پیام نور رضوانشهر

موضوع پروژه:

ساختارهای درختی

نام درس: ذخیره و بازیابی اطلاعات

فهرست

بسمه تعالی

امتحان میان ترم درس آمار و احتمال۲

دانشگاه پیام نور رضوانشهر

سؤال ۱) فرض کنید $X_1, X_2,\ldots,X_n$ متغیرهای تصادفی مستقل و هم توزیع از یک توزیع یکنواخت $(0.\theta)$ و $Y_1,Y_2,\ldots,Y_n$ آماره های ترتیبی مربوط به این نمونه n تایی باشند در این صورت توزیع توام Y_n , Y_1/Y_n را به دست آورید. (۲نمره)

سؤال۲) طول عمر قطعات تولیدی یک کارخانه دارای میانگین ۵ با واریانس ۱می باشد. این کارخانه محصولات خود را در بسته های ۳۶ تایی به مشتریان خود عرضه می کند. یکی از مشتریان کارخانه محصولات را در صورتی قبول می کند که حداقل ۲۵ درصد از بسته های ارسالی میانگین طول عمری بیشتر از ۵/۲۱ داشته باشند. احتمال آن را به دست آورید که یک محموله ۱۲ تایی ارسال شده برای این مشتری پذیرفته شود. (۲ نمره)

سؤال۳) اگر Y,X متغیرهای تصادفی با تابع چگالی توام زیر باشند، توزیع $Z=X-Y$ را به دست آورید. (۲ نمره)

$$f(x,y)=\begin{cases}\frac{1}{25}(\frac{20-x}{x}) & 10<x<20,\frac{x}{2}<y<x\\ 0\end{cases}$$

استفاده از جدول آماری آزاد است

سربلند و پیروز باشید.

ساختارهای درختی

فایل با ساختار درخت جستجوی دودویی

در فایل با ساختار ترتیبی لازمه استفاده از الگوریتم جستجوی دودویی این است که بلاک های داده ای به طور پیوسته ذخیره شده اند اگر بلاک ها به طور ناپیوسته ذخیره و به هم پیوند شده باشند یافتن آدرس بلاک میانی ناممکن است.

فایل با ساختار درخت جستجوی دودویی با n رکورد و کلید اصلی i=1,2,…,n,ki گونه‌ای از درخت دودویی است که دو خاصیت زیر را دارد.

۱- هر گره درخت، بسته به طرز پیاده سازی، حداقل سه یا چهار فیلد در هر دو حالت دو تا از فیلدها حاوی نشانه رو به گره های سمت چپ و سمت راست هستند RPTR, LPTR در حالت وجود سه فیلد، فیلد سوم حاوی خود رکورد است. در غیر این صورت در فیلد سوم کلید رکورد قرار دارد و فیلد چهارم حاوی نشانه روی به بلاک داده ای حاوی رکورد است.

۲- اگر ki کلید یک رکورد باشد کلید تمام رکوردهای موجود در گره های زیردرخت سمت چپ از ki کوچکتر و کلید تمام رکوردهای موجود در گره های زیر درخت سمت راست، از ki بزرگترند،

عملیات در فایل

واکنش رکورد

الگوریتم واکنشی خیلی ساده است سیستم ابتدا به گره ریشه دستیابی پیدا می کند عمل مقایسه بین کلید رکورد مورد نظر و کلید رکورد موجود در گره ریشه انجام می شود، اگر

تساوی برقرار باشد، رکورد پیدا شده است وگرنه، یکی از دو گره سمت راست یا سمت چپ گره ریشه مورد دستیابی قرار می گیرد و عمل مقایسه انجام می شود، این عملیات تا پایان یافتن رکورد مورد نظر یا برخورد به نشانه روی تهی تکرار می شود اگر رکورد مورد نظر در سطح k باشد در حافظه اصلی ذخیره شود برای واکنش رکورد k+1 بار دستیابی مستقیم لازم است.

کارایی این ساختار در واکنشیس رکورد وقتی حداکثر است که ژرفای حداقل باشد و زمانی حداقل است که ژرفای درخت حداکثر باشد.

