جلسه ششم

فایل ترتیبی Sequential:

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

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

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

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

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

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

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

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

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

هنگامی که پردازش ســریالی رکوردها مورد نظر باشــد، ســاختار ترتیبی بهتر از پایل اســت. در بســیاری از سیستم های تجاری که رکوردها به صورت دسته ای (Batch) پردازش می شوند، از ساختار ترتیبی استفاده می شود.

متوسط اندازه رکورد:	$R=a.V$	a تعداد فیلد
کل فایل:	$S=n.a.V$	V طول فیلد
کل فایل با تراکنش:	$S=(n+o).a.V$	n تعداد رکوردها

محاسبه ی TF در فایل ترتیبی:

چقدر زمان طول می کشد به یک رکورد دسترسی پیدا کنیم.

الف) واکشی در فایل ترتیبی اگر جستجو بر مبنای فیلدی غیر کلید باشد:

فایل ترتیبی مشابه یک فایل پایل بوده و لذا مجبوریم که از جست و جوی خطی استفاده کنیم.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

جست و جو با پرش بلاکی (Skipped Block Search):

این روش در واقع بهبود یافته ی روش جست و جوی خطی بوده و هنگامی که آرگومان مورد جست و جو بر مبنای فیلد کلید با شد قابل ا ستفاده ا ست. فرض کنید فایل بر ا ساس کلید به صورت صعودی مرتب شده باشـد. در این حال اولین بلاک را خوانده و کلید مورد جسـت و جو را با کلید آخرین رکورد موجود در بلاک مقایسه می کنیم. اگر کلید مورد جست و جو بزرگ تر با شد، پس حتما در آن بلاک نیست و بنابراین بلاک بعدی را می خوانیم. بدین ترتیب دیگر نیازی نیسـت که بقیه ی رکوردهای موجود در آن بلاک را برسـی کنیم. به همین ترتیب بلاک ها را پشـت سـر هم خوانده و تنها کلید آخرین رکورد هر بلاک را وارسـی می کنیم تا هنگامی که کلید مورد جست و جو از کلید آخرین رکورد بلاکی کوچک تر باشد. در این حالت رکورد مورد جست و جو در آن بلاک بوده و فقط کافی است آن بلاک را با روش خطی یا باینری جست و جو کنیم. بهترین اندازه ی بلاک یا به عبارتی دیگر مقدار بهینه ی B_F برحسـب تعداد رکوردها (n) برای آن که تعداد مقایسه ها در روش جست و جوی بلاکی حداقل باشد برابر است با:

$$B_F = \sqrt{n}$$

زمان دستیابی به رکورد بعدی در فایل ترتیبی TN:

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

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

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

محاسبه TI در فایل ترتیبی:

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

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

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

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

درج فایل های بزرگ:

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

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

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

زمان به هنگام سازی تغییر دهنده:

Tu (Update) , TD (Delete)

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

عملیات ا صلاح دو و ضعیت متفاوت دارد. اگر ا صلاح بر روی فیلد غیر کلید با شد می توان عملیات به هنگام سازی را به صورت درجا انجام داد یعنی:

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

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

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

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

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

برای خواندن کل فایل دو روش داریم:

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

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

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

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

۱: زمان Sort به اندازه ی o رکورد

۲: خواندن فایل اصلی

۳: خواندن فایل T.L.F

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