معرفی روشی کارا برای جستجو درپایگاه داده

کیمیا رضایی کلانتری [۱] ، دکتر حسن رشیدی حرم آبادی [۲]،معصومه بورجندی [۳]

[۱] مدرس دانشگاه آزاد اسلامی واحد ساری، دانشکده فنی و مهندسی
ساری ، ایران
Rezaeikalantari@gmail.com

[۲]استاد دانشگاه آزاد اسلامی واحد قزوین،دانشکده برق و کامپیوتر
قزوین،ایران
Hrashi@gmail.com

[۳] مدرس دانشگاه آزاد اسلامی واحد علی آباد، دانشکده فنی و مهندسی
علی آباد ، ایران
m.boorjandi@yahoo.com

چکیده :

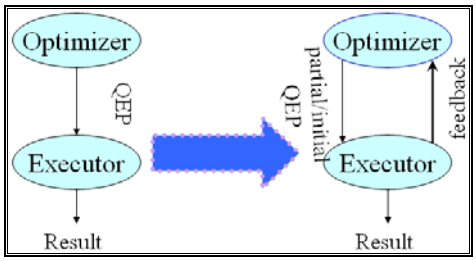
بهینه سا زی تقاضا در هر دوی پایگاه داده های متمرکز و توزیع شده مورد استفاده قرار می گیرد و شامل الگوریتمهای مختلفی می باشد. . مقوله ای که ما در این مقاله مورد بررسی قرار دادیم بهینه سازی تقاضا تحت رتبه بندی می باشد که برای بدست آوردن پاسخهای TOP K بکار می روند. دراین مقاله چند روش برای بهینه سازی تقاضا تحت رتبه بندی را مورد بررسی قرار داده و چگونگی اجرای تطبیقی تقاضای رتبه بندی را بیان می کنیم و در پایان نتیجه حاصل از بکارگیری این روش را با روش سنتی مقایسه می کنیم.

واژه های کلیدی: بهینه سازی تقاضا، رتبه بندی تجمعی، هرس کردن،برنامه نویسی پویا

۱- مقدمه

یک پارامتر مهم در روشهای بهینه سازی تقاضا زمان اجرا و فضای جستجو می باشد. چیزی که واضح است این است که هر چه فضای جستجوی وسیعتری داشته باشیم می توانیم طرح مناسب تری را انتخاب نماییم درعوض در مقابل می بایست زمان بیشتری را صرف نماییم. اما روشی مناسب است که بر اساس نیازهای موجود یک حد میانی بین دقت و سرعت را برای رسیدن به کارایی مطلوب برگزینند. البته این بسته به نوع مسئله، متفاوت است. بهینه سا زی تقاضا بوسیله تخمین هزینه[7,8,9] و بهینه سا زی تقاضا با استفاده از قوانین معادل وجایگزینی آنها با هم می باشد[10]. البته برخی از این روشها از ترکیب ایندو استفاده می کنند. بهینه سا زی تقاضا در هر دوی پایگاه داده های متمرکز و توزیع شده مورد استفاده قرار می گیرد و شامل الگوریتمهای مختلفی می باشد. اما به طور کلی این الگوریتمها به سه دسته اصلی تقسیم می شوند: الگوریتمهای جستجوی جامع، اکتشافی و تصادفی. الگوریتم های برنامه نویسی پویا از معمول ترین نوع الگوریتمهای جستجوی جامع می باشند که این الگورتمها زمان اجرا و فضای جستجو از مرتبه نمایی دارند همچنین با قاعده، قطعی و ساختمند می باشند؛ در نتیجه تضمین می کنند بهترین طرح را بر اساس یک مدل هزینه خاص بدست می آورند. بهینه سازی بر پایه قاعده که در آن فضای جستجو و الگوریتم جستجو را از هم متمایز می کند. بوسیله قوانین فضای جستجو را تعریف می کند و برای جستجو از

الگوریتمهای جستجوی عمومی استفاده می کنند[22].بهینه سازی تقاضا به طور تطبیقی در این نوع روشها یک تعامل بین بهینه سازی و اجرا وجود دارد. در شکل ۱یک نمایی از بهینه سازی تقاضا به طور تطبیقی را مشاهده می کنید. بهینه سازی تقاضای مفهومی یکی دیگر از تحقیقات در زمینه بهینه سازی است که طرح را بر اساس مفهوم تقاضا انتخاب نماییم که در آن از مفاهیم پایگاه داده های منطقی استفاده شده است. مباحث دیگر شامل بهینه سازی تقاضا های چندگانه، بهینه سازی تقاضاها به صورت موازی یا به صورت توزیع شده می باشد[7,2].



شکل۱: بهینه سازی تقاضا به صورت تطبیقی

در بالا بررسی اجمالی برالگوریتمهای بهینه سازی تقاضا را بیان کردیم که اکثر این الگوریتمها برای عمل اتصال پیاده سازی شده اند زیرا زمانبرترین قسمت در یک تقاضا می باشد. مقوله ای که ما مورد بررسی انجام دادیم بهینه سازی تقاضا تحت رتبه بندی می باشد که برای بکار می روند. دراین مقاله چند **TOP K**بدست آوردن پاسخهای روش برای بهینه سازی تقاضا تحت رتبه بندی را مورد بررسی قرار داده **[7,8]**و چگونگی اجرای تطبیقی تقاضای رتبه بندی را بیان می کنیم.

۲- کارهای پیشین

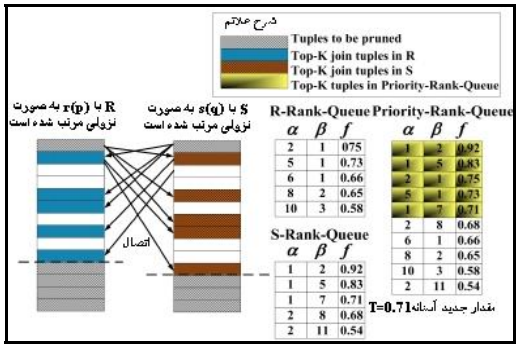
در این بخش به مطالعه زمینه هایی که در بهینه سازی تقاضای تحت رتبه بندی در این چند سال مورد بررسی واقع شده اند می پردازیم.

