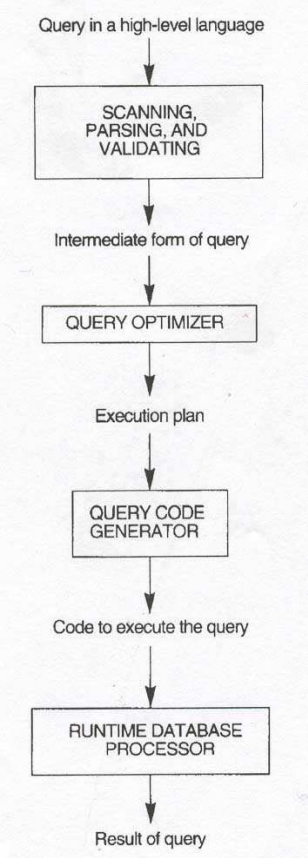
بهینه‌سازی و پردازش پرس و جو:

در این فصل، به تکنیک‌های بکار رفته توسط DMBS برای پردازش، بهینه‌سازی و اجرای پرس و جوهای سطح بالا می‌پردازیم.

پرس و جوی بیان شده در زبان پرس‌و جوی سطح بالا مثل SQL ابتدا باید پویش و تجزیه . معتبر شود. پویشگر (اسکنر) علامت هر زبان، مثل لغات کلیدی SQL، اساس ویژگی، و اساس رابطه، را در متن پرس و جو شناسایی می‌کند، در عوض تجربه کننده، ساختار دستوری پرس و جو را برای تعیین اینکه آیا بر طبق قوانین دستوری زبان پرس و جو تدوین می‌شود یا خیر، چک می‌کند. پرس و جو باید همچنین معتبر شود، با چک کردن اینکه تمام اسامی رابطه و ویژگی معتبر هستند و اسامی معنی‌دار در طرح پایگاه اطلاعاتی ویژهای پرس و جو می‌شوند. نمونه داخلی پرس و جو ایجاد می‌شود، که تحت عنوان ساختار داده‌های درختی بنام درخت پرس و جو می‌باشد. ارائه پرس و جو با استفاده از ساختار داده‌های گراف بنام گراف پرس و جو نیز امکان پذیر است. DOMS باید استراتژی اجرایی برای بازیابی نتیجه پرس و جو از فایل‌های پایگاه اطلاعاتی را هدایت کند.پرس و جو استراتژیهای اجرایی بسیاری دارد. و مرحلهٔ انتخاب، مورد مناسبی برای پردازش پرس وجو تحت عنوان بهینه‌سازی پرس و جو شناخته شده است.

تصویر ۱۸۰۱، مراحل مختلف پردازش پرس و جوی سطح بالا را نشان می‌دهد. قطعه بر نامه بهینه‌ساز پرس وجو، وظیفه ایجاد طرح اجرایی را بعهده دارد و ژنراتور (تولید کننده) که ، کد را برای اجرای آن طرح ایجاد می‌کند. پردازنده پایگاه اطلاعاتی زمان اجرا وظیفه اجرای که پرس و جو را بعهده دارد، خواه در وضعیت کامپایل شده یا تفسیر شده جهت ایجاد نتیجه پرس و جو. اگر خطای زمان اجرا نتیجه شود، پیام خطا توسط پایگاه اطلاعاتی زمان اجرا ایجاد می‌شود.



اصطلاح بهینه‌سازی نام بی مسمایی است چون در بعضی موارد، طرح اجرایی انتخاب شده، استراتژی بهینه نمی‌باشد، آن فقط استراتژی کارآمد معقول برای اجرای پرس و جو است. یافتن استراتژی بهینه، ضامن صرف زمان زیادی است، بجز برای ساده‌ترین پرس و جوها، ممکن است به اطلاعاتی روی

چگونگی اجرای فایل‌ها در فهرست‌های فایل‌ها، اطلاعاتی که ممکن است کاملاً در کاتالوگ DBMS در دسترس نباشد، نیاز باشد. از اینرو، برنامه‌ریزی استراتژی اجرا ممکن است توصیف درست‌تری نسبت به بهینه‌سازی پرس و جو باشد.

برای زبانهای پایگاه اطلاعاتی (دریایی) جهت‌یابی در سطح پایینتر در سیستم‌های قانونی، مثل شبکه DML شبکه‌ای یا MOML سلسله مراتبی، برنامه نویس باید، استراتی اجرای پذیرش و جو را انتخاب کند ضمن اینکه برنامه پایگاه اطلاعاتی را می‌نویسد. اگر DBMS فقط زبان جهت‌یابی را ارائه دهد. فرصت و نیاز محدودی برای بهینه‌سازی پرس وجوی وسیع توسط DBMS وجود دارد، در عوض به برنامه نویس قابلیت انتخاب استراتژی اجرایی بهینه ارائه می‌شود. بعبارت دیگر، زبان پرس و جو در سطح بالا، مثل SQL برای DBMSهای رابطه‌ای یا OQL برای DBMSهای مقصد، در ماهیت تفریطی‌تر است. چون آنچه نتایج مورد نظر پرس و جو است بغیر از شناسایی جزئیات چگونگی بدست آمدن نتیجه، را تعیین می‌کند. بهینه‌سازی پرس و جو برای پرس و جوهایی ضروی است که در زبان پرس و جوی سطح بالا تعیین می شوند. ما روی توصیف بهینه‌سازی پرس و جو در زمینه ROBMS تمرکز می‌کنیم چون بسیاری از تکنیک‌هایی که توصیف می کنیم برای، برای ODBMSها تطبیق یافته‌اند. DBMS رابطه‌ای باید استراتژیهای اجرای پرس و جوی دیگری را ارزیابی کند و استراتژی بهینه یا کارآمد معقولی را انتخاب کند. هر DBMS ، تعدادی الگاریتم دسترسی به پایگاه اطلاعاتی کلی دارد که علامتهای رابطه‌ای مثل SELECT یا JOIN یا ترکیبی از این عملیات ها را اجرا می‌کند. تنها استراتژیهای اجرایی که می‌توانند توسط الگاریتم‌های دسترسی DBMS اجرا شوند و برای طراحی پایگاه اطلاعاتی

فیزیکی ویژه و پرس و جوی خاص بکار روند، می‌توانند توسط قطعه برنامه بهینه‌سازی پرس و جو در نظر گرفته شوند.

ما در بخش ۱۸۰۱ با بحث کلی چگونگی ترجمه پرس و جوهای SQL به پرس و جوهای جبری رابطه‌ای و در بهینه‌شدن آنها کار را شروع می‌کنیم. بعد ما روی الگاریتم‌ها برای اجرای عملیات‌های رابطه‌ای در بخش ۱۸۰۲ بحث می‌کنیم. بدنبال این مطلب، بررسی از استراتژیهای بهینه‌سازی پرس و جو را ارائه می‌دهیم. دو تکنیک اصلی برای اجرای بهینه‌سازی پرس و جو وجود دارد. اولین تکنیک بر اساس قوانین ذهنی جهت ترتیب دادن عملیات‌ها در استراتژی اجرای پرس و جو می‌باشد. ذهن قانونی است که بخوبی در اکثر موارد عمل می‌کند ولی برای کار مناسب در هر مورد کنش تضمین نمی‌شود. قوانین عملیات‌ها را در درخت پرس وجو مجدداً ترتیب می‌دهند. دومین تکنیک شامل برآورد هزینه استراتژیهای اجرای متفاوت و انتخاب طرح اجرایی با پایین‌ترین هزینه برآورد است. دو تکنیک معمولاً در بهینه ساز پرس و جو (باهم ترکیب می‌شوند) بهم ملحق می‌گردند. ما روی بهینه‌سازی ذهنی در بخش ۱۸۰۳ و برآورد هزینه در بخش ۱۸۰٤ بحث می‌کنیم. بعد بررسی مختصری از عوامل در نظر گرفته شده در طول بهینه‌سازی پرس و جو در RDBMS بازرگانی ORACLL= در بخش ۱۸۰۵ را ارائه می‌دهیم. بخش ۱۸۰٦، نوعی بهینه‌سازی پرس و جوی معنایی را ارائه می‌دهد که در آن محدودیت‌های شناخته شده برای پرداختن به استراتژیهای اجرایی پرس و جوی کارآمد استفاده می‌شوند.

۱۸۰۱ – ترجمه پرس و جوهای SQL به پرس و جوهای رابطه‌ای:

در عمل، SQL زبان پرس وجویی است که در اکثر RDBMS های بازرگـانی اسـتفاده مـی‌شـود. پـرس وجوی SQL ، ابتدا به عبارت جبری رابطه‌ای توسعه یافته معادل، نمایانگر ساختار داروهـای درخـت پرس و جو، ترجمه می‌شود و بعد بهینه‌سازی می‌شود. پرس و جوهای SQL به بلوکهای پرس و جـو تجزیه می‌شوند، که واحدهای اساسی را تشکیل می‌دهند که می‌توانند به عملکردهای جبـری ترجمـه شوند و بهینه‌سازی شوند. بلوک پرس و جو شامل عبارت SELECT- FROM-WHERE تکی و بنـدهای Groop By و HAVING است چنانچه این‌ها بخشی از بلوک باشند. از اینرو، پرس و جوهای تو در تـو در پرس و جو بعنوان بلوکهای پرس و جوی مجزا شناسایی می‌شوند. چون SQL شامل عملکردهـای گروهی، مثل MAX ، COUNT,SUM می‌باشد، این عملگرهـا بایـد در پـرس و جـوی جبـری توسـعه یافتـه‌ای شـامل شـوند، همانطوریکـه در بخـش ۷۰۵ توصـیف شـد. پـرس و جـوی SQL در رابطـه EMPLOEE در تصویر ۷۰۵ را در نظر بگیرید:

این پرس و جو شامل، پرس و جوی فرعی تو در تو است و از اینرو به دو بلـوک تجزیـه مـی‌شـود.

بلوک درونی بدین صورت است:

و بلوک بیرونی بدین صورت می باشد:

که C نمایانگر نتیجه حاصله از بلوک درونی است. بلوک درونی به عبارت جبری رابطه‌ای توسعه یافته زیر ترجمه شده است:

و بلوک بیرونی به عبارت زیر ترجمه شده است:

بهینه‌ساز پرس و جو، طرح اجرایی را برای هر بلوک انتخاب می‌کند. ما باید اشاره کنیم به در مثال فوق، بلوک درونی نیاز به ارزیابی شدن دارد تنها زمانی که، حداکثرحقوقی که بعکار می‌رود که بعنوان ثابت C، توسط بلوک بیرونی استفاده می‌شود. ما اینرو پرس و جوی تودرتوی غیرمرتبط نامیدیم (در فصل ۸). آن برای بهینه‌سازی پرس و جوهای تو در توی مرتبط پیچیده‌تر، خیلی سخت‌تر است، جایی که متغیر Tuple از بلوک بیرونی در بند WHERE در بلوک درونی ظاهر می‌شود.

۱۸۰۲- الگاریتم های انسانی برای اجرای عملیاتهای پرس و جو:

RDBMS شامل الگاریتم‌هایی برای اجرای انواع مختلف عملیاتهای رابطه‌ای است که می‌توانند در استراتژی اجرای پرس و جو نمایان شوند، این عملیات‌ها شامل عملیاتهای جبری بیسیک (اصلی) و توسعه یافته مورد بحث در فصل ۷ ، و در بسیاری موارد، الحاقاتی از این عملیات‌ها می‌باشد. برای هر یک از این عملیات ها یا الحاقی از عملیات‌ها، یک یا چند الگاریتم برای اجرای عملیات‌ها در دسترس قرار دارند. الگاریتم ممکن است فقط برای ساختارهای ذخیره خاص مسیرهای دستیابی بکار روند، در اینصورت ، تنها در صورتی استفاده می‌شود که فایل های موجود در عملیات شامل این مسیرهای دستیابی هستند. در این بخش، ما به الگاریتم‌های نمونه بکار رفته برای اجرای SEKECT ، JOIN و دیگر عملیاتهای رابطه‌ای می‌پردازیم. ما بحث مرتب کردن خارجی را در بخش ۱۸۰۲۰۱ آغاز می‌کنیم که در قلب عملیاتهای رابطه‌ای قرار دارد که از استراتژیهای ادغام کردن به مرتب کردن استفاده می‌کند. بعد ما به الگاریتم‌هایی برای اجرای عملیات SELECT در بخش ۱۸۰۲۰۲ می‌پردازیم،

به عملیات JOIN در بخش ۱۸۰۲۰۳ و عملیات PRIJECT و عملیاتهای مجموعه در بخش IE ۱۸۰۲ و عملیات‌های گروهی و جمعی در بخش ۲ .۲ . ۱۸ می‌پردازیم.

۱. ۲. ۱۸- مرتب کردن خارجی:

مرتب کردن، یکی از الگاریتم‌های اولیه بکار رفته در پردازش پرس و جو است. برای مثال، به هر وقت پرس و جوی SQL ، بعد ORDER BY را تعیین می‌کند، نتیجه پرس و جو باید مرتب گردد. مرتب کردن، مؤلفه کلیدی در الگاریتم‌های مرتب کردن- ادغام کردن (مرتب-ادغام) بکار رفته برای Join و عملیاتهای دیگر، دور الگاریتم‌های حذف کپی برای عملیات PROYECT است. ما روی بعضی از این الگاریتم‌ها در بخش ۳. ۲. ۱۸ و ٤. ۰۲ ۱۸ بحث خواهیم کرد. توجه کنید که مرتب کردن در صورتی که اجتناب می‌شود که شاخص مناسب برای امکان دسترسی مرتب شده به ثبت‌ها وجود دارد.

مرتب کردن خارجی به الگاریتم‌های مرتب کردن اشاره می‌کند که برای فایل های بزرگ ثبت های ذخیره شده روی دیسک مناسب هستند که در حافظه اصلی، مثل اکثر فایل های پایگاه اطلاعاتی تناسب نمی یابد. الگاریتم مرتب کردن خارجی نمونه از استراتژی مرتب- ادغام استفاده می‌کند، که با مرتب کردن- فایل های فرعی کوچک بنام اجراها در فایل اصلی شروع می‌شود و بعد اجراها مرتب شده ادغام می‌شوند، فایل های فرعی مرتب شده بزرگتری ایجاد می‌شوند که بترتیب ادغام می‌شوند. الگاریتم ادغام -مرتب، مثل دیگر الگاریتم های پایگاه اطلاعاتی به فاضی بافر در حافظه اصلی نیاز

دارد، جایی که مرتب کردن واقعی و ادغام اجراها انجام می شود. الگاریتم اصلی (سیبک) شرح داده شده در تصویر ۱۸۰۲ ، شامل دو مرحله است: (۱) فاز یا مرحله مرتب کردن و (۲) مرحله ادغام.

در مرحله مرتب کردن، اجراهای فایلی که می‌تواند در فضای باز موجود تناسب یابد در حافظه اصلی خوانده می‌شوند و با استفاده از الگاریتم مرتب کردن داخلی مرتب می‌شود عقب دیسک بعنوان فایل‌های فرعی مرتب شده متوفی نوشته می‌شود. اندازه اجرا و تعداد اجراهای آغازین $\left(\hat{}R\right)$ توسط تعداد بلوکهای فایل (b) و فضای بافر موجود (NB) بیان می‌شود. برای مثال اگر $N_B = 5$ بلوکو اندازه قایل ۱۰۲٤=b بلوک باشد، بعد $nR = \left[\left(b/n_B\right)\right]$ یا ۲۰۵ اجرای آغازین در هر اندازه ۵ بلوک است. از اینرو، بعد از مرحله مرتب کردن، ۲۰۵ اجرای مرتب شده بعنوان فایل‌های فرعی موقتی روی دیسک ذخیره می‌شوند. اجرای مرتب شده بعنوان فایل‌های فرعی موقتی و روی دیسک ذخیره می‌شوند.

