### بسمه تعالي

### ذخيره و بازيابي اطلاعات

### Information storage and retrieval

### حافظ قرآنی

### C:\Users\Ali\Desktop\Ali.jpg

**فهرست مطالب :**

[مقدمه: 4](#_Toc313127151)

[فصل اول: 6](#_Toc313127152)

[مهندسي فايل 6](#_Toc313127153)

[خصوصيات حافظه در معناي عام: 6](#_Toc313127154)

[سلسه مراتب حافظه: 7](#_Toc313127155)

[نوار مغناطيسي: 9](#_Toc313127156)

[تقسيمات نوار 10](#_Toc313127157)

[نحوه ي نشست فايل بر روي نوار: 11](#_Toc313127158)

[پارامتر هاي نوار: 12](#_Toc313127159)

[انواع ديسک ها 13](#_Toc313127160)

[تقسيم بندي ديسک ها: 14](#_Toc313127161)

[مؤلفه هاي آدرس دهي 16](#_Toc313127162)

[پارامتر هاي ديسک: 17](#_Toc313127163)

[فصل دوم : 19](#_Toc313127164)

[مفاهيم مبنايي فايلينگ 19](#_Toc313127165)

[لوکاليتي(locality) : 23](#_Toc313127166)

[Blocking, Deblocking 24](#_Toc313127167)

[روش هاي تشخيص محدوده بلاک: 25](#_Toc313127168)

[روش هاي بلاک بندي: 26](#_Toc313127169)

[مقايسه روش هاي بلاک بندي: 29](#_Toc313127170)

[مزايا و معايب بلاک بندي : 30](#_Toc313127171)

[چگالي لود اوليه: 30](#_Toc313127172)

[درجات Locality: 33](#_Toc313127173)

[سطوح و مکانيزم آدرس دهي به فايل: 33](#_Toc313127174)

[راهنماي فايل ( فهرست فايل ) 37](#_Toc313127175)

[بافر و بافرينگ 38](#_Toc313127176)

[روش هاي بافرينگ 41](#_Toc313127177)

[فصل سوم: 44](#_Toc313127178)

[ارزيابي پارامتريک رسانه ها 44](#_Toc313127179)

[ظرفيت واقعي نوار : 45](#_Toc313127180)

[نرخ انتقال واقعي 46](#_Toc313127181)

[ارزيابي پارامتريک ديسک : 48](#_Toc313127182)

[محاسبه ي درصد استفاده در شيار: 49](#_Toc313127183)

[ارزيابي نرخ انتقال واقعي در ديسک: 49](#_Toc313127184)

[تکنيک هاي کاهش زمان درنگ دوراني: 53](#_Toc313127185)

[تکنيک هاي کاهش زمان استوانه جويي(Seek Time) 55](#_Toc313127186)

[فصل چهارم: 56](#_Toc313127187)

[بررسي تحليلي و پارامتريک مبنايي ساختارهاي فايلينگ : 56](#_Toc313127188)

[ساختارهاي مبنايي فايلينگ 56](#_Toc313127189)

[حافظه ي مصرف شده به ازاي هر رکورد: 58](#_Toc313127190)

[زمان ها 61](#_Toc313127191)

[ارزيابي پارامترها در فايل با ساختارهاي مختلف 68](#_Toc313127192)

[فايل با ساختار پايل : 68](#_Toc313127193)

[ارزيابي ساختار پايل : 71](#_Toc313127194)

[ارزيابي فايل با ساختار ترتيبي(کليدي) 76](#_Toc313127195)

[ارزيابي فايل با ساختار ترتيبي: 79](#_Toc313127196)

مقدمه:

همان طور که از اسم درس پيداست ، درس ذخيره و بازيابي اطلاعات به مباحث ذخيره و بازيابي اطلاعات مي پردازد.

به طور کلي هر گاه بحث از ذخيره و بازيابي اطلاعات مطرح شد ، بحث نوشتن درون فايل ها هم مطرح مي شود.

1. . شناخت و معرفي رسانه هاي ذخيره سازي اطلاعات(نوار و ديسک)
2. . مفاهيم اوليه ي Filing
3. . ارزيابي پارامتريک رسانه ها(نوار و ديسک)

معيار هاي ارزيابي رسانه ها : 1.حجم يا ظرفيت 2.سرعت يا نرخ

1. . ارزيابي ساختار ها(ساختار:شکل و شمايل)

ساختار هاي زير مباني مي باشند، اما مي توان از ترکيب آنها نيز ساختار جديدي نيز ساخت.

1. **پايل**: يعني همين که Data اومد آن را بنويسيم.
2. **ترتيبي**: يعني به يه ترتيبي(اولويتي) فايل ها را مي نويسيم.
3. ترتيبي شاخص دار
4. فايل چند شاخصي
5. فايل مستقيم
6. فايل حلقوي

**معيار هاي ارزيابي ساختار ها:**

1. زمان خواندن يک رکورد(Fetch)()
2. به دست آوردن رکورد بعدي(Get Next)()
3. زمان اضافه کردن يک رکورد جديد(Insert)()
4. زمان به روز رساني يک رکورد(Update)()
5. خواندن کل فايل(سريال و مرتب)()
6. سازمان دهي مجدد ()
7. پارمتر حجم

#### فصل اول:

# مهندسي فايل

تمرين1 :" مهندس کيست؟مهندس چيست؟"

حافظه: به هر دستگاهي که بتوان اطلاعات را در آن ذخيره کرد، به طوري که استفاده کننده از آن بتواند در هر لحظه که لازم داشت به اطلاعات مورد نيازش، دسترسي(Access) داشته باشد.

در بررسي رسانه هاي ذخيره سازي خارجي، دو رسانه ي رايج تر ، يعني نوار مغناطيسي و ديسک مغناطيسي را بررسي مي کنيم.

# خصوصيات حافظه در معناي عام:

حافظه ها براي خود ويژگي هايي دارند، اما خصوصياتي بين آنها مشترک مي باشد. از جمله:

خواندن و نوشتن(Read/Write): هر حافظه اي اين قابليت را دارد که بتوانيم روي آن بنويسيم(Insert يا درج کردن) يا از روي آن بخوانيم(Fetch کردن).

آدرس پذيري(Addressability) :هر حافظه اي مجهز به يک مکانيسم آدرس دهي مي باشد. به عبارتي ديگر مي توان به اطلاعات مورد نظر در حافظه ، آدرسي را نسب داد.(البته واحد آدرس پذير و نحوه ي آدرس دهي به نوع حافظه بستگي دارد.)

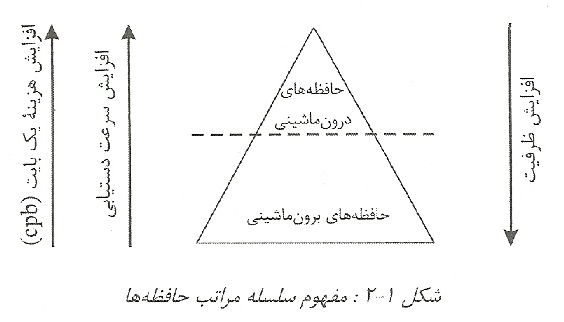
دسترس پذيري(Accessibility): به هر حافظه اي ، از طريق مکانيسم آدرس دهي مي توانيم دسترسي داشته باشيم.(البته دسترسي ممکن است به منظور خواندن/نوشتن باشد.)

ظرفيت(Capacity): هر حافظه اي داراي ظرفيتي است که به 'Bit' يا 'Byte' بيان مي شود.

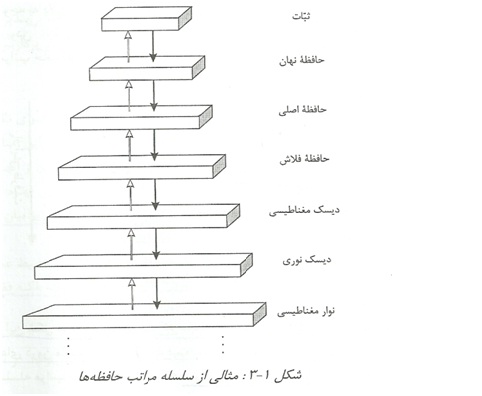
زمان دسترسي(Access Time): مدت زماني باشد ، بين لحظه اي که دستور خواندن/نوشتن داده مي شود و لحظه اي که حافظه ي مورد نظر مورد دسترسي قرار مي گيرد.(منظور از حافظه ي مورد نظر، قسمتي از حافظه است که داده ي مورد نظر در آن ذخيره مي شود.)

نرخ انتقال يا سرعت انتقال(Transfer Rate): به اطلاعاتي که در واحد زمان از حافظه انتقال مي يابد.(و آن را با b/s بيان مي کنند.)

# سلسه مراتب حافظه:

به طور کلي حافظه ها دو دسته مي باشند: 1. درون ماشيني 2.بيرون ماشيني

از اين شکل ساده معلوم مي شود که حافظه هاي برون ماشيني گسترش حافظه هاي درون ماشيني مي باشند( البته با ظرفيت بيشتر و سرعت دسترسي کمتر)



حافظه هاي درون ماشيني:

1. Registers 2.Ram 3. Etc. 4 Catch

### حافظه هاي بيرون ماشيني:

1. نوار کاغذي پانچ شده
2. ديسک نوري(CD)
3. ديسک هاي فشرده
4. کارت پانچ شده
5. نوار مغناطيسي
6. ديسک مغناطيسي
7. طبله يا درام
8. etc

# نوار مغناطيسي:

اهميت نوار مغناطيسي: الف:ساده بودن . ب.رايج بودن

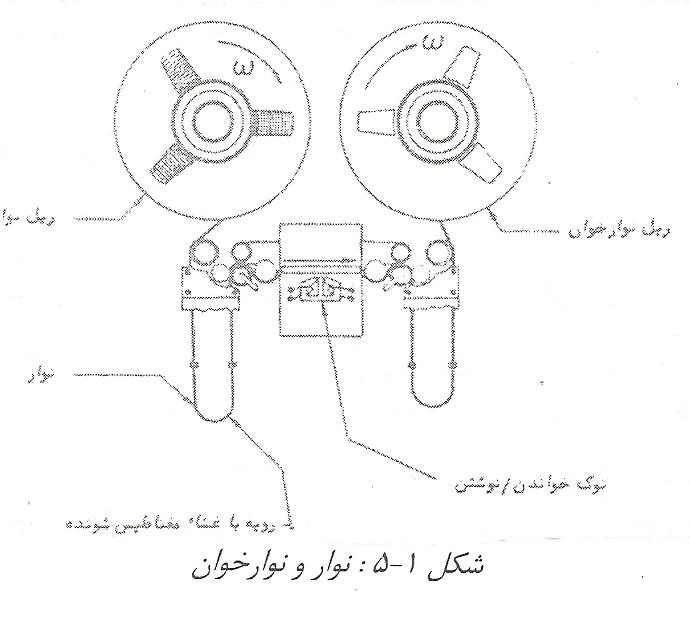
تعريف نوار : يکي از حافظه هاي برون ماشيني است.(در معناي عام)

(اما در معناي خاص) ، رسانه اي است با امکان دستيابي ترتيبي همراه با نوار گردان و کنترلر نوار، که امکان خواندن و نوشتن اطلاعات بر روي نوار را دارد.

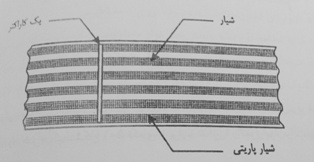
همچنين نوار از نوعي پلاستيک (که معمولاً يک رويه ي آن مغناطيسي شده باشد) ساخته شده است.

تعريف دستيابي ترتيبي: يعني براي رسيدن به داده ي n-اُم ، بايد n-1 داده ي قبلش را پشت سر بگذاريم(مهم ترين تفاوت نوار با ديسک همين است)

تعريف دستگاه نوار خوان: اين دستگاه مجهز به هد R/W مي باشد که مي تواند بوسيله ي آن اطلاعات را روي نوار ضبط کند يا اطلاعات ضبط شده را حس(Sense) کند.

****

**نحوه ي ذخيره سازي داده روي نوار**: يک داده به صورت رشته هاي بيتي بر روي شيارهايي که در سطح نوار وجود دارند ، ذخيره مي شود



نوار از نظر تعداد شيار دو نوع است:

7شياره : براي ذخيره سازي داده هاي 6 بيتي+ 1 بيت Parity

9شياره : براي ذخيره سازي داده هاي 8 بيتي+ 1 بيت Parity

* در نوار 7 بيتي parity يکي از شيار هاي لبه است
* در نوار 9 بيتي parity شيار وسط است

تقسيمات نوار:

بايت بايت هاي ذخيره شده در يک خط نوار

رکورد چند تا بايت(که بستگي به نوع کار ما دارد)

بلاک چند تا رکورد

فايل چند تا بلاک

**تعريف گپ**(Gap): فضايي است بين دو گروه کاراکتر ضبط شده.

وجود گپ براي متوقف کردن نوار و يا حرکت دوباره ي آن ، لازم است.

شکلي از Gap-ها:IRG AND IBG

|  |
| --- |
|  |

Block or record

Inter Block(Record)Gap

دلايل استفاده از Gap :

1.عامل **جداسازي**

2.فضايي است براي **توقف و حرکت**

V

**(زمان توقف و حرکت ،** Start/Stop): به اختلاف زماني بين رَد کردن Gap با توقف و رد کردن Gap بدون توقف مي گويند.(واحد آن ms مي باشد.)

زمان توقف و حرکت يکي از پارامتر هاي زماني مي باشد.

## نحوه ي نشست فايل بر روي نوار:

فايل معمولاً در قالب بلاک هايي به طور پي در پي روي نوار قرار گرفته مي شوند.(در اين حالت اصطلاحاً مي گوييم فايل بلاک بنديBlocked شده است.)

در يک نوار مي توان بيش از يک فايل را ذخيره کرد و هر فايل داراي برچسب(Marker) آغاز و پايان فايل(BOF-EOF) مي باشد.

فايل ها توسط سيستم فايل(File System) بر روي نوار نوشته مي شوند.

# پارامتر هاي نوار:

1. پارامتر هاي ظرفيتي(يعني عاملي که مشخص مي کند ظرفيت يک نوار از نواري ديگر بيشتر است.)

**طول نوار**(واحدش فوت است)(**مثال**: L=1200 ft)

**چگالي نوار**(با D(Density) نمايش مي دهند و يعني تعداد بيت هاي قابل ضبط در هر اينچ از نوار)

(**مثال**: D=6250 bpi(**b**it **p**er **i**nch)

**ظرفيت اسمي** (فرمول: )

ظرفيت اسمي: يعني کاري انجام شود که هيچ انرژي تلف شده اي نداشته باشد.(اما اگر تلف شده ها را نيز حساب کنيم ، مي شود ظرفيت واقعي)

**مثال: حجم فايلي بر روي نواري با چگالي** 1600 bpi **برابر** 800000 Byte **مي باشد. طول اين فايل بر روي نوار را محاسبه کنيد.**

1. پارامتر هاي زماني

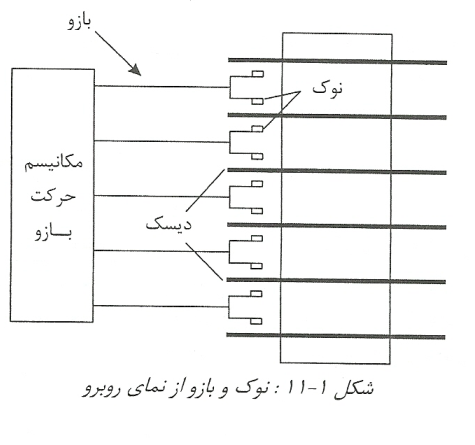
**زمان توقف و حرکت**(واحد آن ms)

**سرعت لغزش نوار**(inch/s)

**نرخ انتقال اسمي**(byte/s)(يعني تعداد بايت هايي که مي توان در هر ثانيه خواند)

تعريف ديسک:(در معناي خاص) ديسک تشکيل شده از تعدادي صفحات گردان حول يک محور ، که مجهز به ديسک گردان و کنترلر ديسک است که امکان خواندن و نوشتن اطلاعات را به ما مي دهد. صفحات ديسک معمولاً در هر دو رويه ي آن مغناطيسي شده است.