ژرفای درخت زمانی حداکثر است که در هر سطح تنها یک گره وجود داشته باشد در این حالت ژرفای درخت N است و متوسط دستیابی (ANA) مستقیم برای واکنشی رکورد برابر است با:

$$ANA = \left(\sum_{1}^{n} k\right)/n \qquad ANA = \frac{n(n+1)}{2n} = \frac{n+1}{2}$$

از طرف دیگر ژرفای درخت زمانی در حداقل است که در هر سطح مثلاً سطح k ام، غیر از سطح ریشه دقیقاً 2^k گروه وجود داشته باشد. اگر ژرفای درخت را x فرض کنیم با فرض پربودن تمام درخت داریم:

$$n=2^x-1$$

و متوسط زمان دستیابی لازم برای واکنشی رکورد برابر است با:

$$T_F = \frac{1}{n}\sum_{k=1}^{x}\left[k.2^{k-1}\right].(r+s+b_{tt})$$

می توان نشان داد که عبارت بالا برابر است با:

$$T_f = (\log_2(n+1)-1)(s+r+b_{tt})$$

عمل درج

اگر درخت خالی باشد، رکورد درج شدنی به آسانی درج می شود. اگر درخت خالی نباشد و کلید رکورد کوچکتر ازکلید رکورد و ریشه باشد، رکورد در سمت چپ ریشه درجه می شود و اگر کلید رکورد از کلید ریشه بزرگتر باشد رکورد در سمت راست ریشه درج می شود. این مقایسه کلیدها در هر سطح دیگر هم تکرار می شود تا نقطه منطقی درج رکورد پیدا شود و عمل جایابی زمانی متوقف می شود که با نشانه روی تهی برخورد شود. بدین ترتیب با درج هر رکورد جدید، یک گره در انتهای یکی از مسیرها ایجاد می شود.

عملیات لازم برای درج رکورد چنین است:

یافتن نقطه منطقی درج

خواندن بلاکی که رکورد باید در آن درج شود (یک بلاک از فضای آزاد)

بازنویسی همین بلاک

بنابراین داریم:

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

حذف رکورد

با حذف رکورد، باید وضعیت ساختاری یک فایل به گونه ای تنظیم شود که ماهیت آن محفوظ بماند، یعنی کماکان یک درخت در جستجوی دودویی باشد. در هر حال گره مربوطه باید حذف شود سه حالت متصور است:

حالت اول: تعداد گره های فرزند گره حذف شدنی صفر باشد (گره فرزند نداشته باشد) در این حالت گره را می توان حذف کرد و درخت کماکان ماهیت خود را حفظ می کند.

حالت دوم: گره حذف شدنی فقط یک گره فرزند داشته باشد.

در این حالت درخت فقط در صورتی ماهیت خود را حفظ می کند که فرزند گره حذف شده، جایگزین آن بشود برای این منظور، فیلد نشانه رو در گره پدر گره حذف شده باید متناسباً تنظیم شود.

حالت سوم: گره حذف شدنی دو فرزند داشته باشد. در این الت، پس از حذف گره مربوطه، رکورد بعد آن، طبق نظم، باید جایگزین آن شود. رکورد بعدی طبق نظم، رکورد سمت چپ در زیر درخت سمت راست گره حذف شدنی است. بدین ترتیب، دیگر نیازی به جستجو در درخت، طبق نظم، برای یافتن رکورد بعدی نیست و به علاوه با حذف رکورد بعدی موضع درخت چنان می شود که یکی از دو حالت اول یا دوم پیش می آید.

خواندن تمام فایل

خواندن سریال رکوردها عملاً بسیار زمانگیر است، اولین رکوردی که باید خوانده شود، رکورد موجود در گره انتهایی سمت چپ ترین شاخه درخت است و بدیهی است که برای خواندن آن رکوردهای پیشین آن باید مورد دستیابی قرار گیرند و به علاوه آدرس این رکوردها نیز باید نگهداری شود تا بتوان آنها را به طور سریال خواند. بنابراین عملیات خواندن سریال علاوه بر زمانگیر بودن به حافظه نسبتاً زیادی نیز نیاز دارد.

برای خواندن پی در پی رکوردها می توان فایل را از ابتدا تا انتها خواند اما با این روش رکوردهای پیشتر حذف شده نیز خوانده می شود. روش بهتری وجود دارد که در آن با n بار دستیابی مستقیم همه رکوردها خوانده می شوند. در این روش ابتدا همه گره های سمت چپ ترین شاخه خوانده می شوند، اگر هر گره ای در این شاخه، نشانه روی راست داشته باشد، آدرس ریشه زیر درخت منشعب از آن گره در یک پشته نگهداری می شود. پس از آنکه همه گره های سمت چپ ترین شاخه خوانده شدند، سمت چپ ترین شاخه از زیر درختی که آدرس ریشه آن، رأس پشته است، پیمایش شده گره هایش خوانده می شوند. عمل خواندن زمانی به پایان می رسد که پشته خالی باشد.

