه لازم (ديسك) براي شاخص است.SI

**درخت جستجوی دودویی:(Bst)**

یک درخت دودویی است که ممکن است تهی باشد،اگر تهی نباشد دارای خاصیت زیر است:

مقدار هر گره، از هر مقدار زیر گره چپ بزرگتر و از هر مقدار زیر درخت راست کوچکتر است.

مثال: 8

 9 8,1,9,5,12,4,18,3

 12 1

 **18 5**

 **4**

 **1.حذف گره بدون فرزند:**به سادگی حذف می شود و هیچ جایگزینی ندارد. 3

**حذف: 2.با یک فرزند:** فرزند جایگزین پدر می شود.

 **3.با دو فرزند:**یا در درخت چپ بزرگترین عدد را جایگزین کرده و یا از سمت راست کوچکترین را

جایگزین پدر می نماییم**.**

**نحوه پياده سازي فايل با استفاده از B - TREE**

هر بلاك شاخص (يا تعدادي بلاك با نامي ديگر) گره اي از درخت در نظر گرفته مي­شود. يك گره، در لود اوليه به تمامي پر نمي­شود،، به عبارت ديگر چگالي آن در لود اوليه كمتر ازصد در صد است. اگر طول هر مدخل، v+p بايت باشد، y اسمي برابر است با : $\left⌊\frac{B}{V+P}\right⌋$ = y

مثال:1,3,5,4,10 B -Tree

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | B | . | A  | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 5 | . | 3 | . |

 B – tree

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . |  | . | 1 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 3 | . | 1 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . |  | . | 3 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . |  | . | 4 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 10 | . | 7 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 5 | . | 4 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . |  | . | 1 | . |

مثال برای مساوی: 1,3,5,4,10 B -Tree

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | B | . | A  | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 5 | . | 3 | . |

 B – tree

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 3 | . | 1 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 3 | . | 1 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . |  | . | 3 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 5 | . | 4 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 10 | . | 7 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 5 | . | 4 | . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| . | 3 | . | 1 | . |

B + -Tree

یکی از بزرگترین عیبهای B –Tree مشکل بودن پیمایش ترتیبی کلیدهاست . B+ -Treeشکل دیگری از

 B –Treeاست که تمام کلیدهای نگهداری شده در برگها به یکدیگر متصل می شوند تا یک مسیر سریالی برای پیمایش کلیدها در درخت را فراهم سازند.

مثال:در B –Tree قبلی B+ -Tree را مشخص کنید؟

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| . | 10 | . | 7 | . |  | . | 5 | . | 4 | . |  | . | 3 | . | 1 | . |

**فايل مستقيم مبنايي**

ساختار مستقيم، شكل بهبود يافته ساختارهاي قبلي نيست. خو ساختار جدايي است كه در آن درج و واكشي ركوردها با استراتژي خاصي صورت مي­گيرد. براي ايجاد فايل در لود اوليه، يكي از صفات خاصه ركورد به عنوان كليد در نظر گرفته مي­شود. براي ايجاد فايل در لود اوليه، يكي از صفات خاصه ركورد به عنوان كليد در نظر گرفته مي­شود.

مقادير اين كليد به سيستم فايل داده مي­شود و سيستم "پردازشي" روي كليد انجام مي­دهد. حاصل پردازش، آدرسي است كه ركورد بايد در آن جاي گيرد.

اين آدرس به آدرس طبيعي يا حفره طبيعي ركورد موسوم است. در واكشي، مجدداً مقدار كليد ركورد مورد نظر به سيستم داده مي­شود و سيستم همان پردازش را انجام مي­دهد و آدرس مكان نشست ركورد را پيدا كرده، آن را واكشي مي­كند، بدين ترتيب استراتژي دستيابي مستقيم به ركوردها، تأمين مي­شود.

در اين ساختار، فايل داراي يك فضاي آدرسي است با m آدرس از 1 تا mيا از صفر تا m-1 . هر آدرس مربوط است به يك حفره، مكان ذخيره­سازي يك ركورد است. در اين فضاي آدرسي بايد n ركورد لود (درج) شوند وnm$\geq $ است. به $\frac{n}{m}$ فاكتور لود مي­گوييم. بنابراين پردازشي كه بايد روي مقادير كليد انجام شود، ماهيتاً يك تبديل كليد به آدرس است، كه معمولاً به كمك يك تابع مبدل انجام مي­شود كه به اين تابع در همساز (يا تابع جاياب) و نيز تابع نگاشتگر مي­گوييم.



**فضاي آدرسي و تبديل كليد به آدرس**

**به طور کلی روشهای دستیابی به دو دسته تقسیم می شوند.**

— ترتیبی

 —تصادفی

**جهت دستیابی تصادفی دو تکنیک مشخص وجود دارد:**

 — شاخص(index)

 — درهم سازی(hashing):کلید آدرس (با یک پردازش)

منظور از در هم سازی این است که کلید به آدرس تبدیل شود.

مثال:

|  |  |  |  |
| --- | --- | --- | --- |
| آدرس | حاصلضرب | کد اسکی دو حرف اول | کلید |
| 290 | 4290 |  65 66  | Bahram |
| 005 | 5005 |  77 65 | amin |
| 478 | 6478 |  79 82 | roeya |

 **— مشکل برخورد(collision) تصادف ،تصادم ، برخورد**

در حالت کلی تصادف زمانی رخ می دهد که کلیدهای مختلف آدرس یکسانی را تولید کنند.