ديسک هم مانند هر رسانه ي ديگري ، داراي کنترلر مخصوص به خود مي باشد ، که به يک کنترلر تعدادي درايور متصل مي شود.



## انواع ديسک ها

1. بر اساس جنس آنها:

**سخت**:(چرا به آنها سخت مي گويند؟)چون جنس صفحات آلياژي از آلومينيوم است.(**مثل**:Hard Disk)

**نرم**:(چرا به آنها نرم مي گويند؟) چون جنس آنها از پلاستيک است.(**مثل: فلاپي**)

1. بر اساس قابليت جابه جا شدن آنها(توجه شود که ديسک ها توسط کاربر معمولي باز و بسته نمي شود.)
2. بر اساس تعداد صفحات دايره اي آنها:

**تک صفحه اي**.(که معمولاً دو رويه اند)

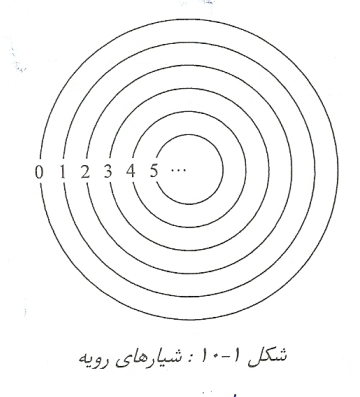
**چند صفحه اي**.(معمولاً به خاطر حفاظت داده ها سطح رويه ي صففحه ي بالا و سطح رويه ي پايين قابل دسترس اند.)

بر اساس نوع بازوي آنها:

**بازوي ثابت** (چنيدن Reader براي هر رويه داريم(

**بازوي متحرک** (تنها يک رويه براي هر رويه داريم(

# تقسيم بندي ديسک ها:

**تعريف** **شيار** :بر روي هر ديسک دواير متحد المرکزي وجود دارد که محل ذخيره سازي داده هاست. به اين محل، شيار مي گويند.

و شيار ها از بيرون به درون ، از صفر تا n شماره گذاري شده اند.

توجه کنيم که در ديسک هاي مغناطيسي سرعت زاويه اي در تمام شيار ها با هم برابر اند ، اما سرعت خطي در هر يک از شيار ها با هم متفاوت مي باشد.

تعريف سکتور(بخش): تقسيماتي باشد از شيار با اندازه ي مساوي، و هر شيار از تعدادي سکتور تشکيل شده است. و دو نوع سکتور وجود دارد:

سکتور سخت افزاري: که توسط سازنده ي ديسک ايجاد مي شود.( Low Level Formatting)

سکتور نرم افزاري: که از طريق نرم افزار(سيستم عامل) قابل ايجاد است و به اين کار فرمت بندي نرم افزاري مي گويند.

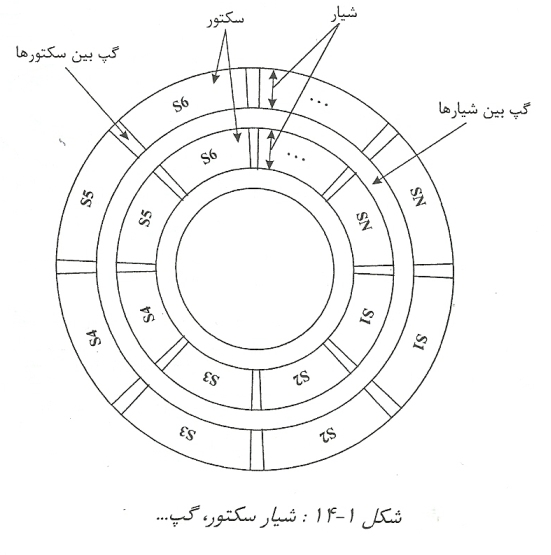
ظرفيت همه ي سکتور ها با هم برابر هستند ، بنابراين بديهي است که چگالي سکتور هاي دروني تر بيشتر است.

آغاز سکتور نرم افزاري بايد همان آغاز سکتور سخت افزاري باشد و اندازه اش مي تواند کمتر از يک سکتور، يک سکتور و يا بيش از يک سکتور باشد.

بنابراين بهتر است که اندازه ي سکتور نرم افزاري مضرب صحيحي از اندازه ي سکتور سخت افزاري باشد تا از به وجود آمدن حافظه ي تلف شده در انتهاي سکتور جلوگيري شود.

**تعريف سيلندر**(استوانه) : شيار هاي هم شماره از رويه هاي مختلف تشکيل سيلندر مي دهند.(يعني استوانه ي i-اُم شامل تمام شيار هاي i-اُم از صفحه هاي مختلف ديسک است.)

دليل استفاده از سيلندر ، ايجاد فضاي سه بعدي است اما نوار دو بعدي است.

بُعد سوم ديسک ها سيلندر است که براي آدرس دهي استفاده مي شود.

مؤلفه هاي آدرس دهي:

1.**شماره ي ديسک**

2.**شماره ي سيلندر**(دروني يا بروني)

3.**شماره ي رويه**(شيار) **در سيلندر**

4.**شماره ي سکتور**

|  |  |  |  |
| --- | --- | --- | --- |
| Sec# | Tr# | Cyl# | D# |

# پارامتر هاي ديسک:

پارامترهاي ظرفيتي(که روي ديسک تأثير مي گذارند) :

1**.ظرفيت** سکتور(1)

همه ي سکتور ها با هم برابراند ، پس اگر ما ظرفيت يک سکتور را داشته باشيم و تعداد سکتور ها در يک شيار را هم داشته باشيم ، آنگاه مي توانيم ظرفيت شيار را حساب کنيم.

2.تعداد سکتور در شيار(2)

3.تعداد شيار در استوانه(تعداد رويه((3)

4.تعداد سيلندر در صفحه(4)

**ظرفيت ديسک** 1\*2\*3\*4

### پارامترهاي زماني:

سرعت گردش ديسک(Revolutions per minute :rpm)

از اين پارامتر، زمان يک دور ديسک به دست مي آيد و آن را با 2r نشان مي دهند.

سرعت درنگ دوراني(واحد:ms)(0<r<2r)

مدت زماني که طول مي کشد تا آغاز داده ي مورد نظر در اثر دوران ديسک به زير نوک R/W برسد.

زمان استوانه جويي(واحد:ms)(S)

يعني زماني که طول مي کشد تا نوک R/W به استوانه ي مورد نظر برسد.

و زمان دسترسي مستقيم مي شود: (s+r)

: به زمان طي کردن يا حرکت کردن از i تا استوانه مي گويند.

توضيح : وقتي روي قرار داريم و به ما گفته اند برو چه قدر زمان مي خواهيم؟

توضيح : وقتي روي هستيم و به ما مي گويند برو چه قدر زمان مي خواهيم؟

نرخ انتقال(واحد: b/s)

تعداد بايتي که در يک ثانيه مي توان انتقال داد و دو نوع نرخ انتقال داريم:

1. **اسمي** : توسط سازنده ي ديسک اعلام مي شود.

2.**واقعي** : قابل محاسبه مي باشد.

نحوه ي ضبط داده ها روي ديسک: داده ها به صورت رشته هاي بيتي روي شيار ها ذخيره مي شوند، البته چگالي ضبط داده ها در شيار هاي بيروني کمتر از شيار هاي دروني مي باشد.

**مثال: در ديسکي با سرعت 6000 دور در دقيقه ، متوسط زمان درنگ دوراني را حساب کنيد.**

فصل دوم :

# مفاهيم مبنايي فايلينگ

**تعريف** **فيلد**: به مکان ذخيره سازي يک واحد معنايي داده(Semantic data unit) که داراي نام باشد، يک فيلد مي گوييم.يا به هر صفت خاصه و تفسير آن يک فيلد مي گويند.

رکورد:از سه ديدگاه مورد بررسي قرار مي دهيم:

از ديدگاه مفهومي(ديد کاربر نسبت به رکورد)

از ديدگاه منطقي(ديد پردازشگر نسبت به رکورد)

از ديدگاه فيزيکي(نحوه ي نشست رکورد روي رسانه ي ذخيره سازي)

توضيح ديدگاه مفهومي رکورد : مجموعه اطلاعاتي است در مورد نمونه هاي مختلف معمولاً يک نوع موجوديت در يک محيط عملياتي.

**تعريف موجوديت** : يک مفهوم کلي هر پديده يا شي اي است در محيط عملياتي. يا به هر يک از مصاديق متمايز يک موجوديت را نمونه مي گويند.

مثال: نوع موجوديت: دانشجو

صفات خاصه: نام و نام خانوادگي،رشته،سال ورود، سال تولد،قد ،وزن و...

تعريف اطلاع: به طور کلي هر صفت خاصه داراي دو مؤلفه مي باشد: 1. اسم صفت خاصه 2. مقدار صفت خاصه. که وقتي اين جفت مؤلفه باهم موجود باشند، مي گوييم اطلاع حاصل شده است.

**مثال**:

x,y**=نام ونام خانوادگي**

اسم صفت خاصه

مقدار صفت خاصه

**c=رشته**

تعريف محيط عملياتي: زير مجموعه اي از دنياي واقعي که ما داريم محيط را در آن تعريف مي کنيم.

صفات خاصه دو نوع اند:

**ذاتي است** ،يعني وابسته به محيط عملياتي نيست. (مثل نام دانشجو ، چون دانشجو قبل از اينکه دانشجو بشود ، نام داشت)

**ساختگي است** ،يعني وابسته به محيط عملياتي هست.(**مثل** کد دانشجويي ،که قبل از اين که دانشجو شويم کد دانشجويي نداشتيم.)

هر رکورد از چندين فيلد تشکيل مي شود.

توضيح ديدگاه منطقي(ديد برنامه ساز): درک ذهني از رکورد بايد در يک قالب و چارچوب مشخص ارائه شود ، که دو طرح(قالب) براي اين منظور وجود دارد:

* طرح (قالب) ثابت مکاني
* طرح(قالب) غير ثابت غير مکاني

توضيح طرح(قالب) غير ثابت غير مکاني : در اين حالت اسم صفت خاصه و مقدار آن ذخيره مي شود.(به شکل زير توجه کنيد.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| پلاک | آدرس | شماره شناسنامه | نام پدر | نام و نام خانوادگي |
| 32 | همين ورا | 2325 | محمد | علي مدني |

|  |  |  |
| --- | --- | --- |
| نام پدر | شماره شناسنامه | نام و نام خانوادگي |
| فريد | 4532 | حسن غلامي |

به نکات زير توجه مي کنيم:

عدم وجود ترتيب در صفات خاصه ممکن است.

برخي از صفات خاصه در بعضي از نمونه ها وجود ندارند.

معمولاً از طرح غير ثابت غير مکاني زماني استفاده مي شود که برخي از مقادير صفات خاصه در برخي از نمونه ها وجود نداشته باشند.(پديده ي non-data)

توضيح طرح(قالب) ثابت مکاني : در اين طرح تناظري بين مکان ذخيره سازي و صفات خاصه وجود دارد که در نتيجه ي آن لازم به ذخيره سازي اسم صفات خاصه نيست.(به شکل زير توجه کنيد.)

|  |  |  |  |
| --- | --- | --- | --- |
| آدرس | تلفن | نام پدر | نام خانوادگي |
| **آدرس 1** | 885523 | سالار | مظلومي |
| **آدرس 2** | 223365 | عباس | مهدوي |

به نکات زير توجه مي کنيم:

در هر دو طرح فوق ممکن است طول رکورد متغيير باشد.(چه چيزي باعث تغيير طول رکورد ها مي شود؟ فيلد ها)

معمولاً زماني از طرح غير ثابت غير مکاني استفاده مي شود که شرايط به گونه اي باشد که طول رکورد ها متغيير باشند.

دلايل متغيير شدن طول رکورد:

* عدم وجود مقدار براي برخي از صفات خاصه.(مثل : شماره موبايل ، نام مادر)
* طول مقادير صفات خاصه ممکن است متفاوت باشند.(مثل : علي زند يا سيد محمد تقي روحاني رامکوهي)
* رخداد پديد ه ي فقره اطلاع شونده(repeating item) يا رخداد گروه اطلاع تکرار شونده(repeating group)

|  |  |
| --- | --- |
| 100 | 2536 |

زماني است که براي يک اسم صفت خاصه ، چندين مقدار را داشته باشيم.(مثل: تعداد يک کالا در انبار)

به چنين فايلي که حاوي اين نوع اطلاعات است ، فايل نامسطح مي گويند.

گروه اطلاع تکرار شونده:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 100000 | 1359 | علي حسيني |

**تمرين**: سه مثال براي گروه اطلاع تکرار شونده و سه مثال براي فقره اطلاع تکرار شونده بزنيد.

توضيح ديدگاه فيزيکي رکورد: نحوه ي نشست فيزيکي رکورد.

رکورد از **دو بخش** تشکيل شده است:

1.بخش پيشوندي رکورد(يا بخش غير داده اي، بخش کنترلي)

2.بخش داده اي

Header

Dd بخش داده ای

توضيح بخش غير داده اي: بخشي است که توسط خود سيستم توليد شده است و تشکيل شده از چند تا فيلد که فيلد هاي آن:

1.فيلد حاوي نوع رکورد

2.فيلد حاوي طول رکورد(در حالتي که طول رکورد متغيير است)

3.فيلد اشاره گر(Pointer)(براي ايجاد ارتباط منطقي بين رکورد ها استفاده مي شود)

4.فيلد حفاظتي:

الف) براي اينکه مشخص شود که چه کاربر يا کاربراني مي توانند به رکورد دسترسي داشته باشند.(کاربران را محدود مي کند)

ب) براي اينکه مشخص شود که چه اعمالي بر روي آن مجاز است(Read-Only)

5. فيلد کد عملياتي

الف) بيانگر عملي است که بر روي رکورد انجام شده است.

ب) بيانگر عملي است که بايد بر روي رکورد انجام شود.

Type

Length

Security

Pointer

Oprational code

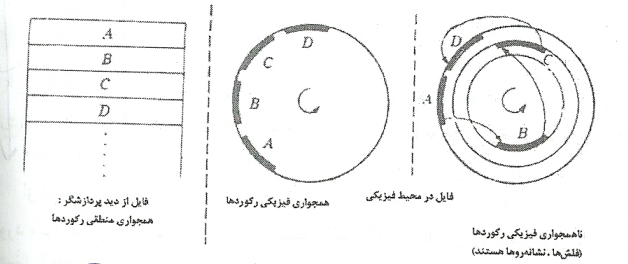
# لوکاليتي(locality) :

فيلد اشاره گر ، به رکورد بعدي اشاره مي کند.

فيلد اشاره گر ، در جايي استفاده مي شود که در قبل و بعد معني داشته باشد.(يعني در جايي که نظم وجود داشته باشد.)

داده هاي پايل(Pile) داراي نظم و ساختار مشخصي نمي باشند ، به همين خاطر ممکن است که فيلد اشاره گر در آنها استفاده نشود.

اصطلاحاً به ميزان هم جواري فيزيکي رکوردهاي منطقاً هم جوار، Locality مي گويند.(يعني اونايي که منطقاً هم جوارند، چه مقدار فيزيکي هم جوار مي باشند.)



بلاک (Block):

مي دانيم که يک بلاک از يک يا چند رکورد تشکيل شده است.

واحد I/O بلاک مي باشد.

طول بلاک ها ثابت مي باشد.(اما رکورد ها ممکن بود که داراي طولي ثابت يا طولي متفاوت باشند.)

اندازه ي يک بلاک را با 'B byte' نمايش مي دهند.

در Windows ، هر 4096 بايت يا 8192 بايت ، يک بلاک مي باشد.

Blocking, Deblocking:

فرض کنيم که در شکل زير برنامه ي ما مي خواهد رکورد دوم از بلاک مشخص شده را بخواند، براي اين کار:

ابتدا دستور Read از Ram بر روي ديسک صادر مي شود تا بلاکي که رکورد مورد نظر ما در آن قرار دارد ، خوانده شود.

بلاک خوانده شده به بافر برده مي شود.(**بافر**:حافظه ي موقت مي باشد.)

