۱ طرح کوئری

DBMS یک کد SQL را به یک طرح کوئری تبدیل می‌کند. عملگرهای طرح کوئری به صورت یک درخت چیده می‌شوند. داده‌ها از برگ‌های این درخت به سمت ریشه این درخت جرکت می‌کنند. خروجی ریشه در درخت نتیجه کوئری است. معمولاً عملگرها باینری (۱-۲ فرزند دارند) هستند. یک درخت پرس و جو می‌تواند به چندین روش تشکیل و اجرا شود.

۲ مدل‌های پردازشی

مدل پردازش DBMS مشخص می‌کند که سیستم چگونه یک برنامه پرس و جو را اجرا کند. این مدل چیزهایی مانند جهت ارزیابی کوئری و نوع داده‌هایی که بین عملگرها در طول مسیر منتقل می‌شود را تعیین می‌کند. مدل‌های پردازش مختلفی وجود دارند که برای بارهای کاری مختلف دارای مزایا و معایب متفاوتی نسبت به هم هستند.

این مدل‌ها همچنین می‌توانند به گونه‌ای پیاده‌سازی شوند که عملگرها را از بالا به پایین یا از پایین به بالا فراخوانی کنند. اگرچه رویکرد بالا به پایین بسیار رایج‌تر است، اما رویکرد پایین به بالا می‌تواند کنترل بهتری بر حافظه‌های نهان (کش) و ثبات‌ها در خطوط لوله پردازنده فراهم کند.

سه مدل اجرایی که در ادامه مورد بررسی قرار می‌گیرند عبارتند از:

- مدل Iterator
- مدل Materialization
- مدل Vectorized-Batch

۱.۲ مدل Iterator

این مدل که به عنوان مدل Pipeline شناخته می‌شود، رایج‌ترین مدل پردازش است و تقریباً توسط هر DBMS مبتنی بر ردیف استفاده می‌شود.

مدل Iterator با پیاده‌سازی یک تابع Next برای هر عملگر در پایگاه داده کار می‌کند. هر Node در برنامه پرس‌وجو تا زمانی که به برگ می‌رسد، Next را بر روی فرزندان خود فراخوانی می‌کند که شروع به ارسال ردیف‌ها به گره‌های پدر خود برای پردازش می‌کنند. سپس هر ردیف تا حد امکان به سمت بالا در برنامه پردازش می‌شود قبل از اینکه ردیف بعدی بازیابی شود. این در سیستم‌های مبتنی بر دیسک مفید است زیرا به ما اجازه می‌دهد هر ردیف را در حافظه به طور کامل استفاده کنیم قبل از این که به ردیف یا صفحه بعدی دسترسی پیدا کند.

عملگرهای برنامه پرس‌وجو در یک مدل Iterator بسیار قابل ترکیب و آسان برای استدلال هستند زیرا هر عملگر می‌تواند به طور مستقل از عملگرهای پدر یا پسر خود در درخت برنامه پرس‌وجو پیاده‌سازی شود، به شرطی که یک تابع Next را به صورت زیر پیاده‌سازی کند:

۱. در هر فراخوانی Next ، عملگر یا یک ردیف را برگرداند یا یک پوینتر null .

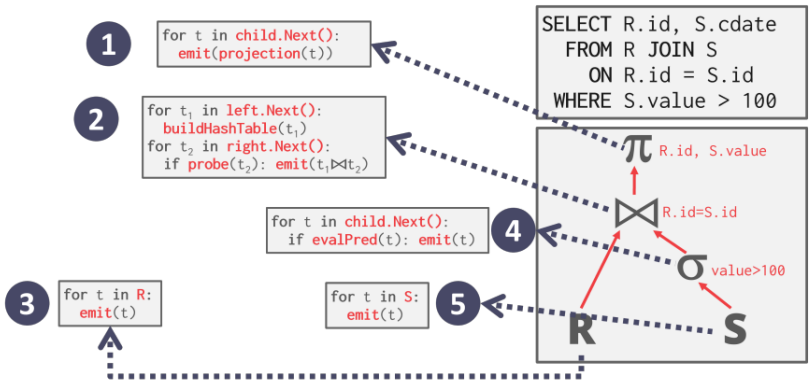
۲. عملگر باید یک حلقه را پیاده‌سازی کند که Next را بر روی فرزندان خود Call می‌کند تا ردیف‌های آنها را بازیابی کند و سپس آنها را پردازش کند.
به این ترتیب، فراخوانی Next بر روی یک والد، Next را بر روی فرزندان خود فراخوانی می‌کند و در پاسخ گره فرزند ردیف بعدی را که والد باید پردازش کند، برمی‌گرداند.