در مرحله ادغام شدن، اجراهای مرتب شده، در طول یکد یا چند گذر ادغام می‌شوند. درجه ادغام شدن $\left(d_m\right)$ تعداد اجراهایی است که می‌توانند با همدیگر در هر گذر ادغام شوند. در هر گذر، یک بلوک بافر، برای حفظ یک بلوک از هر اجرای ادغام شده نیاز می‌باشد، و یک بلوک برای تشکیل یک بلوک نتیجه ادغام لازم است . از اینرو، d_m کوچکتر از $\left(n_B - 1\right)$ و n_R است و تعداد گذرها، $\left[\left(\log_{dm}(n_R)\right)\right]$ است. در مثالها، $d_m = 4$ است. لذا، ۲۰۵ اجرای مرتب شده آغازین در ۲۵ تا در پایان اولیه گذر ادغام می‌شود: که بعد به ۱۲، بعد ٤ بعد یک اجرا ادغام می‌شوند، که بدین معنی است که چهارگذر لازم می‌باشد. حداقل d_m از ۲، عملکرد بدترین مورد الگاریتم را ارائه می‌دهد که بدین قرار است:

$$(z+b)+(z*(b*(\log_2 b))$$

اولین جمله، تعداد دسترسی‌های بلوک برای مرحله مرتب سازی را نشان می‌دهد، چون هر بلوک فایل دو برابر دسترسی می‌شود، یکبار برای خواندن در حافظه، یکبار برای نوشتن ثبت‌ها دیسک بعد از مرتب کردن. دومین جمله، تعداد دسترسی‌های بلوک برای مرحله ادغام کردن را نشان می‌دهد، با فرض اینکه بدترین مورد d_m از ۲ وجود دارد. بطور کلی، ثبت وقایع در مبنای d_m و عبارت برای تعداد دسترسی‌های بلوک نوین قرار می‌شود:

$$(z+b)+(z*(b*(\log d_m b))$$

تصویر ۱۸۰۲- شرح الگاریتم ادغام - مرتب کردن برای مرتب کردن خارجی:

۲. ۲. ۱۸- اجرا و پیاده‌سازی عملیات SELECT :

تعداد Optionهایی (انتخاب‌ها) برای اجرای عملیات SELECT وجود دارد، که بعضی به فایل دارای مسیرهای دستیابی خاص بستگی دارند و تنها برای انواع معین شرایط انتخاب بکار می‌رود. ما به الگاریتم‌هایی جهت اجرای SELECT در این بخش می‌پردازیم. ما از عملیاتهای زیر استفاده می‌کنیم که روی پایگاه اطلاعاتی رابطه‌ای در تصویر ۵۰۷ مشخص شده و بحث ما را روشن می‌سازد:

(opl)

متدهای جستجو برای انتخاب ساده:

تعدادی الگاریتم های جستجو برای انتخاب ثبت‌ها از فایل امکان‌پذیر می‌باشند، چون ثبت‌های فایل نامیده می شوند، چون ثبت‌های فایل را برای جستجو و بازیابی ثبت‌هایی که شرایط انتخاب را

برآورده می‌سازند، پویش می‌کنند. اگر الگاریتم جستجو شامل کاربرد شاخص باشد، جستحوی شاخص پویش شاخص نامیده می‌شد. متدهای جستجوی زیر (S1 تا s6) مثالهایی از الگاریتم‌های جستجو هستند که می‌توانند برای اجرای عملیات انتخاب بکار روند:

- s1 : جستجوی خطی (روش برنامه‌سازی پر قدرت): بازیابی هر ثبت در فایل، و تست اینکه آیا مقادیر ویژگی آن، شرط انتخاب را براورده می‌سازد یا خیر.
- S2: جستجوی بنیادی (دودویی): اگر شرط انخاب شامل قیاس تساوی روی ویژگی کلیدی باشد که روی آن فایل مرتب می‌شود، جستجوی بنیادی، که نسبت به جستجوی خطی کارآمدتر است، می‌تواند بکار رود. مثال OP1 است چنانچه ssn ، ویژگی کلیدی یا شاخص اولیه(یا کلید hash) باشد، برای مثال، SNN-‘123456789’ در opt، شاخص اولیه یا کلید hosh) برای بازیابی ثبت استفاده می‌شود، توجه کنید که این شرط، ثبت تکی را بازیابی می‌کند.
- S4: کاربرد شاخص اولیه برای بازیابی ثبت‌های متعدد: اگر شرط انتخاب شدن قیاس تساوی روی ویژگی غیر کلیدی با شاخص خدشه‌سازی باشد، برای مثال $DNO = 5$ در Op^3 ، شاخص را برای بازیابی کل ثبت‌ها در برآورده ساختن شرط، استفاده کنید.
- S6: بکارگیری شاخص ثانویه (درخت B_+^-) روی قیاس تساوی: این متد جستجو می‌تواند برای بازیابی ثبت تکی بکار رود چنانچه فیلد نمایه‌سازی (شاخص‌سازی) کلید باشد یا برای بازیابی ثبت‌های متعدد بکار می‌رود چنانچه فیلد شاخص‌سازی کلید نباشد، این می‌تواند برای مقایساتی شامل <,>,=>یا => بکار رود. در بخش ۱۸.٤.۳، ما به چگونگی توسعه فرمول‌هایی می‌پردازیم که هزینه‌دستیابی این متدهای جستجو را در اصطلاحات تعداد دستیابی‌های بلوک و

زمان دستیابی برآورد می‌کند. !Method Sبرای هر فایلی استفاده می‌شود ولی تمام متدهای دیگر به داشتن مسیر دستیابی مناسب روی ویژگی‌بکار رفته در شرط انتخاب بستگی دارند. متدهای S4 و 6، می‌توانند برای بازیابی ثبت‌ها در دامنه معین بکار روند برای مثال 3000 <= SALARY <= 305000 پرس و جوها شامل این شرط‌ها، پرس وجوهای دامنه نیامد به می‌شوند.

متدهای جستجو برای انتخاب پیچیده:

اگر شرط عملیات SELECT، شرط تقارنی و مرتبط باشد، در اینصورت اگر از چندین شرط ساده در ارتباط با ارتباط منطقی and مثل op4 فوق تشکیل شود، DBM می‌تواند از متدهای اضافی زیر برای اجرای عملیات استفاده کند:

S7: انتخاب تقارنی یا ارتباطی با استفاده از شاخص اختصاص: اگر ویژگی شامل شده در هر شرط ساده متکی در شرط تقارنی، مسیر دستیابی داشته باشد که به کاربرد یکی از متدهای S2 تا S6 امکان عمل دهد، از آن شرط برای بازیابی ثبت‌های استفاده کنید و بعد کنترل کنید آیا هر ثبت بازیابی شد، شرایط ساده باقیمانده در شرط تقارنی را برآورده می‌کند یا خیر.

S8 : انتخاب تقارنی (ارتباطی) با استفاده از شاخص مرکب: اگر دو یا چند ویژگی در شرایط تساوی در شرط تفاوتی شامل شدند و شاخص مرکب در فیلدهای مرکب وجود داشته باشد، برای مثال اگر شاخص روی کلید مرکب (ESSN, PNO) در فایل Works ON برای OPS ایجاد شده باشد، می توان از شاخص مستقیماً اشاره کرد.

S9: انتخاب تفاوتی با محل تقاطع اشاره گرهای ثبت : اگر شاخص های ثانویه روی بیش از یک فیلد ساقل شده در شرایط ساده در شرط تقارنی در دسترس باشند، و اگر شاخص ها شامل اشاره گرهای ثبت باشند، بعدو دو شاخص می تواند برا بازیابی مجموعه اشاره گرهای ثبت استفاده شود که شرط اختصاصی را برآورده می سازد.

محل تقاطع این مجموعه اشاره گرهای ثبت، اشاره گرهای ثبتی را ارائه می دهد که شرط تقارنی را برآورده می سازد، که بعد برای بازیابی آن ثبت ها مستقیماً استفاده می شوند. اگر فقط بعضی از شرایط ، شاخص های ثانویه داشته باشند،هر ثبت بازیابی شده، برای تعیین اینکه آیا آن شرایط باقیمانده را برآورده می سازد یا خیر، بعداً تست می شود.

هر وقت شرط تکی ، انتخابی مثل OP3, OP2, OP1 را تعیین می کند، می توانیم فقط چک کنیم که آیا مسیر دستیابی روی ویژگی شامل شده در آن شرط وجود دارد یا خیر. اگر مسیر دستیابی وجود داشته باشد،متد مربوط به آن مسیر دستیابی استفاده می شود، درغیر اینصورت،روش جستجوی خطی برنامه سازی پرقدرت (boute force) در متد S1 می تواند استفاده شود. بهینه سازی پرس و جو برای عملیات SELECT برای شرایط انتخاب تفاوتی لازم است، هر وقت بیش از یک ویژگی شامل شده در شرطها، دارای مسیر دستیابی می باشند. بهینه ساز باید، مسیر دستیابی را انتخاب کند که کمترین ثبت ها در کارآمدترین راه با برآورد هزینه های مختلف را بازیابی می کند و متد با حداقل هزینه برآورده شده را انتخاب می کند. وقتی بهینه ساز بین شرط های ساده متعدد در شرط انتخاب تقارنی،انتخاب می کند، آن ، انتخاب پذیری هر شرط را در نظر می گیرد. انتخاب پذیری، بعنوان نسبت مقدار ثبت هایی تعریف می شود که شرط را نسبت به کل تعداد ثبت ها در فایل برآورده می سازد، و لذا تعداد بین

صفر و ۱ است . انتخاب پذیری صفر بدین معنی است که هیچ ثبتی ، شرط را برآورده نمی سازد و یک بدین معنی است که تمام ثبت ها ،شرط را برآورده می سازند. گر چه انتخاب پذیری های دقیق تمام شرط ها ممکن است در دسترس نباشند، برآوردهای انتخاب پذیری ها ،اغلب در کاتالوگ DBMS حفظ می گردند و توسط بهینه ساز استفاده می شوند. برای مثال، برای شرط تساوی روی ویژگی کلیدی رابطه r(R) در رابطه (ثبت ها)تعداد Tople |r(R)| است ،جایی که S=1/|(R)|,(R) است.

برای شرط تساوی روی ویژگی با نامقادیر متمایز، S فقط ||(R)|,/i)|/(|,(R) یا 1/i برآورد می شود، با فرض بر اینکه ثبت ها در میان مقادیر متمایز توزیع می شوند. تحت این فرضیه ، |r(R)1/i ثبت ها ،شرط انتخاب را با انتخاب پذیری s برآورده می سازد در حالت |r(R)*s برآورده می شود. هر چه این برآورد کوچکتر باشد، تمایل استفاده از آن اولین شرط برای بازیابی ثبت ها بیشتر است.

در مقایسه با شرط انتخاب تقارنی (ارتباطی) ، شرط گسسته برای پردازش و بهینه سازی سخت تر است.

براث مثال ، opu را در نظر بگیرید. $\sigma_{DND} = S$:(op 49)

با این شرط ، بهینه سازی اندکی می تواند انجام شود، چون ثبت های برآورد کننده شرط گسسته ، اتحاد یا اجتماع ثبت ها در برآورده ساختن شرط های اختصاصی می باشند. از اینرو، اگر هر یک از شرط ها ، مسیر دستیابی نداشته باشند،ما مجبوریم از روش جستجوی خطی برنامه سازی پرقدرت استفاده کنیم. تنها در صورتی که مسیر دستیابی روی هر شرط موجود باشد،می توان انتخاب را با بازیابی ثبت های برآورد کننده هر شرط، یا id های ثبت آنها بهینه سازی کرد و بعد از عملیات اجتماع برای مرتفع کردن و حذف کپی ها استفاده کرد.

DBMS متدهای زیادی در دسترس دارد و متدهای اضافی نیز دارد. بهینه ساز پرس و جو باید مورد مناسبی را برای اجرای هر عملیات SELECT در پرس و جو استفاده کند. این بهینه سازی از فرمولی استفاده می کند که هزینه ها را برای هر متد دستیابی قابل دسترس برآورد می کند،همانطوریکه در بخش ۱۸۰٤ بحث می شود بهینه ساز، متد دستیابی را با حداقل هزینه برآورد شده،انتخاب می کند.

۳. ۲. ۱۸ : اجرای عملیات JOIN : عملیات JOIN یکی از طولانی ترین عملیات ها در پردازش پرس و جو است. بسیاری از عملیات های اتصال مواجه شده در پرس و جو، انواع EQUIJOIN ، NATURAL JOIN هستند، لذا فقط این دو تا را در اینجا در نظر می گیریم. برای بقیه فصل، اصطلاح اتصال به EQUIJOIN اشاره می کند. راههای ممکن زیادی برای اجرای اتصال دو راهی وجود دارد، که اتصال روی دو فایل است. اتصال ها شامل بیش از دو فایل ، اتصالهای چندراهی نامیده می شوند. تعدادی راههای ممکن برای اجرای اتصال های چندراهی بسرعت رشد می کنند. در این بخش، ما روی تکنیک هایی برای اجرای فقط اتصال های دوراهی بحث می کنیم. برای نشان دادن بحث خود، به طرح رابطه ای تصویر ۵ .۷ در رابطه های PROJECT, DEPARTMENT, EMPLOYEE اشاره می کنیم. الگاریتم هایی که در نظر می گیریم ، جهت عملیاتهای اتصال بفرم زیر است $Rx_{A=B}$ S که A,B ویژگیهای R و S حوزه سازگار است. متدهایی که بحث می کنیم، بفرم کلی تر اتصال توسعه می یابند. ما چهار تا از متداول ترین تکنیک ها را برای اجرای این اتصال ، با استفاده از عملیاتهای مثال زیر نشان می دهیم:

(op6):

(op7):

متدهای برای اجرای اتصال ها:

J1 : اتصال با حلقه تودرتو (برنامه سازی پرقدرت) : برای هر ثبت t در R (حلقه بیرونی) هر ثبت s را از S بازیابی کنید (حلقه درونی) و تست کنید آیا دو ثبت ، شرط اتصال t[A]=s[B] را برآورد می سازند یا خیر.

J2 : اتصال با حلقه تکی: اگر شاخص (یا کلید hosl) برای یکی از دوویژگی اتصال B از S ، وجود داشته باشد، هر ثبت t را در R (حلقه تکی) بازیابی کنید و بعد از ساختار دستیابی برای بازیابی تمام ثبت های تطبیق پذیری s از S که s[B] = t[A] را برآورده می سازند، استفاده کنید.

J3 : اتصال مرتب (کردن) – ادغام : اگر ثبت های R و S توسط مقدار ویژگی های اتصال B,A مرتب شوند، می توان اتصال را در کارآمدترین راه ممکن اجرا کرد. هر دو فایل در ترتیب ویژگی های اتصال تطبیق پذیری ثبت هایی پویش می شوند که دارای مقادیر یکسانی برای B,A داشتند. اگر فایل ها مرتب نشوند، آنها ابتدا با استفاده از مرتب کردن خارجی مرتب می شوند. در این متد، جفت های بلوکهای فایل در بافرهای حافظه در ترتیب کپی می شوند و ثبت های هر فایل تنها یکبار هر یک برای تطبیق پذیری با فایل دیگر، پویش می شوند، مگر اینکه هر دو B,A ویژگیهای غیر کلیدی باشند، که در آن مورد، متد نیاز به تعدیل شدن دارد. طرح الگاریتم اتصال مرتب کردن – ادغام در تصویر (a) ۳ . ۱۸ ارائه شده است. ما از R(i) برای اشاره به ثبت i ام در R استفاده می کنیم.

تغییرات اتصال ادغام شدن – مرتب کردن می تواند زمانی بکار رود که شاخص های ثانویه روی هر دو ویژگی های اتصال وجود دارند. شاخص ها،قابلیت دسترسی به ثبت ها در ترتیب ویژگی های اتصال را ارائه می دهند، ولی ثبت ها در سراسر بلوکهای فایل پخش می شوند، لذا این متد کاملاً غیر

کارآمد است، تحت عنوانی که هر دستیابی به ثبت ممکن است شامل دسترسی به بلوک دیسک متفاوتی باشد.

تصویر ۳. ۱۸:

```
(a)  sort the tuples in R on attribute A; (*assume R has n tuples (records) *)
     sort the tuples in S on attribute B; (*assume S has m tuples (records) *)
     set i←1, j←1;
     while (i ≤ n) and (j ≤ m)
     do{   if R(i)[A] > S(j)[B]
                 then     set j←j+1
           elseif R(i)[A] < S(j)[B]
                 then     set i←i+1
                 else {   (* R(i)[A] = S(j)[B], so we output a matched tuple*)
                          output the combined tuple <R(i), S(j)> to T;
                                (*output other tuples that match R(i), if any*)
                                set l←j+1;
                                while (l ≤ m) and (R(i)[A] = S(l)[B])
                                do   {        output the combined tuple <R(i), S(l)> to T;
                                              set l←l+1
                                      }
                                (*output other tuples that match S(j), if any*)
                                set k←i+1;
                                while (k ≤ n) and (R(k)[A] = S(j)[B])
                                do   {        output the combined tuple <R(k), S(j)> to T;
                                              set k←k+1
                                      }
                                set i←k, j←l
                       }
     }

(b)  create a tuple t[<attribute list>] in T' for each tuple t in R;
          (*T' contains the projection result before duplicate elimination*)
     if <attribute list> includes a key of R
          then T←T'
          else {   sort the tuples in T';
                   set i←1, j←2;
                   while i ≤ n
                         do   {   output the tuple T'[i] to T;
                                  while T'[i] = T'[j] and j ≤ n do j←j+1; (*eliminate duplicates*)
                                  i←j; j←i+1
                              }
```

اتصال hash : ثبت های فایل های S,R ،هر دو در فایل hash یکسانی ، با استفاده از تابع hashing یکسانی روی ویژگی های اتصالی A از B,R از S بعنوان کلیدهای hash ، hash می شوند. ابتدا ، گذر تکی از طریق فایل با ثبت های کمتر، ثبت های خود را در مخازن فایل hash ، hash می کند، این مرحله تقسیم کردن (پارتیشن)نامیده می شود، چون ثبت های R به مخازن hash تقسیم می شوند.