فایل با ساختار درخت جستجوی دودویی نخ کشی شده (TBST)

می توان برای تسریع پردازش سریال رکورد ها (بر اساس نظم صعودی یا نزولی مقادیر کلید) درخت جستجوی دودویی را کامل تر کرد به این ترتیب که به جای نشانه روی تهی در هر گره، نشانه رو به رکورد پیشین و بعدی ایجاد کنیم. انگار گره ها را نخ کشی کرده باشیم با این تغییر، فیلد نشانه روی چپ به گره پیشین و فیلد نشانه روی راست به گره بعدی نشانه می رود.

نخ کشی گره ها سبب تسریع عملیات بازیابی بعدی و بازیابی قبلی می شود، مشخص است که پردازش سریال رکوردها با انجام یک سلسله عملیات بازیابی بعدی امکان پذیر می شود، در عوض عملیات درج و حذف رکورد زمانگیر تر می شود.

فایل با ساختار درخت صفحه بندی شده

می توان گره های درخت را بلاک بندی کرد در این صورت می گوییم درخت صفحه بندی شده داریم.

بلاک بندی های گره های درخت، امکان می دهد تا مثلاً ریشه درخت و احیاناً گره فرزند چپ یا راست یا هر دو با یک بار دستیابی به دست آیند .

می توان فاکتور بلاک بندی را چنان انتخاب کرد که هر بلاک حاوی یک گره ریشه و دو زیر درخت هر یک به عمق $h=x_{subtree}$ باشد، به سهولت می توان دریافت که:

$$B_f = \sum_{i=0}^{h=1} 2^i = 2^n - 1$$

مزیت اصلی درخت صفحه بندی شده این است که عمق درخت به نسبت h-1 کاهش می یابد و در نتیجه متوسط زمان جستجو کاهش می پذیرد، اما این ساختار مشکلاتی دارد، از جمله:

- بروز حافظه هرز در صفحه ها، شکل۹ سه صفحه (بلاک با سه گره) کاملاً استفاده شده اند. در بقیه صفحه ها، فضای هرز پدید آمده است به علاوه با افزایش فاکتور بلاک بندی، فضای هرز نیز زیاد می شود.
- بروز فزونکاری در سیستم به ویژه در عملیات درج و حذف (زیرا محتوای صفحه‌ها باید متناسباً تنظیم شوند).

فایل با ساختار درخت متعادل(B-TREE)

معرفی ساختار

ساختارB-TREE ، با رتبهm که آن را باB_m نمایش می دهیم یک درخت جستجوی 2M+1 راهه است که در آن همه گره ها (غیر از گره ریشه) حداقل نیمه پر بود و عمق تمام شاخه های آن یکسان است.

منظور از2M+1 راهه این است که در هر گره2M+1 فیلد نشانه رو وجود دارد.

اگر بخواهیم فایلی با ساختارB-TREE ایجاد کنیم باید هر گره درختB_M را به صورتی طراحی کنیم که هر گره درخت6M+2 فیلد دارد. فیلداول، فیلدC حاوی یک عدد صحیح است نشان دهنده تعداد کلیدهای موجود در گره، پس یک شمارنده است.

هر گره غیر از گره ریشه حداقل نیمه پر است بنابراین مقدار فیلدC یعنی C بین2M,M است. مقادیر کلیدKi ها دو فیلد دردو طرف دارد: فیلدPi و فیلدri (I=1,2,…,2M) جفت فیلدهای (Pi,Ki) در واقع مدخلی را تشکیل می دهند که به گره ای در سطح پایین تر اشاره می کند که مقادیر کلید موجود در آن ازki کوچکتر است. فیلدri حاوی نشانه رویی است به بلاک داده ای حاوی رکورد.

بالاخره یک فیلد نشانه رو در انتهای گره داریم.P_{2M+1} که به گره ای در سطح پایین اشاره می کند که در آن مقادیر کلید ازK_{2M} بزرگترند.

با توجه به ساختمان یک گره می بینیم که تعداد نشانه روهای ساختاری یعنیPi در هر گره باید1+C باشند. بنابراین، هر گره، غیر از گره ریشه حداقل1+M گره فرزند دارد در حالی که گره ریشه حداقل دو گره فرزند دارد.

بنابر آنچه گفته شد خصوصیات فایل با ساختارB-TREE از رتبه M را می توان چنین برشمرد:

- یک درخت جستجوی2M+1 راهه است.
- ژرفای تمام شاخه ها یکسان است.
- گره ریشه حداقل دو گره فرزند دارد.
- گره های غیر ریشه حداقل M+1 گره فرزند دارد و حداکثر فرزندان2M+1 است.
- تعداد کلیدها در هر گره یکی کمتر از تعداد فرزندان آن گره است.