۲-۱- یک دستاورد مبتنی بر هرس کردن برای پشتیبانی اتصال تقاضاها یی با K جواب بهتر[16]

یک موضوع مهم در مجتمع سازی اطلاعات در مقیاس بالا، انتخاب جوابهای رتبه بندی K جواب بهتر ازچندین منبع می باشد به طوریکه هزینه انتقال کمینه گردد. ایده اصلی آن به این صورت است که پارامترهای ورودی دستاورد رتبه بندی K جواب بهتر مبتنی بر هرس کردن شامل: رابطه های R,S ، شرط اتصال $r(a) \quad \theta \quad s(b)$، تابع رتبه بندی افزایشی یکنواخت $f(r(p),s(q))$ و K تعداد جوابهای مطلوب اتصال می باشند. $r(a),s(b)$ صفتهای اتصال و $r(p),s(q)$ صفتهای رتبه بندی می باشند.

رابطه R و S بر اساس صفتهای رتبه بندیشان به صورت نزولی مرتب می شوند، ایده اصلی پیشنهاد شده این است که مقدار رتبه بندی K جواب بهتر، تابع رتبه بندی با هرس کردن مکرر تاپلهایی از R و S که نتیجه اتصالی با مقدار رتبه بندی کمتر از مقدار آستانه دارند، بدست می آید. سرانجام، تاپلها بر طبق مقدار رتبه بندی با هم الحاق می شوند. در شکل۲ نمونه ای از آنرا مشاهده می کنید.

T حد پایین تابع رتبه بندی f می‌باشد. α شماره سطر رکورد در R می باشد. β شماره سطر رکورد در S می باشد. TOP K join tuples in R تاپلهایی با مقدار رتبه بندی بیشتر در R که می‌توانند با اولین تاپل در S الحاق شوند. اگر تعداد تاپلهایی از R که می توانند با اولین رکورد در S الحاق شوند کمتر از K باشد، این عمل با تاپلهای بعدی در S ادامه می یابد تا زمانیکه K تا تاپل برای اتصال یافت شود یا S به پایان برسد. TOP K join tuples in S تاپلهایی با مقدار رتبه بندی بیشتر در S که می توانند با اولین تاپل در R الحاق شوند. اگر تعداد تاپلهایی از S که می توانند با اولین رکورد در R الحاق شوند کمتر از K باشد، این عمل با تاپلهای بعدی در R ادامه می یابد تا زمانیکه K تا تاپل برای اتصال یافت شود یا R به پایان برسد. $R\_Rank\_Queue(\alpha,\beta,f)$ این صف شماره سطر TOP K join tuples در R، شماره سطر متناظر TOP K join tuples در S و مقدار رتبه بندی نتایج اتصال را نگهداری می‌کند. $S\_Rank\_Queue(\alpha,\beta,f)$ این صف شماره سطر TOP K join tuples در R، شماره سطر متناظر TOP K join tuples در S و مقدار رتبه بندی نتایج اتصال را نگهداری می‌کند. $Priority\_Rank\_Queue(\alpha,\beta,f)$ این صف اجتماع $R\_Rank\_Queue,\ S\_Rank\_Queue$ را نگهداری می کند.



شکل۲. مثالی از روش هرس کردن برای تقاضاها یی با K جواب بهتر

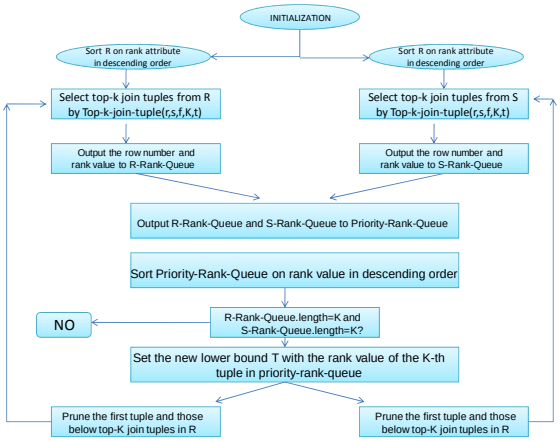
۲-۲. بهینه سازی تقاضای تحت رتبه بندی**[10]**

رتبه بندی یک خصوصیت مهم است که می بایست توسط موتورهای تقاضای رابطه ای فعلی پشتیبانی شود. اخیراً چندین عملگر تقاضای اتصال رتبه بندی بر پایهٔ الگوریتم های رتبه بندی تجمعی پیشنهاد شده است. عملگر های اتصال رتبه بندی تدریجاً هنگام عمل اتصال، نتایج اتصال را رتبه بندی می کنند. در این مقاله یک محیط بهینه سازی تقاضا آگاه از رتبه بندی بوجود آورده شده است که بطور کامل عملگرهای اتصال رتبه بندی را درون موتورهای تقاضای رابطه ای قرار داده است. محیط بر پایه توسعه سیستم الگوریتم برنامه سازی پویا R در دو زمینهٔ شمارش و هرس می باشد. بر خلاف عملگرهای اتصال

سنتی، بهینه سازی برای عملگرهای اتصال رتبه بندی وابسته به تخمین کاردینالیتی ورودی این عملگرها می باشد. در این مقاله ابتدا اندازه ورودی مورد نیاز برای الگوریتم های تجمعی رتبه بندی به طور بهینه ارزیابی می شود. در نهایت به طور عملی یک محیط طراحی شده که با آزمایشها صحت محیط و دقت مدل تخمینی پیشنهاد شده را مورد بررسی قرار دادند.

۳-۲ معماری کلی روش:

معماری کلی روش پیشنهادی در شکل ۳ نشان داده شده است .



شکل۳.معماری کلی روش

۳- الگوریتم های ارائه شده برای رتبه بندی

رتبه بندی یک خصوصیت مهم است که می بایست توسط موتورهای تقاضای رابطه ای فعلی پشتیبانی شود. اخیراً چندین عملگر تقاضای اتصال رتبه بندی بر پایهٔ الگوریتم های رتبه بندی تجمعی پیشنهاد شده است. عملگر های اتصال رتبه بندی تدریجاً هنگام عمل اتصال ، نتایج اتصال را رتبه بندی می کنند. در این مقاله یک محیط بهینه سازی تقاضا آگاه از رتبه بندی را بوجود آورده شده است که بطور کامل عملگرهای اتصال رتبه بندی را درون موتورهای تقاضای رابطه‌ای قرار داده است. محیط بر پایه توسعه سیستم الگوریتم برنامه سازی پویا R در دو زمینهٔ شمارش و هرس می باشد. بر خلاف عملگرهای اتصال سنتی، بهینه سازی برای عملگرهای اتصال رتبه بندی وابسته به تخمین کاردینالیتی ورودی این عملگرها می باشد. در این مقاله یک مدل احتمالی برای تخمین کاردینالیتی ورودی فراهم آورده شده است و از این رو هزینه عملگر اتصال رتبه بندی تخمین زده می شود .