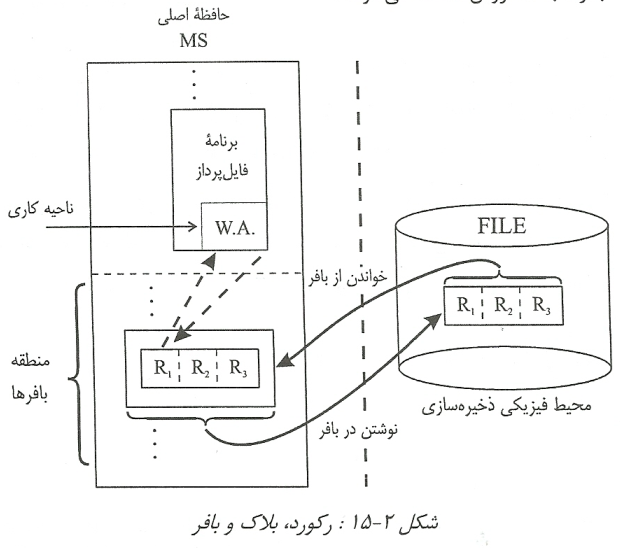
در قسمت بافر بلاک ما باز مي شود و حالا مي تواند به رکورد مورد نظر دسترسي داشته باشد.(Deblocking)

وقتي کار برنامه ي ما با رکورد دوم تمام شد ، بلاک ما دوباره بسته مي شود.(Blocking)

پس هر موقع که ما با فايلي کار داريم File System ، عمل Blocking و Deblocking را براي فايل مورد نظر ما انجام مي دهد.

براي فايل مورد نظر ما انجام مي دهد.

File System: مديريت نوشتن ها و خواندن ها را از ديسک بر عهده دارد.



# روش هاي تشخيص محدوده بلاک:

به نکات زير توجه مي کنيم:

در صورت ثابت بودن طول رکورد ، نيازي به تکنيک خاصي نداريم.

اگر در بين رکورد ها گپ داشته باشيم ، بلاک بندي مفهومي ندارد(چون گپ يا بين رکورد هاست يا بين بلاک ها)

سه روش براي **تشخيص محدوده ي بلاک ها** وجود دارد:

1. فيلد طول در بخش پيشوندي باشد.(يعني هر رکورد همان اولش بگويد که من به چه اندازه هستم)

R2

R1 L2

L1 L3 R3

1. استفاده از يک نشانگر(Marker) خاص در انتهاي رکورد ها(مثلا خط فاصله)

R1 - R2 - …

1. استفاده از جدول مکان نما در انتهاي بلاک(يعني آدرس هاي رکورد ها را در يک جدول نگه داريم)

R3

R2

R1

1200 800 0

* حالت خاص روش سوم:

L3

L1

R3

R2

R1

L L2

L1

L3

2

R1  R2 R3

|R1|=|R2|=L1

تمرين: "چه روش هاي ديگري ممکن مي باشد؟

# روش هاي بلاک بندي:

1. بلاک بندي رکورد با طول ثابت يک تيکه W1 )تلف) IBG

...

R

R

R

R

R

R

WS (فضايي که هيچ بلاکي در آن جا نمي شود(

W1 :ناشي از نگنجيدن رکوردي ديگر در بلاک

WS: ناشي از نگنجيدن بلاکي ديگر در شيار

L1 L2

R1

R2

L3

R3

R3

pointer

1. بلاک بندي رکورد با طول متغيير و دو تيکه

بلاک بندي رکورد ها با طول متغيير و طول يک تيکه

R1  L3 …

L1

R2

L2

R3

W1

براي ارزيابي هر روش بلاک بندي، قرارداد زير را داريم:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **قرار داد** | | | | | |
| اندازه ي گپ |  |  | تعداد رکورد ها در يک بلاک |  |
| حافظه ي تلف شده به ازاي يک رکورد |  | تعداد بلاک ها در يک شيار(Track) |  |
| حافظه ي تلف شده به ازاي يک بلاک |  | طول رکورد |  |
| اندازه ي اشاره گر يا اندازه ي فيلد طول در بخش پيشوندي |  | طول بلاک |  |

**فرمول هاي روش اول** :

فرمول هاي روش دوم :

به احتمال زيادي در هر بلاک ، يک اشاره گر به رکورد بعدي وجود دارد.

فرمول هاي روش سوم :

# مقايسه روش هاي بلاک بندي:

مقايسه ي روش اول با روش دوم و سوم

حسن ها:

داراي پردازش ساده تر مي باشد.(چون طول رکورد ها ثابت مي باشند)

حجم نرم افزار کمتر مي باشد.(منظور ما نرم افزاري مي باشد که وظيفه ي خواندن و نوشتن روي ديسک را دارد(System file))

معايب :

انعطاف پذيري کمتر مي باشد.

روش دوم انعطاف پذيري بيشتري نسبت به روش اول دارد.(چون در اين روش براي طول رکورد محدوديتي وجود ندارد)

حداکثر طول رکورد در روش سوم به اندازه ي طول مقدار آن مي باشد.

به طور متوسط اتلاف حافظه در روش دوم کمتر از روش سوم مي باشد.

روش سوم به طور متوسط براي خواندن يک رکورد ، تعداد I/O-ِ کمتري لازم دارد.

مديريت و بلاک بندي در روش دوم به مراتب سخت تر مي باشد ، به عبارتي پردازش پيچيده تر ، حجم نرم افزار بيشتر ، و مديريت سخت تر مي باشد.

## مزايا و معايب بلاک بندي :

مزايا:

صرفه جويي در مصرف حافظه(زيرا به جاي گپ بين رکوردي ، گپ بين بلاکي داريم.)

افزايش سرعت I/O(چون در هر I/O به جاي خواندن يک رکورد ، تا رکورد مي خوانيم.)

معايب:

اتلاف حافظه(به دليل ايجاد انواع حافظه هاي تلف شده ي درون بلاکي)

براي انجام بلاک بندي(Blocking) و از بلاک در آوردن(Deblocking) نياز به نرم افزار داريم.

اتلاف حافظه ي اصلي(Ram)(به دليل نياز به بافر ، چرا که لازمه ي Blocking ، Buffering مي باشد.)

با افزايش طول داده ي انتقالي ، احتمال رخ داد خطا هم افزايش پيدا مي کند.( البته با توجه به پيشرفت هاي تکنولوژي امروز، اين احتمال صفر مي باشد.)

**تعريف** **باکت**: يکي از تقسيمات منطقي فايل مي باشد و از چندين بلاک تشکيل شده است و در صورت وجود واحد I/O در سطح منطقي مي باشد(ناحيه کنترلي – سکشن- ترن-پارتيشن).(واحد I/O در سطح فيزيکي همان Block مي باشد.)

# چگالي لود اوليه:

**تعريف** **چگالي لود اوليه** : نسبت حافظه ي پر شده به کل حافظه ي بلاک مي باشد.

معمولاً در لود اوليه تمامي بلاک پر نمي شود.

فرمول :

شکل :

Ld خالی

دلايل چگالي لود کمتر از 100% :

مزاياي چگالي لود کمتر از 100%:

1. ساده شدن و سرعت دادن به انجام برخي از اعمال بر روي فايل.

فرض کنيم که مي خواهيم فايل ما مرتب باشد و رکورد هاي زير هم موجود مي باشند.(و چگالي لود=100% و )

I

F

E

A

B

D

اگر بخواهيم رکورد C را درج کنيم ، بايد:

براي اين کار بايد همه ي رکورد هاي به اندازه ي يک رکورد به انتهاي فايل شيفت کنند تا جا براي رکورد C باز شود.

حالا فرض مي کنيم که رکورد G هم مي خواست بيايد ، دوباره بايد همه ي رکورد ها را به سمت راست شيفت مي داديم و اگر رکورد I هم مي خواست برود ، دوباره بايد همه ي رکورد ها را به سمت چپ شيفت مي داديم.

براي جلوگيري از اين همه شيفت ، يک مقدار طول لود را کمتر از 100% مي گيريم تا اگر اتفاقي هايي مانند اتفاق مثال ما افتاد ، با مشکل کمتري رو به رو شويم.

F

E

B

A

شکل راه حل:

1. حفظ Locality

براي فهميدن اينکه چه درصدي از چگالي لود خوب مي باشد ، بستگي به اين دارد که(**مثل اتوبوس**) :

جايي که احتمال درج بعدي زياد مي باشد، چگالي لود را پايين تر مي گيريم.

چگالی لود

جايي که احتمال درج بعدي کم مي باشد، چگالي لود را زياد تر مي گيريم.

معايب چگالي لود کمتر از 100%

1. اتلاف حافظه هايي که ممکن است رزرو شده اصلاً پر نشود (در نتيجه حافظه ي تلف شده در جاجاي فايل به وجود مي آيد.)
2. افزايش زمان خواندن فايل(چون تعداد بلاک هاي ما زياد شده است.)

نکته : اساساً در محيط هاي خطي و ايستا ، استفاده از چگالي لود اوليه ي کمتر از 100% پيشنهاد نمي شود

مثال: در فايلي با 1000 رکورد که طول رکورد ها 40 بايت و طول بلاک 200 بايت است، در صورتي که چگالي لود اوليه را در دو حالت 80% و 100% در نظر بگيريم، تعداد بلاک اشغالي توسط اين فايل چند است؟

به ميزان همجواري فيزيکي رکوردهاي منطقاً هم جوار Locality مي گويند.(**مثل يک گوني سيب زميني**)

Locality يک طيف مي باشد از ضعيف تا قوي. هر Locality قوي تر باشد ، انجام بعضي از اعمال سريع تر و راحت تر مي باشد.

# درجات Locality:

1. رکورد منطقاً بعدي در همان بلاکي مي باشد که رکورد فعلي قرار دارد ، و رکورد در بافر مي باشد.(يعني I/O نداريم)
2. رکورد منطقاً بعدي در همان بلاکي مي باشد که رکورد فعلي قرار دارد.(يعني با يک I/O هر رکورد به دست مي آيد و بلاک لزوماً در بافر نمي باشد.)
3. رکورد منطقاً بعدي در بلاکي باشد همجوار با بلاک حاوي رکورد فعلي.(در اين حالت زمان درنگ دوراني صفر مي باشد.())
4. رکورد منطقاً بعدي در همان استوانه اي باشد که رکورد فعلي وجود دارد.( )
5. رکورد منطقاً بعدي در استوانه اي هم شماره با استوانه حاوي رکورد فعلي باشد.(البته از يک ديسکي ديگر.)

تکنيک فايل توزيع شده ي چند Pack-ي يکي از تکنيک ها براي شبيه سازي ديسک با شانه ي ثابت مي باشد.

1. رکورد منطقاً بعدي در استوانه ي همجوار قرار دارد.
2. رکورد منطقاً بعدي در جايي مي باشد که آدرس آن را در اختيار داريم.()
3. آدرس رکورد بعدي با انجام محاسبات به دست مي آيد.(همان حالت 7 مي باشد باضافه ي يکسري پردازش)
4. آدرس ها در يک فايل ديگري مي باشد که با خواندن آن فايل آدرس ها به دست مي آيند.

نکته:ممکن است در لود اوليه Locality قوي باشد اما به تدريج با انجام عمليات جديد بر روي آن، Locality ضعيف شود که باشد احيا گردد.(به عمل احيا کردن Locality ، **سازماندهي مجدد** (Reorganization) مي گويند.)

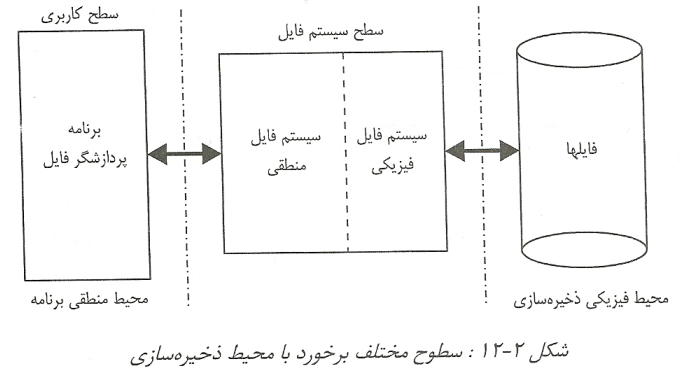
1 2 3 4 5 6 7 8 9 10

1 2 4 5 6 9 10

# سطوح و مکانيزم آدرس دهي به فايل:

سطوح آدرس دهي به سه سطح تقسيم مي شوند:

1. سطح کاربري(منظور برنامه ي کاربردي)
2. سطح سيستم فايل منطقي
3. سطح سيستم فايل فيزيکي



### سطح کاربري:

در اين سطح آدرس دهي معمولاً به يکي از سه صورت زير انجام مي شود:

1. Symbolic : يعني کاربر نام فايل و نام رکورد را وارد مي کند.
2. **آدرس دهي نسبي** : شماره ي نسبي رکورد در فايل داده مي شود.
3. **آدرس دهي محتوايي**: مقدار يک يا چند صفت خاصه را مي دهيم.

رکورد يا رکورد هايي که صفت خاصه ي آنها با آن مقدار داده شده مساوي باشد، را مورد جستجو قرار مي دهيم.

### سطح فيزيکي:

در اين سطح سيستم بايد مؤلفه هاي فيزيکي را به کنترلر فايل(نرم افزار) بدهد تا رکورد به دست آيد.

البته بايد توجه داشته باشيم که در اينجا تمام نرم افزار هايي که با فايل ها کار مي کنند ، کاربر به حساب مي آيند.(**مثل**: Word , Excel)

### سطح منطقي:

تبديل برنامه ي درخواست کلان کاربر به سطح سيستم فايل فيزيکي ، توسط سيستم فايل منطقي انجام مي شود.

سيستم فايل منطقي ديد خاصي نسبت به کل محيط ذخيره سازي دارد . به اين صورت که کل محيط را به صورت خطي مي بيند و تشکيل شده از تعدادي بلاک مي باشد.

بلاک

...

هر بلاک داراي يک شماره ي منحصر به فرد مي باشد که اصطلاحاً به آن RBA(يا شماره ي نسبي بلاک) مي گويند.

سيستم فايل با استفاده از RBA بلاک ، مؤلفه هاي آدرس فيزيکي را به شرحي که خواهيم ديد استخراج خواهد کرد.

RBA در کل محيط ذخيره سازي مطرح مي باشد. هر رسانه(نوار، ديسک) يک محدوده ي RBA براي خود دارد.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **شماره ي ديسک** | **تعداد استوانه** | **تعداد شيار(رويه)** | **تعداد بلاک ها** | **دامنه ي RBA** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**مثال: فرض کنيم که سه نوع ديسک داريم**:

**نحوه ي محاسبه ي RBA**:

را سيستم در اختيار دارد ، پس بايد را محاسبه کنيم:

نکته: شماره ي نسبي از صفر شروع مي شود

**نحوه ي محاسبه ي مؤلفه هاي آدرس دهي از روي RBA**:

که اين کار بر عهده ي سيستم فايل فيزيکي مي باشد.

فرض کنيم x شماره ي RBA مورد نظر در کل فضاي ذخيره سازي مي باشد. با مقايسه ي x و دامنه ي RBA -ِ هر يک از ديسک ها ، شماره ي ديسک مشخص مي شود.

**مثال: فرض کنيد که سه ديسک با شرايط زير موجود مي باشد**:

|  |  |  |  |
| --- | --- | --- | --- |
| **ديسک** | **تعداد استوانه** | **تعداد شيار** | **تعداد بلاک در شيار** |
|  | 100 | 20 | 10 |
|  | 30 | 50 | 20 |
|  | 50 | 50 | 20 |

**الف)**

**اولاً: محدوده ي** RBA**-ها را مشخص کنيد.**

**ثانياً: آدرس فيزيکي رکورد با شماره ب نسبي درون فايلي را در حالي که و باشد را حساب کنيد.**

**ب) اگر** n=10,000**(تعداد رکورد هاي فايل) ، مطلوبست آدرس** EOF**.**

# راهنماي فايل ( فهرست فايل )

هر سيستم فايل، معمولا از يک راهنماي فايل براي مديريت فايل ها استفاده مي کند.

راهنماي فايل جدولي است که داراي تعدادي(Entry) مي باشد.

در هر Entry ، نام فايل ، صفات خاصه ي فايل و آدرس هايي که داده هاي فايل در آن ها ذخيره شده اند ، نگهداري مي شود.