بقیه تصویر ۳ .۱۸:

```
(c)  sort the tuples in R and S using the same unique sort attributes;
     set i←1, j←1;
     while (i ≤ n) and (j ≤ m)
     do  {     if R(i) > S(j)
               then  {     output S(j) to T;
                           set j←j+1
                     }
               elseif R(i) < S(j)
               then  {     output R(i) to T;
                           set i←i+1
                     }
               else set j←j+1 (*R(i)=S(j), so we skip one of the duplicate tuples*)
         }
     if (i ≤ n) then add tuples R(i) to R(n) to T;
     if (j ≤ m) then add tuples S(j) to S(m) to T;

(d)  sort the tuples in R and R using the same unique sort attributes;
     set i←1, j←1;
     while (i ≤ n) and (j ≤ m)
     do  {     if R(i) > S(j)
               then        set j←j+1
               elseif R(i) < S(j)
               then        set i← i+1
               else  {     output R(i) to T; (* R(i)=S(j), so we output the tuple *)
                           set i←i+1, j←j+1
                     }
         }

(e)  sort the tuples in R and S using the same unique sort attributes;
     set i←1, j←1;
     while (i ≤ n) and (j ≤ m)
     do  {     if R(i) > S(j)
               then        set j←j+1
               elseif R(i) < S(j)
               then  {     output R(i)to T; (* R(i) has no matching S(j), so output R(i)*)
                           set i←i+1
                     }
               else        set i←i+1, j←j+1
```

در دومین مرحله بنام مرحله آزمایش، گذر تکی از طریق فایل دیگر (S) هر یک از ثبت های خـود را برای آزمایش مخزن مناسب hash می کند و آن ثبت با تمام ثبت های تطبیق پـذیر و برابـر از R در آن مخزن ترکیب می شود. این توصیف ساده شده از اتصال hash ،فرض می کند کـه دو فایـل کـوچکتر کاملاً در مخازن حافظه بعد از اولین مرحله تناسب می یابند. ما روی تغییرات اتصال – hash بحث می کنیم که به این فرضیه زیر نیازی ندارد.

در محل، تکنیک های J1 تا J4 ،با دسترسی به کل بلوکهای دیسک فایل بغیر از ثبت های اختصاصی اجرا می شوند. بسته به فضای بافر موجود در حافظه ،تعداد بلوکهای خوانده شده از فایل می تواند تنظیم گردند.

اثرات فضای بافر موجود و عامل انتخاب اتصال روی عملکرد اتصال: فضای بافر موجود، اثر مهمی روی الگاریتم های گوناگون اتصال دارد. ابتدا ، روش حلقه تو در تو را در نظر بگیرید (J1) . دوباره به عملیات OP6 فوق توجه کنید، فرض کنید که تعداد بافرهای موجود در حافظه اصلی برای اجرای اتصال $n_B = 7$ بلوک است. برای مثال ، فرض کنید که فایل DEPARTMENT شامل $r_D = 50$ بلوک دیسک است و فایل Employee شامل $r_E = 6000$ ثبت ذخیره شده در $b_E = 2000$ بلوک دیسک است. این مزیتی در خواندن بلوک های بسیار در حد ممکن در حافظه از فایلی است که ثبت های آن برای حلقه بیرونی استفاده می شوند. بعد الگاریتم می تواند یک بلوک را در زمان برای فایل حلقه درونی بخواند و از ثبت های خود برای آزمایش بلوکهای حلقه بیرونی در حافظه برای برابری ثبت ها استفاده کند. این ، کل تعداد دستیابی های بلوک را کاهش می دهد. بلوک بافر اضافی برای شامل شدن ثبت های حاصله بعد از اینکه بهم متصل می شوند، لازم است و فهرست و محتویات این بلوک بافر در فایل نتیجه ضمیمه می شوند، فایل دیسکی که شامل نتیجه اتصال است، هر زمانی که آن بایگانی می شود. این بلوک بافر برای حفظ ثبت های نتیجه اضافی مجدداً استفاده می شود.

در اتصال حلقه تودرتو آن تفاوتی را ایجاد می کند که کدام فایل برای حلقه بیرونی انتخاب شود و کدامیک برای حلقه درونی انتخاب شود. اگر Employee برای حلقه بیرونی بکار رود، هر بلوک

Employee یکبار خوانده می شود، که بدین قرار است: کل تعداد بلوکهای دستیابی شده برای فایل بیرونی = b_E

تعداد دفعات $(n_B = 2)$ بلوک فایل خروجی بارگیری می شوند = $[(b_E = /(n_B - 2)]$

کل تعداد بلوکهای دستیابی شده برای فایل درونی = $b_D * [b_E /(n_B - 2)]$

از اینرو، ما کل تعداد دستیابی های بلوک زیر را بدست می آوریم:

$$B_E + ([b_E / C$$

بعبارت دیگر، اگر ما از ثبت های DEPARTMENT در حلقه بیرونی استفاده کنیم، با تقارن ، کل تعداد دستیابی های بلوک زیر را بدست می آوریم:

$$b_D + (b_D /($$

الگاریتم اتصالی از بافر برای حفظ ثبت های متصل شده در فایل نتیجه استفاده می کند. وقتی بافر بایگانی می شود، آن در دیسک نوشته می شود و مجدداً استفاده می شود. اگر فایل نتیجه عملیات اتصال، b_{RES} بلوک دیسک داشته باشد، هر بلوک یکبار نوشته می شود، لذا ، b_{RES} دستیابی بلوک اضافی باید به فرمول های قبلی، به م نظور برآورد کل هزینه عملیات اتصال، افزوده گردد. کنترل های یکسانی برای فرمول ها بعداً برای الگاریتم های اتصال دیگر توسعه یافت. همانطوریکه این مثال نشان می دهد، آن مزیتی برای استفاده فایل با بلوکهای کمتر بعنوان فایل حلقه بیرونی در اتصال حلقه تودرتو است.

عامل دیگری که روی عملکرد اتصال اثر می گذارد ، بویژه متد حلقه تکی J2 ، درصد ثبت ها در فایلی است که با ثبت ها در فایل دیگر متصل می شود. ما اینرا فاکتور انتخاب اتصال فایل با توجه به

شرط اتصال مساوی با فایل دیگر می نامیم. این فاکتور به شرط اتصال مساوی ویژه بین دوفایل بستگی دارد. برای نشان دادن این ، عملیات op7 را در نظر بگیرید، که هر ثبت DEPARTMENT را با ثبت Employee برای مدیر آن دپارتمان متصل می کند. در اینجا، هر ثبت DEPARTMENT انتظار می رود با ثبت تکی employee متصل شده، ولی بسیاری از ثبت های Employee متصل نخواهند شد. فرض کنید که شاخص های ثانویه در هر دو ویژگی های SSN از Employee و MGRSSN از OEPARTMENT ، با تعداد سطوح شاخص $X_{SSN} = ٤$ و $X_{MGRSSN} = ٢$ وجود دارد. ما درو انتخاب برای اجحرای متد J2 داریم. اولی هر ثبت Empyzb را بازیابی می کند و بعد از شاخص روی MGRSSN از DEPARTMGNT برای یافتن ثبت تطبیق پذیری DEPARTMGNT برای یافتن ثبت تطبیق پذیری DEPARTMGNT استفاده می کند. در این مورد، هیچ ثبت تطبیق پذیری برای کارمندانی که بر دپارتمان مدیریت نمی کنند، مشاهده نمی شود. تعداد دستیابی های بلوک برای این مورد تقریباً بدین قرار است:

$$b_E + (r_E * ($$

دومین انتخاب هر ثبت DEPARTMGNT را بازاریابی می کند و بعد از شاخص روی EMPLOYEE برای یافتن ثبت EMPLOYEE میر تطبیق پذیری استفاده می کند. در این خصوص، هر ثبت DEPARTMGNT، یک ثبت تطبیق پذیری Empoyee دارد. تعداد دستیابی های بلوک برای این مورد، بدین قرار میشود:

$$b_D + (r_D + (\ $$

دومین انتخاب کارآمدتر است چون فاکتور انتخاب اتصال DEPARTMGNT با توجه به شرط اتصال SSN=MGRSSN ، یک است، در عوض فاکتور انتخاب اتصال Employee با توجه به همان شرط اتصال یکسان (S0/S000) یا ۰/۰۱ است. برای متد J2 ، یا فایل کوچکتر از فایلی که برای هر ثبت تطبیق و برابری دارد، باید در حلقه اتصال (بیرونی) استفاده شود. این برای ایجاد شاخص برای اجرای عملیات اتصال امکان پذیر است اگر یک وجود نداشته باشد. اتصال ادغام – مرتب کردن J3، کاملاً کارآمد است چنانچه هر دو فایل توسط ویژگی اتصال خود مرتب شوند. فقط گذر تکی از طریق هر فایل ایجاد می شود. از اینرو تعداد بلوکهای دستیابی شده، ؟؟ با مجموع تعداد بلوک ها در هر دو فایل است. برای این متد، هر دو op7, op6 به دستیابی های بلوک $b_E + b_D = 2000 + 2010$ نیاز دارند. بهر حال، هر دو فایل به مرتب شدن توسط ویژگی های اتصال نیاز دارند، اگر یکی یا هر دو اینطور نباشند، آنها ممکن است برای اجرای عملیات اتصال مرتب شوند. اگر هزینه مرتب کردن فایل خارجی را با ($b\log_2 b$) دستیابی بلوک برآورد کنیم و اگر هر دو فایل نیاز به مرتب شدن داشته باشند، کل هزینه اتصال ادغام – مرتب کردن می تواند توسط $(b_E + b_D + b_E \log_2 b_E + b_D \log_2 b_D)$ برآورد شود.

اتصال hash پارتیشن و اتصال hash دو رگه :متد اتصال hash ، J4 نیز کاملاً متفاوت است. در این مورد ، فقط گذر تکی از طریق هر فایل صورت می گیرد، خواه فایل ها مرتب باشند یا مرتب نباشند. اگر جدول hash برای کوچکتر از دو فایل در حافظه اصلی بتواند بعد از hashing روی ویژگی اتصال آن حفظ گردد، اجرا بسادگی صورت می گیرد. بهر حال اگر بخشهایی از فایل hash روی دیسک ذخیره شوند، متد پیچیده تر می شود، و تعدادی تغییرات برای بهبود کارآیی پیشنهاد شده است. ما روی دو

تکنیک بحث می کنیم: اتصال hash پارتیشن ، و تغییراتی بنام اتصال hash دو رگه ، که کـاملاً کارآمـد نشان داده شده است.

در الگاریتم اتصال hash پارتیشن ، هر فایلی ابتدا به M پارتیشن با استفاده از تابع hash تقسـیم کـردن، روی ویژگیهای اتصال، تقسیم می شود. بعد هر جفت پارتیشن ها متصل می شوند. برای مثال ، فرض کنید رابطه های S,R را روی ویژگیهای اتصال R.A , S.B متصل می کنیم:

$$Rx_{A=B}S$$

در مرحله تقسیم کردن ، R به M پارتیشن $R_1,R_2,...R_M$ به M پارتیشـن $R_1,R_2,...R_M$ تقسـیم می شود. ویژگی هر جفت پارتیشن های متناظر R_i,S_i در این است که ثبت ها در R_i فقط به متصل شدن با ثبت ها در S_i نیاز دارند و برعکس. این ویژگی با استفاده از همان تابع hash برای تقسیم کردن هر دو فایل روی ویژگیهای اتصال آنها، ویژگی A برای R و ویژگی B بـرای S ، تضـمین مـی گـردد. حداقل تعداد بافرهای مورد نیاز حافظه برای مرحله تقسیم کردن، M تـا اسـت. هـر فایـل S,R بطـور مجزایی تقسیم می شوند. برای هر پارتیشنی ، بافر در حافظه تکی ، که اندازه آن یـک بلـوک دیسـک است، برای ذخیره ثبت هایی تخصیص می یابد که نسبت به این پارتیشن hash می باشـد. هـر وقـت بافر در حافظه برای پارتیشن پر می شود، محتویات آن به فایل فرعی دیسک ضمیمه مـی گردنـد کـه این پارتیشن را ذخیره می کند. مرحله تقسیم کردن دو تکرار دارد. بعد از اولین تکرار ، اولین فایـل R ،به فایل های فرعی $R_1,R_2,...R_M$ تقسیم می شود، که تمام ثبـت هـایی کـه بـه همـان بـافر hash شدند، در همان پارتیشن قرار دارند. بعد از دومین تکرار، دومین فایل S بطور مشابه تقسیم می شود.

در دومین مرحله ، بنام مرحله آزمایش یا اتصال یـافتن، M تکـرار لازم اسـت. در طـول تکـرار ، i دو پارتیشن Si,Ri متصل می شوند. حداقل تعداد بافرهای مورد نیاز بـرای تکـرار i ،تعـداد بلـوک هـا در کوچکتر از دو پارتیشن کوچکتر یعنی R_i بعلاوه دو بافر اضافی می باشد. اگر از اتصال حلقـه تودرتـو در طول تکرار i استفاده کنیم، ثبت ها از کوچکتر از دو پارتیشن R_i به بافرهای حافظه کپی می شوند، بعد تمام بلوک ها از پارتیشن دیگر S_i خوانده می شوند، در یک زمان، هر ثبت برای آزمایش پارتیشن R_i جهت تطبیق پذیری ثبت ها استفاده می شود. هر ثبت تطبیق پذیری در فایل نتیجه متصل می شـود و نوشته می شود برای بهبود کارآیی آزمایش در حافظه ، کاربرد جدول hash در حافظه بـرای ذخیـره ثبت ها در پارتیشن R_i با استفاده از تابع hash متفاوت از تابع hash تقسیم کردن متداول است. مـا مـی توانیم هزینه این اتصال hash پارتیشن را بعنوان $3*(b_R + b_S) + b_{RES}$ برای مثـال خـود بـرآورد کنیم، چون هر ثبت یکبار خوانده و به عقب دیسک در طول مرحله تقسیم کردن نوشته می شـود. در طول مرحله اتصال (آزمایش)، هر ثبت، دومین بار برای انجام اتصال خوانده می شود. مشـکل اصـلی این الگاریتم برای تضمین این مطلب است که تابع hash تقسیم کـردن یکنواخـت و یکسـان اسـت و اندازه های پارتیشن تقریباً مساوی هستند. اگر تابع تقسیم کردن، غیر یکنواخت و اریـب باشـد، بعـد بعضی پارتیشن ها ممکن است برای تناسب در فضای حافظه موجود برای دومین مرحله اتصال خیلی بزرگ باشند. توجه کنید که اگر فضای بافر در حافظه موجـود $n_B > (b_R + 2)$ باشـد، جـایی کـه b_R تعداد بلوک ها برای کوچکتر از دو فایل متصل شده، یعنی R ، است، بعد هیچ دلیلی برای انجام تقسیم کردن وجود ندارد چون در این مورد، اتصال می تواند بطـور کامـل در حافظـه بـا اسـتفاده از

بعضی تغییرات اتصال حلقه تودرتو براساس hashing و آزمایش اجرا شود. برای مثال فرض کنید ما عملیات اتصال opb را انجام می دهیم که بدین صورت تکرار شده است.

(op6)

در این مثال، فایل کوچکتر، فایل DEPARTMENT است، از اینرو، اگر تعداد بافرهایی در حافظه موجود $n_B > (b_D + 2)$ باشد، کل فایل DEPARTMENT می تواند در حافظه اصلی خوانده شود و به جدول hash روی ویژگی اتصال سازماندهی شود. هر بلوک Employee در بافر خوانده می شود و هر ثبت Employee در بافر روی ویژگی اتصال خود hash می شود. و برای آزمایش مخزن در حافظه متناظر در جدول DEPARTMENT , hash استفاده می شود.

اگر ثبت تطبیق پذیری مشاهده شود، ثبت ها متصل می شوند، و ثبت های حاصله در بافر نتیجه، نوشته می شوند و در فایل نتیجه نوشته می شوند. هزینه در اصطلاحات دستیابی های بلوک $(b_D + b_E)$ بعلاوه b_{res} هزینه نوشتن فایل نتیجه است.

الگاریتم اتصال hash، دو رگه ،تغییرات اتصال hash پارتیشن است، که مرحله اتصال برای یکی از پارتیشن ها در مرحله تقسیم کردن شامل می شود. برای نشان دادن این مطلب، فرض کنید که اندازه بافر حافظه، یک بلوک دیسک است که n_B از این بافرها در دسترس هستند، و آن تابع hash بکار رفته h(k)=kmodm است تا اینکه M پارتیشن ایجاد می شود، جاییکه $M < n_B$ است. برای مثال ، فرض کنید که ما عملیات اتصال op6 را انجام می دهیم. در اولین گذر مرحله تقسیم کردن، وقتی الگاریتم اتصال hash دو رگه، در کوچکتر از دو فایل تقسیم می شود. الگاریتم ، فضای بافر در میان M پارتیشن را تقسیم می کند تا اینکه تمام بلوک های اولین پارتیشن DEPARTMENT کاملاً در حافظه اصلی باقی

می ماند. برای هر پارتیشن دیگر، فقط بافر رد حافظه تکی که انـدازه آن یـک بلـوک دیسـک اسـت. تخصیص می یابد. بقیه پارتیشن در دیسک بعنوان اتصال hash پارتیشن منظم نوشته می شود. از اینرو در پایان اولین گذر مرحله تقسیم کردن، اولین پارتیشن DEPARTMENT در حافظه اصلی کـاملاً بـاقی می ماند،در عوض هر یک از پارتیشن های دیگر DEPARTMENT در فایل فرعی دیسک باقی می ماند برای دومین مرحله تقسیم کردن،ثبت های دومین فایل متصل شـده، فایـل بزرگتـر، Employee در ob6 تقسیم می شوند. اگر ثبـت در اولـین پارتیشـن hash شـود، آن بـا ثبـت تطبیـق پـذیری (برابـری) در DEPARTMENT متصل می شود و ثبت های متصل شده در بافر نتیجه نوشـته مـی شـوند. اگـر ثبـت Employee در پارتیشن بغیر از اولین پارتیشن hash شود، بطور معمول تقسـیم مـی شـود. از اینـرو، در پایان دومین گذر مرحله تقسیم کردن، تمام ثبت هایی که در اولین پارتیشن hash مـی شـوند، متصـل شده اند. اکنون m-1 جفت پارتیشن روی دیسک وجود دارد. لذا، در طول دومـین مرحلـه اتصـال یـا آزمایش ، m-1 تکرار بجای m لازم می باشد.