۳-۱- رتبه بندی تجمعی

الگوریتم های رتبه بندی تجمعی پایگاه داده را به عنوان چندین لیست می بینند. هر لیست شامل رتبه بندی تعدادی اشیاء می باشد و به هر شی در یک لیست یک امتیاز اختصاص می یابد که رتبه آنرا در درون لیست مشخص می کند و سپس لیست خروجی طبق ترکیب امتیازات مرتب می شود. برای بدست آوردن رتبه بندی تجمعی، الگوریتم به طور افزایشی یک وضعیت موقتی که شامل همه امتیازات اشیاء دیده شده می باشد، را نگهداری می نماید. الگوریتم اشیاء را از لیستها تا زمانیکه اطلاعات کافی برای بدست آوردن K جواب بهتر را داشته باشد، بازیابی می کند. به طور کلی، الگوریتم های تجمعی پیشنهادی می توانند بر طبق دو معیار اصلی گروه بندی شود. اولین گروه بندی بر طبق نوع دسترسی بر روی لیستهای ورودی است که می تواند دسترسی مرتب شده یا دسترسی تصادفی باشد. در دومین گروه بندی برپایه فرضیات بر روی اشیاء رتبه بندی شده می باشد که هر ورودی می تواند شامل مجموعه اشیاء متفاوتی باشد.[9,10,11].

۳-۲- عملگرهای تقاضای اتصال رتبه بندی

برای پشتیبانی الگوریتم های تجمعی رتبه بندی شده در یک سیستم پایگاه داده، می توان این الگوریتم ها را در سطح برنامه های کاربردی به عنوان توابع تعریف شده توسط کاربر یا آنها را به عنوان عملگرهای تقاضا در هسته DBMS پیاده سازی کرد. عملگرهای اتصال رتبه بندی به ورودی های رتبه بندی شده، به تابعی برای فرآهم کردن تاپل بعدی و به تابع یکنواخت رتبه بندی که مجموع امتیازها را محاسبه می کند، احتیاج دارند

پیاده سازی ممکن اتصال رتبه بندی شامل HRJN , NRJN می‌باشند.HRJN دارای۲ جدول پراکندگی برای دو ورودی اتصال و همچنین یک صف اولویت می باشد که اتصالهای معتبر که بر اساس ترکیب امتیازات مرتب شده اند را نگهداری می نماید. در HRJN یک الگوریتم رتبه بندی تجمعی وجود دارد. الگوریتم یک مقدار آستانه را نگهداری می کند که حد بالای امتیاز همهٔ ترکیبات اتصال که هنوز دیده نشده اند را تعیین می کند. یک نتیجه اتصال اگر امتیازی بزرگتر یا مساوی مقدار آستانه داشته باشد به عنوان جواب انتخابی بعدی گزارش می شود. از طرف دیگر، الگوریتم بوسیله خواندن تاپلها از ورودی های چپ و راست ادامه می یابد در هر مرحله الگوریتم تصمیم می گیرد کدام ورودی بر اساس استراتژی های مختلف انتخاب شود. NRJN شبیه به HRJN است بجز اینکه آن برای بدست آوردن نتایج اتصال معتبر از استراتژی حلقه تو در تو استفاده می نماید. وضعیت داخلی NRJN شامل فقط یک صف اولویت از همه ترکیبات اتصال دیده شده می باشد. همانند NRJN نیز یک مقدار آستانه دارد که حد بالای امتیاز های همه نتایج اتصال دیده نشده، می باشد.[9,10]

۳-۳- طرح شمارش با استفاده از برنامه نویسی پویا

بهینه ساز می بایست طرح هایی با اتصال n تایی را به صورت دنباله ای از اتصالهای دوتایی بوسیله عملگرهای اتصال دودویی، تبدیل کند سپس متد اتصال بهینه برای هر اتصال دودویی پیدا کند. اساس دستاورد برنامه نویسی پویا بر پایه این فرض است که مدل هزینه اساس بهینه گی را برآورده می سازد. برای جلوگیری از طرح های اضافی بوجود آمده ، برنامه نویسی پویا یک ساختار را برای نگهداری

طرحهای هرس نشده بکار می برد. هر ورودی این ساختار برابر با یک زیر مجموعه از جدولها در تقاضا می باشد. الگوریتم در یک روش پایین-بالا بوسیله بوجود آوردن طرح ها برای جدولهای تنها، اجرا می شوند به همین ترتیب تا کل تقاضا ادامه می یابد. طرح ها با مجموعه جدول های بزرگتر از طرح های با مجموعه جدولهای کوچکتر تشکیل می شوند. الگوریتم یک طرح با هزینه ی بیشتر را در صورتیکه طرحی با هزینه ی کمتر یا با خصوصیات عمومی بیشتر در این ساختار قرار داشته باشد، هرس می کند و در نهایت کم هزینه ترین طرح برای اتصال N جدول برگردانده می شود. [10]

۳-۴- توسعه فضای شمارشی:

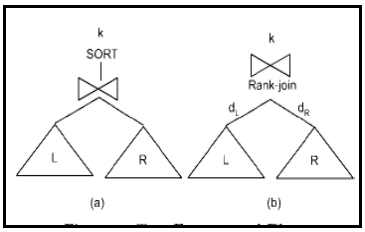
از آنجاییکه ترتیبهای مطلوب اکنون شامل خصوصیاتی که در تابع رتبه بندی نقش دارند نیز می باشند، بهینه ساز می بایست در طرحهای شمارش در سطح های بالاتر، بر اساس عبارت رتبه بندی، ترتیبهای مطلوبی که ایجاد می شود را برای استفاده عملگرهای اتصال رتبه بندی سطوح بالاتر در نظر بگیرد (شکل ۵).

برای یک تقاضا با n رابطه ورودی، T_1 تا T_n فرض کنید که یک تابع رتبه بندی $f(S_1,S_2,...,S_n)$ وجود دارد که S_i یک عبارت رتبه بندی بر روی رابطه T_i می باشد. برای دو رابطه ورودی L و R، فضای طرحهای اتصال رتبه بندی در صورت دارا بودن قابلیتهای شایستگی اتصال، انتخابهای اتصال و ترتیب اتصال، توسعه می یابد.