راهنماي فايل حاوي اطلاعاتي در مورد فايل مي باشد و معمولاً اين اطلاعات به هنگام دسترسي به فايل مورد استفاده قرار مي گيرد.

در يک راهنماي فايل معمولاً اطلاعات زير وجود دارد:

* و معمولاً (يعني ممکن است وجود نداشته باشد و محاسباتي است. )
* تعداد رکورد ها و فاکتور بلاک بندي (n و )
* طول رکورد(وقتي که طول رکورد ثابت باشد)
* نام صاحب فايل
* تاريخ ايجاد و تاريخ فايل(Last Access Date)
* اندازه ي جاري فايل(مثلاً الان اين فايل 20 kb مي باشد.)
* حداکثر اندازه ي مجاز فايل
* نرخ(سرعت، مقدار) رشد فايل
* ...

نحوه ي اجراي درخواست کاربر به صورت زير مي باشد:

1) سر زدن به بافر I/O و جستجوي رکورد در آن (زيرا ممکن است در اثر I/O-ِ قبلي ، بلاک مورد نظر در بافر موجود باشد)

2) بررسي معتبر بودن و مجاز بودن کاربر(انجام محاسبات مانند مثال قبل براي استخراج مؤلفه هاي آدرس فيزيکي مي باشد.)

3) انجام I/O و خواندن بلاک در بافر و در اختيار گذاشتن رکورد (عمليات Deblocking انجام مي شود.)

# بافر و بافرينگ

**تعريف** **بافر** : بافر حافظه اي واسط(ميان گير) در عمليات I/O روي فايل مي باشد.

وقتي بلاکي خوانده مي شود ، به بافري که در حافظه ي اصلي قرار دارد منتقل مي شود.

دليل استفاده از بافر : براي ايجاد هماهنگي بين CPU و I/O Device(چون که اختلاف سرعت در بين آنها وجود دارد.)

طبق شکل زير ، بافر هاي مورد نياز از بخش خاصي از حافظه ي اصلي(Ram) در اختيار عمل I/O قرار داده مي شود که در اصطلاح به اين بخش ناحيه ي بافر ها(Buffer Pool) گفته مي شود.

خواندن در بافر

حافظه اصلی

**MS**

منطقه بافرها

نوشتن در بافر

محیط فیزیکی ذخیره سازی

**R1**  **R2 R3**

فایل برنامه پرداز

**W.A.**

محیط فیزیکی ذخیره سازی

**مديريت بافر**: بافر يک منبع سيستمي مي باشد و بايد مانند ساير منبع هاي سيستمي مديريت شود.

به طور کلي **منبع هاي سيستمي** عبارتند از:

CPU

I/O Device

حافظه ي اصلي

I/O Channel

Buffer

براي مديريت هر منبع سيستمي بايد به **چهار** سؤال "چه" پاسخ داده شود:

"به چه برنامه اي ، به چه مدتي، چه منبعي را ، چگونه تخصيص دهد؟ "

براي مديريت بافر **دو روش** اصلي وجود دارد:

* 1. مديريت بر عهده ي سيستم فايل باشد.
  2. مديريت بر عهده ي مدير منابع OS باشد.(Resource Manager)

بافر ها به **دو دسته** تقسيم مي شوند:

1. بافرهاي نرم افزاي
2. بافر هاي سخت افزاري

**تعريف** **بافر هاي نرم افزاري** : بافر هايي مي باشند که از ناحيه ي Buffer Pool-ِ موجود در حافظه ي اصلي به برنامه ها اختصاص داده مي شوند.

**تعريف** **بافر هاي سخت افزاري** : بافر هايي مي باشند که در خود I/O Device وجود دارند.

تکنيک هاي دسترسي به محتواي بافر: (**دو روش** اصلي وجود دارد)

1. تکنيک يا اسلوب انتقالي(Move Mode)
2. تکنيک يا اسلوب مکاني(Locate Mode)

توضيح **تکنيک انتقالي** : پس از انتقال بلاک به بافر ، عمليات Deblocking توسط سيستم فايل انجام مي شود و رکورد ها به برنامه داده مي شوند. در اين روش کليه ي عميات بر عهده ي سيستم فايل مي باشد و برنامه تنها رکورد ها را دريافت مي کند. وظيفه Blocking بعد از پردازش نيز به عهده سيستم فايل است .

توضيح **تکنيک مکاني** : پس از انتقال بلاک به بافر ، آدرس بافر در اختيار برنامه ي کاربردي گذاشته مي شود و مسؤليت کليه ي اعمال بر عهده ي برنامه ي کاربر مي باشد.

آدرس بافر مي تواند

الف) در يک رجيستر خاص گذاشته شود.

ب) در يک حافظه ي خاص گذاشته شود. (که معمولاً کاربر مشخص ميکند )

نکته : حالت پيش فرض تکنيک انتقالي مي باشد ، مگر اينکه کاربر تقاضاي تکنيک مکاني را کند.

تعريف بافرينگ: بافرينگ يعني انجام چهار کار زير:

1.ايجاد بافر 2.تخصيص بافر

3.مديريت بافر 4.به کارگيري بافر

**اهميت بافرينگ از نظر مصرف حافظه :**

مثال : فرض کنيم که ظرفيت حافظه ي اصلي ما 2000 بلاک باشد و 30 بلاک هم برنامه ي کاربر مي خواهد . هر برنامه ي کاربر هم با 3 فايل ديگر کار مي کند و براي هر فايلي که برنامه ي کاربر دارد با آنها کار مي کند ، يک بافر نياز دارد که حداقل در آن 2 بلاک جا شود ,پس براي هر فايل هم 2 بافر نياز داريم.

حداقل اندازه ي بافر 180 بلاک مي شود،

پس حدوداً 10% از حافظه ي اصلي براي بافر از دست مي رود. **(رجوع شود به معايب بلاک بندي)**

روش هاي بافرينگ :

1- بافرينگ ساده (ياتکي)(Single Buffering)

2- بافرينگ مضاعف يا دوتايي(Double Buffering)

3- بافرينگ چرخشي(Cycle Buffering)

**توضيح بافرينگ ساده** : در اين حالت سيستم فايل تنها يک بافر در اختيار عمل I/O قرار مي دهد.

بافر پر میشود

بافر پر میشود





**I/O**

**CPU**

**بنابراين در يک بازه ي زماني مشخص :**

* يا بافر در اختيار عمل I/O مي باشد.
* يا محتواي آن در حافظه نوشته مي شود.
* يا از حافظه ي روي هارد خوانده مي شود.
* يا در اختيار CPU براي عمل پردازش مي باشد.

ملاحظه مي شود که در اين نوع بافرينگ توازي عملياتي وجود ندارد.(يعني چند عميات نمي توانند به طور همزمان انجام شوند.)

بنابراين در سيستم هاي چند کاربره(Multi User) يا سيستم هاي چند وظيف اي(Multi Task) مواقع بي کاري CPU مي تواند به بقيه ي پردازش ها اختصاص داده شود.

**توضيح بافرينگ مضاعف** : در اين نوع بافرينگ به ازاي هر فايل 2 بافر در اختيار عمل I/O قرار مي گيرد و طبق شرايطي امکان ايجاد توازي عملياتي وجود دارد.(يعني چند تا عمليات مي توانند به طور هم زمان انجام شوند.

**نمودار عمليات در پردازش انبوه در اين نوع بافرينگ به صورت زير مي باشد:**

CPU

Aخواندن در بافر

B خواندن در بافر

B پردازش بافر

Aپردازش بافر

Wait

Wait

آماده سازی اولیه

درخواست کاربر

I/O

CB

**سؤال : "*در چه شرايطي همروندي يا توازي عملياتي خواهيم داشت؟*" بافرينگ مضاعف با شرط کارايي**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **فرضيات** | | | | | |
| زمان پردازش يک بلاک به طور متوسط |  |  | طول بلاک |  |
| زمان انتقال يک بلاک |  | اندازه ي گپ |  |
| زمان گذر از يک گپ |  | نرخ انتقال | t |

اگر باشد(که معمولاً هست)، مي توان موازي کاري انجام داد.

#### فصل سوم:

# ارزيابي پارامتريک رسانه ها

منظور از پارامتر ها : 1.ظرفيت انتقال 2.نرخ انتقال

منظور از رسانه ها : 1.نوار 2.ديسک

براي محاسبه ي سرعت انتقال و ظرفيت واقعي ، ابتدا پارامتر ها را يادآور مي شويم:

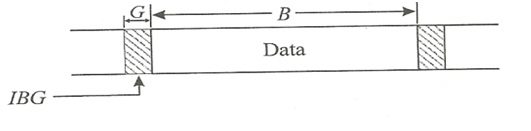
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| پارامتر ها | | | | | |
| نرخ انتقال واقعي |  |  | تعداد رکورد ها در يک بلاک |  |
| اندازه ي طول نوار |  | تعداد بلاک ها | b |
| چگالي نوار |  | اندازه ي گپ |  |
| زمان توقف و حرکت |  | نرخ انتقال اسمي |  |

چندين مثال:

* بخش پيشوندي رکورد
* حافظه ي تلف شده ناشي از نگنجيدن رکوردي ديگر در بلاک.
* گپ بين بلاک ها
* حافظه ي تلف شده ي ناشي از نگنجيدن بلاکي ديگر در شيار.
* بخش پيشوندي فايل(راهنماي فايل)
* و ...

# ظرفيت واقعي نوار :

با داشتن ظرفيت اسمي، مي توان ظرفيت واقعي را ارزيابي کرد.

عامل اصلي که نقش کاهنده را در ميزان استفاده ي واقعي از ظرفيت ذخيره سازي نوار دارد، همان گپ بين بلاک ها مي باشد. 

نکته: فرض مي کنيم B بايت يک بلاک حاوي داده ي خاص مي باشد و براي B بايت ، B+G بايت حافظه مصرف مي کنيم:

ظرفیت اسمی

مثال:در نواري با طول 600 اينچ و چگالي 800 bpi و با بلاک 200 بايتي و گپ بين بلاکي 50 بايت، ظرفيت اسمي و واقعي اين نوار را محاسبه کنيد.

مثال: با فرض اين که B=2000 و G=20 ، مطلوبست درصد استفاده؟

مثال:با مقدار هاي B هاي مختلف زير و G=100 مطلوبست درصد استفاده؟

B=100 درصد استفاده ميشود 50%

**100%**

**اندازه بلاک**

B=200 درصد استفاده ميشود 66%

B=2000 درصد استفاده ميشود 95%

B=3000 درصد استفاده ميشود 97%

B=4000 درصد استفاده ميشود 97.5%

نکته: ملاحظه مي شود که با اندازه ي گپ ثابت و با افزايش طول بلاک ، درصد استفاده ي واقعي افزايش پيدا مي کند . اما به دو دليل اندازه ي بلاک را خيلي بزرگ در نظر نمي گيريم:

1.ملاحظه مي شود که از يک اندازه ي خاصي به بعد ، نرخ استفاده از بلاک بسيار کند مي شود.

2.قبلاً گفته شد که لازمه ي بلاکينگ ، بافرينگ مي باشد. هر چه اندازه ي بلاک بزرگ تر باشد ، اندازه ي بافر بزرگتر مي شود و در نتيجه حافظه ي لازم براي بافرينگ بيشتر مي شود. (معايب باک بندي)

# نرخ انتقال واقعي

يعني با چه سرعتي داده انتقال مي يابد.

نرخ انتقال معمولاً با حجم در بازه ي زمان اندازه گيري مي شود.

1 . زمان خواندن يک بلاک:

يعني خواندن از بين اين دو گپ.

مي دانيم زمان اسمي براي خواندن يک بلاک برابر است با:

*اما زمان واقعي براي خواندن يک بلاک چه قدر است؟*(بايد نمودار بکشيم.)

نرخ انتقال واقعی

نرخ انتقال اسمی

2 . زمان خواندن براي n بلاک:

مي دانيم که زمان اسمي براي خواندن n بلاک برابر است با:

توضيح: ما براي خواندن يک بلاک ترمز داريم و براي خواندن n بلاک هم يک ترمز داريم ، پس از بيشتر است.

در حالت کلي داريم:

t اسمی

N=1

يعني هر چه قدر تعداد بلاک هاي خوانده شده بيشتر شود ، نرخ انتقال اسمي هم بيشتر مي شود.

# ارزيابي پارامتريک ديسک :

محاسبه ي ظرفيت واقعي ديسک : براي اين کار بايد تمام حافظه هاي تلف شدن را از تمام ظرفيت اسمي کم کنيم.

انواع حافظه هاي هرز در ديسک:

1. بخش پيشوندي رکورد
2. بخش پيشوندي بلاک
3. حافظه ي هرز درون بلاکي
4. حافظه ي هرز ناشي از گپ ها
5. حافظه ي هرز ناشي از نگنجيدن بلاکي ديگر در شيار
6. بخش پيشوندي فايل

نکته : بايد يک راه حل اساسي براي محاسبه ي مجموع حافظه هاي هرز به دست آوريم.

نکته : بايد يک راه حل عملي براي محاسبه ي مجموع حافظه هاي هرز به دست آوريم.

## محاسبه ي درصد استفاده در شيار:

با فرض خالص بودن b بايد يک بلاک داريم:

## ارزيابي نرخ انتقال واقعي در ديسک:

نکته: نرخ انتقال واقعي در ديسک به شرطي که خواهيم ديد ، به عوامل زير بستگي دارد:

* به نحوه ي پردازش بلاک ها
* به نحوه ي چينش بلاک ها
* به نرخ انتقال اسمي
* و به نوع بافرينگ

در دو حالت پردازش انبوه و بلاکي و در دو نوع بافرينگ ساده و مضاعف ، ارزيابي را انجام مي دهيم.

### پردازش بلاکي:

در پردازش يک بلاک ، بافرينگ مضاعف معنايي ندارد و بافرينگ فقط ساده مي باشد.

يادآوري : نرخ انتقال واقعي يعني چه ميزان داده ي خالص در واحد زمان منتقل مي شود.

نکته ي اصلي اين است که فرض کنيم زمان پردازش از زمان چرخش يک دور ، کمتر است.

(يعني فرض کنيم: )

مي دانيم که زمان اسمي براي خواندن يک بلاک برابر است با :

اما زمان واقعي صرف شده براي خواندن يک بلاک ، برابر است با:

هم براي بافرينگ ساده و هم براي بافرينگ مضاعف در پردازش بلاکي.

### پردازش انبوه:

در پردازش انبوه براي ما بافرينگ مهم مي باشد ، پس داريم:

* بافرينگ ساده

در اين حالت در هر دور از چرخش ديسک ، فقط مي توان يک بلاک را خواند.

* بافرينگ مضاعف

دو حالت براي بافرينگ مضاعف وجود دارد:

الف) داراي شرط کارايي

ب)فاقد شرط کارايي

**توضيح** **فاقد شرط کارايي**: از لحاظ معنايي يعني تا ما از روي گپ رد شويم ، CPU فرصت نمي کند تا پردازش بلاک قبلي را تمام کند تا ما بتوانيم Data را از روي بلاک بعدي بخوانيم.

و از لحاظ فرمولي يعني:

**توضيح** **داراي شرط کارايي** : يعني

نرخ انتقال واقعي حتي از نصف نرخ انتقال اسمي هم کمتر مي باشد.

يک راه حل عملي:

فرض کنيد که رکورد ها ، حاوي داده ي خالص مي باشند و به طور متوسط اندازه ي آنها R بايت است و در هر استوانه K شيار و در هر شيار n رکورد موجود مي باشد.

داده ي واقعاً خوانده شده از يک استوانده برابر است با:

در يک مسأله ي مشخص با در نظر گرفتن زمان هاي استوانه جويي(S) ، زمان درنگ دوراني(r) مي توان نرخ انتقال واقعي را محاسبه کرد.

**نتيجه** : اگر رسانه ي ما ديسک باشد، S و r در نرخ انتقال واقعي مؤثر است. پس بايد براي بهبود نرخ انتقال واقعي اين دو مؤلفه را کاهش دهيم.