جهت سادگی در نمایش، گره ها را چنین نشان می دهیم.

عملیات در ساختار

فرض می کنیم که فایل با ساختار درختB_M است. رکوردها بلاک بندی نشده اند. برای تسریع پردازش سریال رکوردها بهتر است که رکوردها طبق نظم مورد نظر به یکدیگر پیوند شوند.

واکنش رکورد

پس از یافتن گره مربوط به رکورد با استفاده از نشانه روri رکورد مورد دستیابی قرار می‌گیرد. جستجوی گره مربوط به رکورد تا زمانی ادامه می یابد که گره مورد نظر یافت شود و یا به نشانه روی تهی برخورد شود که حالت عدم موفقیت است.

تعداد دفعاتI/O برای یافتن رکورد بستگی دارد به عمق درخت و یا به سطحی که گره ناظر به رکورد قرار دارد. اگر عمق درخت راX فرض کنیم، تعداد دستیابی مستقیم لازم برای واکنشی رکورد چنین است. $2 \le NA \le X+1$

بنابراینT_f بستگی دارد به سطحی که گره ناظر رکـورد در آن قـرار دارد و حـداکثر برابـر است با:

$$T_f=(x+1)(r+s+b_{tt})$$

خواندن سریال تمام فایل

چون رکوردها را ترتیبی پیوندی در نظر گرفتیم. برای خواندن سریال تمام فایل، بهn بـار دستیابی مستقیم نیاز است. (رکورد بلاک بندی شده اند) بدیهی اسـت ایجـاد و مـدیریت این لیست پیوندی برای سیستم فزونکاری دارد.

عمل درج

عمل حذف

برای بررسی عمل حذف درخت$B2$ شکل قبل را در نظر می گیریم و تعدادی رکـورد را حذف می کنیم و طی این عملیات، حالات مختلف عمل حذف را نشان می دهیم.

- حذف رکورد با کلید۴۰

این حذف ساده ترین حالت از حالات حذف است در بلاک حاوی این رکـورد، بـیش از دو رکورد ذخیره شده است. بنابراین با حذف این رکورد، هنوز سه رکورد در ایـن بـلاک جای دارد عمل ادغام یا توزیع مجدد کلیدها پیش نمی آید پـس از حـذف ایـن رکـورد، درخت به صورت زیر در می آید.

- حذف رکورد با کلید۷

در این حالت یکی از گره ها سبکبار می شود. ابتدا باید امکـان توزیـع مجـدد کلیـدها را بررسی کرد. بلاک سمت راست بلاک حاوی رکورد حذف شدنی یعنـی بـلاک همـراه را

بررسی می کنیم اگر مجموع کلیدهایاین دو بلاک از2M بزرگتر باشد (مثال٤) یک کلید باید از بلاک همراه خارج شود (اولین مقدار کلید از این بلاک که۱۲ است) و به جای اولین رکورد بلاک بلافاصله بالاتر قرار داده شود (به جای۹) این رکورد خود از بلاک بلافاصله بالاتر خارج شده و به بلاک حاوی رکورد حذف شدنی منتقل می شود و در عین حال رکورد حذف شدنی نیز از این بلاک حذف می شود.

حذف رکورد با کلید۹

این حالت شبیه حالت قبل از البته با یک تفاوت شباهت در این است که بلاکی که سبکبار می شود، یک گره انتهایی است و تفاوت در این است که در این حالت توزیع مجدد کلیدهای امکان پذیر نیست، با حذف رکورد با کلید۹ بلاک حاوی این رکورد- که اینک سبکبار است- با بلاک همراهش ادغام می شود و ضمناً کلید ۱۲ از بلاک بالاتر نیز در بلاک حاصل ادغام وارد می شود.

با جابجا کردن کلید ۱۲ از بلاک نیز سبکبار می شود و در نتیجه باید با بلاک سمت راست خود ادغام شود .

- حذف رکورد با کلید۳۲

در این حالت رکوردی حذف می شود که کلید آن در گره انتهایی نیست بلکه در گره ای است که تعدادی از آن منشعب شوند در این حالت کلید بعدی حذف شده جایگزین می‌شود.

(بعدی در پیمایش نظمند) در این مثال کلید بعدی ۳۵ است و جایگزین کلید ۳۲ می‌شود اگر گره انتهایی در اثر این جابجایی سبکبار شود باید مثل حالات قبلاً دیده عمل کرد.