هدف، اتصال بعنوان ثبت های زیاد در طول مرحله تقسیم کردن و صرفه جویی در هزینه ذخیره ایـن ثبت ها در پشت دیسک و خواندن مجدد آنها در دومین بار در طول مرحله اتصال است.

۱۸.۲.٤ – اجـرای عملیاتهـای PROJECT و Set : عملیـات Project (R), attribute list> بـرای اجـرا مستقیم است چنانچه <Matribute, list شامل کلیه رابطه R_i باشد، چون در این مورد، نتیجه عملیات ، تعداد tuple های یکسانی تحت عنوان R دارد ولی با فقط مقادیر بـرای ویژگیهـا در <attribute list> در هر tuple اگر <attribute list> شامل کلید R نباشد، کپی و تکثیر tuple ها باید حذف گردد. این با مرتب کردن نتیجه عملیات و بعد حذف duplicate toples صورت می گیرد که در نتیجه بعد از مرتـب کـردن

نمایان میشود. طرح الگاریتم در تصویر c(b) ۱۸ ارائـه شـده اسـت. hashing مـی توانـد بـرای حـذف تکثیرها بکار رود: تحت عنوانی که هر ثبت در مخزن فایل hash در حافظه hash و درجه می شـود، آن در برابر موارد قبلی در مخزن چک می شود، اگر آن کپی و تکثیر باشد (duplicate) ، درج نمـی شـود. آن برای فراخواندنی در پرس و جوهای SQL در اینجا مفید است، پیش فرض بـرای حـرف duplicate ها از نتیجه پرس و جو نمی باشد، تنها اگر Distinct کلمه کلیدی را شامل شـود، duplicate هـا حـذف شـده از نتیجـه پـرس و جـو هسـتند. عملیاتهـای , CARTESIAN rrODUCT, SET DIFFERENCE INTERSECTION, UNION, SET گاهـاً اوقـات بـرای اجـرا پرهزینـه هسـتند. بـویژه CARTESIAN RXS PRODUCT گران است، چون نتیجه آن شامل ثبت برای هر الحاق ثبت ها از S,r است. علاوه بر آن ، ویژگیهای نتیجه شامل تمام ویژگیهای S,R است. اگر N, R ثبت و ناویژگی داشته باشد،S ، m ثبت و k ویژگی داشته باشد، رابطه حاصله، n*m ثبت و j+k ویژگی دارد.

از اینرو، آن برای اجتناب از عملیات CARTESIAN PRODUCT و جانشینی با عملیاتهای معـادل دیگـر در طـول بهینـه سـازی پـرس و جـو مهـم اسـت. ۳ عملیاتهـای SET دیگـر ، SET DIFFERENCE, INTERSECTION, UNION فقط برای رابطه های اجتمـاع سـازگار بکـار مـی رونـد، کـه تعـداد یـک ویژگیهای یکسان در حوزه های ویژگی یکسانی دارند. راه مرسوم بـرای اجـرای ایـن عملیـات هـا ، کاربرد تغییرات تکنیک ادغام – مرتب کردن است. دو رابطه روی ویژگیهای یکسان مرتب می شـوند و بعد از مرتب شدن،پوشش تکی از طریق هر رابطه برای ایجاد نتیجه کافی است. برای مثال، مـا مـی توانیم ، عملیات RUS, UNION ، را با پویش و ادغام هر دو فایل مرتب شده بطور همزمان انجام دهیم و هر وقت TUPLE یکسانی در هر دو رابطه وجود دارد، فقط یکی در نتیجـه ادغـام شـده حفـظ مـی

گردد. برای عملیات RUS, INTERSECTION ، ما در نتیجه ادغام شده فقط آن TUPLE هـایی را حفـظ می کنیم که در هر دو رابطه ظاهر می شوند. تصویر (C) ۳. ۱۸ تا (C) اجـرای ایـن عملیـات هـا را بـا مرتب کردن و ادغام کردن طراحی می کند. اگر مرتب کـردن روی ویژگیهـای کلیـدی منحصـر بفـرد صورت گیرد، عملیات ها، بیشتر ساده مـی شـوند. hashing مـی توانـد همچنـین بـرای اجـرای SET DIFFERENCE, INTERSECTION بکار رود. یک جدول تقسیم می شود. و جدولی دیگر برای آزمایش پارتیشن مناسب استفاده می شود. برای مثال برای اجرای RUS ابتدا ثبت های R را HASH می کند بعد ثبت های S را hash می کند، ولی ثبت های duplicate را در مخزن ها درج نمی کند. برای اجرای RNS ، ابتدا ثبت های R به فایل hash تقسیم می شود. بعد در حالیکه هر ثبت S hash می شود، برای کنتـرل آزمایش می شود اگر ثبت قابل شناسایی از R ، در مخزن یافت شود، و اگر ثبت به فایل نتیجه اضـافه شود. برای اجرای R-S ، ابتدا ثبت های R به مخازن فایل hash, hash می شود در حالیکه هر ثبـت S ، hash می شود، اگر ثبت شناسایی در مخزن یافت شود، آن ثبت از مخزن برداشته می شود.

۵ . ۲ ۱۸– اجرای عملیاتهای گروهی : عملگرهـای گروهـی (Sum, Average, count , Max, Min) زمـانی برای کل جدول بکار می روند، می توانند توسط پذیرش جدول یـا بـا اسـتفاده از شـاخص مناسـبی محاسبه شوند. برای مثال، پرس و جوی SQL زیر را در نظر بگیرید:

SELECT MAX (SALARY)

FROM EMPLOYEE:

اگر شاخص روی SALARY برای رابطه Employee وجـود داشـته باشـد، بهینـه سـاز مـی توانـد روی استفاده از شاخص برای جستجوی بزرگترین مقدار با دنبال کردن اشاره گر بیشتر به راست در هر هر

گره شاخص از ریشه به برگ راست تصمیم گیری کند. آن گره شامل بزرگترین مقدار SALARY بعنوان ثبت آخر آن است. در اکثر موارد، این نسبت به پویش کل جدول Employee کارآمدتر است. چون هیچ ثبت واقعی نیاز به بازیابی شدن ندارد. گروه MIN در حالت مشابه ای می تواند کنترل شود، بجز اینکه اشاره گر چپ از ریشه تا برگ چپ دنبال شود.آن گره شامل کوچکترین مقدار SALARY بعنوان اولین ثبت آن است.

شاخص برای گروههای SUM, AVERANGE, COUNT بکار می رود ولی فقط در صورتی که شاخص dense باشد، در اینصورت اگر ثبت شاخص برای هر ثبت در فایل اصلی وجود داشته باشد. در این خصوص، محاسبه مربوط برای مقادیر در شاخص بکار می رود. برای شاخص nondense تعداد واقعی ثبت ها در ارتباط با هر ثبت شاخص باید برای محاسبه درست بکار رود. وقتی بند Group By در پرس و جو بکار می رود، عملگر گروهی باید بطور مجزایی برای هر گروه tuple بکار رود. از اینرو، جدول ابتدا باید به مجموعه های فرعی tuple تقسیم شود، جایی که هر پارتیشن (گروه) مقدار یکسانی برای ویژگیهای گروهی دارد. در این مورد، محاسبه پیچیده تر است. پرس و جوی زیر را در نظر بگیرید:

```
SELECT DNO , AVG (SALARY)

FROM employee

Group By DNO;
```

تکنیک معمول برای این پرس و جوها ، ابتدا جهت کاربرد مرتب کردن یا hashing روی ویژگیهای گروهی برای تقسیم فایل به گروههای مناسب است. بعد الگاریتم، تابع گروهی (جمعی) را برای tuple

ها در هر گروه محاسبه می کند،که مقدار ویژگی گروهی یکسان دارند. در مثال پرس و جو ، مجموعه tuple ها، برای هر تعداد دپارتمان ، با همدیگر در پارتیشن و میانگین محاسبه شده برای هر گروه گروهبندی می شود. توجه کنید که اگر شاخص خوشه سازی روی ویژگیهای گروهی وجود داشته باشد، بعد ثبت ها به زیرمجموعه های مناسبی تقسیم می شوند. در این مورد، فقط بکارگیری محاسبه برای هر گروه ضروری است.

۶ .۲ .۱۸– اجرای اتصال بیرونی: در بخش ۳ . ۵. ۷ ، عملیات اتصال بیرونی ارائه شده است با ۳ تغییرات آن: اتصال بیرونی چپ، اتصال بیرونی راست و کل اتصال بیرونی. ما در فصل ۸ بحث کردیم که چطور این عملیات ها می تواند در SQL2 تعیین شوند. در زیر مثالی از عملیات اتصالی بیرونی چپ در SQL2 آمده است:

SELECT LNAME, FNAME, DNAME

FROM (Employeeleft outer join Department on Dno= Dnumber)

نتیجه این پرس و جو ، جدول اسامی کارمندان و دپارتمانهای مربوط به آنها است. آن برای نتیجه اتصال منظم (درونی) مشابه است، بجز اینکه اگر Employee tuple ، دپارتمان مربوطه نداشته باشد، نام کارمند در جدول حاصله ظاهر نمی شود، ولی نام دپارتمان ، برای این tuple ها در نتیجه پرس و جو null یا پوچ یا تهی است. اتصال بیرونی می تواند یا تعدیل کپی از الگاریتم های اتصال مثل اتصال حلقه تودرتو، یا اتصال حلقه تکی محاسبه شود. برای مثال ، برای محاسبه اتصال بیرونی چپ از رابطه چپ بعنوان حلقه بیرونی یا حلقه تکی استفاده می کنیم چون هر tuple ، در رابطه چپ باید در نتیجه ظاهر شود. اگر tuple های تطبیق پذیری در رابطه دیگری وجود داشته باشد، tuple ها متصل

شده در نتیجه ایجاد می شود و ذخیره می شوند. بهر حال اگر هیچ tuple تطبیق پذیری یافت نشود، tuple در نتیجه هنوز وجود دارد ولی با مقدار پوچ یا تهی پر می شود. الگاریتم های ادغام – مرتب کردن، اتصال hash می توانند برای محاسبه اتصالات بیرونی توسعه یابند. بدین ترتیب ، اتصال بیرونی می تواند با اجرای الحاق عملگرهای جبری رابطه ای محاسبه شود. برای مثال ، عملیات اتصال بیرونی چپ نشان داده شده با فوق معادل با توالی عملیاتهای رابطه ای زیر است:

۱– محاسبه JOIN (درونی) در جداول Employee, Department :

۲– یافتن tupleهای employee که در نتیجه JOIN ظاهر نمی شود:

۳– پر کردن هر tuple در temp2 با فیلد ONAME :

٤– بکارگیری عملیات UNION برای Temp2, temp1 برای ایجاد نتیجه RESULT LEFT OUTER JOIN

هزینه اتصال بیرونی تحت عنوان محاسبه شده فوق، مجموعه هزینه های مراحل مربوطه است. بهرحال توجه کنید که مرحله ۳ می تواند بعنوان رابطه موقتی انجام گیرد که در مرحله ۲ ساخته می شود. در اینصورت می توان هر tuple حاصله را با تهی پوشش داد یا پر کرد. علاوه بر آن در مرحله ٤ ، می دانیم که دو اپراند اجتماع ، غیر متصل می شوند لذا هیچ نیازی به حذف duplicate نمی باشد.

۱۸ . ۲ . ۷ – الحاق و ترکیب عملیات ها با استفاده از لوله ای کردن: پرس و جوی تعیین شده در SQL به عبارت جبری رابطه ای ترجمه می شود که توالی عملیاتهای رابطه ای است. اگر عملیات تکی را اجرا کنیم، باید فایل های موقتی روی دیسک برای حفظ نتایج حاصله از این عملیات های موقتی ایجاد کنیم که سربار اضافی را ایجاد می کند. ایجاد و مرتب کردن فایل های موقتی بزرگ روی دیسک ضامن صرف زمان ا ست و در بسیاری موارد می تواند غیر ضروری باشد، چون این

فایل ها فوراً بعنوان ورودی در عملیات بعدی استفاده می شوند. برای ایجاد تعدادی فایل موقتی، ایجاد کد اجرای پرس و جو متداول است که در ارتباط با الگاریتم ها برای الحاقات عملیات ها در پرس و جو است.

برای مثال، نسبت به اجرای مجزا، JOIN می تواند با دو عملیات SELECT روی فایل های ورودی و عملیات نهایی PROJECT روی فایل حاصله ترکیب شود، همگی توسط یک الگاریتم با دو فایل ورودی و فایل خروجی تکی اجرا می شود. نسبت به ایجاد چهار فایل موقتی، ما الگاریتم را مستقیماً بکار می بریم و فقط یک فایل نتیجه بدست می آوریم. در بخش ۱ . ۳. ۱۸، ما روی چگونگی گروهبندی بهینه سازی جبری رابطه ای ذهنی عملیاتها با همدیگر برای اجرا بحث می کنیم. این پردازش مبتنی بر جاری کردن یا لوله ای کردن نامیده می شود. ایجاد کد اجرای پرس و جو برای پیاده سازی عملیاتهای متعدد متداول است. کد ایجاد شده برای ایجاد پرس و جو چندین الگاریتم را ترکیب می کند که مربوط به عملیاتهای اختصاصی می باشند. از آنجائیکه truple های نتیجه از یک عملیات ایجاد می شوند، آنها بعنوان ورودی برای عملیاتهای بعدی ارائه می شوند. برای مثال اگر عملیات اتصال دو عملیات انتخاب را روی روابط مبنا دنبال کند، tupleهای ناشی از هر انتخاب بعنوان ورودی برای الگاریتم اتصال در جریان یا لوله که ایجاد می شوند، ارائه می گردند.

۳. ۱۸- بکارگیری قوانین ذهنی در بهینه سازی پرس و جو: در این بخش،ما روی تکنیک های بهینه سازی بحث می کنیم که از قوانین ذهنی برای تعدیل نمونه داخلی پرس و جو استفاده می کنند که معمولاً به فرم درخت پرس و جو یا ساختار داده های گراف پرس و جو، برای بهبود عملکرد اجرایی آن است. تجزیه کننده پرس و جو در سطح بالا ابتدا، نمونه داخلی آغازین را ایجاد می کند که بعد بر

طبق قوانین ذهنی بهینه سازی می شود. بدنبال آن، طرح اجرای پرس و جو برای اجرای گروههای عملیات ها براساس مسیرهای دستیابی موجود روی فایل ها در پرس و جو ،ایجاد می شود.

یکی از قوانین ذهنی اصلی، جهت بکارگیری عملیات های SELECT , PROJECT قبل از بکارگیری عملیات های JOIN و بنیادی دیگر است. این بدین علت است که اندازه فایل ناشی از عملیات بنیادی، مثل JOIN ، معمولاً تابع مضروب اندازه های فایل های ورودی است. عملیات های SELECT و PROJECT ، اندازه فایل را کاهش می دهند و از اینرو باید قبل از عملیات JOIN (اتصال) و عملیات بنیادی دیگر بکار گرفته شود.

ما در بخش ۱. ۳. ۱۸ با ارائه درخت پرس و جو و نمادهای گراف پرس و جو کار را آغاز می کنیم. اینها بعنوان پایه و اساس برای ساختارهای داده هایی استفاده می شوند که برای نمونه داخلی پرس و جوها بکار می روند. درخت پرس و جو برای ارائه عبارت جبری رابطه ای توسعه یافته یا عبارت جبری رابطه ای بکار می رود،در عوض گراف پرس و جو برای نشان دادن عبارت حسابان رابطه ای استفاده می شود. وقتی در بخش ۲ . ۳. ۱۸ نشان می دهیم که چطور قوانین بهینه سازی ذهنی برای تبدیل درخت پرس و جو به درخت پرس و جوی معادل بکار می روند، که عبارت جبری رابطه ای متفاوتی را نشان می دهد که برای اجرا کارآمدتر است ولی همان نتیجه اولیه را ارائه می دهد. ما روی تعادل و توازن عبارات جبری رابطه ای گوناگون بحث می کنیم. بالاخره در بخش ۳ .۳. ۱۸ به ایجاد طرح های اجرای پرس و جو می پردازیم.