تکنيک هاي کاهش زمان درنگ دوراني (3 تکنيک دارد)

تکنيک هاي کاهش زمان استوانه جويي (4 تکنيک دارد)

# تکنيک هاي کاهش زمان درنگ دوراني:

1. ايجاد BOT-ها به صورت غير متعارف(Begin Of Track)
2. چيدن بلاک ها به صورت n در مياني(يک درميان، دو در ميان و ...)
3. پراکنده خواني بلاک ها

**توضيح** 1 : چون ديسک در حال چرخش مي باشد و ما در حال خواندن داده در شيار مي باشيم ، مثلاً FS به هد مي گويد که برو شيار 1 داده را بخوان.

بايد BOT را کمي جلوتر قرار دهيم تا هد کارش را راحت تر باشد.

متعارف

غیر متعارف

**توضيح** 2 **:** زماني مي توان از اين تکنيک استفاده کرد که شرايط زير برقرار باشد:

محدوديت بافر داريم.(يعني بافرينگ ساده است.)

خواندن سريال بلاک ها مورد نظر باشد.(سريال: يعني پشت سر هم و با ترتيب باشد.)

B2

B3

B4

B5

B1

B6

بلاک ها بر روي رسانه مرتب باشند.

**مثال:** و

B3

B5

B2

B4

B1

B6

2 در میانی

B4

B2

B5

B3

B1

B6

1 در میانی

تمرين: بار اول فرض کنيد و بار دوم نيز فرض کنيد . براي دو حالت زير نحوه ي چيدن بلاک ها را مشخص کنيد.

**توضيح** 3**:** بايد شرايط فراهم باشد تا از اين تکنيک استفاده کنيم:

محدوديت بافر نداريم.

پي در پي خواندن بلاک ها مورد نظر است.

در اين حالت هر بلاکي که به زير هد مي آيد ، خوانده مي شود. بنابراين در يک دور چرخش ديسک مي توان همه ي بلاک هاي يک شيار را خواند.(فرض بر اين است که در بافر محدوديتي وجود ندارد.)

# تکنيک هاي کاهش زمان استوانه جويي(Seek Time)

1. استفاده از ديسک با بازوي ثابت (يا ديسک هاي سريع تر، تکنيک سخت افزاري)
2. استفاده از فايل توزيع شده ي چند پکي(تکنيک نرم افزاري)
3. اعمال ملاحظات خاص لوکاليتي (يعني درجه ي لوکاليتي را افزايش دهيم)

که به روش هاي زير امکان پذير مي شود:

* ذخيره سازي داده ها با بسامد بالاي دستيابي ، روي ديسک هاي سريع تر(بسامد:تعداد رخ داد در واحد زمان مي شود.تکنيک سخت افزاري)
* ذخيره سازي داده ها با بسامد بالاي دستيابي در استوانه هاي مياني ديسک(تکنيک نرم افزاري)

1. استفاده از الگوريتم هاي کاراتر براي حرکت دادن هِد ديسک(تکنيک نرم افزاري)

* الگوريتم FCFS(First Come First Service) : صف. ساده ترين الگوريتم براي پيمايش استوانه و بر اساس رديف درخواستي عمل مي کند .
* الگوريتم SSTF(Shortest Seek Time First):يعني در هر لحظه دستوري را انجام مي دهد که کمترين جابجايي را براي پاسخ گويي نياز داشته باشد.
* الگوريتم Scan : يعني در جهتي حرکت مي کنيم که مي خواهيم بيشترين درخواست را انجام دهيم.

#### فصل چهارم:

# بررسي تحليلي و پارامتريک مبنايي ساختارهاي فايلينگ :

اهداف اصلي استفاده از هر سيستم فايلي ، عبارت است از:

* افزايش سرعت عمليات ها.
* صرفه جويي در مصرف حافظه.
* دلايلي ديگر ...

نکته: اين دو دليل اصلي معمولاً با هم تضاد دارند و بايد بين آنها تعادل و صلح ايجاد کرد.

هر سيستم فايل ، فايل ها را بر اساس ساختارهايي ايجاد مي کند.

ساختار هاي مختلف ، در دو عمل اساسي، يعني بازيابي و ذخيره سازي داده ها با هم تفاوت دارند و رسيدن به هدف هاي سيستم فايل،مستقيماً بستگي به ساختاري که سيستم فايل ايجاد مي کند دارد.

# ساختارهاي مبنايي فايلينگ

1. فايل با ساختار پايل : فايلي است فاقد نظم ساختاري و غالب رکورد ها غير ثابت غير مکاني هستند.
2. فايل با ساختار ترتيبي : فايلي است که رکورد هاي آن ثابت مکاني اند و فايل بر اساس يکي از صفات خاصه مرتب مي باشد.
3. فايل شاخص دار(Index-دار): فايلي است که هر موضوع و هر مطلب براي خود يک Index جداگانه دارد.
4. فايل چند شاخصي(Multi Index) : که به نوعي حالت عمومي تر 3 مي باشد.
5. فايل با ساختار مستقيم(Direct File): مفهوم Hash.
6. فايل با ساختار حلقوي : اين ساختار براي پايگاه داده اي شبکه اي مناسب مي باشد.

نکته: بعضي از بزرگان عقيده دارند که ساختار هاي 4 و 6 مبنايي نمي باشند.

مطالعه ي ساختار ها در اين فصل به دو طريق انجام مي شود:

1.تفصيلي(مفصل) : براي شناخت کلي ساختار .

2.تحليلي : در راستاي شناخت تفصيلي مي باشد و به منظور ارزيابي ساختار بر اساس يکسري پارامتر ها استفاده مي شود.

معمولاً معيار ها و ضوابط ارزيابي به دو عامل بر مي گردد:

* زمان انجام عمليات بر روي فايل (عمليات: نوشتن و خواندن)
* حافظه ي مصرفي به ازاي هر رکورد.

عمليات اصلي بر روي فايل به ترتيب زير مي باشد:

* حافظه ي مصرفي
* زمان عمليات

1. زمان Fetch کردن تک رکورد()
2. زمان به دست آوردن رکورد بعدي()
3. زمان درج يک رکورد()
4. زمان به هنگام رساني يک رکورد()
5. زمان خواندن کل فايل()
6. زمان سازماندهي مجدد()

پس در کل 7 پارامتر مورد بررسي قرار مي گيرد: 6 پارامتر زماني + 1 پارامتر ظرفيتي

عمليات درج، به هنگام سازي و سازماندهي مجدد را عمليات تغيير دهنده ي محيط فيزيکي ذخيره سازي مي گويند.

انجام عمليات شش گانه، در نهايت منجر به انجام سه عمل اساسي در محيط فيزيکي ذخيره سازي، مي شود که عبارتند از:

1.مکان يابي

2.خواندن فيزيکي

3.نوشتن فيزيکي

# حافظه ي مصرف شده به ازاي هر رکورد:

عبارتند از:

1. بخش پيشوندي رکورد
2. بخش داده اي رکورد
3. انواع حافظه هاي تلف شده به ازاي يک رکورد
4. متراکم يا نا متراکم بودن فايل
5. پديده ي افزونگي

نکته:انواع حافظه هاي تلف شده سامل مواردي باشد که تا به حال ديده ايم و همچنين ساير مواردي که هنوز بررسي نکرده ايم.(مثل شاخص)

تعريف فايل متراکم: فايلي است که تمام صفات خاصه در تمامي رکورد ها داراي مقدار باشند.(يعني پديده ي none-data نداريم.)

توجه:در زماني که فايل داراي ساختار ثابت مکاني باشد ، احتمال متراکم بودن فايل کم است.(چون احتمال رخ دادن پديده ي none-data بالاست.)

تعريف فايل نامتراکم : فايلي است که متراکم نباشد.(پديده ي none-data رخ مي دهد و در جاي جاي فايل حافظه ي تلف شده داريم.)

**تعريف پديده ي افزونگي** :فايلي را داراي افزونگي گوييم که مقادير بعضي از صفات خاصه، بيش از يک بار در محيط فيزيکي، ذخيره سازي شده باشند.(افزونگي: يعني يک مقدار را دوبار ذخيره کرده ايم.)

افزونگي بر دو نوع مي باشد: 1.طبيعي(يا ذاتي) 2.تکنيکي

**توضيح افزونگي طبيعي**: ذات و ماهيت داده هاي محيط به گونه اي باشند که ما را مجبور مي کند تا بعضي از مقادير را چند بار ذخيره کنيم.

مثال:

|  |  |
| --- | --- |
| Cr # | Student num |
| Fr100,Math200,ISR301 | 7927 |
| Fr100 | 7930 |
| Fr100,Math200 | 8035 |
| Fr100,ISR301 | 7820 |
| Fr100,ISR302 | 7931 |

با روش هاي فشرده سازي اطلاعات مي توان اين مشکل را حل کرد ، بخصوص وقتي که پديده ي فقره اطلاع تکرار شونده داريم و افزونگي طبيعي هم داريم و مجموع عناصر هم محدود است.

البته مي توان از تکنيک فشرده سازي ماتريس بيتي نيز استفاده کرد:

|  |  |
| --- | --- |
| Fr100 Math200 ISR301 ISR302 | Student num |
| 1 1 1 0 | 7927 |
| 1 0 0 0 | 7930 |
| 1 1 0 0 | 8035 |
| 1 0 1 0 | 7820 |
| 1 0 0 1 | 7931 |

توضيح افزونگي تکنيکي: عبارت است از تکرار بعضي از مقادير صفت خاصه در محيط فيزيکي ذخيره سازي ،که به خاطر استفاده از يک روش کاراتر براي دسترسي به وجود آمده.(مثلاً براي ايجاد شاخص)

|  |  |
| --- | --- |
|  | 7000 |
|  | 7020 |
|  | 7835 |
|  | 7926 |
|  | 7927 |
|  | 8001 |
|  | 8030 |
|  | 9002 |

|  |  |
| --- | --- |
|  | 7927 |
|  | 8030 |
|  | 7835 |
|  | 7926 |
|  | 7020 |
|  | 7000 |
|  | 8001 |
|  | 9002 |

در اين حالت زمان جستجو کم مي شود ، اما شاخص ما هم بايد ذخيره شود.(حافظه ي تلف شده)

# زمان ها

1. توضيح واکشي تک رکورد : عملي است محتوايي، يعني کاربر يک آرگومان جستجو به صورت به ما مي دهد و رکوردي را طلب مي کند .

لازمه ي جستجو کردن در فايل، دسترسي به بلاک حاوي رکورد مورد نظر و خواندن آن مي باشد.

اگر بخواهيم به رکورد ها از طريق کليد اصلي دسترسي داشته باشيم، دو روش وجود دارد:

دسترسي ترتيبي:در اين روش، سيستم بايد تعدادي بلاک را به صورت پي در پي بخواند تا به بلاک حاوي رکورد مورد نظر برسد.(جستجوي خطي)

دسترسي تصادفي:در اين روش، سيستم مي تواند آدرس بلاک حاوي رکورد مورد نظر را محاسبه کند. و پس از محاسبه ي آدرس، مي تواند بلاک حاوي رکورد مورد نظر را به طور مستقيم بخواند.

يادآوري: در ارزيابي زمان عمليات ششگانه روي فايل، هر گاه بلاکي با شيوه ي دسترسي ترتيبي خوانده شود، زمان خواندن مي باشد.

و اگر بلاکي با شيوه ي دسترسي تصادفي خوانده شود، زمان خواندن مي باشد.

چه چيزهايي آدرس را به ما مي دهند:

* با شاخص (Index) به دست آيد.
* با ارتباط خاص بين رکورد ها به دست آيد.
* با محاسبه به دست آيد.

که پيمايش مي تواند داراي حالات زير باشد:

* خطي
* جستجوي دودويي
* جستجوي بلاکي با پرش

مثال:فرض کنيم فايلي داريم گه:

الف)فاقد هر گونه نظم باشد ، آنگاه زمان واکشي تک رکورد چه قدر است؟

ب)اگر فايل بر اساس صفت خاصه ي ii مرتب شده باشد و ما جستجو را بر اساس همان صفت خاصه انجام دهيم، زمان واکشي تک رکورد چه قدر مي شود؟

2.توضيح بدست آوردن رکورد بعدي() : عملي است ساختاري و نه محتوايي.(يعني کاربر آرگومان جستجو را نمي دهد و رکورد را تلف مي کند.)

با توجه به نظم ساختاري بايد رکورد بعدي به کاربر داده شود.

عمل getNext در بعضي از ساختارها ناممکن است .(در ساختار هايي که فاقد نظم ساختاري باشند.)

براي انجام عمل getNext وجود نظم لازم است ولي کافي نيست.

نکته: هر چه لوکاليتي قوي تر باشد ، عمل getNext سريع تر انجام مي شود.(البته getNext در همه جا امکان پذير نيست.)

مثال: فايل دانشجو-درس را در نظر مي گيريم. اگر اين فايل بر اساس نظم صعودي مقادير شماره ي دانشجويي مرتب شده باشد، و مقدار شماره ي دانشجوي x را واکشي کرده باشيم،رکورد بعدي، رکورد دانشجويي باشد که مقدار شماره دانشجويي اش بلافاصله از مقدار x بزرگتر است.

3.توضيح درج تک رکورد : منظور از اين عمل، درج يک رکورد جديد، بعد از لود اوليه ي فايل در حافظه مي باشد.

در درج تک رکورد ، دو حالت ممکن است پيش آيد:

الف) فايل فاقد هرگونه نظم ساختاري مي باشد و بلاک خاصي را براي درج رکورد، در نظر نداريم.در اين حالت ساده ترين جا براي درج انتهاي فايل مي باشد و زمان آن مي شود:

ب ) فايل داراي نظم باشد. در اين حالت رکورد بايد در بلاک خاصي درج شود.(به اين نقطه اصطلاحاً نقطه ي منطقي درج مي گوييم.)

ريز عمليات درج:

1. خواندن نقطه ي منطقي درج(در حالتي که نظم نداريم ، نقطه ي منطقي درج در انتهاي فايل مي باشد.)
2. کار در بافر
3. بازنويسي بلاک
4. در برخي از ساختار ها و در صورت لزوم اصطلاح استراتژي دستيابي

### 4.توضيح به هنگام سازي تک رکورد():

يعني مقدار يک يا چند صفت خاصه در يک رکورد مشخص تغيير پيدا مي کند.

با توجه به ساختار ، ممکن است اين به هنگام سازي باعث جا به جا شدن رکورد شود.

مثال:

1

7

5

4

3

6

1

7

5

4

3

6

اصل کلي : رکورد به هنگام در آمدني بايد واکشي شود.(پس زمان اين عمل ، به زمان Fetch وابستگي مستقيم دارد.)

Update يک رکورد در حالت کلي به دو صورت مي تواند انجام شود: 1. درجا 2.برون از جا

توضيح Update-ِ درجا (InPlace): يعني رکوردِ Update شده ، دوباره در همان محل قبلي اش نوشته مي شود.(عمل Replace)

توضيح Update-ِ برون از جا (OutPlace): يعني رکوردِ Update شده ، در جاي ديگري غير از جاي قبلي ذخيره شود.

### دلايل Update برون از جا :

1. نظم ساختاري
2. افزايش طول بر اساس Update و نگنجيدن در جاي قبلي(در صورت متغيير بودن طول رکورد)
3. دليل تکنيکي و ساده شدن پياده سازي(مثلاً در ساختار دوم يعني ترتيبي)

ريز عمليات Update-ِ درجا:

1. Fetch کردن رکوري که بايد Update شود.
2. کار در بافر(زمان= 0)
3. بازنويسي()
4. در برخي از ساختارها و در صورت لزوم اصلاح استراتژي دستيابي

بازنويسي يعني آن داده اي که در بافر هست را در هارد بنويسيم.(نه پاک کردن و دوباره نوشتن)

توجه شود که در داخل محاسبه شده است.

بازنويسي در حالتي رخ مي دهد که يا ديسک با بازوي ثابت باشد يا هد تغيير مکان نداده باشد.

### ريز عمليات Update-ِ برون از جا:

1. Fetch کردن رکوردي که بايد Update شود.
2. کار در بافر
3. بازنويسي رکورد قبلي
4. درج رکورد جديد
5. اصلاح استراتژي دستيابي(در برخي از ساختار ها و در صورت لزوم)

### 5. .توضيح خواندن کل فايل()::

گاه بنا به دلايلي لازم است که کل فايل را بخوانيم. و اين دلايل به دو دسته تقسيم مي شوند:

1. بنا به درخواست کاربر(مثال:ليست گيري ها ، حقوق کامندان،Copy)
2. بنا به نياز سيستمي(مثال: سازمان دهي مجدد)

در اين عمل، تمام بلاک هاي فايل بايد خوانده شوند و وجود حافظه هاي تلف شده اي که به هر دليل به وجود مي آيند ، سبب افزايش زمان اين عمل مي شود.