۳-۵- طرح های هرس:

یک زیر طرح P_1 نسبت به زیر طرح P_2 هرس می شود اگر و تنها اگر P_1 هزینه بیشتر و خصوصیات کمتری نسبت به P_2 دارد. البته برای دو طرح P_1 و P_2 با خصوصیت فیزیکی یکسان اگر P_1 یک طرح *pipeline* (مثل طرح اتصال حلقه تو در تو) باشد و P_2 یک طرح غیر *pipeline* می باشد (طرح اتصال مرتب سازی- ادغام) باشد. P_1 نمی تواند نسبت به P_2 هرس شود حتی اگر P_2 از P_1 کم هزینه تر باشد. در دنیای واقعی، هزینه مدل برای عملگرهای تقاضای مختلف، خیلی پیچیده می باشد و به پارامترهای زیادی وابسته است. پارامترها شامل کاردینالیتی ورودی، بافرهای در دسترس، نوع مسیرهای دسترسی و بسیاری از پارامترهای دیگر می باشد. اگر چه هزینه مدلها می تواند خیلی پیچیده باشد، یک جزء اصلی ارزیابی دقیق، دقت تخمین اندازه نتایج میانی می باشد. عملگر اتصال رتبه بندی همه ورودی هایش را استفاده نمی کند، اندازه ورودی واقعی وابسته به خود عملگر می باشد از این رو، کاردینالیتی ورودی وابسته به تعداد نتایج اتصال رتبه بندی درخواست شده از عملگر می باشد. بنابراین هزینه عملگر اتصال رتبه بندی به مواردی همچون تعداد نتایج مورد نیاز K وتعداد تاپلهایی از ورودی که اطلاعات کافی برای عملگر را برای گزارش K جواب مورد نیاز فراهم می کندو انتخاب پذیری عمل اتصال وابسته است.

در شکل ۶، هر دو طرح خصوصیت ترتیب یکسانی را فراهم می آورند طرح b ممکن است *pipeline* وابسته به زیر طرح های L و R باشد یا نباشد. در همه موارد هزینه دو طرح نیاز به مقایسه برای تصمیم گیری برای هرس کردن دارند. تخمین هزینه طرح (b) سخت است زیرا وابستگی قوی نسبت به تعداد نتایج رتبه بندی مورد نیاز دارند. بنابراین برای ارزیابی طرح (b) ما نیاز داریم گسترش مقدار k در *pipeline* را تخمین بزنیم. تخمین برای عمق ها به وسیله k و انتخاب پذیری عمل اتصال انجام می شود. توجه داشته باشید در طرح a هزینه طرح تقریباً مستقل از تعداد تاپلهای خروجی، خارج شده از طرح می باشد. زیرا آن یک طرح مرتب سازی *blocking* می باشد.



شکل۶: دو طرح شمارش

برای هر طرح اتصال رتبه بندی، هزینه طرح وابسته به k و انتخاب پذیری اتصال (S) می باشد. هرس کردن یک طرح اتصال رتبه بندی در مقابل طرح اتصال رتبه بندی دیگر وابسته به کاردینالیتی ورودی رابطه ها، هزینه روش اتصال، مسیرهای دسترسی و آمارهای در دسترس بر روی امتیازهای ورودی می باشد. گسترش مقدار k نهایی برای یک زیر طرح خاص وابسته به مکان زیر طرح، در طرح کامل، تخمین می باشد. k^*، مقداری از k است که در آن هزینه دو طرح با هم برابر است و کاردینالیتی خروجی طرح a، na نامیده می شود[7,8].

برای تخمین عمقی که بتوان K جواب بهتر را بدست آورد، می بایست DL و DR را تعیین کرد.

تخمین کمترین DL و DR: تاکنون هیچ فرضی بر روی توزیع رتبه بندی L و R نداشتیم. ما نشان دادیم که dL و dR وابسته به CL و CR بر حسب امتیازهای تاپلها در این عمق ها می باشند.

به صورت یک مدل احتمالی می توان مقدار کاردینالیتی ورودی را برای هر طبقه بدست آورد.

فرض شده است که امتیازهای رتبه بندی در L و R براساس توزیع یکنواخت باشند. اجازه دهید x میانگین کاهش *slab* برای L (یعنی میانگین تفاوت بین امتیازهای دو شی رتبه بندی شده در L می باشد) و y نیز میانگین کاهش *slab* برای R باشد. از این رو، مقدار مورد انتظار $\delta_L(C_L)$ برابر xC_L می باشد و مقدار مورد انتظار $\delta_R(C_R)$ برابر yC_R میباشد.مامیبایست $\delta = \delta_L(C_L)+\delta_R(C_R) = xC_L + yC_R$ را کمینه کنیم. برای این منظور می بایست $xC_L + yC_R$ را کمینه کنیم، در رابطه $SC_LC_R \geq K$ را داریم. کمینه سازی با قرار دادن

$C_L=\sqrt{(yK)/(xs)}$ و $C_R=\sqrt{(xK)/ys}$ بدست می آید. در این مورد $d_L=C_L+(y/x)C_R$ و $d_R=C_R+(x/y)c_L$ می باشند. برای سادگی فرض شده است، هر دو رابطه از یک توزیع یکنواخت می باشند یعنی $x=y$ است. در نتیجه $c_L=c_R=\sqrt{K/S}$ و $d_L=d_R=2\sqrt{K/S}$ می باشد.

در یک اتصال سلسله مراتبی خروجی یک عملگراتصال رتبه بندی به عنوان ورودی یک عملگر دیگر به کار می رود، توزیع امتیاز در سطح دوم اتصال بیشتر از توزیع یکنواخت نیست. فرض کنید تابع امتیازدهی مجموع دو امتیاز باشد. امتیاز اتصال رتبه بندی با دو توزیع یکنواخت از یک توزیع مثلثی پیروی می کند. نظر به اینکه ما در اتصال سلسله مراتبی به بالا می رویم. توزیع منجر به توزیع نرمال توسط قضیه حد مرکزی می شود.