۱ . ۳. ۱۸- نماد برای درخت های پرس و جو و گراف های پرس و جو: درخت پرس و جو ، ساختار داده های درختی است که مربوط به عبارت جبری رابطه ای است. آن رابطه های ورودی

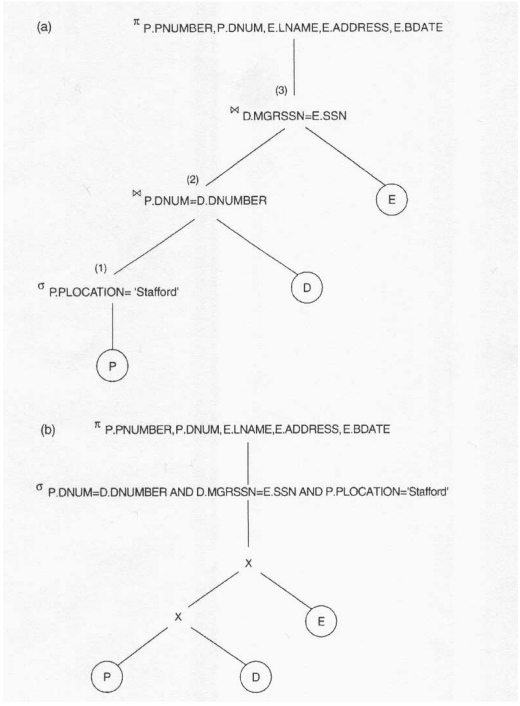
پرس و جو را بعنوان گروه های برگ های درخت نشان می دهـد و عملیاتهـای جبـری رابطـه ای را بعنوان گروه های داخلی نشان می دهد. اجرای درخت پرس و جو شامل اجرای عملیات گره داخلـی است هر وقت اپراند آن در دسترس باشد و بعد جایگزینی آن گره داخلی بـا رابطـه ای اسـت کـه از اجرای عملیات حاصل می گردد. اجرا زمانی خاتمه می یابد که گره ریشـه اجـرا مـی شـود و رابطـه نتیجه برای پرس و جو ایجاد می شود. تصویر (a) ٤ـ ١٨ ، درخت پرس و جو را برای پرس و جـوی QL از فصل های ٧ تا ٩ نشان می دهد. برای هر پروژه واقع در Stafford ،تعداد پروژه ، تعداد دپارتمان در حال کنترل ، و نام فامیل، آدرس و تاریخ تولد مدیر دپارتمان بازیابی میشود. این پرس و جو روی طرح رابطه ای تصویر ٥ ـ٧ تعیین شده و مربوط به عبارت جبری رابطه ای زیر است:

TTPNOMBER, DNUM

QZ: SELECT P.pnumber, p.

FROM

WMERE



تصویر ٤ ـ١٨

در تصویر (a) ٤ .١٨ ، ٣ رابطه Gmployee, Department , Project توسط گره های برگهای E,D,P نشان داده شده اند، در حالیکه عملیاتهای جبری رابطه ای عبارت توسط گره های درخت داخلی نشان داده شده اند. وقتی این درخت پرس و جو اجرا می شود، گره با علامت (١) در تصویر (a) ٤ . ١٨ باید اجرا را قبل از گره (٢) آغاز کند چون tuple های حاصله از عملیات (١) باید در دسترس باشند قبل از اینکه ما بتوانیم عملیات اجرایی (٢) را آغاز کنیم. بطور مشابه گره (٢) باید نتایج اجرایی و تولیدی را آغاز کند قبل از اینکه گره (٣) بتواند اجرا را آغاز کند و بدین ترتیب همانطوریکه مشاهده می شود،درخت پرس و جو ، ترتیب خاص عملیاتها را برای اجرای پرس و جو نشان می دهد. نمونه خنثی تر پرس و جو،نماد گراف پرس و جو است. تصویر (c) ٤ .١٨ ، گراف پرس و جو را برای پرس و جوی QL نشان دهد.

رابطه ها در پرس و جو توسط گره های رابطه نشان داده می شوند،که بعنوان چرخه های تکی به نمایش در می آیند. مقادیر ثابت ، از شرایط انتخاب پرس و جو ، توسط گره های ثابت نشان داده شده اند، که بعنوان چرخه های دوبل به نمایش در می آیند. شرایط انتخاب و اتصال توسط لبه های گراف نشان داده شده اند، همانطوریکه در تصویر (c) ٤. ١٨ آمده است. بالاخره، ویژگیهای بازیابی شده از هر رابطه در پرانتزهای مربعی بالای هر رابطه به نمایش در آمده اند.

نمونه گراف پرس و جو، ترتیبی را مشخص نمی کند که روی آن عملیاتها ابتدا اجرا می شوند. فقط گراف تکی مربوط به هر پرس و جو وجود دارد. اگر چه تعدادی تکنیک های بهینه سازی براساس گراف های پرس و جو بودند، اکنون پذیرفته می شود که درختان پرس و جو ارجحیت دارند چون

در عمل، بهینه ساز پرس و جو نیاز به نشان دادن ترتیب عملیات ها برای اجرای پرس و جو دارد که در گرافهای پرس و جو امکان پذیر نمی باشد.

۱۸.۳.۲– بهینه سازی ذهنی درخت های پرس و جو: بطور کلی، بسیاری از عبارات جبری رابطه ای متفاوت و از اینرو درختان پرس و جوی متفاوت می توانند معادل باشند، که می توانند در ارتباط با پرس و جوی یکسانی باشند. تجزیه کننده پرس و جو، درخت پرس و جو آغازین استانداردی را برای ارتباط با پرس و جوی SQL، بدون انجام هر گونه بهینه سازی، ایجاد می کند. برای مثال، برای پرس و جوی انتخاب پروژه – اتصال ، مثل Q2 ، درخت آغازین در تصویر ۱۸.٤ (b) نشان داده شده است. CARTESIAN PRODUCT رابطه های تعیین شده در بند FROM ، ابتدا استفاده می شود، بعد شرط های انتخاب و اتصال بند HERE بکار می روند که توسط طرح روی ویژگیهای بند SELECT دنبال می شود. این درخت پرس و جوی متعارف ، عبارت جبری رابطه ای را نشان می دهد که غیرکارآمد است چنانچه مستقیماً اجرا شود، که بعلت عملیاتهای CARTESIAN PRODUCT (X) است. برای مثال ، اگر رابطه های EMPLOYEE, DEPARTMENT, PROJECT ، اندازه های ثبت ۱۰۰ ، ۸۰ ، ۱۸۰ بایت داشته باشند و شامل ۱۰۰ ، ۲۰ و TUPLE 5000 باشند، نتیجه CARTESIAN PRODUCT شامل ۱۰ میلیون TUPLE در اندازه ثبت ۳۰۰ بایتی هر یک می باشند. بهر حال، درخت پرس و جو در تصویر ۱۸.٤ (b) ، فرم استاندارد ساده ای است که می تواند بآسانی ایجاد شود. اکنون آن کار بهینه ساز پرس و جوی ذهنی برای تغییر شکل دادن این درخت پرس و جوی آغازین به درخت پرس و جوی نهایی است که برای اجرای کارآمد است. بهینه ساز باید شامل قوانینی برای تعادل در میان عبارات جبری رابطه ای باشد که می تواند برای درخت آغازین استفاده شود. قوانین بهینه سازی پرس

و جوی ذهنی از این عبارات متعادل برای تغییر شکل درخت آغازین به درخت نهایی استفاده می کند و درخت پرس و جو را بهینه سازی می کند. ما ابتدا روی چگونگی تغییر شکل درخت پرس و جو با استفاده از قانون ذهنی بحث می کنیم. بعد به قوانین تغییر شکل کلی و عمومی می پردازیم و نشان می دهیم چطور آنها در بهینه ساز ذهنی جبری بکار می روند. مثال تغییر شکل پرس و جو : پرس و جو Q زیر را روی پایگاه اطلاعاتی تصویر ۷.۵ در نظر بگیرید: نام خانوادگی کارمندان متولد شده بعد از ۱۹۵۷ را که در پروژه Aquarius کار می کنند را پیدا کنید. این پرس و جو می تواند در SQL بقرار زیر تعیین شود:

Q: SELECT LNAME

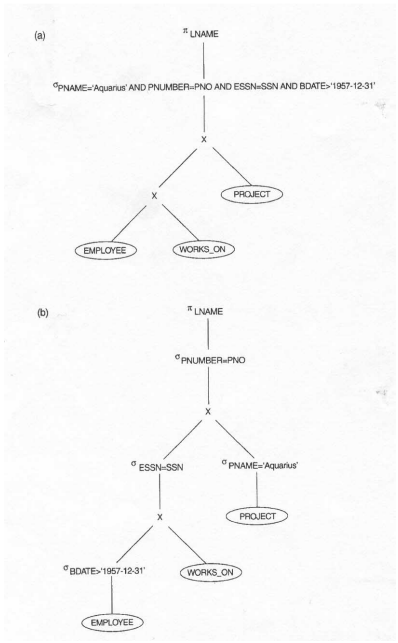
FROM...

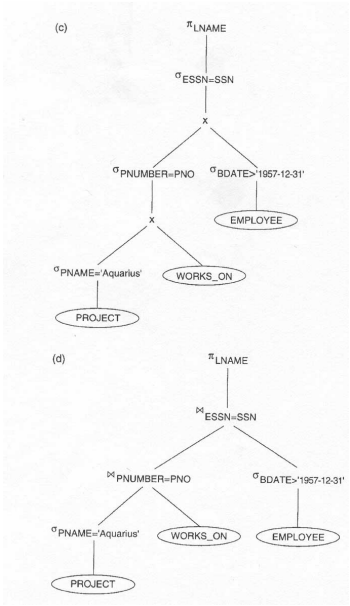
WHERE

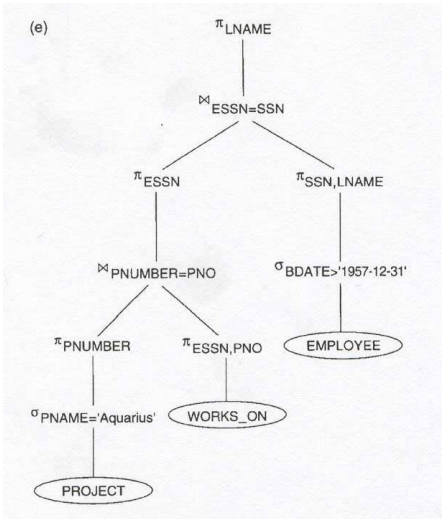
درخت پرس و جوی آغازین برای Q در تصویر ۱۸.۵ (a) نشان داده شده است. اجرای این درخت مستقیماً ابتدا فایل خیلی بزرگی شامل CARTESIAN PRODUCT از کل فایل های PROJECT, UORUS - ON, EMPLOYEE را ایجاد می کنند بهر حال ، این پرس و جو نیاز به فقط یک ثبت از رابطه PROJECT ،برای پروژه AQUARIUS دارد و فقط ثبت های EMPLOYEE برای آن متولد بعد از ۳۱-۱۲-۱۹۵۷ دارد. تصویر ۱۸.۵ (b) درخت پرس و جوی بهبود یافته را نشان می دهد که از عملیاتهای SELECT برای کاهش تعداد Tuple ها که در CARTESIAN PRODUCT ظاهر می شوند، استفاده می کند.

تصویر ۱۸.۵

(a)







بهبود بیشتر با سوئیچینگ موقعیت های رابطه هـای Project, Employee در درخـت بدسـت مـی آیـد، همانطوریکه در تصـویر (c) ۱۸.۵ نشـان داده شـده اسـت. ایـن از اطلاعـاتی اسـتفاده مـی کنـد کـه PNUMBER ویژگی کلیدی رابطه پروژه است، و از اینرو عملیات SELECT روی رابطه Project ، فقـط ثبت تکی را بازیابی خواهد کرد. ما می توانیم درخت پرس و جو را بیشـتر بـا جـایگزین کـردن هـر

عملیات CARTESIAN PRODUCT که توسط شرط اتصال با عملیات JOIN دنبال می شود، بهبود بخشیم. همانطوریکه در تصویر (d) ۵. ۱۸ نشان داده شده است. بهبود دیگر برای حفظ ویژگیهای مورد نیاز توسط عملیاتهای بعدی در رابطه های واسطه ،توسط شامل شدن عملیاتهای Project (II) در حد ممکن در درخت پرس و جو است،همانطوریکه در تصویر (e) ۵. ۱۸ نشان داده شده است. این ویژگیهای (ستونهای) رابطه های واسطه را کاهش می دهد، در عوض ، عملیاتهای SELECT ، تعداد tuple ها (ثبت ها) را کاهش می دهند.

همانطوریکه از مثال قبلی مشاهده می شود، درخت پرس و جو می تواند مرحله به مرحله به درخت پرس و جوی دیگری تغییر شکل یابد که برای اجرا کارآمد تر است. بهر حال، ما باید مطمئن شویم که مراحل تغییر شکل همیشه منتهی به درخت پرس و جوی معادل می شوند.

برای انجام این کار ، بهینه ساز پرس و جو باید بداند که کدامیک از قواین تغییر شکل این تعادل را حفظ می نماید. ما به بررسی این قوانین تغییر شکل بعداً می پردازیم.

قواینن کلی تغییر شکل برای عملیاتهای جبری رابطه ای: قوانین زیادی برای تغییر شکل عملیاتهای جبری رابطه ای به عملیاتهای معادل وجود دارد. در اینجا ، ما به معنی عملیات ها و رابطه های حاصله علاقمند می باشیم. از اینرو، اگر دو رابطه مجموعه ویژگیهای یکسانی در ترتیب متفاوت داشته باشد، ولی دو رابطه، اطلاعات یکسانی را نشان می دهد، ما معادل رابطه ها را در نظر می گیریم. در بخش ۷. ۱. ۲ ما تعریف دیگری از رابطه ارائه دادیم که ترتیب ویژگی های غیر مهمی را ایجاد می کند، از این تعریف در اینجا استفاده می کنیم. اکنون قوانین تغییر شکل را بیان می کنیم که در بهینه سازی پرس و جو بدون اثبات آنها ، مفید هستند.

۱- آبشاری : شرط انتخاب تقارنی (همبستگی) می تواند به آبشار عملیاتهای اختصاصی تقسیم شود:

σ_{c}1AND

۲- تعویض پذیری σ : عملیات σ، تعویض پذیر است.

σ_{c1}(

۳- آبشار II : در آبشار عملیاتهای II همگی بجز مورد آخری می توانند نادیده گرفته شود:

III_{List1}(

٤- تبادل و تعویض σ با II : اگر شرط انتخاب c شامل فقط آن ویژگی های A1, ... an در لیست طرح باشد،دو عملیات بدین قرار تعویض می شوند:

٥- تعویض پذیری ∞(x,) : عملیات ∞ تعویض پذیر است، تحت عنوان عملیات X بدین قرار :

R∞_{c}S ≡

Rx

توجه کنید که گر چه ترتیب ویژگی ها در رابطه های ناشی از دو اتصال یکسان نمی باشند، معنی یکسان است چون ترتیب ویژگی ها در تعریف رابطه دیگر مهم نمی باشد.

٦- تعویض پذیری σ با ∞ (یا x) :اگر تمام ویژگی ها در شرط انتخاب c شامل فقط ویژگی های یکی از رابطه های متصل شده یعنی R ، باشند، دو عملیات بقرار زیر مبادله می شود:

σ_{c}(

بدین ترتیب اگر شرط انتخاب c بتواند بعنوان (c1 ANDC2) نوشته شود، جایی که شرط C1 شامل فقط ویژگیهای R ، و شرط C2 شامل فقط ویژگی های S است،عملیاتها بقرار زیر مبادله می شوند:

$\sigma_c(R \infty S) \equiv$

قوانین یکسانی بکار می روند چنانچه ∞ با عملیات X جایگزین شود.

۷- مبادله II با ∞ (یا X) : فرض کنید که لیست طرح $(A_2 \ldots A_n, B_1 \ldots, B_m\}$ است، که $A_1, \ldots . A_n$ ویژگی های R و $B_1 \ldots, B_m$ ویژگیهای S هستند. اگر شرط اتصال C شامل فقط ویژگی های در L باشد، دو عملیات می تواند بقرار زیر مبادله شوند:

$II_L(R \infty_c S) \equiv$

اگر شرط اتصال c شامل ویژگیهای اضافی که در L نمی باشد، باشد، اینها باید به لیست طرح اضافه شوند و II عملیات نهایی نیاز است برای مثال اگر ویژگیهای در s شرط اتصال c شامل شوند ، ولی در لیست طرح L نباشند، عملیات ها بقرار زیر مبادله می شوند.

$II_L(R$

برای X ، هیچ شرط c وجود ندارد، لذا اولین قانون تغییر شکل همیشه توسط جایگزینی ∞_c یا X بکار می رود.

۸- تعویض پذیری عملیاتهای مجموعه :عملیاتهای مجموعه تعویض پذیر هستند ولی - اینطور نمی باشد.