خواندن کل فايل به دو صورت مي تواند انجام شود:

1. به صورت پي در پي():يعني بلاک به بلاک از آغاز فايل تا انتهاي آن بخوانيم.
2. به صورت سريال():در خواندن سريال بر اساس نظم ساختاري رکورد ها خوانده مي شوند.

نکته: خواندن سريال، مجموعه اي از اعمال GetNext مي باشد.

هر چه لوکاليتي قوي تر باشد ، خواندن سريال سريع تر است( اما در برخي از ساختار ها خواندن سريال ناممکن است.)

نکته: به طور کلي ناممکن دو جنبه دارد: 1. فيزيکي 2.منطقي

ممکن است که منطقاً انجام عمل ممکن نباشد ، اما فيزيکي ممکن باشد.(ناممکن فيزيکي)

ممکن است که منطقاً انجام عمل ممکن نباشد ، و فيزيکي هم ممکن نباشد.(ناممکن منطقي)

نکته: هر جا GetNext ناممکن باشد ، خواندن سريال هم ناممکن مي باشد.

### 6.توضيح سازماندهي مجدد :

هر فايلي پس از لود اوليه در حافظه، دوره ي حياتي دارد که طي آن عملياتي را اعم از بازيابي يا ذخيره سازي تحمل مي کند.

در اثر تغييراتي که به تدريج در فايل ايجاد مي شود، ممکن است به تدريج فايل کارايي اوليه اش را از دست بدهد و براي همين بايد سازمان دهي انجام داد.

### دلايل کاهش تدريجي کارايي فايل به طور کلي عبارت است از:

* از بين رفتن نظم(يا وضع) اوليه
* به وجود آمدن فضاي تلف شده در فايل
* به وجود آمدن وضعيت نامطلوب در استراتژي دستيابي(Access Strategy)(که مفهوم آن را در فايل هاي شاخص دار خواهيم ديد.)

### دلايل سازمان دهي مجدد:

1. احياي نظم ساختاري اوليه
2. استرداد(پس گرفتن) حافظه هاي تلف شده

* ناشي از Update برون از جا و زدن مارکر D به رکورد قبلي
* ناشي از حذف يک رکورد

1. اصلاح ويا ساخت استراتژي دستيابي

نکته: بر اثر انجام عمليات بر روي فايل ، به تدريج ساختار کارايي خود را از دست مي دهد و بايد به طور مجدد سازماندهي شود.

### ريز عمليات سازماندهي مجدد:

1. خواندن کل فايل
2. کار در بافر(براي بلاک بندي مجدد فايل جديد)
3. درج رکورد هاي فعال(يعني اون هايي که مارکر D دارند ، نمي نويسيم.)
4. اصلاح/ ساخت استراتژي دستيابي

### زمان بازنويسي بلاک

زمان عمل نوشتن بلاک را که همان انتقال محتواي بافر به ديسک است، در يک ارزيابي کلي، مي توان مساوي عمل خواندن بلاک در نظر گرفت.(يعني:)

و يا زمان نوشتن بلاک ها به طور انبوه برابر است با:

و زمان نوشتن يک رکورد برابر است با:

اما زمان بازنويسي يک بلاک، چون در مکان سابقش انجام مي شود، مي توان تحت شرايطي زمان آن را برابر با 2r در نظر گرفت.

# ارزيابي پارامترها در فايل با ساختارهاي مختلف

# فايل با ساختار پايل :

ساختاري مي باشد که هيچ گونه نظم ساختاري اي ندارد، يعني:

* غالب رکورد ها، غير ثابت غير مکاني هستند.(يعني طول رکورد ها متغيير مي باشد.)
* هيچ گونه نظمي در ساختار قابل تصور نيست(يعني هيچ ارتباطي بين مقادير صفت خاصه و ترتيب آنها وجود ندارد.)

به شکل زير توجه کنيد:

### موارد استفاده ي فايل با ساختار پايل به ترتيب زير مي باشد:

1. در محيط هايي که در آن ها داده ها نظم پذير نباشند و پردازشي از قبل روي داده ها انجام نشده باشد. يا فايل اساساً براي بايگاني ايجاد شده باشد.
2. در محيط هايي با داده هاي استراتژيک وقتي که ايمني داده ها مورد نظر باشد، بي نظمي مي تواند ايمني فايل را افزايش دهد.
3. براي ايجاد ساختار هاي مؤثرتر(يعني ابتدا فايل با ساختار پايل ايجاد مي شود و سپس فايل با ساختار اصلي ايجاد مي گردد.)
4. جنبه ي آموزشي دارد.(جهت درک بهتر ساختارهاي ديگر)

### ارزيابي ساختارها :

|  |  |
| --- | --- |
| قرار داد | |
| متوسط حافظه ي مصرفي براي ذخيره سازي نام صفت خاصه |  |
| متوسط حافظه ي مصرفي براي ذخيره سازي مقدار صفت خاصه |  |
| تعداد صفت خاصه(حداکثر) |  |
| ميانگين تعداد صفت خاصه |  |
| تعداد رکورد هاي درج شده دقيقاً قبل از عمل سازماندهي مجدد |  |
| تعداد رکورد هاي درج شده در طول حيات فايل از سازمان دهي مجدد |  |
| تعداد رکورد هاي حذفي |  |
| اندازه بلاک |  |
| تعداد رکورد |  |
| فاکتور بلاک بندي |  |
| تعداد بلاک |  |
| زمان پردازش يک بلاک |  |

بديهي است که

مثال:

**نام**:مهرداد

**رشته**:کامپیوتر

**شماره**:12

**نام**: حسن

**نام خانوادگی**:حسینی

**تولد**:50

**آدرس**:آدرس 1

**تلفن**:2222

**نام** **خانوادگی**: زهرایی

## ارزيابي ساختار پايل :

### الف-1) متوسط حافظه ي مصرفي براي تک رکورد (R=?)

چون قالب رکوردها غير ثابت غير مکاني است هم نام و هم مقدار صفت خاصه بايد ذخيره شود براي ذخيره سازي هر صفت خاصه A+V+2 بايت حافظه مصرف ميشود .به طور متوسط تا صفت خاصه داريم پس :

### ب-1)زمان واکشي تک رکورد :

طبيعي است که فايل فاقد هر گونه نظم است و بايد به طور متوسط نصف رکوردهاي فايل خوانده شود و بايد پيمايش خطي انجام دهيم

ملاحظه مي شود که زمان فچ کردن تک رکورد در اين ساختار بسيار زياد است، مخصوصاً زماني که تعداد رکورد ها زياد باشد.

سؤال: "آيا مي توان زمان خواندن تک رکورد را کاهش داد؟"

جواب: بله، براي کاهش زمان فچ کردن تک رکورد ، از تکنيک request baching استفاده مي کنيم. در اين تکنيک درخواست هاي همگن(درخواست هايي که صفت جستجو در آنها يکسان باشد) جمع شده و در يک بار خواندن کل فايل به آنها پاسخ داده مي شود.

فرض کنيم که يک دسته درخواست همگن شامل l درخواست باشد:

### ب- 2) زمان بدست آوردن رکورد بعدي :

گفته شد فايل پايل فاقد هرگونه نظم ساختاري است و getnext عملي است محتوايي و نه ساختاري پس getnext بي معني است . اگر کاربر خودش آرگومان جستجو را بدهد در اين صورت عمل همان عمل fetch ميشود و با getnext فرق دارد .

رکورد بعدي در اين ساختار مفهومي ندارد، چون هيچ گونه ارتباط ساختاري بين رکورد فعلي و بعدي آن برقرار نمي باشد.

اين عمل ،در ساختار پايل به درد نخور مي باشد.

### ب-3)زمان درج کردن تک رکورد()

چون فايل فاقد هر گونه نظم است، پس رکورد جديد به انتهاي فايل اضافه مي شود.

ريز عمليات درج کردن در ساختار پايل به ترتيب زير مي باشد:

آخرين بلاک خوانده مي شود(براي اين که آدرس آخر فايل را داشته باشيم.)

کار در بافر(يعني Deblocking ، درج رکورد ، Blocking)

بازنويسي بلاک(بازنويسي ، يعني آدرس جاي نوستن براي ما معلوم مي باشد و در همان جا مي نويسيم.)

پس زمان کل درج کردن در ساختار پايل به صورت زير محاسبه مي شود:

یعنی بهترین زمان درج در تمام ساختار ها.

اين عمل در ساختار پايل به درد به خور مي باشد.

### ب-4) به هنگام سازي تک رکورد()

قرارداد : در اين ساختار، به هنگام سازي را برون از جا انجام مي دهيم.(چون احتمال جا نشدن رکورد در مکان قبلي زياد مي باشد.)

ريز عمليات به هنگام سازي تک رکورد در فايل با ساختار پايل:

Fetch کردن رکوردي که بايد Update شود.

کار در بافر

بازنويسي رکورد قبلي

درج رکورد جديد

پس زمان کل Update کردن در فايل با اين ساختار به صورت زير محاسبه مي شود:

ملاحظه مي شود که هم مانند زمان زيادي دارد.

عمل حذف حالت خاصي از عمل به هنگام سازي مي باشد، با اين تفاوت که درج رکورد جديد در آن انجام نمي شود و زمان آن برابر است با:

تمرين: زمان به هنگام سازي درجا چه قدر است؟

### ب-5) زمان خواندن کل فايل()

در دو حالت بررسي مي کنيم:

خواندن پي در پي :يعني از BOF تا EOF بلاک به بلاک خوانده مي شود و خواندن به صورت دسته اي انجام مي شود.

خواندن سريال : چون فايل فاقد هر گونه نظم است ، پس خواندن سريال امکان ندارد.(يادآوري اينکه خواندن سريال خود جزوي از اعمال getNext بود و ديديم که عمل getNext در اين ساختار امکان پذير نبود.)

نکته: با توجه به اينکه اکثر فايل ها غير ثابت غير مکاني اند، ممکن است بعضي از رکورد ها آن صفت خاصه ي مورد نظر ما را نداشته باشند.

### ب-6) زمان سازمان دهي مجدد فايل()

اين فايل از نظر احياي نظم ساختاري اوليه، نيازي به سازماندهي مجدد ندارد.

ولي به خاطر عمل به هنگام سازي و همچنين عمل حذف،ممکن است در جاي جاي فايل حافظه ي تلف شده پديد آيد.

دلايل سازماندهي مجدد فايل

احياي نظم ساختاري اوليه(🗷)

پس گرفتن حافظه هاي تلف شده(🗹)

اصلاح استراتژي دستيابي(در صورت لزوم)(🗷)

لازم است براي پس گرفتن حافظه هاي تلف شده که به خاطر مارکر d ( به علت حذف رکورد يا به علت به هنگام سازي برون از جا) به وجود آمده اند، سازماندهي مجدد انجام مي شود.

ريز عمليات سازمان دهي مجدد:

1. خواندن کل فايل
2. کار در بافر(به منظو خارج کردن رکورد هاي غير فعال داراي مارکر d و بلاک بندي مجدد رکورد هاي فعال)
3. بازنويسي رکورد هاي فعال

پس زمان کل سازمان دهي مجدد به صورت زير محاسبه مي شود:

# ارزيابي فايل با ساختار ترتيبي(کليدي)

فايل ترتيبي دو نوع مي باشد:

* 1. فايل ترتيبي کليدي :در اين نوع فايل، رکورد ها بر اساس کليدشان ذخيره مي شوند.
  2. فايل ترتيبي زماني : در اين نوع فايل، رکورد ها به ترتيب ورود به سيستم ذخيره مي شوند و نوع خاصي از فايل پايل مي باشد که طول رکورد ها در آن ثابت است.

در ادامه با درباره ي فايل ترتيبي کليدي صحبت خواهيم کرد نه ترتيبي زماني.

اگر در ساختار فايل پايل تغيير ايجاد کنيم، فايل ترتيبي به دست مي آيد که در آن:

غالب رکورد ها، ثابت مکاني اند.

رکورد ها بر اساس مقادير يکي از صفات خاصه ، نظم داده مي شوند(که به اين صفات خاصه، کليد گفته مي شود.)

مزاياي ناشي از اصلاحات(بهبود ها):

به دليل اينکه غالب رکورد ها ثابت مکاني اند، پردازش ساده تر مي باشد.(و پردازش ساده تر يعني سرعت انجام برخي از اعمال بيشتر است.)

به دليل وجود نظم براي جستجو، امکان اعمال الگوريتم هاي جستجو وجود دارد.

به دليل وجود نظم ، امکان عمل getNext و همچنين امکان خواندن سريال فايل به وجود آمد.

صرفه جويي در مصرف حافظه به وجود آمده(يا عدم وجود افزونگي ناشي از تکرار نام صفات خاصه)

### معايب ناشي از اصلاحات(بهبود ها):

اتلاف حافظه(به دليل اينکه قالب رکورد ها ثابت مکاني اند، و امکان رخ داد پديده ي none-data وجود دارد. يعني حافظه هاي تلف شده در جاي جاي فايل)

### کاهش انعطاف پذيري ساختار

که اين کاهش از دو نظر مطرح مي شود:

الف) از نظر طول رکورد،يعني ساختار اجازه ي تغيير طول رکورد را نمي دهد و اگر قرار باشد طول بعضي نمونه رکورد ها تغيير پيدا کند، فايل بايد دوباره طراحي و ايجاد شود.

ب) از نظر عمليات تغيير دهنده(يعني درج، حذف، به هنگام سازي)

پديده ي عدم تقارن

در اين ساختار صفات کليد با صفات خاصه ي غير کليد فرق دارند.

کليد نقش محوري بازي مي کند و ويژگي خاصي نسبت به بقيه دارد. حتي اگر ساختار بر اساس ويژگي هاي بيان شده نتواند عمل کند.اما در اينجا برابري بين صفات خاصه وجود ندارد.

به دليل وجود نظم، انجام برخي عمليات زمان گير تر و الگوريتم ها پيچيده تر.

توصيف فايل با ساختار ترتيبي: از لحاظ تئوري ، وقتي که تعداد رکورد هاي فايل کم است مي توان از اين ساختار استفاده کرد.

F

D

B

A

اين ساختار داراي دو ناحيه مي باشد:

F

D

C

B

A

ناحيه ي اصلي

TLF(Trancetion Long File)

توضيح ناحيه ي اصلي: اين ناحيه فقط خواندني(Read-Only) مي باشد.(يعني Update نمي شود.)

ناحيه ي اصلي بر اساس کليد مرتب شده است.(Ordered)

توضيح قسمت TLF: تغييرات در فايل ، در اين قسمت ذخيره مي شود تا زماني که سازمان دهي مجدد انجام شود.

ظرفيت TLF را طراح فايل با توجه به حجم عمليات تغيير دهنده و نياز داشتن کاربر به خروجي هاي مرتب ، تعيين مي کند.

در زمان سازمان دهي مجدد، تغييرات بر روي ناحيه ي اصلي اعمال مي شود. بنابراين وجود TLF باعث مي شود تا نظم ساختاري از بين برود.

TLF

ناحیه ی اصلی

زماني که ناحيه ي اصلي Read-Only باشد.

زماني که ناحيه ي اصلي Read-Only نباشد: در اين حالت فايل فقط داراي يک حالت اصلي مي باشد که مي توان تغييرات را بر روي آن اعمال کرد.

کاربردهاي فايل با ساختار ترتيبي:

زماني که Fetch کردن سريع تک رکورد ، مورد نظر نباشد.

خواندن سريال فايل به صورت پريوديک

براي ايجاد ساختار هاي بهتر و کاراتر

## ارزيابي فايل با ساختار ترتيبي:

### الف-1) متوسط حافظه ي مصرفي()

براي محاسبه داريم:

برای ذخیره ی n رکورد

سهم هر رکورد از ناحیه ی TLF

### ب-1) زمان واکشي تک رکورد()

دو حالت ممکن است اتفاق بيافتد:

* اگر کاربر بر اساس کليد جستجو کند.
* اگر کاربر براساس کليد جستجو نکند و جستجو را بر اساس صفت خاصه ي ديگري غير از کليد انجام دهد.