x و y دو متغیر تصادفی مستقل از توزیع یکنواخت $[\circ,n]$ باشند. ما به این توزیع یکنواخت u_1 را نسبت می دهیم. ما مجموع j متغیر تصادفی مستقل از u_1 را با u_j نشان می دهیم. متغیر تصادفی $z=x+y$ دارای توزیع $u_۲$ می باشد که یک توزیع مثلثی بر روی $[\circ,2n]$ با یک قله در n می باشد. اگر ما n عنصر از توزیع $u_۲$ را انتخاب کنیم، امتیاز عنصر i ام $(i\le n/2)$، در یک ترتیب نزولی از امتیازها انتظار داریم. $2n-\sqrt{2in}$ باشد. به طور کلی، اگر ما m عنصر از u_j را در بازه ی $[\circ,jn]$ انتخاب کنیم سپس امتیاز عنصر i ام به صورت مقابل است:

$$Score_i=jn-(j!in/m)^{1/j}$$ رابطه۱

با استفاده از امتیازهای توزیع توصیف شده، ما مقادیر C_L و C_R که منجر به مقدار کمینه d_L و d_R برای طرح اتصال رتبه بندی کلی تخمین زنده می شود. فرض کنید خروجی L، خروجی اتصال رتبه بندی l رابطه رتبه بندی شده باشد و خروجی R، خروجی اتصال رتبه بندی r رابطه رتبه بندی شده باشد. k تعداد نتایج رتبه بندی خروجی مورد نیاز هر زیر طرح باشد و S انتخاب پذیری اتصال باشد. سپس کمیته کردن $\delta=\delta_L(CL)+\delta_R(CR)$ منجر به کمینه کردن $\delta=(l!C_Ln^{l-1})^{1/L}+(r!C_Rn^{r-1})^{1/r}$ می شود. ما $C_R=\dfrac{K}{SC_L}$ را جایگزین می کنیم و δ را نسبت به C_R کمینه می کنیم. پس از کمینه سازی نتایج زیر را داریم:

$$c_L^{r+l}=\frac{(r!)^lK^ln^{r-l}l^{r^l}}{s^l(l!)^rr^{r^l}},\quad c_R^{r+l}=\frac{(l!)^rK^rn^{l-r}r^{r^l}}{s^r(r!)^ll^{r^l}}$$ رابطه۲

$$d_L=c_L[1+r/l]^l,\quad d_R=c_R[1+l/r]^r$$ رابطه۳

توجه کنید که dR و dL حد بالای اکید و در بدترین حالت می باشد. برای تحلیل میانگین، فرض می کنیم که L از توزیع u_1 و R از توزیع u_r با داشتن n تاپل پیروری می کنند. اتصال L و R یک رابطه دیگری به نام G را با توزیع U_{L+r} و تاپلهای Sn^2 بوجود می آورد. با استفاده از رابطه ۲ و قرار دادن $j=l+r$ و $m=Sn^2$، امتیاز k امین عنصر از K جواب بهتر به صورت $Score_K=(l+r)n-\left((l+r)!kn^{(1+r-2)}/S\right)^{1/(l+S)}$ می‌باشد از این رو در L (یا R) به میزان تاپلی که در اتصال با R (یا L) بتواند $Score_K$ را فراهم آورد، احتیاج داریم. در نتیجه به طور میانگین d_L و d_R به صورت رابطه ۴ می باشد.

رابطه۴

$$d_L^{l+r}=\frac{((l+r)!)^lK^ln^{r-l}}{(l!)^{l+r}s^l},\quad d_R^{l+r}=\frac{((l+r)!)^rK^rn^{l-r}}{(r!)^{l+r}s^r}$$

۴- بهینه سازی تطبیقی تقاضا های تحت رتبه بندی در پایگاه داده‌های رابطه ای[9]

از آنجاییکه استراتژی های اجرایی بهینه ای که بوسیله بهینه سازهای ایستا تقاضا، اتخاذ می شود، ممکن است در هنگام رخ داد خطا ها یا تغییرات غیر منتظره در محیط های محاسباتی، بهینه گی خود را از دست دهند، در این مقاله چند استراتژی اجرای تطبیقی برای تقاضا های K جواب بهتر معرفی شده است که به این تغییرات غیر منتظره و خطاها پاسخ می دهند. این تکنیکهای بهینه سازی مجدد طرح اجرا را در زمان اجرا برای بهبود مستقیم، کارایی تقاضاهای در حال اجرا، تغییر می دهد.

۴-۱- بهینه سازی تقاضای محدود شده بهK [18]

برای پشتیبانی چنین تقاضایی در پایگاههای رابطه ای، روشی که پیشنهاد می شود این است که به جای عمل اتصال بر روی رابطه ها، عمل جستجو را بر روی ایندکس هایی که توسط تقاضا بر روی رابطه ها حاصل می شوند، داشته باشیم. برای درک این مفهوم دو جنبه، جستجوی وضعیت گسسته و تابع بهینه سازی پیوسته ترکیب گردید که برای ترکیب دو جنبه، چار چوب OPT^* ارائه داده شد. که بهینه سازی محدود شده به K را به عنوان یک جستجو A^* بر روی ترکیب فضای چند ایندکس کد گذاری می کند، که به طور کارا یک تابع هدف g و اندازه بازیابی K را مشخص می کند. تابع هدف g شامل عبارت محدود کننده منطقی B و عبارت بهینه سازی عددی G می باشد. به طور مفهومی آن یک تقاضای بهینه سازی محدود شده K بر روی پایگاه داده D می باشد که تابع هدف g را بر روی دامنه تعریف شده بوسیله D، بهینه می کند. و K تاپل بهتر که $t\in D$ را پیدا می کند به طوریکه g(t) ماکزیمم شود[18].

در تقاضاهای رتبه بندی دو نوع محدودیت وجود دارد: شرط های منطقی(B) و عبارت رتبه بندی(O)، بهینه سازهای سنتی، بهینه سازی آگاه از O,B را با هم ندارند آنها اغلب بهینه سازی بر اساس ایندو عبارت را مرحله به مرحله پردازش می کنند، یک جستجو ی مجتمع کلی در فضای تابع g وجود ندارد. برخی از الگوریتم های

موجود برای حل مسئله بهینه سازی، بر روی این فرض استناد می کنند که تابع g یکنواخت است. ایده اصلی در این مقاله این است که یک تقاضا بر اساس معانی واقعی تقاضا پردازش شود. برای تفهیم این مفهوم، دو جنبه در نظر گرفته شده است: اولا از جنبه استفاده از ایندکس ها، برای جستجوی تاپلهای ماکزیمم کننده بر روی گره های ایندکس وضعیتهای گسسته ایجاد شده است و از اینرو جنبه جسستجوی وضعیت گسسته مورد بحث می باشد. ثانیا از دید ماکزیمم کردن تابع هدف، برای بهینه کردن g و از اینرو جنبه بهینه سازی تابع به طور پیوسته وجود دارد. یک راه حل مناسب، یکپارچگی در این دو پارامتر است. نکته مهم OPT^* تبدیل صحیح مشکل بهینه سازی تقاضا به جستجو بر روی ایندکس ها می باشد. از طرف دیگر، OPT^* مطابق جنبه جستجوی وضعیتها ساخته شده است. OPT^* مطابق بهینه سازی تابعی برای اندازه گیری چشم انداز، مرتب سازی مجدد را انجام می دهد. همچنین فضای وضعیت با تابع اکتشافی مناسب و وضعیت های اولیه، پیکربندی انجام می شود. OPT^* همچنین به طور مفهومی با چندین چهارچوب پیشنهادی قبلی مثل KNNاتصالهای فضایی در تقاضاهای فضایی و TA در تقاضای K جواب بهتر می تواند یکی شود.