۹- پویندی : این چهار عملیات پیوندی هستند که اگر θ برای هر یک از این چهار عملیات قرار گیرد بدین صورت می شد:

۱۰- تعویض پذیری σ با عملیاتهای مجموعه : عملیات σ با $\cup, \cap$ و - معاوضه می شود چنانچه برای هر یک از این ۳ عملیات قرار گیرد که بدین قرار داریم:

$\sigma_c(R$

$II_L(RUS)$

۱۲- تبدیل توالی (σ, x) به ∞ : چنانچه شرط c از σ که آن x مربوط به شرط اتصال را دنبال می کند.

توالی (σ, x) را به ∞ بدین قرار تبدیل می کند.

$\sigma_c(R\ x$

تغییر شکل های ممکن دیگیری وجود دارند. برای مثال، شرط اتصال یا انتخاب c می تواند به شرط معادل با استفاده از قوانین زیر تبدیل می شود:

Not (cy ANSD

Not (

تغییر شکل های اضافی بحث شده در فصل های ۷ و ۹ در اینجا تکرار نمی شوند. ما بعداً در مورد چگونگی استفاده از تغییر شکل ها در بهینه سازی ذهنی بحث می کنیم:

شرح کلی الگاریتم بهینه سازی جبری ذهنی: اکنون می توانیم مراحل الگاریتمی را شرح دهیم که از قوانین فوق برای تغییر شکل درخت پرس و جوی آغازین به درخت بهینه سازی شده استفاده می کند که برای اجرا کارآمدتر است. (در اکثر موارد)

الگاریتم منتهی به تغییر شکل هایی مشابه به موارد مورد بحث در مثال ما در تصویر ۵ .۱۸ می شود.

مراحل الگاریتم بقرار زیر است:

۱- کاربرد Rule1 ، تفکیک هر عملیات SELECT با شرط های تفارنی بـه آبشـار عمیقهـای SELECT . این به درجه بیشتری از آزادی در حرکت رو به پایین عمیقهای SELECT با عملیتهای دیگـر، حرکـت هر عملیات SELECT در پایین درخت پرس و جو، توسط ویژگی های شامل شده در شرط انتخـاب ، اجازه داده می شود.

۳- کاربرد قوانین ۵ و ۹ مربوط به تعویض پذیری و پیوندی عمیقیهای بنیـادی ، گـره هـای برگهـای درخت را با استفاده از معیارهای زیر دوباره ترتیب می دهد. اول اینکه موقعیت گره برگها مربوط بـه اکثر عملیاتهای محدود SELECT است، لذا آنها ابتدا در نمونه درخت پرس و جـو اجـرا مـی شـوند. تعریف اکثر SELECT محدود به معنی مواردی است که رابطه ای با کمترین Tuple یـا بـا کـوچکترین اندازه مطلق ایجاد می کند. احتمال دیگر، جهت تعریـف محـدودترین SELECT بعنـوان مـوردی بـا کوچکترین انتخاب پذیری است ، این عملی تر است چون برآوردهای انتخاب پـذیری هـا اغلـب در کاتالوگ DBMS در دسترس هستند. دوم اینکه مطمئن شوید که مرتب کردن گره های برگهـا موجـب عملیات های CARTESIAN PRODUCT نمی شوند،چنانچه دو رابطه با محـدودترین SELECT ، شـرط اتصال مستقیمی بین آنها ندارد، تغییـر دادن ترتیـب گـره هـای برگهـا بـرای اجتنـاب از محصـولات cartesian مطلوب است.

٤- بکارگیری قانون ۱۲ ، الحاق عملیـات CARTESIAN PRODUCT بـا عملیـات بعـدی SELECT در درخت در عملیات JOIN است، اگر شرط، شرط اتصال را نشان دهد.

۵- بکارگیری قوانین ۳ ، ٤ ،۷ ،۱۱ مربـوط بـه آبشـاری PROJECT و تعـویض پـذیری PROJECT بـا عملیات های دیگر ، تفکیک و حرکت لیست های ویژگی های طـرح رو بـه پـائین درخـت در حـد

ممکن با ایجاد عملیات های جدید PROJECT در صورت لزوم است. فقط آن ویژگیهای مورد نیاز در نتیجه پرس و جو و درعملیات های بعدی در درخت پرس وجو باید بعد از هر عملیات Project حفظ شود.

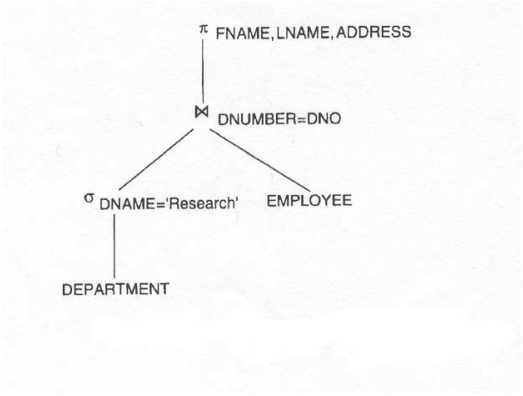
٦- شناسایی درختهای فرعی که گروههای عملیات هایی را نشان می دهند که می توانند با الگاریتم تکی اجرا شوند. در مثال ها ، تصویر (b) ۵. ۱۸ درخت تصویر (c) ۵. ۱۸ را بعد از بکارگیری مراحل ۱ و ۲ الگاریتم نشان می دهد، تصویر (c) ۵. ۱۸ درخت را بعد از مرحله ۳ نشان می دهد و توصیر (d)۵. ۱۸ بعد از مرحله ٤ و تصویر(e) ۵. ۱۸ بعد از مرحله ۵ را نشان می دهد. در مرحله ٦ ، ما با همدیگر عملیات هایی را در درخت فرعی گروهبندی می کنیم که ریشه آنها عملیات II_{ESSN} در الگاریتم تکی است. ما عملیاتهای باقیمانده را به درخت فرعی دیگری گروهبندی می کنیم، جایی که tuple های ناشی از اولین الگاریتم جایگزین درخت فرعی می شود که ریشه آن عملیات II_{ESSN} است،چون اولین گروهبندی بدین معنی است که این درخت فرعی ابتدا اجرا می شود.

خلاصه قانون ذهنی برای بهینه سازی جبری : اکنون قانون ذهنی را برای بهینه سازی جبری بطور خلاصه بیان می کنیم. قانون ذهنی اصلی جهت بکارگیری عملیاتهایی است که اندازه نتایج واسطه را کاهش می دهد. این شامل اجرای عملیاتهای SELECT در حد ممکن برای کاهش تعداد tuple ها و عملیاتهای PROJECT برای کاهش تعداد ویژگی ها است. این با حرکت عملیاتهای PROJECT, SELECT در حد ممکن در پایین درخت صورت می گیرد. علاوه بر آن، عملیاتهای JOW, SELECT که محدودترین هستند، که در رابطه با با کمترین tuple ها یا با کوچکترین اندازه مطلق نتیجه می شود، باید قبل از دیگر عملیات های مشابه اجرا شوند. این با مرتب کردن مجدد گره های برگهای درخت

در میان خودشان صورت می گیرد در حالیکه از محصولات Cartesian جلوگیری می شود. بقیه درخت بخوبی تعدیل و تنظیم میشود.

۱۸.۳.۳- تبدیل درختان پرس و جو به طرح های اجرای پرس و جو: طرح اجرا برای عبارت جبری رابطه ای که بعنوان درخت پرس و جو نشان داده شده شامل اطلاعاتی در مورد متدهای دستیابی موجود برای هر رابطه و نیز الگاریتم های استفاده شده در معادله عملگرهای رابطه ای نشان داده شده دردرخت است. بعنوان یک مثال ساده، پرس و جوی Q1 از فصل ۷ را در نظر بگیرید که عبارت جبری رابطه ای مربوط بدین قرار است:

IIfname, LNAME



درخت پرس و جو در تصویر ۱۸.۶ نشان داده شده است. برای تبدیل این به طرح اجرایی، بهینه ساز ، جستجوی شاخص را برای عملیات SELECT ، پویش جدول را بعنوان متد دستیابی برای Employee ، الگاریتم اتصال حلقه تودرتو برای اتصال و پویش نتیجه JOIN را برای عملگر PROJECT انتخاب می کند. علاوه بر آن، روش در نظر گرفته شده برای اجرای پرس و جو، ارزیابی لوله ای شده و تحقق یافته را تعیین می کند.

با ارزیابی تحقق یافته ،نتیجه عملیات بعنوان رابطه متوفی ذخیره می شود. برای مثال، عملیات اتصال می تواند معاوضه شود و کل نتیجه بعنوان رابطه موقتی ذخیره شود، که بعد بعنوان ورودی توسط الگاریتمی خوانده می شود که عملیات PROJECT را معاوضه می کند که جدول نتیجه پرس و جو را ایجاد می کند. بعبارت دیگر، با ارزیابی لوله ای شده، همانطوریکه tuple های حاصله عملیات ایجاد می شوند، آنها مستقیماً بسوی عملیات بعدی در توالی پرس و جو پیش می روند. برای مثال، همانطوریکه tuple های انتخاب شده از DEPARTMENT توسط عملیات SELECT ایجاد می شوند، آنها در بافر قرار می گیرند، الگاریتم عملیات JOIN ، tuple ها را از بافر مصرف میک ند، و آن tuple هایی که از عملیات JOIN ناشی می شوند در الگاریتم عملیات طرح لوله ای می شوند. مزیت لوله ای کردن، هزینه مقرون بصرفه است که نتایج واسطه در دیسک و خواندن آنها در عقب برای عملیات بعدی را ندارد.

٤. ١٨- بکارگیری انتخاب پذیری و برآوردهای هزینه در بهینه سازی پرس و جو: بهینه ساز پرس و جو نباید به قوانین ذهنی منحصراً وابسته باشد، آن باید هزینه های اجرای پرس و جو را با استفاده از استراتژی های اجرایی برآورد و مقایسه کند و باید ، استراتژی را با حداقل برآورد هزینه انتخاب کند. برای کار با این روش، برآوردهای هزینه درست لازم می باشند، تا اینکه استراتژی های متفاوتی مقایسه شوند. علاوه بر آن، ما باید تعداد استراتژی های اجرایی در نظر گرفته شده را محدود کنیم، در غیر اینصورت، زمان زیادی صرف ایجاد برآوردهای هزینه برای استراتژیهای اجرایی ممکن خواهد شد. از اینرو، این روش برای پرس و جوهای کامپایل شده مناسب تر است، جایی که بهینه سازی در زمان کامپایل انجام می شود و کد استراتژی اجرایی حاصله در زمان اجرا مستقیماً ذخیره و اجرا می

شود. برای پرس و جوهای تفسیر شده، که کل مرحله نشان داده شده در تصویر ۱.۱۸ در زمان اجـرا رخ می دهد، بهینه سازی در مقیاس کامل، زمان واکنش را کند می کند. بهینه سازی ماهرانه تـر بـرای پرس و جوهای کامپایل شده تعیین می شود، در عوض، بهینه سازی با صرف زمان کمتر، در بهتـرین حالت برای پرس و جوهای تفسیر شده عمل می کند. ما این روش را بهینه سازی پرس و جو مبتنـی بر هزینه می نامیم و آن از تکنیک های بهینه سازی سنتی استفاده می کند که فضای راه حل و جـواب برای مسئله را برای راه حلی جستجو می کند که تابع هدفمند و واقعـی (هزینـه) را بـه حـداقل مـی رساند. توابع هزینه بکار رفته در بهینه سازی پرس و جـو، برآوردهـا هسـتند و توابـع هزینـه دقیقـی نیستند، لذا، بهینه سازی، استراتژی اجرایی پرس و جو را انتخاب می کنـد کـه مـورد بهینـه ای نمـی باشد. در بخش ۱ .٤ .۱۸ ما روی مؤلفه های هزینه اجرایی پرس و جو بحث می کنیم، این اطلاعـات در کاتالوگ DBMS حفظ می شود. در بخش ۳ .٤ .۱۸. ما روی مؤلفه های هزینه اجرایی پرس و جـو بحث می کنیم، این اطلاعات در کاتالوگ DBMS حفظ مـی شـود. در بخـش ۳ . ٤. ۱۸ ، مثالهـایی از توابع هزینه برای عملیات SELECT را ارائه می دهیم و در بخش ٤ .٤. ۱۸ ، روی توابـع هزینـه بـرای عملیات های JOIN دو راهه بحث می کنیم.

بخش ۵. ٤. ۱۸ روی اتصالات چند راهی بحث می کند و بخش ٦. ٤. ۱۸ مثالی را ارائه می دهد.

۱. ٤. ۱۸– مؤلفه های هزینه برای اجرای پرس و جو: هزینه اجرای پرس و جو شامل مؤلفه های زیـر است:

۱– هزینه دستیابی به ذخیره ثانویه : این هزینه جستجو ، خواندن و نوشتن بلوکهای داده هـایی اسـت که در ذخیره ثانویه روی دیسک باقی می ماند. هزینه جستجوی ثبت ها در فایل به نـوع سـاختارهای

دستیابی روی فایل مثل مرتب کردن،hashing ، شاخص های اولیه یا ثانویه بستگی دارد. علاوه بـر آن، عواملی مثل خواه بلوکهای فایل روی سیلندر دیسک یکسانی اختصـاص مـی یابنـد یـا روی دیسـک پخش می شوند روی هزینه دستیابی اثر می گذارد.

۲- هزینه ذخیره : این هزینه ذخیره هر نوع فایل واسطه ای است که با استراتژی اجرایی برای پرس و جو ایجاد می شود.

۳- هزینه محاسباتی:این هزینه اجرای عملیات ها در حافظه روی بافرهـای داده هـا در طـول اجـرای پرس و جو می باشد.

این عملیات ها شامل جستجو و مرتب کردن ثبت ها ، ادغام ثبت ها برای اتصال، و اجرای محاسبات روی مقادیر فیلد است.

٤- هزینه کاربرد حافظه : این هزینه مربوط به تعداد بافرهای حافظه مورد نیاز در طول اجرای پرس و جو می باشد.

٥- هزینه ارتباطات : این هزینه حمل پرس و جو و نتایج حاصـلهٔ آن از محـل پایگـاه اطلاعـاتی بـه سایت یا ترمینال است جایی که پرس و جو نشأت می گیرد.

برای پایگاههای اطلاعاتی بزرگ ، تأکید اصلی روی به حداقل رساندن هزینه دستیابی به ذخیره ثانویه است. توابع هزینه ساده، عوامل دیگر را نادیده مـی گیرنـد و اسـتراتژی هـای اجرایـی پـرس و جـو متفاوت را در اصطلاحات تعداد انتقال های بلوک بین دیسک و حافظه اصلی مقایسه می نمایند. برای پایگاههای اطلاعاتی کوچکتر، جایی که اکثر داروها در فایل های موجود در پرس و جـو مـی تواننـد کاملاً در حافظه ذخیره شوند، تأکید روی به حداقل رساندن هزینـه محاسـباتی اسـت. در پایگاههـای

اطلاعاتی توزیع شده، جایی که بسیاری از سایت ها شامل می شوند، هزینه ارتباطات باید به حداقل برسند. شامل شدن تمام مؤلفه های هزینه در تابع هزینه بعلت شکل جایگزینی اوزان مناسب در مؤلفه های هزینه، مشکل است. بدین علت است که بعضی از توابع هزینه ، فقط عامل تکی، دستیابی دیسک را در نظر می گیرند. در بخش بعدی روی بعضی از اطلاعاتی بحث می کنیم که برای تدوین توابع هزینه لازم می باشد.

۲ .٤. ۱۸– اطلاعات کاتالوگ بکار رفته در توابع هزینه:برای برآورد هزینه های استراتژیهای گوناگون اجرایی، ما باید مسیر هر نوع اطلاعاتی را حفظ کنیم که برای توابع هزینه لازم می باشد. این اطلاعات ممکن است در کاتالوگ DBMS ذخیره شوند، جایی که توسط بهینه ساز پرس و جو دستیابی می شود. اول اینکه، ما باید اندازه هر فایل را بدانیم. برای فایلی که ثبت های آن، همگی از نوع یکسانی هستند، تعداد ثبت ها (tuple ها) (γ) ،اندازه ثبت (میانگین) (R) ،و تعداد بلوک ها (b) لازم می باشند. فاکتور یا عامل بلوکه کردن (bf) برای فایل ممکن است لازم باشد. ما باید مسیر متد دستیابی اولیه و ویژگیهای دستیابی اولیه را برای هر فایل حفظ کنیم. ثبت های فایل ممکن است مرتب نشوند یا توسط ویژگی با یا بدون شاخص خوشه ای سازی یا اولیه یا hash روی ویژگی کلیدی مرتب شوند. اطلاعات روی تمام شاخص های ثانویه و ویژگیهای شاخص دهی حفظ می شوند. تعداد سطوح (x) هر شاخص چند سطحی، برای توابع هزینه ای نیاز است که تعداد دستیابی های بلوک را برآورد می کند که در طول اجرای پرس و جو رخ می دهد، در بعضی از توابع هزینه ، تعداد بلوکهای شاخص یک سطحی (b_{II}) لازم می باشد.