مدل Iterator اجازه می‌دهد تا جایی که ممکن است، DBMS یک ردیف را از طریق عملگرهای مختلف پردازش کند قبل از اینکه ردیف بعدی را بازیابی کند.

سری کارهایی که برای یک ردیف در برنامه پرس‌وجو انجام می‌شود، یک خط لوله نامیده می‌شود.

برخی عملگرها تا زمانی که فرزندان تمام ردیف‌های خود را ارسال کنند، مسدود می‌شوند. نمونه‌هایی از این عملگرها شامل joins ، زیردرخواست‌ها و مرتب‌سازی‌ها هستند. این عملگرها به عنوان شکننده‌های خط لوله شناخته می‌شوند.

دستور LIMIT به راحتی با این روش کار می‌کند، زیرا یک عملگر می‌تواند فراخوانی Next بر روی فرزندان خود را زمانی که همه ردیف‌های مورد نیاز خود را دارد متوقف کند.



در اینجا شبه‌کد برای توابع Next برای هر یک از عملگرها آورده شده است. این توابع Next به طور اساسی حلقه‌های for هستند که بر روی خروجی عملگر فرزند خود تکرار می‌شوند. به عنوان مثال،ریشه Next را بر روی فرزند خود، عملگر join فراخوانی می‌کند. پس از پردازش همه ردیف‌ها، یک پوینتر null (یا پوینتر دیگری) ارسال می‌شود که به گره‌های والد اطلاع می‌دهد که به مرحله بعد بروند.

۲.۲ مدل Materialization

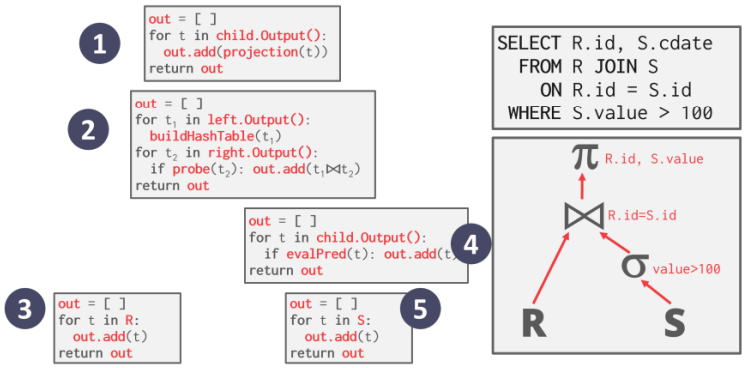
مدل ماده‌سازی یک تخصصی از مدل تکرارگر است که در آن هر عملگر ورودی خود را به طور کامل پردازش کرده و سپس خروجی خود را به طور کامل منتشر می‌کند. به جای داشتن یک تابع Next که یک تاپل را برمی‌گرداند، هر عملگر همه تاپل‌های خود را هر بار که فراخوانی می‌شود، برمی‌گرداند. برای جلوگیری از اسکن تاپل‌های بیش از حد، DBMS می‌تواند اطلاعاتی درباره تعداد تاپل‌های مورد نیاز به عملگرهای بعدی منتقل کند. خروجی می‌تواند یا یک تاپل کامل (NSM) یا یک زیرمجموعه‌ای از ستون‌ها (DSM) باشد.

هر عملگر برنامه پرس‌وجو یک تابع Output را پیاده‌سازی می‌کند که:

۱. عملگر تمامی ردیف‌ها از فرزندان خود را یکجا پردازش می‌کند.

۲. نتیجه بازگشتی از این تابع تمامی ردیف‌هایی است که عملگر هرگز ارسال خواهد کرد. زمانی که عملگر اجرای خود را به پایان می‌رساند، DBMS هرگز نیازی ندارد که برای بازیابی داده‌های بیشتر به آن بازگردد.