**— برای کم کردن تعداد برخوردها از روش های زیر استفاده می شود:**

1.استفاده از تابع درهم ساز بهتر

2.استفاده از حافظه ی اضافی

3.ذخیره بیش از یک رکورد در یک آدرس (تکنیک باکت بندی)

**انواع تابع درهم ساز**

توابع مختلفي براي ايجاد فايل مستقيم وجود دارد كه بروز تصادف در همه آنها كم و بيش محتمل است. برخي از اين توابع عبارتند از :

الف : انتخاب ارقام مياني مربع كليد

ب : تقسيم كردن

ج : شيفت دادن

د: تبديل مبنا

 اينك بطور اجمال روش­هاي نامبرده را به كمك مثال­هايي توضيح مي­دهيم.

**الف –انتخاب ارقام مياني مربع كليد**: در اين روش بخش عددي كليد را به توان دو رسانده، ارقام مياني عدد حاصله را انتخاب و با توجه به m تنظيم مي­شود.

مثال : 7000m= و 172148 ki=

029634933904(ki)2 =

چهار رقم مياني را انتخاب مي­كنيم و در حالت كلي نياز به تنظيم دارد، پس در ضريب 7/0 ضرب مي­كنيم، حاصل قطعاً از 7000 كوچكتر خواهد بود.

ai = 7/0\*3493 = 2445

**ب\_ تقسيم**: در اين روش بخش عددي كليد را بر عددي تقسيم مي­كنند. مقسوم عليه، نزديكترين عدد اول به m است (و بهتر است خود m عدد اول باشد). باقيمانده تقسيم آدرس ركورد است.

4220= ($\frac{172148}{6997}$) باقيمانده = آدرس

4220 = آدرس ai

توجه داريم كه باقيمانده به تنظيم ندارد.

**ج –شيفت دادن :**در اين روش، ارقام طرفيني كليد را به داخل شيفت مي­دهند، اعداد حاصله را با هم جمع و نتيجه را در طيف آدرس­ها، تنظيم مي­كنند.

59$|$2073$|$17

 + 2017

5973

7000 ai = (ضريب مناسب) \*7990

**د –تبديل مبنا**: در اين روش، كليد كه خود در مبنايي است، به مبناي ديگر برده مي­شود و با توجه به طيف فضاي آدرسي، عمل تنظيم هم صورت مي­گيرد.

 (11)172148ki =

 7/0 \* 6373 = ai 6373/26 = 11 \* 8+ ...+114 \* 7 + 115 \* 1

در اين مثال، فرض مي­كنيم كه مقدار كليد در مبناي 11 هست و آن را به مبناي 10 مي­بريم.

**ضوابط انتخاب تابع درهم ساز**

براي انتخاب تابع بهتر، ضوابطي وجود دارد. هرچند در يك سيستم فايل مشخص، پردازشگر فايل معمولاً تابع را انتخاب نمي­كند، بلكه سيستم ايجاد كننده فايل بر اساس تابع تعيين شده­اي، پيرو درخواست كاربر، فايل را ايجاد مي­كند. ولي مواردي هست كه در آنها خود پردازشگر فايل بايد تابع را معرفي كند و در اين صورت بايد بتواند تابع بهتر را انتخاب را انتخاب نمايد. به علاوه طراح سيستم فايل كه نرم­افزار سيستم فايل مستقيم را طراحي و تهيه مي­كند. بايد از ضوابط انتخاب تابع آگاهي داشته باشد تا بخواهد مناسترين تابع را برگزيند. ضوابط انتخاب تابع بهتر عبارتند از:

1. تابع بايد چنان باشد كه بتوان آن را روي تمام اجزاء كليد (تمام ارقام كليد) اعمال كرد.
2. توزيع يكنواخت­تر ركوردها
3. كمتر بودن تعداد تصادفي­ها و در نتيجه كمتر بودن متوسط تعداد عمليات ورودي/ خروجي لازم براي واكشي يك ركورد دلخواه، ما اين ضوابط را در مثالي نشان مي­دهيم.

مثال: فايلي داريم با ده ركورد، پس از اعمال دو تابع h1 و h2 آدرس­هاي زير بدست آمده­اند:

2 ، 1 ، 8 ، 7 6 ، 5 ، 4 ، 3 ، 2 ، 1 = h1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |

1 ، 1 ، 8 ، 7 6 ، 5 ، 4 ، 3 ، 2 ، 1 = h2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |

فرض مي­كنيم كه هر دو تابع، روي تمام ارقام كليد اعمال شده­اند و لذا ضابطه اول براي هر دو وجود دارد.

 در تابع h1 ، ركورد نهم با ركورد يكم و ركورد دهم با ركورد دوم تصادم كرده است و در تابع h2 ، ركورد نهم با ركورد يكم و ركورد دهم با ركورد يكم.

مي­بينيم كه توزيع ركوردها در حفره­ها، در هر حالت يكسان است و تعداد تصادفي­ها نيز مساوي، پس بايد متوسط تعداد عمليات ورودي/ خروجي لازم براي واكشي يك ركورد را بدست آورد. فرض مي­كنيم ركوردهاي تصادفي هر آدرس بهم زنجير شده باشند.

با تابع h1: 2/1 = $\frac{12}{10}$ و 12 = 2 \* 1 + 2 \* 1 + 1 \* 8

با تابع h2 : /1 = $\frac{13}{10}$ و 13 = 3 \* 1 + 2 \* 1 + 1 \* 8

پس تابع h1 از تابع h2 مناسبتر است.

**راه­حل­هاي مشكل تصادف**

1. ايجاد يك فايل جداگانه و درج تصادفي­ها در اين فايل
2. درنظر گرفتن ناحيه­اي جداگانه در خود فايل و درج تصادفي­ها در اين ناحيه
3. كاوش خطي و درج تصادفي در اولين باكت جادار
4. احتمالاتي كردن مجدد و استفاده از تابعي ديگر براي توليد آدرس
5. ايجاد زنجيره