فایل درختی

تاکنون ساختارهایی دیده ایم مبتنی بر درخت که در آنها یک بخش شاخص جدا از فایل داده ای وجود داشت در این قسمت به ساختاری می پردازیم که در آن به نحوی بخش داده ای فایل در بخش شاخص ادغام شده است. این فایل را فایل درختی می نامیم.

شرح ساختار

در این ساختار ارتباط درختی بین خود رکوردهای داده ای برقرار می گردد. سریالیتی فقط بر اساس مقادیر یکی از صفات خاصه تأمین می شود و روش پردازش سریال این فایل همان الگوریتم (پیمایش نظم اند) مطروحه در مباحث مربوطه به درخت دودویی است.

اجمال الگوریتم مزبور چنین است.

Proceed: go to the left subtree if not terminal black then- proceed else process the records of the terminal black go up to the root for this svbtree.

زمان واکنشی رکورد

عمده ترین مزیت این ساختار این است که «افزونگی» ندارد رکوردهای فایل (طبعاً با مقدار کلید در خود رکورد) همان درخت جای دارن و کلید هر رکورد فقط یکبار ذخیره می شود (وجود شاخص باعث بروز افزونگی در فایل است) ما در این ارزیابی فرض می‌کنیم که تمام بلاک ها پر هستند و رکوردها یک پاره اند. هر چند با داشتن بلاک های دارای بخش رزو، عملیات بهنگام سازی تسهیل می شود و کاراتر است.

عیب این ساختار این است که ظرفیت بلاک های آن در مقابل ظرفیت بلاک ها در ساختارهای با شاخص به طور قابل ملاحظه ای کاهش می یابد. به نحوی که زمان واکنش رکورد را افزایش می دهد (به طور متوسط) در این ساختار رکوردها می توانند طول متغیر داشته باشند زیرا الگوریتم های دستیابی بستگی به تعداد رکوردهای بلاک ندارد.

با فرض ثابت بودن مقدارR داریم:

$$y = \left\lfloor \frac{B - P}{R + P} \right\rfloor + 1$$

حتی اگر به بلاک های پایین ترین سطح نیاز به نشانه رو نباشد، فیلد مربوطه اش را ایجاد می کنند. زیرا سبب تسهیل در «رشد فایل» می شود. از این رو مقدارy در این سطح همان است که در سطوح بالاتر تعداد رکوردها در هر سطح برابر است با:

$$n_i = (y-1)b_i = (y-1)y^{x-i}$$

i: شماره سطح

b_i: تعداد بلاک ها در سطحi

که در آنx تعداد سطوح است و می توان آن را چنین برآورد کرد:

تعداد رکوردها

در سطح۱: $n_1 - n.(y-1)\sum_{i=x}^{2} y^{x-i}$

$$x = \lceil \log yn \rceil$$

زمان واکنشی یک رکورد بستگی دارد به اینکه رکورد در چه سطحی از درخت باشد، لذا متوسط زمان لازم برای واکنشی یک رکورد را محاسبه می کنیم، برای این منظور زمان

واکنشی رکوردهای هر سطح را محاسبه و حاصل جمع این زمان ها را بر r تقسیم می‌کنیم.

$$T_F = \frac{1}{n}(s + r + b_{tt})(1y - 1)\sum_{i=x}^{2}(x - i + 1)(y^{x-i} + xn_1)$$

مقایسه کارآیی فایل درختی با کارایی فایل دارای شاخص درختی و کارایی فایل ترتیبی شاخص دار.

ما این مقایسه را به کمک یک مثال در دو حالت رکوردهای کوتاه و رکوردهای بلند انجام می دهیم.

مفروضات مثال:

الف: در حالت رکوردهای کوتاه، رکورد نوع کارمند دارای دو فیلد شماره کارمند، ۹ کاراکتر و مهارت، ٦ کاراکتر است، باید R=۱۵.

ب: در حالت رکوردهای طولانی، فرض طول R=۱۸۰۰ بایت می باشد (که چندان هم طولانی نیست) و داریم، بایت B=۱۰۰۰ رکورد n=۲۰۰۰ بایت D=٤

با توجه به مفروضات بالا، محاسبات لازم را برای سه ساختار و در دو حالت انجام می‌دهیم.

I: فایل درختی

- جمع بندی، با مقایسه سه روش در این مثال درمی یابیم که روش اول یعنی فایل درختی وقتی که رکوردها طول کوچک دارند، بیشترین کارایی را دارد، ولی با رکوردهای طولانی، کمترین.