پارامتر مهم دیگر، تعداد مقادیر متمایز (d) ویژگی و انتخاب پذیری آن (Sl) است، که کد ثبت هایی است که شرط تساوی روی ویژگی را برآورده می سازد. این به برآورد اصلیت انتخاب (S=sl*r) ویژگی امکان عمل می دهد که میانگین تعداد ثبت هایی است که شرط انتخاب تساوی روی آن ویژگی را برآورده می سازد. برای ویژگی کلیدی ، S=1 , d=1 , SL=1/r است. برای ویژگی غیر کلیدی، با ایجاد فرضیه ای که مقادیر متمایز d ، در میان ثبت ها بطور یکنواختی توزیع می شوند، ما SL=(1/d) را برآورد می کنیم و لذا S=(r/d) است. اطلاعاتی از قبیل تعداد سطوح شاخص برای حفظ آسان است چون آن اغلب تغییر نمی کند. بهر حال ، اطلاعات دیگری ممکن است مکرراً تغییر کند، برای مثال، تعداد ثبت ها r در فایل هر زمان که ثبت درج یا پاک می شود، تغییر می کند. بهینه ساز پرس و جو به بستن نیاز دارد ولی ضرورتاً به مقادیر بیش از دقیقه این پارامترها برای استفاده در برآورد هزینه استراتژیهای اجرایی گوناگون نیاز نمی باشد.

در دو بخش بعدی، ما چگونگی استفاده این پارامترها در توابع هزینه برای بهینه ساز پرس و جو مبتنی بر هزینه را بررسی می کنیم.

۳ . ٤. ۱۸ – مثالهای توابع هزینه برای SELECT : اکنون توابع هزینه را برای الگاریتم های انتخاب Sy تا S8 بحث شده در بخش ۲ . ۲. ۱۸ در اصطلاحات تعداد انتقال های بلوک بین حافظه و دیسک ارائه می دهیم. این توابع هزینه، برآوردهایی هستند که زمان محاسباتی، هزینه ذخیره، و عوامل دیگر را نادیده می گیرند.

هزینه برای متد Si به دستیابی های بلوک CSi اشاره می کند.

S1= روش جستجوی خطی (برنامه سازی پرقدرت) : ما تمام بلوکهای فایل را برای بازیابی تمام ثبت ها در برآورده ساختن شرط انتخاب جستجو می کنیم و $C_{sta}=b$ است برای شرط تساوی روی کلید، تنها نیمی از بلوکهای فایل در حد میانگین جستجو می شوند قبل از یافتن ثبت لذا $C_{S1b}=(b/2)$ چنانچه ثبت مشاهده شود، چنانچه هیچ ثبتی، شرط را برآورده نسازد، $C_{s1b}=b$ است.

S2 = جستجوی بنیادی:این جستو تقریباً به $C_{S2} = \log_2 b + [cs/bfr)1-1$ بلوک فایل دستبای پیدا می کند. این به $\log_2 b$ کاهش می یابد در صورتی که شرط تساوی روی ویژگی منحصر بفرد (کلیدی) قرار داشته باشد، چون در این مورد s=1 است.

S3 : بکارگیری شاخص اولیه (s3a) یا کلید hash (s3b) برای بازیابی ثبت تکی:برای شاخص اولیه ، یک بلوک بیشتر از تعداد سطوح شاخص بازیابی می شود از اینرو $C_{S3a} = x + 1$ است. برای hashing ، تابع هزینه تقریباً برای hashing استاتیک یا hashing خطی $C_{s3b}=1$ است و آن برای hashing قابل توسعه ۲ می باشد.

S4 : بکارگیری شاخص مرتب کردن برای بازیابی ثبت های متعدد :اگر شرط قیاس <, >, <= یا >= روی فیلد کلیدی با شاخص مرتب کردن باشد، نیمی از ثبت های فایل، شرط را برآورده می سازند. این تابع هزینه $C_{S4} = 2 + (b/2)$ را ارائه می دهد. این برآورد سختی است،گر چه در حد میانگین درست است، در موارد اختصاصی کاملاً نادرست است.

S5 = بکارگیری شاخص خوشه ای سازی برای بازیابی ثبت های متعدد : با توجه به شرط تساوی، S ثبت شرط را برآورده می سازد، جایی که s ، اصلی بودن انتخاب ویژگی شاخص دهی است. بدین معنی است که به [(s/bfr)] بلوک فایل دسترسی پیدا می شود و $C_{s5}=x+[(s/bfr)]$ ارائه می گردد.

S6 : بکارگیری شاخص ثانویه $(B^+ - tree)$:روی قیاس تساوی، s ثبت، شرط را برآورده می کند، جاییکه s ، اصلیت انتخاب ویژگی شاخص دهی است. بهر حال ، چون شاخص،غیر خوشه ای می شود، هر یک از ثبت ها روی بلوک متفاوتی باقی می مانند،لذا برآورد هزینه (بدترین مورد) ، $C_{sta} = x + s$ است. این تا x+1 برای ویژگی کلیدی شاخص دهی کاهش می یابد. اگر شرط قیاس <,>= یا <= باشد و نیمی از ثبت های فایل برای برآورده ساختن شرط فرض شود، بعد نیمی از بلوکهای شاخص اولیه سطح بعلاوه نیمی از ثبت های فایل از طریق شاخص دستیابی می شوند. برآورد هزینه برای این مورد،تقریباً $C_{s6b} = x + (b_{I1}/2) + (r/2)$ است. فاکتور r/2 می تواند اصلاح شود چنانچه برآوردهای انتخاب پذیری بهتری در دسترس باشند.

S7 = انتخاب تقارنی (همبستگی) : ما می توانیم از s1 یا یکی از متدهای S2 تا S6 فوق استفاده کنیم. در مورد دومی ،از یک شرط برای بازیابی ثبت ها و بعد کنترل بافر حافظه استفاده می کنیم که آیا هر ثبت بازیابی شده، شرایط باقیمانده در ارتباط را برآورده می سازد یا خیر.

S8 = انتخاب تقارنی (همبستگی) با استفاده ازشاخص مرکب :در همان S3ap ، S5 یا S6a به نوع شاخص بستگی دارد.

مثال بکارگیری توابع هزینه: در بهینه ساز پرس و جو، شمارش استراتژی های ممکن گوناگون برای اجرای پرس و جو و برآورد هزینه برای استراتژیهای مختلف متداول است. تکنیک بهینه سازی،مثل برنامه نویسی پویا، برای یافتن برآورد هزینه بهینه (حداقل) استفاده می شود، بدون درنظرگرفتن تمام استراتژیهای اجرایی ممکن.

ما روی الگاریتم های بهینه سازی در اینجا بحث نمی کنیم، ما از مثال ساده ای برای نشان دادن چگونگی استفاده از برآوردهای هزینه ، استفاده می کنیم. فرض کنید که فایل Employee در تصویر F,S ، r_E=10000 ثبت ذخیره شده در b_E=2000 بلوک دیسک با فاکتور بلوکه کردن bfr_E=5 ثبت / بلوک دارد و مسیرهای دستیابی بقرار زیر می باشد:

۱- شاخص خوشه ای سازی روی SALARY ، با سطوح $X_{SALARY} = 3$ و میانگین اصلیت انتخاب $S_{SALARY} = 20$ است.

۲- شاخص ثانویه روی ویژگی کلیدی SSN ، با $X_{SSN} = (S_{SSN} = 1)$ است.

۳- شاخص ثانویه روی ویژگی غیر کلیدی DNO ، با $X_{DNO} = 2$ و بلوکهای شاخص سطح اول $b_{I1DNO} = 4$ است.

$d_{DNO} = 125$ مقادیر متمایز برای DNO وجود دارد،لذا اصلیت انتخاب $S_{DNO} = (r_E / d_{DNO}) = 80$ DNO است.

٤- شاخص ثانویه روی SEX ، با $X_{sex} = 1$ است. $d_{sex} = 2$ مقادیر برای ویژگی sex (جنسیت) وجود دارد، لذا میانگین اصلی انتخاب $S_{SEX} = (r_E / d_{SEX}) = 5000$ است.

ما کاربرد توابع هزینه را با مثالهای زیر نشان می دهیم:

(op1):

(op2):

(op3):

(op4):

هزینه انتخاب جستجوی خطی S1 بعنوان $C_{s1a} = b_E = 2000$ یا $C_{s1b} = (b_E / 2) = 1000$ برآورد خواهد شد. برای op1 ، می توان از متد s1 یا متد S6a استفاده کرد، برآورد هزینه بریا Sba ، $C_{s6a} = x_{ssN} + 1 = 4 + 1 = 5$ است و روی متد S1 انتخاب می شود، که هزینه میانگین آن ، C_{s1b}=1000 است. برای op2 می توان از متد s1 یا متد S6b استفاده کرد، لذا روش جستجوی خطی را برای op2 انتخاب می کنیم.

برای op3 ، می توان از متد s1 یا متد Sba استفاده کرد، لذا متد Sba را انتخاب می کنیم.

بالاخره، op4 را در نظر بگیرید که شرط انتخاب همبستگی (تقارنی) دارد. ما به برآورد هزینه با کاربرد هر یک از ۳ مؤلفه در شرط انتخاب برای بازیابی ثبت ها ، بعلاوه روش جستجوی خطی نیاز داریم. دومی، برآورد هزینه C_{s1a}=2000 را ارائه می دهد. بکارگیری شرط اول (DNO=5) ، برآورد هزینه C_{sba}=82 را ارائه می دهد. بکارگیری شرط اول SALARY>30,000 برآورد هزینه $C_{S4} = X_{SALARY} + (b_E / 2) = 3 + (2000/2) = 1003$ را ارائه می دهد. بکارگیری شرط اول(SEX=F) برآورد هزینه $C_{S6a} = X_{SEX} + S_{Sex} = 1 + 5000 = 5001$ را ارائه م یدهد. بهینه ساز، متد Sba را روی شاخص ثانویه روی DNO انتخاب می کند چون پایین ترین برآورد هزینه را دارد. شرط (DNO=S) برای بازیابی ثبت ها بکار می رود، و بخش باقیمانده شرط همبستگی (تقارنی) برای هر ثبت انتخاب شده چک می شود بعد از اینکه در حافظه بازیابی می شود.

٤. ٤. ١٨– مثالهای توابع هزینه برای JOIN : برای توسعه توابع هزینه درست برای عملیاتهای JOIN ، به داشتن برآورد برای اندازه فایلی نیاز داریم که بعد از عملیات JOIN حاصل می گردد. این بعنوان نسبت اندازه فایل اتصال حاصله به اندازه فاصل محصول Cartesian حفظ می گردد، چنانچه هر دو

برای همان فایل ورودی یکسان بکار روند، و انتخاب پذیری اتصال (Ji) نامیده می شود. اگر ما به تعداد tuple ها در رابطه R به |R| اشاره کنیم، فرمول زیر را داریم:

Js=|(R

اگر هیچ شرط اتصالی (c) وجود نداشته باشد، بعد Js=1 می شود و اتصال با CARTESIAN PRODUCT یکی است. اگر هیچ TUPLE (یا ثبتی) از رابطه ها ، شرط اتصال را برآورده نکند، بعد jS=0 است. بطور کلی، $0 \leq JS \leq 1$ است. برای اتصال در جایی که شرط C قیاس تساوی r.a=s.b است ما دو مورد خاص زیر را بدست می آوریم:

۱- اگر a کلید R باشد، پس $|s| \leq |(R \infty_c s)| 1$ ، لذا $hajs \leq (1/|R|)$ است.

۲- اگر B کلید S باشد، پس $|R| \leq |(R \infty_c S)/$ لذا $Js \leq (1/|s|)$ است.

داشتن برآورد انتخاب پذیری اتصال برای شرط های اتصال رخداده،به بهینه ساز پرس و جو امکان می دهد تا اندازه فایل حاصله را بعد از عملیات اتصال برآورد کند، اندازه های دو فایل ورودی را با استفاده از فرمول $|(R \infty_c s)| = js * |R| * |S|$ ارائه داده می شود. اکنون می توان توابع هزینه تقریبی نمونه را برای برآورد هزینه بعضی از الگاریتم های اتصال ارائه شده در بخش ۳ . ۲. ۱۸ ارائه کرد. عملیاتهای اتصال بفرم زیر هستند.

$R \infty_{A=B} S$

که B,A ویژگیهای حوزه سازگار S , R هستند. فرض کنید که R ، b_R بلوک دارد و آن S، b_S بلوک دارد:

J1 : اتصال حلقه تو در تو : فرض کنید که ما از R برای حلقه بیرونی استفاده می کنیم بعد تابع هزینه زیر را برای برآورد تعداد دستیابی های بلوک برای این متد، با فرض ۳ بافر حافظه بدست می آوریم. ما فرض می کنیم که فاکتور بلوکه کردن برای فایل حاصله ، bfr_{RS} است،و آن انتخاب پذیری اتصال بدین قرار است:

$C_{J1}= b_R+$

آخرین قسمت فرمول، هزینه نوشتن فایل حاصله در دیسک است. این فرمول هزینه می تواند برای در نظر گرفتن تعداد متفاوتی بافر حافظه تعدیل گردد، همانطوریکه در بخش ۱۸.۲.۳ بحث شده است.

J2 : اتصال حلقه تکی : اگر شاخصی برای ویژگی اتصال B از S با سطوح شاخص X_B وجود داشته باشد، می توان هر ثبت s در R را بازیابی کرد و بعد از شاخص برای بازیابی کل ثبت های تطبیق پذیری t از S استفاده کرد که t[B]=[A] را برآورده می سازد. هزینه به نوع شاخص بستگی دارد. برای شاخص ثانویه، جایی که s_B ،اصل انتخاب برای ویژگی اتصال B از S است، فرمول زیر بدست می آید:

$$C_{J2a} = b_R +$$

برای شاخص خوشه ای سازی جایی که SB اصل انتخاب B است فرمول زیر را داریم:

$C_{J2b}=b_R+$

برای شاخص اولیه، فرمول زیر بدست می آید:

$C_{J2c}=b_R+$

اگر کلید hash برای یکی از دو ویژگی اتصال، B از S ، وجود داشته باشد فرمول زیر بدست می آید:

$C_{J2d}=b_R+($

که $b \geq 1$ میانگین تعداد دستیابی های بلوک برای بازیابی ثبت با توجه به مقدار کلیدی hash آن است.

J3: اتصال ادغام – مرتب کردن: اگر فایل ها قبلاً روی ویژگیهای اتصال مرتب شده باشند، تابع هزینه برای این متد بدین قرار است:

$C_{J3a}=b_R+$

اگر باید فایل ها را مرتب کنیم، هزینه مرتب کردن باید بدان اضافه شود. می توان هزینه مرتب کردن را توسط $(2*b)+(2*b*\log_2 b)$ برای فایل از b بلوک تقریب و تخمین زد. از اینرو تابع هزینه زیر بدست می آید.

$C_{J3b}=(2*b_R*$

مثال بکارگیری توابع هزینه : فرض کنید که فایل Employee توصیف شده در مثال بخش قبلی را داریم ، و فرض کنید که فایل Department در تصویر ۵. ۷ شامل r_D = ۱۲۵ ثبت ذخیره شده در b_D =۱۳ بلوک دیسک را داریم. عملیات اتصال زیر را در نظر بگیرید:

(op6):

(op7):

فرض کنید که شاخص اولیه روی DNUMER از Department با سطح $x_{Dnumber}$ =۱ و شاخص ثانویه روی Mgrssn از Department با اصل انتخاب S_{Mgssn} =۱ و سطوح x_{mgrssn} =۲ را داریم. فرض کنید که

انتخاب پذیری برای op6 ، Js_{op6}= (1/1Department)=۱/۱۲۵ است چـون DNUMBER کلیـد Department است. همچنین فرض کنید که فاکتور بلوکه کردن برای فایل اتصال حاصله brf_{ED} =٤ ثبت در هر بلوک است. می توان هزینه را برای عملیات JOIN در OP6 با استفاده از متدهای عملـی J1 و J2 بقـرار زیـر برآورد کرد:

۱- بکارگیری متد J1 با Empuyee بعنوان حلقه بیرونی:

$C_{J1}=b_E+$

۲- بکارگیری متد J1 با Department بعنوان حلقه بیرونی

$C_{J1}=b_D+$

۳- بکارگیری متد J2 با Empleyee بعنوان حلقه بیرونی:

$C_{J2c}=b_c+$

٤- بکارگیری متد J2 با Department بعنوان حلقه بیرونی:

$C_{JLa}=b_D+($

مورد ٤ پایین ترین برآورد هزینه را دارد و انتخاب خواهد شد. توجه کنید که ۱۵ بـافر حافظـه بـرای اجرای بجای فقط دو تا در دسترس بوده اند، ۱۳ تا از آنهـا بـرای حفـظ کـل رابطـه Department در حافظه استفاده شد، و یکی بعنوان بافر روی نتیجه اسـتفاده شـده و هزینـه بـرای مـورد ۲ ، بـه فقـط $b_E+b_D+((Js_{op6}$ * r_E*r_D) / bfr_{ED}) یا 4SB کاهش یافته کـه در بخـش ۳. ۲ .۱۸ بحـث شـده اسـت. بعنـوان تمرین خواننده باید ، تحلیل مشابهی را برای op7 انجام دهد.