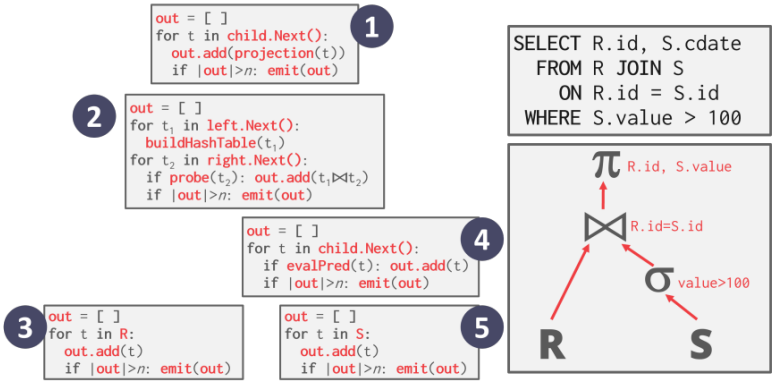
این رویکرد برای بارهای کاری OLTP بهتر است زیرا پرس‌وجوها معمولاً تنها به تعداد کمی از ردیف‌ها در هر زمان دسترسی دارند. بنابراین، فراخوانی‌های کمتری برای بازیابی ردیف‌ها وجود دارد. مدل Materialization برای پرس‌وجوهای OLAP با نتایج میانی بزرگ مناسب نیست زیرا ممکن است DBMS نیاز داشته باشد که آن نتایج را بین عملگرها به دیسک منتقل کند.



در مدل Materialization ، اجرای پرس‌وجو از گره ریشه شروع می‌شود و تابع Output فرزند فراخوانی می‌شود که عملگرهای زیرین را فراخوانی می‌کند و تمامی ردیف‌ها را به سمت بالا باز می‌گرداند. در شکل فوق شبه‌کد برای چگونگی اجرای این فرآیند برای هر عملگر آمده است.

۳.۲ مدل Vectorization

مانند مدل iterator ، هر عملگر در مدل برداری یک تابع Next را پیاده‌سازی می‌کند. با این حال، هر عملگر یک دسته بردار از داده‌ها را به جای یک تاپل منتشر می‌کند. پیاده‌سازی حلقه داخلی عملگر برای پردازش دسته‌های داده به جای یک مورد در هر زمان بهینه شده است. اندازه دسته می‌تواند بر اساس سخت‌افزار یا ویژگی‌های کوئری متفاوت باشد.



مدل بردارسازی بسیار شبیه به مدل Iterator است، به جز اینکه در هر عملگر، یک بافر خروجی با اندازه انتشار مورد نظر مقایسه می‌شود. اگر بافر بزرگتر باشد، یک دسته از ردیف‌ها به بالا ارسال می‌شود. این مدل با پردازش دسته‌ای از ردیف‌ها به جای پردازش تک‌تک ردیف‌ها در هر زمان، بهره‌وری بیشتری را به دست می‌آورد.

مدل بردارسازی به دلیل اینکه تعداد فراخوانی‌های تابع Next کمتر است، برای پرس‌وجوهای OLAP که نیاز به اسکن تعداد زیادی از ردیف‌ها دارند ایده‌آل است. این مدل به عملگرها اجازه می‌دهد تا به راحتی از دستورالعمل‌های برداری (SIMD) برای پردازش دسته‌ای از ردیف‌ها استفاده کنند.

۴.۲ جهت پردازش

- رویکرد ۱: از بالا به پایین
  - با ریشه شروع کنید و داده‌ها را از فرزندان به والدین "کشیده" کنید.
  - تاپل‌ها همیشه با فراخوانی توابع منتقل می‌شوند.
- رویکرد ۲: از پایین به بالا
  - با برگ شروع کنید و داده‌ها را از فرزندان به والدین "هل" دهید.
  - این رویکرد امکان کنترل دقیق‌تر حافظه‌های کش و رجیسترها در خطوط لوله عملگرها را فراهم می‌کند.

۳ روش‌های دسترسی

به این که DBMS چگونه داده‌های ذخیره شده در یک جدول دسترسی پیدا می‌کند، روش دسترسی میگویند.

به طور کلی دو روش برای مدل‌های دسترسی وجود دارد: داده‌ها یا از یک جدول خوانده می‌شوند یا با یک اسکن ترتیبی از یک شاخص.