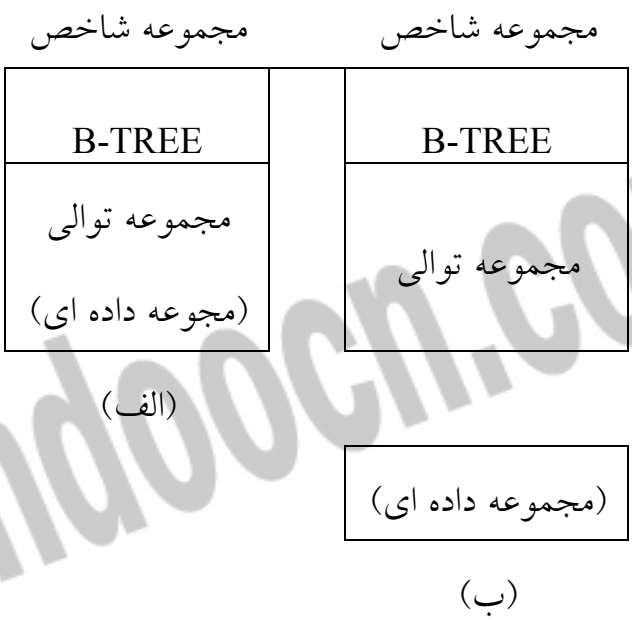
ساختارI.S بیشتر دیده شده که در این مثال هم مورد مقایسه با سایر روش ها قرار گرفت، روش مناسبی است. ولی نه در مواردی که روش اول، بهینه می نماید. (مثل فایل با رکوردهای کوتاه) وقتی که رکوردها طولانی باشند باز هم می توان در یک فایل جداگانه جای داد. در این صورت برای واکنشی رکوردهای فایل به یک دستیابی دیگر نیاز است. لذا تنها هنگامی کارایی دارد که عملیات روی بخش کلیدی رکوردها امر رایجی باشد.

متعادل کردن درخت

کارایی فایل درختی بعد از عملیات حذف و درج ممکن است کاهش یابد زیرا این عملیات روی بلاک های دیگر نیز تأثیر گذاشته آنها را دچار تغییرات می کنند لذا برای حفظ کارایی آنها باید درخت را به صورت متعادل درآورد یعنی عمق هر شاخه از درخت (از رأس تا پایین ترین سطح) X و یاX-1 باشد متعادل کردن، معمولاً در اثناء عمل بهنگام سازی و یا در عمل سازماندهی مجدد انجام می گیرد.

فایل با ساختار درخت B^+

در فایل با ساختار درخت متعادل دیدیم که در هر گره از هر سطح درخت نشانه روهایی به رکوردها وجود داشت. هزینه ایجاد و نگهداری چنین ساختاری نسبتاً زیاد است و به علاوه هر دو مجموعه شاخص و داده با هزینه بالا باید ایجاد شوند. فایل با ساختار درخت B^+ چنین مشکلاتی را ندارد. در شکل زیر دو طرح برای فایل با ساختار درخت B^+ دیده می شود.



می بینیم که در طرح (الف) مجموعه توالی همان مجموعه داده ای است یعنی رکوردها در گره های انتهایی ذخیره می شوند، اما در طرح (ب) در مجموعه توالی فقط مدخل‌هایی با دو فیلد مقدار کلید و نشانه رو به رکورد وجود دارد. همانگونه در فصل ساختارهای شاخص دار دیدیم، در هر دو طرح دیگری نیازی به نشانه رو در گره های غیرانتهایی درخت برای نشانه روی به رکوردهای داده ای نیست.

توجه داریم که پیاده سازی درخت B^+ بر اساس طرح (ب) قبل نسبت به طرح (الف) این مزیت مهم را دارد که به علت بالا بودن تعداد مدخل ها در هر گره (ظرفیت نشانه روی گره) ژرفای درخت کاهش می یابد و در نتیجه کارایی ساختار افزایش می یابد.

خصوصیات ساختار درخت B^+_M چنین است:

- تمام گره های انتهایی همسطح هستند (ارتفاع تمام شاخه ها یکسان است)
- گره ریشه حداقل دارای ۲ فرزند است در حالی که هر گره دیگر حداقل $M+1$ فرزند دارد.

حداکثر تعداد فرزند برای هر گره $2M+1$ است.

- تعداد کلیدها در هر گره یکی کمتر از فرزندان آن گره است.
- درخت تشکیل شده از گره های انتهایی یک درخت B_M است.

با توجه به وجود مجموعه توالی در آن ساختار می توان آن ساختار را یک پیاده سازی ممکن از فایل با ساختار درخت B^+ دانست، چون تمام عملیات در فایل را در آن ساختار بررسی کردیم.