یک بهینه سازی تقاضا محدود شده به K مثل Q بر روی خصوصیات تقاضا A_i که ۱ از یک تا m است و اتصال n رابطه D_1 تا D_n تعریف می شود بطوریکه $Dom(A_i)$ دامنه مقادیر خصوصیات A_i می باشد، $Rel(Ai)$ مشخص می کند که خصو صیت A_i به رابطه ای مانند D_j تعلق دارد و $D=D_1\times D_2\times\ldots\times D_n$ می باشد. بهینه سازی تقاضا محدود شده به K برای یک تقاضایی مثل Q، یک دو تایی به صورت $<g , K>$ می باشد که: تابع هدف g: $(Dom(A_1)\times\ldots\times Dom(A_m))\rightarrow R^+$ است که یک تاپل t با m صفت را به یک امتیاز عددی مثبت نگاشت می نماید. که شامل عبارت بهینه سازی $O=Dom(A1)\times\ldots\times Dom(A_m)\rightarrow R^+$ و عبارت منطقی $B(Dom(A_1)\times\ldots\times Dom(A_m))\rightarrow\{0,1\}$ می باشد. K، تعداد جواب بهتر در یک تقاضا می باشد.

هدف جستجو برروی تاپلها در D است که تابع هدف g با کمترین هزینه ممکن، بیشینه شود. برای نیل به هدف اولین احتیاج، جستجو کردن بر روی تاپلهای پایگاه داده در D می باشد که برای این منظور، استفاده از متدهای دسترسی پیشنهاد می شود. مثل پیمایش جدول، ایندکس، اگر از پیمایش جدول استفاده کنیم احتیاج به جستجوی جامع داریم و بنابراین قابل بهینه سازی نیست. استفاده از ایندکس ها سریعتر می باشد. احتیاج دوم، کمینه کردن هزینه برای الگوریتم جستجو است که باعث حداکثر کردن مقدار تابع g می شود

۴-۲- استنتاج فضای وضعیت ایندکس:

برای استنتاج فضای وضعیت ایندکس می بایست نگاشت فضا صورت گیرد و وضعیت هدف مشخص باشد.

به منظور نگاشت فضا ابتدا احتیاج داریم یک گراف اتصال را مانند یک نگاشت فضایی که می بایست برای پاسخ گویی تقاضا جستجو شود بوجود آوریم. یک گراف ترکیبی به عنوان یک ضرب دکارتی گرافهای جستجوی منفرد می باشد که همه مسیرها برای دستیابی به تاپل ها را توصیف می کند. مخصوصا برای بوجود آوردن فضای وضعیت از یک مجموعه از گراف ایندکس $I=<I_1,\ldots,I_m>$ احتیاج داریم یک گراف $I=(V,E)$ تعریف شده است که V مجموعه وضعیتها و E مجموعه انتقال ها بین وضعیتها می باشد که این نگاشت، بر اساس گره های مختلف، وضعیتهای متفاوتی دارد.

وضعیتها در یک گراف جستجو مکانهای مقادیر را در پیمانه های مختلف از بزرگ تا کوچک نشان می دهد سر انجام به تاپلها در پایگاه داده دسترسی پیدا می کنیم.

وضعیت ناحیه: در حالیکه یک گره ایندکس منفرد یک بازده از مقادیر را برای یک خصوصیت تعریف می کند، ترکیبی از چندین گره ایندکس یک ناحیه را تعریف می کند. به طور کلی هر ترکیب گره‌های ایندکس یک وضعیت ناحیه را بوجود می آورند. شبیه به گرهها در ایندکس، وضعیتهای نواحی درگراف ترکیب می توانند به وضعیتهای برگ و وضعیتهای داخلی گروه بندی شود. وضعیتی برگ می باشد که همه گره های تشکیل دهنده آن, گره های برگ در ایندکسهای متناظر باشند. در غیر این صورت آن یک وضعیت داخلی می باشد. به همین صورت، برای مولفه تاپل T در یک گراف ایندکس، هر تاپل همچنین یک وضعیت تاپل را مشخص می کند اگر همه ایندکسهای متناظر، به آن تاپل اشاره کنند.

انتقالها بین وضعیتها برروی لبه ها در گرافهای ایندکس تعریف می شوند. انواع مختلف انتقالها منجر به رفتارهای مختلف پیمایش فضا می شود مثل: پیمایشهای سلسله مرتبی و میان برگی. برای هر دو وضعیت $v,u\in V$ اگر $next(u)$ باشد در این صورت یک انتقال $transition(u,v)$ وجود دارد. برای وضعیت داخلی R وضعیت های قابل دسترس بوسیله اتصال های پدر-فرزند در گره های ایندکس R بوجود می آیند، چنین انشعابی، باعث بوجود آمدن جستجوی بالا-پایین در گراف می گردد که از یک ناحیه ریشه شروع می شود و به تدریج به پاسخ های تقاضا تمرکز می کند. گره های برگ، به هردوی گره های برگ وتاپل ها دسترسی دارند برای بوجود آوردن وضعیتهای همسایه در گره های برگ، اشاره گرهای بردار به گره های برگ مجاور اشاره می کنند. وضعیتهای برگ همسایه بوسیله ترکیب هر گره برگ جدید با گره های برگ از ایندکس های دیگر بوجود می آیند. چنین گسترشی انتقالی، جستجو پایین- بالا را بوجود می آورد که از یک وضعیت برگ شروع شده و به تدریج برای رسیدن به جواب ها گسترش می یابد.