در حالت اول براي Fetch کردن رکورد بايد جستجوي خارجي(External Search) انجام داد و زمان بستگي دارد به الگوريتم جستجو.

يکي از الگوريتم هاي رايج براي جستجو، الگوريتم جستجوي دودويي مي باشد که در اين جا بايد در يک محيط منظم خارجي(آرايه اي خارجي) اجرا شود.

توجه داريم که براي اعمال اين الگوريتم، فايل علاوه بر مرتب بودن ، بايد در فضاي پيوسته ذخيره شده باشد. اما اگر فايل ناپيوسته يا بند بند شده باشد، نمي توان هر بار آدرس بلاک وسطي را به دست آورد و در نتيجه الگوريتم جستجوي دودويي را نمي توان به کار برد.

### جستجوي دودويي

جستجوي دودويي، در يک محيط منظم خارجي بايد در دو سطح انجام گيرد:

در سطح اول، جستجويي در فايل داريم تا بلاک مورد نظر پيدا شود.(البته واحد جستجو را در اين سطح بلاک مي گيريم.) که براي اين کار بايد بلاک هاي فايل خوانده شوند.

در سطح دوم، براي هر بلاکي که به بافر آورده مي شود، يک جستجوي درون بلاکي داريم.

اين دو سطح هر در ارزيابي زمان دخالت دارند:

چون واحد جستجو بلاک مي باشد، تعداد دفعات مراجعه ي سيستم فايل به فايل خواهد بود.

در حالت دوم ، فايل عملاً تبديل مي شود به حالت خاصي از پايل و با جستجوي خطي در ناحيه ي اصلي خواهيم داشت:

حال اگر زمان جستجو در TLF را هم دخالت دهيم، ارزيابي دقيقتر مي شود.

توجه: حتي پس از يافتن نسخه اي از رکورد مورد نظر در ناحيه ي اصلي، باز هم بايد TLF به تمامي بررسي شود، چون ممکن است بعد از لود اوليه ، رکورد مورد نظر تغيير کرده و يا حذف شده باشد.

اگر فرض کنيم که ظرفيت TLF به نعداد رکورد باشد و در لحظه ي Fetch ، تا رکورد در آن وجود داشته باشد.( به سمت صفر ميل مي کند، و به طور متوسط )

### ب-2)زمان به دست آوردن رکورد بعدي()

اگر رکورد فعلي و رکورد بعدي ، هر دو در ناحيه ي اصلي باشند، به دست آوردن رکورد بعدي امکان پذير مي شود.(پس اگر رکورد بعدي در ناحيه ي اصلي نباشد، مي گوييم امکان پذير نيست.)

پس با خواندن يک بلاک -تا رکورد به دست مي آيد که هر کدام بعدي ديگري مي باشد. پس داريم:

توجه شود که ناحيه ي اصلي مرتب مي باشد. پس اگر رکورد بعدي در ناحيه ي اصلي نباشد، عمل getNext ناممکن مي شود.

**ب-3)زمان درج کردن تک رکورد**()

ناحیه ی اصلی

**TLF**

ساختارش پایل می باشد

**Read-Only**

رکوردي را که مي خواهيم درج کنيم را نمي توانيم به انتهاي فايل اصلي اضافه کنيم ، چون بايد در نقطه ي خاصي درج شود و اين کار در فايل هاي بزرگ زمان گير است.

در عمل ، رکورد در TLF درج مي شود تا در سازمان دهي مجدد ، فايل بر اساس نظم مورد نظر بازآرايي شود

براي ارزيابي زمان درج، دو حالت را در نظر مي گيريم:

1. درج در فايل کوچک
2. درج در حالت کلي

در حالت اول ، مي توان در فايل کوچک ، نقطه ي درج را يافت و رکورد را در آن نقطه درج کرد. که در اين صورت بقيه ي رکورد ها بايد به سمت پايين فايل(EOF) شيفت داده شوند.

### ريز عمليات درج در ناحيه ي اصلي:

1. پيدا کردن نقطه ي منطقي درج
2. شيفت دادن بقيه
3. درج تک رکورد

نکته: در حالتي که فايل کوچک است ، توجه داريم که درج در ناحيه ي اصلي صورت مي گيرد و در اين حالت ناحيه ي اصلي Read-Only نيست.

در حالت دوم ، فايل ترتيبي فايلي منظم مي باشد، بنابراين نقطه ي منطقي درج در آن مطرح مي شود. از طرفي در فايل هاي بزرگ ناحيه ي اصلي Read-Only مي باشد.

پس رکوردي که بايد درج کنيم را در ناحيه ي TLF درج مي کنيم. و زمان اين عمل همان زمان درج در فايل با ساختار پايل است. به علاوه ي .

توضيح در مورد : هر چند که عمل سازمان دهي مجدد ، هم زمان با عمل درج انجام نمي شود، ولي به هر حال اين کار در فايل ترتيبي به خاطر احياي نظم اوليه، انجام مي شود.

با اين هدف که رکوردِ درج شده در TLF ، در محل منطقي خود جاي داده شوند.

سازمان دهي مجدد رکورد ، زماني را مصرف مي کند که اين زمان بايد بين رکورد درج شده در TLF سرشکن مي شود. چون فقط پس از سازماندهي مجدد است که هر رکورد ، در نقطه ي منطقي اش درج مي شود.

### ريز عمليات درج در TLF:

1. خواندن آخرين بلاک از TLF
2. کار در بافر
3. بازنويسي

### ب-4) به هنگام سازي تک رکورد()

اگر مقدار کليد عوض نشود، مي توان رکورد را به صورت درجا Update کرد. ولي معمولاً براي Update نيز از TLF استفاده مي شود.

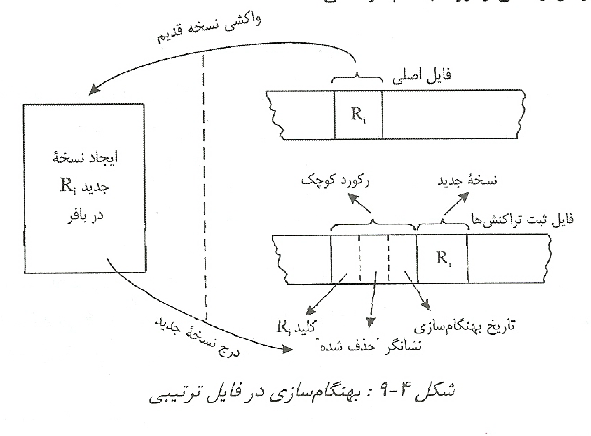
به اين ترتيب که رکوردي که بايد Update شود ، Fetch مي شود در بافر Update مي شود و همراه يک رکورد کوچک ديگر، همزمان در TLF درج مي شود.

رکورد کوچک نشان مي دهد رکورد درج شده ، نسخه ي جديد رکوردي است که نسخه ي قبلي آن همچنان در ناحيه ي اصلي مي باشد.

### ريز عمليات Update تک رکورد:

1. خواندن رکوردي که بايد Update شود.
2. کار در بافر(ساختن رکورد جديد بر رکورد کوچک)
3. درج رکورد جديد

اين نوع Update باعث مي شود تا نظم ساختاري از بين برود ، اما هزينه ي سرشکن شده ي سازمان دهي مجدد در عمل درج رکورد جديد، اعمال شده است.



### ب-5) زمان خواندن کل فايل()

در دو حالت بررسي مي کنيم:

خواندن پي در پي: چون ترتيب ندارد، Master و TLF خوانده مي شوند.

خواندن سريال :خواندن سريال در اين ساختار منطقاً ناممکن است، چون getNext همواره ممکن نيست.

Master

TLF

n تا رکورد دارد

O تا رکورد دارد

يک راه حل عملي براي خواندن سريال: Sort کردن Master و TLF و خواندن پي در پي آنها.

ناحيه ي TLF، Sort مي شود و هم توالي با ناحيه ي اصلي خوانده مي شود.(که به اين نوع خواندن، هم توالي خواندن مي گويند.)

### ب-6) زمان سازمان دهي مجدد فايل()

دلايل سازماندهي مجدد فايل:

احياي نظم ساختاري اوليه(🗹)

پس گرفتن حافظه هاي تلف شده(🗹)

اصلاح استراتژي دستيابي(در صورت لزوم)( 🗷)

### ريز عمليات سازماندهي مجدد:

1. مرتب کردن TLF
2. هم توالي خواندن Master و TLF
3. کار در بافر
4. نوشتن رکورد هاي فعال

شاخص مفهوم آشنايي است.مثلاً شاخص انتهاي کتاب و يا فهرست مطالب آن همه شاخص مي باشند.

شاخص اساساً براي دسترسي سريع به رکورد ها ، به ويژه در فايل هاي بزرگ ايجاد مي شود.

اين شيوه در شيوه هاي دسترسي، جزو شيوه هاي دسترسي تصادفي به حساب مي آيد و دسترسي تصادفي از طريق آدرس رکورد امکان پذير مي شود.

اين ساختار براي پوشش دادن عيب اصلي دو ساختار قبلي آمده ، که عبارت بود از:

زمان Fetch کردن يا جستجو : دو ساختار قبلي داراي زمان جستجوي زيادي بودند که در سيستم هاي عملياتي مورد قبول نمي باشند.

فايل ترتيبي شاخص دار از چند جز تشکيل شده است:

ناحيه ي اصلي(ترتيبي)

ناحيه ي سر ريزي: براي انجام عمليات ذخيره سازي پس از لود اوليه استفاده مي شود.

Pointer:که ناحيه ي اصلي و سرريزي را به هم مرتبط مي کند و شاخص را به ناحيه ي اصلي مرتبط مي کند.

استراتژي دستيابي، موسوم به شاخصIndex

بررسي رفتار فايل با ساختار شاخص دار:

خواندن پي در پي: با توجه به اينکه در خواندن پي در پي نظم اهميت ندارد، ناحيه ي اصلي و ناحيه ي سرريزي باهم خوانده مي شوند.(در خواندن پي در پي شاخص مورد استفاده قرار نمي گيرد.)

خواندن منطقي پي در پي: در اين حالت اول ناحيه ي اصلي را مي خوانيم و بعد ناحيه ي سرريزي را مي خوانيم.

خواندن سريال: در اين حالت از ناحيه ي اصلي که مرتب است شروع مي کنيم و مي خوانيم تا به Pointer-ي به ناحيه ي سرريزي برسيم. سپس به ناحيه ي سرريزي رفته و زنجيره ي رکورد هاي سرريزي را دنبال مي کنيم. پس از اتمام زنجيره به ناحيه ي اصلي بر مي گرديم و عمليات خواندن ناحيه ي اصلي را تا رسيدن Pointer-ي به ناحيه ي سرريزي ادامه مي دهيم. و بعد دوباره رفتن به ناحيه ي سرريزي و ...

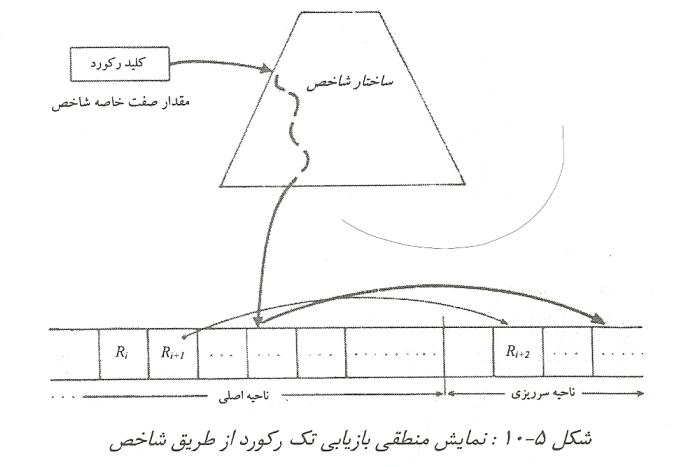
بنابراين ملاحظه مي شود که در خواندن سريال ، سيستم بين ناحيه ي اصلي و سرريزي ترافيک دارد.

در نتيجه براي اين که ساختار کاراتر باشد، بايد رکورد هاي سرريزي با لوکاليتي قوي تر درج شوند.

نکته: شاخص در خواندن سريال هم مورد استفاده قرار نمي گيرد.

واکشي تک رکورد:

اگر کاربر کليد را بدهد، از شاخص استفاده مي شود. در اين حالت در شاخص غواصي کرده تا به Pointer-ي به ناحيه ي اصلي مي رسيم.(رسيدن از سر شاخص به برگ، يعني رفتن به ناحيه ي اصلي و در صورت پيدا نکردن رکورد، رفتن به ناحيه ي سرريزي توسط Pointer-ي از ناحيه ي اصلي سرريزي.)

کليد، صفت خاصه اي مي باشد که فايل بر اساس آن مرتب شده و شاخص هم بر اساس آن ساخته مي شود.

به نکات زير توجه کنيم:

شاخص در Fetch کردن تک رکورد قابل استفاده مي باشد، اگر کاربر کليد را به عنوان آرگومان جستجو بدهد.

شاخص فقط و فقط به رکورد هاي ناحيه ي اصلي اشاره مي کند، نه به رکورد هاي ناحيه ي سرريزي.

همين مسأله ساختار را خراب مي کند. چون هدف اين ساختار کاهش زمان Fetch کردن تک رکورد بود، در حالي که پيمايش Link List در ناحيه ي سرريزي زمان گير مي باشد و با هدف ما در تضاد است.

شاخص(Index):

شاخص يا فايل شاخص، مجموعه اي مي باشد که از تعدادي entry تشکيل شده است.

و هر کدام از entry-ها به صورت شکل زير مي باشند:

Pointer

Value

در ادامه توضيح مي دهيم که در فيلدِ Pointer و فيلدِ Value چه چيزهايي مي تواند قرار گيرد:

فيلدِ Value(مقدار): در اين فيلد مقدار هر صفت خاصه اي مي تواند قرار گيرد. و در حالت خاص، مقدار کليد اصلي در اين فيلد گذاشته مي شود.

فيلدِ Pointer(آدرس): اين فيلد ، داراي يک pointer به يک رکورد يا گروهي از رکورد ها در فايل داده اي است.

پس entry شاخص را مي توان يک رکورد کوچک دانست که طول آن V+P بايت مي باشد.

در مثال زير شاخص را بر اساس کد دانشجويي ساخته ايم:

3525

3527

1

2000

تعريف شاخص اصلي:هرگاه در انتخاب صفت خاصه ي شاخص ، کليد انتخاب شود ، شاخص ما اصلي مي باشد.

تعريف شاخص ثانويه:هرگاه در انتخاب صفت خاصه ي شاخص، غير کليد انتخاب شود، شاخص ما ثانويه خواهد بود.

شاخص در حالت کلي ممکن است متراکم يا نامتراکم باشد:

توضيح شاخص متراکم: شاخصي مي باشد که به ازاي هر رکورد در ناحيه ي اصلي ، يک Entry در شاخص وجود دارد.

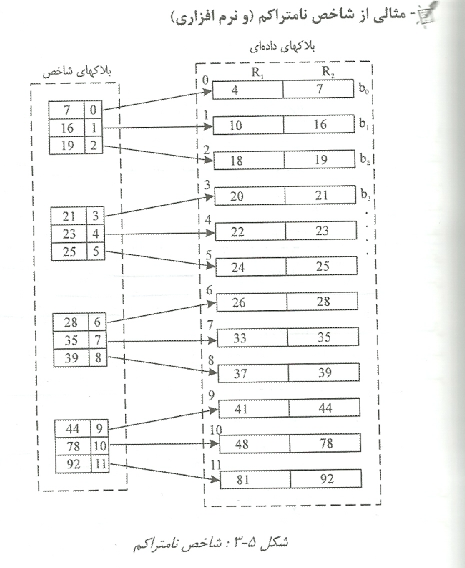
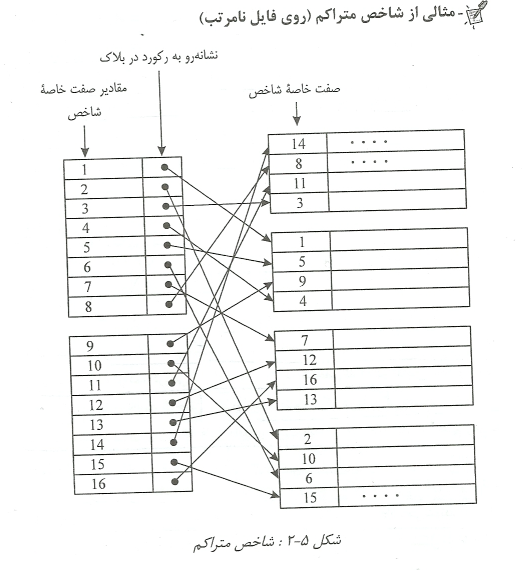
حسن شاخص متراکم اين است که يک ضريب براي داده ي مورد نظر پيدا مي شود.