فایل با ساختار درخت 1k-d

این ساختار گونه ای از ساختار درخت جستجوی دودویی است، دیدیم که در ساختار درخت جستجوی دودویی در هر گره یک رکورد وجود دارد و هر گره می تواند یک فرزند در سمت چپ و یک فرزند در سمت راست داشته باشد. در فایل با ساختار درخت جستجوی دودویی فیلد کلید (صفت خاصه کلید) در تمام رکوردهای تمام سطوح واحد و یکسان است.

اما تفاوت فایل با ساختار درختk-d با فایل با ساختار درخت جستجوی دودویی در این است که فیلد کلید (فیلد حاوی نشانوند مقایسه) در سطوح مختلف یکسان نیست. اگر رکورد k فیلد$F_k,\ldots,F_2,F_1$ داشته باشد. در این صورت از سطحi ام از فیلدfi به عنوان نشانوند مقایسه استفاده می شود.(I=1,2,…,n) و اگر تعداد سطوح بیش از تعداد فیلدها شود به طور چرخشی عمل می شود یعنی در سطحk+1 دوباره فیلدf_1 بلاک مقایسه و جستجو می گردد و الی آخر. اگرN گره ای باشد در سطحی که فیلدfi به آن متناظر شده است و محتوای فیلدfi در رکورد ذخیره شده در گره N مقدارX باشد دراین صورت فرزند سمت چپ N و تمام پی آیندگانش (اخلافش) باید در فیلدFi مقدار کوچکتر ازx داشته باشد و فرزند سمت راست و تمام پی آیندگانش باید در فیلدFi مقدار بزرگتر یا مساویx داشته باشد.

- مثال- می خواهیم مختصات ۹ نقطه را در نه رکورد از یک فایل ذخیره کنیم:

عرض=y	طول=x
6	3
7	6
1	1
6	5
3	4
0	6
1	5
4	0
2	7

در اینجا هر رکورد ۲ فیلد دارد:

فیلد عرض=F_2 و فیلد طول=F_1

$(F_1)=X,(F_2)=Y$

به طور کلی تعداد درخت های k-d برای یک مجموعه واحد از رکوردها می توان بیش از یک باشد. انتخاب یک درخت از درخت های ممکن بستگی به نظمی دارد که بر اساس آن می خواهیم رکوردها را درج کنیم.

بررسی نحوه ساختن این درخت و چگونگی جستجو در آن را به خواننده وامی گذاریم.

از این ساختار در پاسخگویی به پرس وجوهای طیفی می توان استفاده کرد.

مثال۱: می خواهیم مجموعه نقاطی را پیدا کنیم که عرض آنها ۱ است.(y=1)

ابتدا از ریشه شروع می کنیم (سطحi=1) چونF1 فیلد طول است X,(X) نشانوند جستجو نیست باید در هر دو شاخه چپ و راست رکورد واقع در ریشه جستجو کنیم (و البته خود رکورد موجود در ریشه را نیز) در سمت چپ رکورد ریشه، رکورد(1,1) را می‌یابیم که یک جواب است.

در این سطحi=2 تنها باید در شاخه راست جستجو کنیم زیرا نشانوند جستجو و مقایسه y=1 است و در شاخه چپ این گره فقط رکوردهای باy<1 می توانند وجود داشته باشند (در اینجا زیر درخت سمت چپ تهی (بی گره) است در سمت راست این گره رکورد(0,4) را داریم که جواب نیست اما چون در این سطح نشانوند مقایسهx است باید در هر دو شاخه گره جستجو کنیم. ولی زیر درخت آن تهی است.

حالت جستجو را در شاخه سمت راست رکورد ریشه با شروع از گره(6,7) ادامه می‌دهیم در اینجا چون نشانوند مقایسهy است و 7>1 پس تنها باید در شاخه چپ این گره جستجو را ادامه دهیم و بنابراین به گره (٦و٥) می رسیم در اینجا نشاوند مقایسهX است باید در هردو شاخه کار جستجو انجام شود. در سمت چپ به گره (٣و٤) می‌رسیم

که جواب نیست در سمت راست گره (۵و۰) را بررسی می کنیم. این رکورد جواب نیست ولی چون نشانوند مقایسهy است و 1>0 پس تنها باید در سمت راست می رویم و به رکورد (٦و۱) می رسیم که جواب است. در این سطح نیز باید در هر دو مسیر جستجو کنیم. در سمت چپ زیر شاخه تهی است و در سمت راست به گره (۲و۷) می‌رسیم که جواب نیست بنابراین مجموعه جواب چنین است: {(1,1),(6,1)}

فایل با ساختار ترای

ساختارtrie گونه ای از درخت است که به ویژه برای جستجو در فایل وقتی که نشانوند جستجو کلمات با طول دلخواه و متغیر باشد استفاده می شود به عنوان مثال فرض کنیم که می خواهیم رکورد فردی به نامSMYTH را از فایل حاوی مشخصات افراد پیدا کنیم اگر بتوان به نحوی به قسمت اسامی آغاز شونده با کاراکترS در فایل رفت و جستجو را از آن قسمت ادامه داد. طبعاً جستجو سریع تر می شود در ادامه جستجو باید نامی را جستجو کنیم که کاراکتر دومشM باشدو... پس از یافتن نام مورد نظر می توان به رکورد مربوط دستیابی داشت.