۴-۳- وضعیت هدف:

در میان وضعیتها در فضا، وضعیتهای هدف واقعی برابر با تاپلها یی هستند که تابع هدف g را ماکزیمم می کنند، بنابراین مسیراصلی پیدا

کردن وضعیتهای تاپل بهینه، بیشترین امتیاز g می باشد. برای بکار بردن A^* برای بهینه سازی محدود شده به K احتیاج داریم، مسئله را به پیدا کردن وضعیتهای تاپل بهینه، برای یافتن مسیر بهینه برای دستیابی به وضعیت هدف، تبدیل کنیم. اول از همه برای اینکه یک وضعیت تاپل را به یک مسیری که وضعیت را به سوی یک هدف هدایت می کند، تبدیل کنیم احتیاج داریم یک شبه هدف t^* را به عنوان وضعیت هدف اضافه کنیم و هر تاپل را به این شبه هدف t^* متصل نمائیم. با این شبه هدف t^*، هر مسیر t را بر اساس یک وضعیت تاپل یکتا بدست می آورد زیرا بین وضعیتهای تاپل انتقالی وجود ندارد. علاوه بر این، برای تبدیل وضعیت تاپل بهینه به کوهتاترین مسیر منتهی شده به این وضعیت، احتیاج داریم فاصله‌های مناسبی را به لبه های بین وضعیت ها اختصاص دهیم، برای اینکه کمترین فاصله را به بیشترین امتیاز برگردانیم، از اینرو می بایست فاصله بین وضعیت تاپل t را به عنوان قرینه امتیاز g (امتیاز وضعیت تاپل) در نظر بگیریم یعنی $d(t,t^*) = -g(t)$ علاوه بر این ما احتیاج داریم مطمئن شویم که مسیری که به تاپل بهینه منتهی می شود، کوتاهترین مسیر بر روی همه مسافت طی شده می باشد برای این منظور برای همه لبه های داخلی $i = 1,...,n-1, d(vi, vi+1) = 0$ می باشد[18].

هنگامیکه ایندکس ها یک نقشه ایستا از فضای وضعیت را استنتاج می کنند، علاوه بر احتیاج یک چشم انداز بر روی نقشه، می بایست جستجو به طور کارآیی به سمت هدف راهنمایی شود. به طور مفهومی معیارهای چشم انداز، کیفیت نسبی وضعیتعهای مختلف را نسبت به تابع هدف بدست می آورند و از اینرو پیکربندی پویا تقاضا را بر روی فضای ایستا ارائه می دهند. برای کد گذاری تقاضای بهینه سازی محدود شده به K توسط جستجوی A^*، پیکربندی شامل دو جنبه می باشد: اولا تعریف پارامتر های اکتشافی مناسب بر پایه تابع هدف برای راهنمایی جستجو و پیکربندی فضای وضعیت نسبت به پارامترهای اکتشافی می باشد. ثانیا یک مجموعه از وضعیتهای اولیه قطعی توسط تابع هدف برای شروع جستجو به طور مناسب، می‌بایست شناسایی شود. برای این منظور از تابع پیوسته بهینه سازی استفاده شده است. ابتدا پارامترهای اکتشافی برای راهنمایی جستجو A^* تعریف می شود. به تابع بهینه سازی برای تخمین حد بالای امتیاز وضعیتهای تاپل قابل دستیابی از وضعیت جاری احتیاج داریم. ثانیا برای شناسایی یک مجموعه از وضعیتهای اولیه می بایست نقطه های بهینه محلی تابع هدف را بدست آوریم. برای بدست آوردن نقطه های بهینه محلی، سه گروه تکنیک وجود دارد. گروه اول، متدهای تحلیلی، مشتقهای تابع هدف (یا برای توابع چند متغیره شیب تابع هدف) را بدست می آورند و یک مقدار ماکزیمم یا مینیمم را هنگامیکه مشتقها برابر صفر است بدست می آورند. گروه دوم: متدهای بر پایه جستجو مثل hill climbing می باشد که مقدارextremum را بوسیله تقریب زدن آنها به طور تدریجی در فضای مقادیر ساخت یافته منظم بدست می آورند. گروه سوم: متدها بر پایه قالب بطوریکه نزدیک به گره های extremum می باشند. برای بکاربردن جستجوی آگاه، الگوریتم آن A^* به یک تخمین اکتشافی برای کم هزینه ترین مسیر برای دست یابی به هدف احتیاج دارد. برای تضمین کامل بودن الگوریتم A^* پارامتر اکتشافی می‌بایست قابل قبول و نزولی باشد. قابل قبول بودن به این معنی است در یک حالت معین r ، پارامتر اکتشافی می بایست به طور بهینه امتیازهای تاپلهای قابل دستیابی از r را ارزیابی کند. برای ساختن تخمین دقیق پارامتر اکتشافی می بایست، حد بالای اکید را برای وضعیت در نظر گرفت. خصوصیت نزولی: تخمین اکتشافی هرگز نباید در طول مسیر بازدید شده توسط A^* از وضعیت اولیه تا وضعیت هدف کاهش پیدا کند در مسئله ما آن به این معنی است که امتیاز ارزیابی شده تابع هرگز نباید افزایش بیابد و از این رو آن را خصوصیت نزولی می گوییم. هدف ما پشتیبانی همه مسیر های در دسترس می باشد، اما در ایندکس ها علاوه بر انتقال های پدر و فرزند انتقال برادر نیز وجود دارد لینکهای برادر وضعیت های برگ را به طور کامل در فضای وضعیت به هم متصل می کند.

خصوصیات قابل قبول و نزولی هنگامیکه در جستجو بین وضعیت‌های برگ لبه هایی با افزایش مقدار داریم نقض می شوند بنابراین ما همه black link ها را حذف کنیم.black link ها لبه هایی در بین وضعیتهای برگ که باعث نقض خصوصیات قابل قبول و نزولی می شوند. توسط یک تابع می بایست، مجموعه ای از وضعیت ها یافت که به وسیله آن ها می توان به هر وضعیت تاپل در فضا دسترسی داشت. شاید شروع از وضعیت برگ مناسب تر باشد بنابراین توسط تابع شروع، نقطه یا نقاط نزدیک تر به هدف، پیدا می شود، بوسیله شروع جستجو با وضعیتهای بهینه محلی، جستجو A^* به طور اتوماتیک از blackلinkها چشم پوشی می کند زیرا جستجو همیشه بهترین وضعیت در دسترس در وضعیت جاری را برای پردازش انتخاب می‌نماید.