۵. ٤. ۱۸- پرس و جوهای رابطه متعدد (چندگانه) و مرتب کردن اتصال: قوانین تغییر شکل جبری در بخش ۲ .۳. ۱۸ شامل قانون تبادلی و قانون پیوندی برای عملیات اتصال است. با این قوانین ، بسیاری از عبارات اتصال معادل می توانند ایجاد شوند. در نتیجه ، تعدادی درخت های پرس و جوی دیگری بسرعت رشد می کنند تحت عنوان که تعداد اتصالات در پرس و جو افزایش می یابد. بطور کلی، پرس و جویی که ۸ رابطه را متصل می کند، N-1 رابطه اتصال دارند. و از اینرو ، تعداد زیاد ترتیب های اتصال متفاوت دارد.

تصویر ۷. ۱۸- دو درخت پرس و جوی کامل چپ

برآورد هزینه درخت اتصال ممکن برای پرس و جو با تعداد اتصالات زیاد ، به میزان اساسی زمان توسط بهینه ساز پرس و جو نیاز دارد. از اینرو، مختصر کردن این درختان پرس و جوی ممکن لازم می باشد. بهینه سازیهای پرس و جو، ساختار درخت پرس و جو را به آن درختان کامل چپ محدود می سازند. درخت کامل چپ، درخت نیازی است که بچه راست هر گره بدون برگ همیشه رابطه مبنا است. بهینه ساز، درخت کامل چپ ویژه را با پایین ترین هزینه برآورد شده انتخاب می کند. دو مثال از درختان کامل چپ در تصویر ۷ .۱۸ نشان داده شده است. با درختان کامل چپ، بچه راست در رابطه درونی در نظر گرفته می شود زمانی که اتصال حلقه تودرتو اجرا می شود. یک مزیت درختان کامل چپ (یا کامل راست) در این است که آنها برای لوله ای کردن قابل اصلاح هستند، همانطوریکه در بخش ۳. ۳. ۱۸ بحث شد. برای مثال، اولین درخت کامل چپ در تصویر ۷ .۱۸ را در نظر بگیرید و فرض کنید که الگاریتم اتصال ، متد حلقه تکی است، در این مورد، صفحه دیسک tuple ها در رابطه بیرونی ، برای آزمایش رابطه برای tuple (ثبت) تطبیق پذیر بکار می رود. همانطوریکه

بلوک حاصله از tuple ها از اتصال R1 و R2 ایجاد می شود، آن برای آزمایش R3 استفاده می شود. همانطوریکه صفحه حاصله از tuple ها از این اتصال ایجاد می شود، آن برای آزمایش R4 استفاده می شود. مزیت دیگر درختان کامل چپ در داشتن رابطه مبنا بعنوان یکی از ورودی ها در هر اتصال است که به بهینه ساز امکان می دهد تا از هر مسیر دستیابی روی آن رابطه استفاده کند که ممکن است در اجرای اتصال مفید باشد.

اگر تحقق پذیری بجای لوله ای کردن استفاده شود، نتایج اتصال بعنوان رابطه های موقتی تحقق می پذیرد و ذخیره می شود. ایده کلیدی از نقطه نظر بهینه ساز با توجه با مرتب کردن اتصال، یافتن مرتب کردنی است که اندازه نتایج موقتی را کاهش خواهد داد، چون نتایج موقتی توسط عملگرهای بعدی بکار می روند، و از اینرو روی هزینه اجرایی آن عملگرها اثر می گذارند.

۶.٤.۱۸– مثالی برای نشان دادن بهینه سازی پرس و جو مبتنی بر هزینه: ما پرس و جوی Q2 و درخت پرس و جوی آنرا در نظر می گیریم که در تصویر (a) ٤.۱۸ برای نشان دادن بهینه سازی پرس و جوی هزینه ای ارائه شده است.

تصویر ۸. ۱۸

Qz:SELECT

FROM

WHERE

a)

b)

c)

فرض کنید که اطلاعات آماری در مورد رابطه های نشان داده شده در تصویر ۸. ۱۸ داریم. فرمت اطلاعات نمونه کاتالوگ در بخش c ۱۷ را دنبال می کند آمار Low-value - High- value برای روشن شدن نرمال سازی شده است. درخت در تصویر (a) ٤. ۱۸ برای ارائه نتیجه مرحله بهینه سازی ذهنی جبری و شروع بهینه سازی مبتنی بر هزینه فرض می شود. اولین بهینه سازی مبتنی بر هزینه برای درنظرگرفتن، مرتب کردن اتصال است. همانطوریکه قبلاً بدان اشاره شد، فرض می کنیم بهینه ساز، فقط درختان کامل چپ را در نظر می گیرد، لذا ترتیب های اتصال بالقوه، بدون محصول Cartesian بدین قرارند:

1- Project ∞ Department ∞ Emplyee

2-

3-

4-

فرض کنید که عملیات انتخاب برای رابطه Project بکار رفته است. روش تحقق پذیرفته شده، (مادی) را در نظر بگیریم، بعد رابطه موقتی جدیدی بعد از هر عملیات اتصال، ایجاد می شود. برای بررسی هزینه ترتیب اتصال (۱) ، اولین اتصال بین Project, Department قرار دارد. هر دو متد اتصال و متدهای دستیابی برای رابطه های ورودی باید مشخص شوند. چون Department هیچ شاخصی بر طبق تصویر ۸. ۱۸ ندارد، تنها متد دستیابی موجود، پویش جدولاست. رابطه Project، عملیات انتخاب اجرا شده قبل از اتصال را دارد لذا دو انتخاب وجود دارد: پویش جدول یا بکارگیری شاخص Proj-Ploc آن، لذا

بهینه ساز باید هزینه های برآورد شده آنها را مقایسه کند. اطلاعات آماری روی شاخص Proj-Ploc تعداد سطوح شاخص x=2 را نشان می دهد. شاخص غیر منحصربفرد است، لذا، بهینه ساز، توابع داده های یکنواخت را در نظر می گیرد و تعداد اشاره گرهای ثبت برای هر مقدار Plocation در ۱۰ را برآورد می کند. این از جداول در تصویر ۸. ۱۸ با ضرب Selectivity * Num Rows محاسبه می شود، که Selectivity توسط 1/Num-Distinct برآورد می شود. لذا هزینه کاربرد شاخص و دستیابی به ثبت ها تا ۱۲ دستیابی برآورد می شود. هزینه پویش جدول تا ۱۰۰ دستیابی بلوک برآورد می شود، لذا دستیابی شاخص در حد انتظار کارآمدتر است.

در روش تحقق یافته، فایل موقتی Temp1 در اندازه ۱ بلوک برای حفظ نتیجه عملیات انتخاب ایجاد می شود. اندازه فایل با تعیین فاکتور بلوکه کردن با استفاده از فرمول Num-Rows/ Blocks محاسبه می شود که 2000/100 یا ۲۰ ردیف در هر بلوک را ارائه می دهد. از اینرو ۱۰ ثبت انتخاب شده از رابطه Project در بلوک تکی تناسب خواهد یافت. اکنون می توان هزینه برآورد شده اتصال اول را محاسبه کرد. ما فقط متد اتصال حلقه تو در تو را در نظر می گیریم، که رابطه بیرونی، فایل موقتی است، Temp1 ، و رابطه درونی Department است. چون کل فایل temp1 در فضای بافر موجود تناسب می یابد، ما به خواندن هر یک از ۵ بلوک جدول Department فقط برای یکبار نیاز داریم، لذاهزینه اتصال ٦ دستیابی بلوک بعلاوه هزینه نوشتن فایل نتیجه موقتی، Temp2 ، بهینه ساز باید اندازه temp2 را مشخص کند. چون ویژگی اتصال Dnumber ، کلیدی برای Department است، هر مقدار DNUM از Temp1 مساوی با تعداد ردیف ها در temp1 یعنی ۱۰ است. بهینه ساز اندازه ثبت را برا temp2 تعیین می کند و تعداد بلوکهای مورد نیاز برای ذخیره این ۱۰ ردیف را تعیین می کند.

بطور خلاصه ، فرض کنید که فاکتور بلوکه کردن برای Temp2 ، ۵ ردیف در هر بلوک است، لذا کل دو بلوک برای ذخیره temp2 لازم می باشند. بالاخره ، هزینه آخرین اتصال لازم است برآورد شود.

ما می توانیم از اتصال حلقه تکی روی temp2 استفاده کنیم چون در این مورد ، شاخص Emp-SSN می تواند برای آزمایش و قرارگیری ثبت های برابری از employee بکار رود. از اینرو، متد اتصال شامل خواندن در هر بلوک temp2 و جستجو هر ۵ مقدار mGRSSN با استفاده از شاخص EMP-SSN است. هر جستجوی شاخصی به دستیابی ریشه، دستیابی برگها، و دستیابی بلوک داده ها نیاز دارد. لذا ، ۱۰ جستجو به ۳۰ دستیابی بلوک نیاز دارد. افزودن دو دستیابی بلوک برای temp2 ، کل ۳۲ دستیابی بلوک برای این اتصال را فراهم می آورد. برای طرح نهایی، فرض کنید لوله ای کردن برای ایجاد نتیجه نهایی استفاده می شود، که به دستیابی های بلوک اضافی نیازی ندارد، لذا، کل هزینه برای ترتیب اتصال (۱) بعنوان مجموع هزینه های قبلی برآورد می شود. بهینه ساز هزینه ها را در حالت مشابه برای ۳ ترتیب اتصال دیگر برآورد می کند و یکی را با حداقل برآورد انتخاب می کند. ما این را بعنوان تمرین برای خواننده ارائه می دهیم.

۵. ۱۸- مرور و بررسی بهینه سازی پرس و جو در ORACLE DBMS : ORACLE (نسخه ۷) دو روش متفاوت را برای بهینه سازی پرس و جو ارائه می کند. روش مبتنی بر قانون و مبتنی بر هزینه، با روش مبتنی بر قانون ، بهینه ساز، طرح های اجرایی مبتنی بر عملیات های طبقه بندی شده را انتخاب می کند. ORACLE جدول ۱۵ مسیر دستیابی طبقه بندی شده (درجه بندی شده) را حفظ می کند که درجه بندی پایین تر دلالت بر روش کارآمدتر دارد. مسیرهای دستیابی از دستیابی جدول تا RawID ردیف می شود، جایی که ROWID ، آدرس فیزیکی ثبت را تعیین می کندتا شامل فایل داده ها ، بلوک

داده ها و offset ردیف در بلوک برای پوشش کامل جدول است، کـه کـل ردیـف هـا در جـدول بـا خواندن چند بلوکی جستجو می شوند. بهر حال ، روش مبتنی بـر قـانون ، بواسـطه روش مبتنـی بـر هزینه مرحله بندی می شود که بهینه ساز، مسیرهای دستیابی دیگری و الگاریتم های عملگر را بررسی می کند و طرح اجرایی را با حداقل هزینه برآورد شـده انتخـاب مـی کنـد. جـداول کاتـالوگ شـامل اطلاعات آماری درحالت مشابهی استفاده می شوند، همانطوریکه در بخش ٦. ٤. ١٨ بحث شده است. هزینه پرس و جوی برآورد شده به تناسب زمان (منقضی) سپری شدهمورد انتظار و مورد نیـاز بـرای اجرای پرس و جو با طرح اجرایی مورد نظر است. بهینه ساز ORACLE این هزینه را براساس کـاربرد برآورد شد، منابع مثل I/O ، زمان Cpu و حافظه مورد نیاز ، محاسبه می کند. هدف بهینه سازی مبتنـی بر هزینه ORACLE به حداقل رساندن زمان (منقضی) سپری شده برای پردازش کل پرس و جو است. افزودن جذابیت به بهینه ساز پرس و جوی ORACLE ، قابلیت برای توسـعه دهنـده برنامـه کـاربردی جهت تعیین اشارهها برای بهینه ساز است. ایده این است که توسعه دهنده برنامه کاربردی،اطلاعـات بیشتری در مورد داده ها نسبت به بهینه ساز دارد. برای مثال ، جـدول Employee نشـان داده شـده در تصویر ٥ .٧ را در نظر بگیرید. ستون Sex آن جدول فقط دو مقدار متمـایز دارد. اگـر ١٠٠٠٠ کارمنـد وجود داشته باشد، بعد بهینه ساز برآورد می کند که نیمی زن و نیمی مرد هستند، با فرض توزیع داده های یکنواخت. اگر شاخص ثانویه وجود داشته باشد، بیشتر از مورد بکار رفته نمی باشد. بهر حـال ، اگر توسعه دهنده برنامه کاربردی بداند که فقط ١٠٠ کارمند مرد وجود دارد اشاره در پـرس و جـوی SQL مشخص می شود و شرط بند WHERE آن Sex=M است تا اینکـه شـاخص مربـوط در پـردازش پرس و جو بکار رود. اشاره های گوناگونی مشخص می شوند که بدین قرارند:

- روش بهینه سازی برای بیانیه SQL

- مسیر دستیابی برای جدول دستیابی شده بیانیه

- ترتیب اتصال برای بیانیه اتصال

- عملیات اتصال ویژه در بیانیه اتصال

بهینه سازی مبتنی برهزینه ORACLE8 ، مثال خوبی از روش مهم و برجسته در نظر گرفته شده برای بهینه سازی پرس و جوهای SQL در RDBMS های بازرگانی است.

۱۸.٦- بهینه سازی پرس و جوی معنایی: روش متفاوت برای بهینه سازی پرس و جو ، بنام بهینه سازی پرس و جوی معنایی، پیشنهاد شده است. این تکنیک که ممکن است درت رکیب با تکنیک های بحث شده قبلی بکار رود،از محدودیت های تعیین شده روی طرح پایگاه اطلاعاتی مثل ویژگیهای منحصر بفرد و محدودیت های پیچیده تر دیگر، به منظور تعدیل یک پرس و جو به پرس و جوی دیگر که برای اجرا کارآمدتر است، استفاده می کند. ما روی این روش به تفضیل بحث نمی کنیم ولی فقط با مثال ساده ای آنرا روشن می سازیم. پرس و جوی SQL را در نظر بگیرید.

SELECT

FROM

WHERE

این پرس و جو ، اسامی کارمندانی را بازیابی می کند که بیشتر از ناظرین و سرپرستان خود عایدی بدست می آورند. فرض می شود که ما محدودیتی روی طرح پایگاه اطلاعاتی داشتیم که بیان کرده که هیچ کارمندی نمی تواند بیشتر از سرپرست خود عایدی بدست آورد. اگر بهینه ساز پرس و جوی

معنایی موجودیت این محدودیت را چک کند، به اجرای پرس و جو ابداً نیازی نمی باشد چون می داند که نتیجه پرس و جو خالی و تهی خواهد بود.

این زمان قابل ملاحظه ای را صرف می کند چنانچه کنترل محدودیت بطور کارآمدی بتواند انجام شود. بهرحال، جستجواز طریق محدودیت های بسیار در یافتن آنهایی که عملی هستند برای پرس و جوی ارائه شده و بهینه سازی می شوند، می تواند ضامن صرف زمان باشد. با توجه به قوانین فعال در سیستم های پایگاه اطلاعاتی، تکنیک های بهینه سازی پرس و جوی معنایی در DBMS های درآینده به ثبت می رسند.

۱۸.۷- خلاصه : در این فصل، مروری بر تکنیک های بکار رفته توسط DBMS ها در پردازش و بهینه سازی پرس و جوهای سطح بالا را ارائه کردیم. ابتدا روی چگونگی ترجمه پرس و جوهای SQL به عملیات جبری رابطه ای و بعد چگونگی اجرای عملیات های جبری رابطه ای گوناگون توسط DBMS بحث کردیم. ما مشاهده کردیم که بعضی از عملیات ها، بویژه ، SELECT و JOIN انتخاب های اجرایی بسیاری دارند ما روی چگونگی الحاق عملیات ها در طول پردازش پرس و جو برای ایجاد اجرای مبتنی بر جریان یا لوله ای کردن بجای اجرای تحقق یافته (مادی شده) بحث کردیم. بدنبال آن، روشهای ذهنی برای بهینه سازی پرس و جو را توصیف کردیم که از قوانین ذهنی و تکنیک های جبری برای بهبود کارآیی اجرای پرس و جو استفاده می کند. ما نشان دادیم که چطور درخت پرس و جویی که عبارت جبری رابطه ای را ارائه می کند می تواند با سازماندهی مجدد گره های درخت و تغییر شکل آن به درخت پرس و جوی معادل دیگر که برای اجرا کارآمدتر است، بهینه سازی شود. ما قوانین تغییر شکل معادل را ارائه کردیم که برای درخت پرس و جو بکار می روند.

بعد طرح های اجرایی پرس و جو را برای پرس و جوهای SQL ارائه کردیم که طرح های اجرایی متد را به عملیاتهای درخت پرس و جو اضافه کرد. بعد روی روش مبتنی بر هزینه برای بهینه سازی پرس و جو بحث کردیم. و نشان دادیم که چطور توابع هزینه برای بعضی از الگاریتم های دستیابی به پایگاه اطلاعاتی توسعه می یابند و چطور این توابع هزینه برای برآورد هزینه های استراتژیهای اجرایی گوناگون بکار می روند. ما مروری بر بهینه ساز پرس و جوی ORACLE را ارائه کردیم و به تکنیک بهینه سازی پرس و جوی معنایی اشاره کردیم.