در همین مثال ساده می بینم که جستجو در این ساختار اساساً با روش های دیگر جستجو که تا کنون دیده ایم تفاوت دارد چون در این روش جستجو کاراکتر به کاراکتر انجام می شود. اگر رکورد مورد نظر موجود باشد جستجو به اندازه فایل بستگی ندارد به علاوه لزوماً همیشه همه کاراکترهای نشانوند جستجو در مقایسه دخالت داده نمی شوند.

در این ساختار فرض بر این است که همه مقادیر کلید (نشانوند جستجو) رشته هایی از یک الفبای مشخص هستند مثلاً اگر مقادیر کلید، عددی باشند این مقادیر به صورت رشته هیا کاراکتر در نظر گرفته می شوند که الفبای آن {0,1,...} است. الفبا همیشه N>1 کاراکتر دارد.

Trie گونه ای از درخت است ولی نوع و معنای گره های موجود در مسیرها با گره های انتهایی فرق دارد. گره های مسیری حاوی «شاخص ها» هستند در حالی که گره های انتهایی، حاوی نشانه روهایی به رکوردهای ذخیره شده هستند. بنابراین ساختارTrie یک استراتژی دستیابی به فایل داده ای است. ضمناً وجود گره انتهایی یعنی رکورد یافته و تمام طول آن پیموده شده است.

مثال: فایل کوچکی فقط حاوی دو رکورد با مقادیرSMYTHNSON, SMITH را در نظر می گیریم. ساختارTrie برای این فایل صورتی است که مقادیر کلید ممکن است قسمت پیشوندی مشترک داشته باشند. یادآوری می شود که اگر دو رشته نا تهیX,Y از یک الفبا را در نظر بگیریم رشتهX را پیشوند رشتهY می گوییم و تنها اگر یک رشته ناتهی دیگر مثل Zاز همان الفبا وجود داشته باشد کهY=X,Z

کلیدهایی که پیشوندی کلیدهای دیگر هستند می توانند مشکل آفرین باشند. برای مثال اگر بخواهیم رکورد با کلیدSM را درTrie شکل بالا درج کنیم باید تدبیری بیاندیشیم. یک راه ساده این است که وقتی بخواهیم چنین رکوردی را درج کنیم در سمت راست کلید آن یک کاراکتر خاص که در الفبا نباشد اضافه کنیم.

بهبود ساختار

اگر بخواهیم رکوردی که کاراکتر I ام نام آن با کارکترI ام کلید رکورد مورد نظر یکسان باشد را پیدا کنیم باید عمل پویش انجام شود اما اگر در ساختار شکل بالا تغییری ایجاد کنیم هزینه عمل پویش کاهش می یابد به عبارت دیگر تنها با یکبار دستیابی برای پی بردن به وجود یا عدم وجود کلید مورد نظر لازم می شود برای ایجاد بهبود در ساختار هر گره از مسیرN+1 فیلد نشانه رو در نظر می گیریم. تبدیل یکی از این نشانه رو ها بهNIT یعنی رکورد با پیشوند پویش شده تا آنجا وجود ندارد وگرنه عمل جستجو در سطح بعدی ادامه می یابد.

برای پیاده سازی این بهبود در هر گره مسیری حداقلN+2 فیلد در نظر گرفته می شود. در حالی که هر گره انتهایی تنها دو فیلد باید داشته باشد.

فیلد او در هر دو گره نوع آن را مشخص می کند: مقدار0 یعنی گره انتهایی و مقدار 1 یعنی گره مسیری است. در حالت اول فیلد دیگر(r1) یک نشانه رو به بلاک حاوی رکورد وجود دارد که کلید یا پیشوند کلیدش تا سطح قبلی پویش شده است.

در حالت دوم هر گره ازn+1 فیلد نشانه رو تشکیل شده است به نحوی که نشانه روی I ام معرف کاراکتر aiالفبا است و نشانه رویn+1 ام نشان دهنده کاراکتر بلانک است.