توضيح شاخص نامتراکم: شاخصي مي باشد که به ازاي هر گروه از رکورد ها، يک Entry در شاخص دارد.(معمولاً يک بلاک)

حسن شاخص نامتراکم اين است که شاخص ما کوچکتر مي شود.

در واقع در شاخص متراکم ، واحد شاخص پذير يک رکورد ، و در شاخص نامتراکم، گروهي از رکورد ها مي باشد.

در شاخص نامتراکم، فايل داده اي بايد روي مقدار هاي صفت خاصه ي شاخص مرتب باشد تا بتوان رکورد ها را گروه بندي کرد.

اما در شاخص متراکم، لازم نيست که فايل داده اي مرتب باشد و روي فايل نامرتب هم ايجاد مي شود.

تعريف لنگرگاه شاخص(Anchor): نقطه اي از فايل داده اي که شاخص به آن اشاره مي کند را لنگرگاه شاخص مي گويند.

در حالتي که شاخص متراکم باشد، لنگرگاه شاخص رکوردي و در حالتي که نا متراکم باشد، لنگرگاه شاخص گروهي(بلاکي) مي باشد.

نکته: اگر فايل داده اي بزرگ باشد، چه شاخص متراکم باشد و چه نامتراکم، شاخص بزرگ مي شود و در نتيجه زمان جستجو در آن زياد خواهد شد.

براي حل مشکل دو راه حل وجود دارد:

استفاده از الگوريتم هاي کاراتر براي جستجو در شاخص(که حتماً محدوديت دارد.)

ايجاد شاخص براي خود شاخص

بديهي باشد که شاخص هاي سطح بالاتر ، شاخص هاي نامتراکم خواهند بود.

سطح آخر شاخص در حد يک بلاک بوده و در حافظه ي اصلي مقيم مي شود.

گروه در شاخص نا متراکم

گروه در شاخص نا متراکم دو حالت دارد:

سخت افزاري: يعني گروه بر اساس تقسيمات سخت افزاري گروه مشخص مي شود.(مثل:سکتور، شيار، استوانه)

نرم افزاري: يعني گروه بر اساس تقسيمات نرم افزاري گروه مشخص مي شود.(مثل:به هر تعداد رکورد با هر نامي، بلاک، باکت)

مثال :فرض کنيد که مي خواهيم از يک شاخص سخت افزاري براي گروه بندي رکورد ها استفاده کنيم، پس داريم:

v200

3

2

v100

v1

1

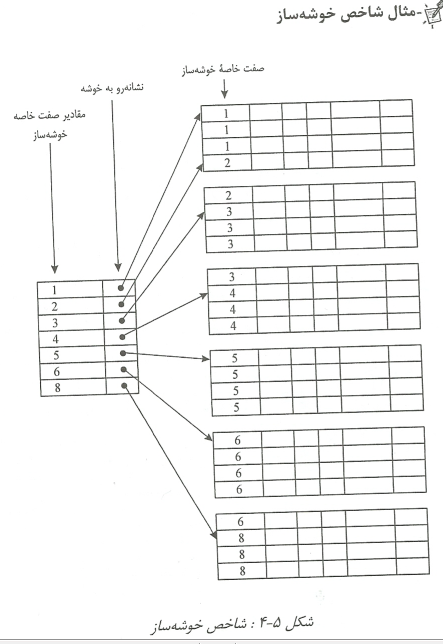
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | اسنوانه 1 | استوانه 2 | استوانه 3 | ... |
| شيار 1 |  |  |  |  |
| شيار 2 |  |  |  |  |
| شيار 3 |  |  |  |  |
| شيار 4 |  |  |  |  |

البته توجه شود که هر دو گروه شاخص ممکن است سخت افزاري باشند.

شاخص خوشه ساز

مي دانيم که فايل داده اي بايد مرتب بايد تا بتوان رکورد ها را در گروه هايي جاي داد ، اما گاه ممکن است مقادير صفت خاصه ، تکراري باشند.

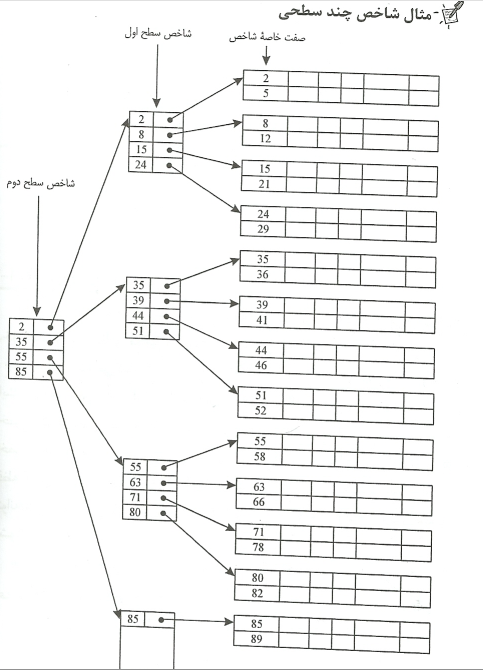
صفت خاصه اي که مقاديرش در فايل تکراري است ، امکان مي دهد تا رکوردها در خوشه هايي(دسته هايي) جاي گيرند. به چنين صفت خاصه اي، صفت خاصه ي خوشه ساز مي گويند.

اگر فايل داده اي بر حسب مقادير يک صفت خوشه ساز مرتب شود، ساخص نا متراکم ايجاد شده روي اين صفت ، شاخص خوشه ساز ناميده مي شود.

شاخص چند سطحي

وقتي که تعداد entry-هاي شاخص زياد باشند، جستجو در شاخص براي يافتن entry شاخص مورد نظر ، زمان گير مي شود.

براي سريع تر شدن عمل جستجو در شاخص ، آن را در چند سطح ايجاد مي کنند.

تعداد سطح هاي شاخص را ، عمق شاخص مي نامند و آن را با x نمايش مي دهند.

معايب شاخص بندي

معايب ساختار هاي شاخص دار ، به ويژه در حالت چند شاخصي عبارتند از:

حافظه به خاطر ايجاد شاخص ، مصرف مي شود.

در عمليات ذخيره سازي over head پيش مي آيد.

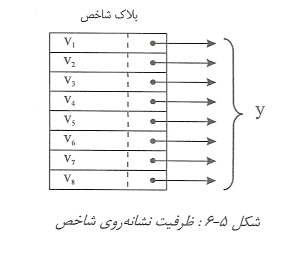
براي حل مثال زير به نکات زير توجه مي کنيم:

تعداد Entry-هاي سطح 1 شاخص و سطح 2 شاخص و همين طور براي بقيه ي سطوح، از فرمول زير به دست مي آيد:

تعداد بلاک هاي فايل اصلي= تعداد Entry-هاي سطح 1 شاخص

نکته: گفته شد که فايل شاخص نيز مانند فايل داده اي ، بلاک بندي مي شود. اندازه ي بلاک ها معمولاً يکسان است. در اينجا بايد مفهومي به نام Fan-Out را تعريف مي کنيم.

تعريف Fan-Out(يا ظرفيت نشانه روي) : ظرفيت نشانه روي يا تعداد Entry-هاي يک بلاک شاخص را Fan-Out مي گويند و با y نشان مي دهند.



حجم کل شاخص از فرمول زير به دست مي آيد:

مثال: فرض کنيد که يک فايل ترتيبي داراي رکورد مي باشد. براي اين فايل با مشخصات زير:

اولاً: سطح هاي شاخص را مشخص کنيد.

ثانياً: حجم شاخص را اندازه بگيريد.

B=2000 byte

R=200 byte

v= 14 byte

p=6 byte

n=

# تکنيک هاي درج در سرريزي

سؤال:" تخصيص فضا به صورت فيزيکي چگونه مي باشد؟"

جواب:

در نظر گرفتن ناحيه اي از فايل داده اي در انتهاي آن که به طور مشترک استفاده مي شود.

در نظر گرفتن يک يا چند شيار پاياني هر استوانه، براي درج سرريزي هاي همان استوانه.

نکته 1: "آيا تعداد شيار هاي در نظر گرفته شده براي سرريز، در انتهاي تمام استوانه ها با هم برابراند؟"

جواب:در عمل چنين است.

نکته 2:"اگر ناحيه ي سرريز استوانه اي پر شد، چه کار کنيم؟"

جواب: دو کار مي توانيم انجام دهيم:

سازمان دهي مجدد انجام دهيم.

در نظر گرفتن ناحيه ي سرريزي ثانويه در انتهاي فايل، که بين تمام استوانه ها به طور مشترک استفاده مي شود.

نکته 3:"عمل درج چگونه انجام مي گيرد؟"

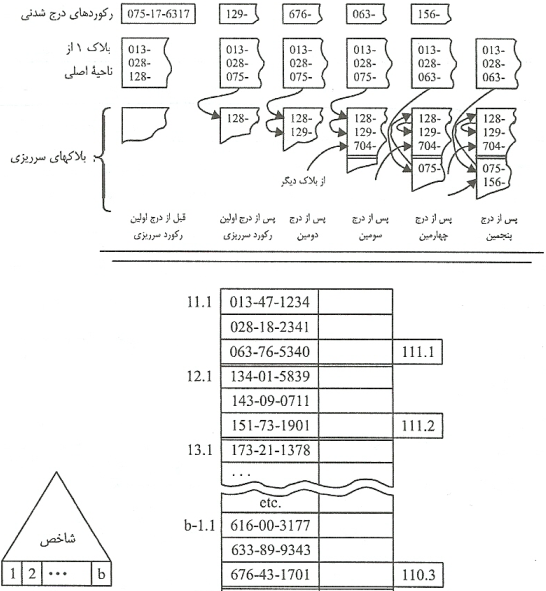
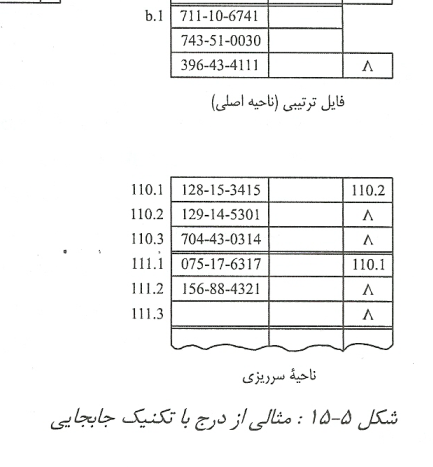
جواب : براي درج رکورد دو روش وجود دارد:

1.رکورد درجي، در اولين بلاک جادار در ناحيه ي سرريزي استوانه اي ، وارد شود و رکورد قبلي به آن اشاره کند.

در اين حالت ، هر رکورد از ناحيه ي اصلي و ناحيه ي سرريزي ، داراي اشاره گر مي باشد.

2.در اين روش رکورد درجي سعي مي کند تا در ناحيه ي اصلي وارد شود و رکورد ديگري را به ناحيه ي سر ريزي بفرستد.(به اين تکنيک ، تکنيک Push Throw مي گويند.)

در اين حالت، هر بلاک از ناحيه ي اصلي به هر رکورد در ناحيه ي سرريزي ، يک pointer دارد.



نکته: در ساختار سوم، شاخص ايستا مي باشد. يعني به ازاي تغييرات در فايل اصلي ، شاخص به روز نمي شود. بنابراين از تکنيک push throw براي درج استفاده مي کنيم.و براي حفظ اعتبار شاخص، اولين رکورد بلاک ها نبايد جابه جا شود.

نکته: مقدار کليد ، تعيين مي کند که رکورد درجي بايد در ناحيه ي اصلي درج شود يا در ناحيه ي سرريزي.

مقايسه ي دو تکنيک :

تعداد زنجيره ها در تکنيک دوم ، کمتر از تعداد زنجير ها در تکنيک اول مي باشد.

حجم حافظه ي مصرفي براي pointer-ها در تکنيک دوم کمتر است.

تأمين خواندن به صورت سريال، در تکنيک دوم راحت تر مي باشد.

طول زنجيره ها در تکنيک اول کمتر از تکنيک دوم است

جمع بندي: تکنيک دوم بهتر مي باشد.

**ارزيابي فايل با ساختار شاخص دار**

فرضيات:

ساختار شاخص، ايستا و هم توالي با فايل داده اي است.

سطح اول شاخص ، نامتراکم و گروه ، بلاک مي باشد.

تکنيک درج، push throw است.

ناحيه ي سرريزي استوانه اي داريم.(يعني تکنيک دوم)

ناحيه ي سرريزي ثانويه نداريم.

تکنيک حذف ، مارکر Delete مي باشد.

سطوح مختلف شاخص بر روي يک استوانه ذخيره مي شود.

در غواصي سطوح از (x-1) تا سطح 1 را بايد بخوانيم و در طول اين غواصي زمان استوانه جويي نداريم.

**الف-1)محاسبه ي متوسط حافظه ي مصرفي به ازاي هر رکورد()**

ب-1)زمان واکشي تک رکورد()

دو حالت قابل تصو است:

کاربر کليد را بدهد.

کاربر کليد را ندهد.

ريز عمليات حالت 1:

غواصي در شاخص تا رسيدن به سطح 1 و pointer-ي به ناحيه ي اصلي

خواندن بلاکي از ناحيه ي اصلي

(به احتمالي) رفتن به ناحيه ي سرريزي و خواندن به طور متوسط نصف زنجيره ي رکورد هاي سرريزي

حالت 2: در اين حالت فايل تبديل به يک فايل پايل مي شود و ديگر شاخص به درد نمي خورد.(بدبخت مي شويم!)

ب-2)زمان به دست آوردن رکورد بعدي()

6 حالت بين رکورد اصلي و رکورد سرريزي قابل تصور است. که مجموع آنها رابطه ي زير مي شود:

تمرين: "اين 6 حالت کدام اند؟"

ب-3)زمان درج تک رکورد()

دو حالت ممکن است رخ دهد:

رکورد درجي ، بتواند عامل push شود.

رکورد درجي، مستقيماً در ناحيه ي سرريزي درج شود.(نتواند عامل push شود.)

ريز عمليات 1: ()

يافتن نقطه ي منطقي درج(بلاکي از ناحيه ي اصلي)

کار در بافر(يعني قرار دادن رکورد در بلاک و خواندن رکورد push شدني)

خواندن بلاکي از ناحيه ي سرريزي براي قرار دادن رکورد جديد در آن(سيستم آدرس آخرين بلاک آزاد را دارد)

کار در بافر(تنظيم اشاره گر ها)

بازنويسي بلاک ناحيه ي سرريزي

بازنويسي بلاک ناحيه ي اصلي

ريز عمليات 2: ()

خواندن بلاک جادار از ناحيه ي سرريزي

کار در بافر

خواندن رکورد پيشين

کار در بافر

بازنويسي رکورد ناحيه ي درجي

بازنويسي رکورد ناحيه ي پيشين

نکته: با فرض نامتراکم بودن شاخص و گروه بودن بلاک، فايل داده اي داراي تا گروه مي باشد و تعداد سطح هاي شاخص(عمق شاخص) از فرمول زير به دست مي آيد:

البته در هنگام ذخيره سازي فايل ممکن است که شاخص در يک فايل جدا و يا در يک فايل داده اي ذخيره شود.

تمرين: براي مثال فوق با شاخص 4 سطحي، چند رکورد از فايل را مي توان آدرس دهي کرد؟

مثال: مثال قبل را براي اعداد زير حل کنيد.

B=1800

R=200

v= 15 byte

p=2 byte

n=