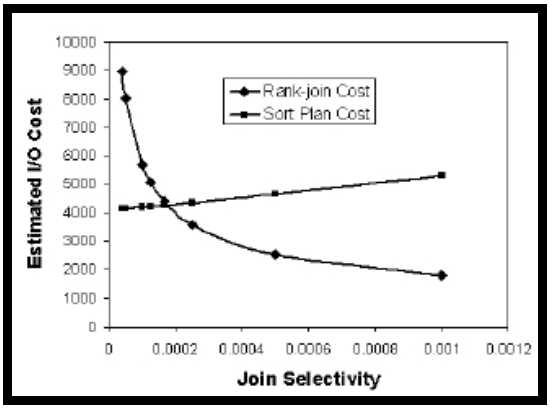
۴-۴- الگوریتم OPT^*:

الگوریتم دو صف اولویت رانگهداری می کند. صف ToDoQ وضعیتهای ناحیه ای که بعدا رسیدگی می شوند و TupLeQ که دنباله ای از وضعیتهای تاپل K جواب بهتر را درمیان تمامی وضعیتهای بازدیده شده تا به حال نگهداری مینماید. علاوه براین الگوریتم جدول پراکندگی Have_DonQ را برای همه وضعیتهای برگ باز دید شده، نگهداری می نماید. TODOQ با یک وضعیتهای اولیه توسط تابع OPT_POINT مقدار دهی اولیه می شودکه مجموعه نقاط بهینه محلی می باشند.برای توضیح دقیقتر این الگوریتم به [18] مراجعه کنید.

۵- نتیجه گیری

. عملگر اتصال رتبه بندی ممکن است همیشه بهترین روش برای فراهم کردن نتایج رتبه بندی مورد نیاز نباشد. در حقیقت ، وابسته به بسیاری پارامترها (برای مثال انتخاب پذیری اتصال ، مسیرهای قابل

دسترسی و اندازه حافظه) می باشد که یک طرح اتصال سپس مرتب سازی ممکن است یک روش بهتری برای فرآهم کردن نتایج رتبه بندی باشد. باید بسته به شرایط روش بهتر انتخاب شود شکل زیر ارزیابی هزینه I/O را برای مقادیر مختلف انتخاب پذیری اتصال برای دو طرح (طرح مرتب سازی و طرح اتصال رتبه بندی) نشان می دهد.



شکل۴: ارزیابی هزینه I/O دو طرح مرتب سازی و اتصال رتبه بندی

همانطور که مشاهده کردیم، هر یک از روشهای سنتی و ارائه شده برای یکسری مقادیر K مناسب تر می باشند. بر خلاف عملگرهای تقاضای سنتی، تخمین هزینه عملگرهای اتصال رتبه بندی سخت می باشد زیرا آنها خروج زودرس دارند . هر وقت که نتایج top k گزارش شود اجرا بدون استفاده از همه ورودی ها متوقف می شود. در این روش مفهوم خصوصیات مطلوب در بهینه سازی تقاضا را را که شامل عبارات رتبه بندی مطلوب است توسعه داده شده است و سپس بر اساس هزینه طرحهای نا مطلوب هرس می شود همچنین بوسیله مدلی احتمالی کاردینالیتی ورودی تخمین زده می شود. در واقع یک مجتمع سازی بین الگوریتم های رتبه بندی تجمعی به سبک بازیابی اطلاعات و پردازش تقاضاهای رابطه ای رایج اعمال شده است.

۶- منابع:

[1] Bennet, Kristin, “A Genetic Algorithm for Database Query Optimization”, Technical Report, University of Wisconsin,1997.
[2] Bernstein, P. A., N. Goodman, “Query Processing in a System for Distributed Database ”, ACM Transactions Database System, 6(4): 602-625, December 1981.
[3] Bitton, D., H. Boral, D. J. Dewitt, W. K. Wilkinson, “Parallel Algorithms for the Execution of Relational Database Operations”, ACM Transactions Database System, 8(3): 324-353, Sept. 1983.
[4] Chen, Zhiyuan, “Query Optimization in Compressed Database Systems”, In Proceedings of the ACM SIGMOD, May 2001.
[5] Connolly, Thomas, “Database Systems”, 3rd ed., Addison-Wesley, USA, 2002.
[6] Date, C.J., “An Introduction to Database Systems”, 7th ed., Addison-Wesley, USA, 2000.
[7] Graefe, G., D. Dewitt. “The EXODUS optimizer generator”, In Proceedings of the ACM SIGMOD Conference on Management of Data, 160-172, May 1987.
[8] Graefe, G., W. J. Mckenna, “The volcano optimizer generator: Extensibility and efficient search”, In Proceedings of the 9th International Conference on Data Engineering, 209-218, April 1993.
[9] Ilyas, I. F., W. G. Aref, A. K. Elmagarmid, H. G. Elmongui, R. Shah and J. S.Vitter, “Adaptive Rank-aware Query Optimization in Relational Databases”, ACM Transactions on Database Systems, 2006.
[10] Ilyas, I. F., R. Shah, W. G. Aref, J. S. Vitter, and A. K. Elmagarmid, “Rank-aware query optimization”, In Proceedings ACM SIGMOD International Conference on Management of Data, 203–214, 2004.
[11] Ilyas, I. F., W. G. Aref, A. K. Elmagarmid, “Supporting Top-k Join Queries in Relational Databases”, In Proceedings 29th International Conference on Very Large Data Bases, 754–765, 2003.
[12] Ilyas, I. F., C. Li, K. Chang and S. Song, “Ranksql: Query algebra and optimization for relational top-k queries”, In Proceedings ACM SIGMOD International Conference on Management of Data, 2005.
[13]Ioannidis, Y. E., Y. C. Kang, “Randomized algorithms for optimizing large join queries”, In Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, 312-321, May 1990.
[14] Jarke, Matthlas, Jijrgen Koch, “Query Optimization in Database Systems”, ACM Computing Surveys, 16(2), June 1984.
[15] Kossmann Donnald, Konrad Storcker, “Iterative Dynamic Programming: A New Class of Query Optimization Algorithms”, ACM Transactions on Database Systems, 25(1): 43–82, March 2000.
[16] Liu, Jie, Liang Feng, and Yunpeng Xing, “A Pruning-based Approach for Supporting Top-K Join Queries”, ACM Transactions on Database Systems, Edinburgh, Scotland, May 2006.
[17] Silberschatz, Henry F., “Database System Concepts”, 3th ed., WCB/Mc Graw Hill, USA, 1999.
[18] Zhang, Z., S.Hwang, K. ChenChuan, Ch.M. Wang, Ch. A. Lang, Y. Chang, “Boolean + Ranking: Querying a Database by KConstrained Optimization”, In Proceedings of the ACM SIGMOD, Chicago, Illinois, USA, June 2006.
[۱۹]. روحانی رانکوهی، سید محمد تقی، ”سیستمهای مدیریت پایگـاه داده(مفاهیم و تکنیکها)“ .