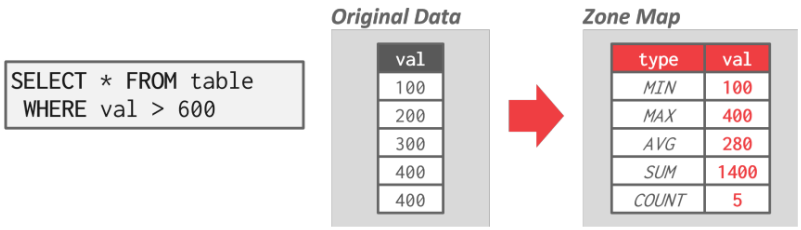
۱.۳ اسکن ترتیبی

عملگر اسکن ترتیبی روی هر صفحه در جدول تکرار می‌کند و آن را از حافظه پنهان بافر بازیابی می‌کند. هنگامی که اسکن روی تمام تاپل‌های هر صفحه تکرار می‌کند، گزاره را ارزیابی می‌کند تا تصمیم بگیرد که آیا تاپل را به عملگر بعدی منتشر کند یا نه.

اسکن ترتیبی جدول تقریباً همیشه ناکارآمدترین روش برای اجرای یک پرس و جو توسط یک DBMS است.

تعدادی بهینه‌سازی وجود دارد که به افزایش سرعت اسکن‌های ترتیبی کمک می‌کنند:

- پیش‌دریافت: صفحات بعدی را از قبل دریافت کنید تا DBMS مجبور نباشد هنگام دسترسی به هر صفحه بر روی I/O ذخیره‌سازی متوقف شود.
- دور زدن حافظه پنهان بافر: عملگر اسکن صفحات دریافتی از دیسک را در حافظه محلی خود ذخیره می‌کند تا از سیل ترتیبی جلوگیری کند.
- موازی‌سازی: اسکن را با استفاده از چندین نخ / فرآیند به صورت موازی اجرا کنید.
- ماده‌سازی دیرهنگام: DBSM های از نوع DSM می‌توانند به تأخیر انداختن به هم دوختن توپل‌ها تا بخش‌های بالای طرح پرس و جو را انجام دهند. این اجازه می‌دهد تا هر عملگر حداقل اطلاعات مورد نیاز را به عملگر بعدی منتقل کند.
- خوشه‌بندی پشته: توپل‌ها با استفاده از یک شاخص خوشه‌بندی در صفحات پشته ذخیره می‌شوند.
- پرس و جوهای تقریبی: پرس و جوها را بر روی یک زیرمجموعه نمونه‌برداری شده از کل جدول اجرا کنید تا نتایج تقریبی تولید کنید. این به طور معمول برای محاسبه تجمیع‌ها در یک سناریو انجام می‌شود که اجازه خطای کم را می‌دهد تا یک پاسخ تقریباً دقیق تولید شود.
- نقشه‌بندی منطقه‌ای: پیش‌محاسبه تجمیع‌ها برای هر ویژگی توپل در یک صفحه. DBMS سپس می‌تواند با بررسی نقشه منطقه‌ای خود ابتدا تصمیم بگیرد که آیا نیاز به دسترسی به یک صفحه دارد یا نه.



در مثال بالا، کوئری Select از Zone-Map متوجه می‌شود که حداکثر مقدار در داده‌های اصلی تنها ۴۰۰ است. سپس، به جای آن که مجبور باشد هر زوج مرتب در صفحه را بررسی کند، پرس و جو می‌تواند از دسترسی به صفحه به‌طور کامل اجتناب کند زیرا هیچ یک از مقادیر بزرگتر از ۶۰۰ نخواهند بود.

۲.۳ اسکن شاخص

در یک اسکن شاخص، DBMS یک شاخص را انتخاب می‌کند تا توپل‌هایی که یک پرس و جو نیاز دارد را پیدا کند.

```
SELECT * FROM students
 WHERE age < 30
   AND dept = 'CS'
   AND country = 'US'
```

Scenario #1

There are 99 people under the age of 30 but only 2 people in the CS department.

Scenario #2

There are 99 people in the CS department but only 2 people under the age of 30.

جدولی با ۱۰۰ زوج‌مرتب و دو ایندکس را در نظر بگیرید: سن و دپارتمان. در سناریوی اول، بهتر است از ایندکس دپارتمان در اسکن استفاده شود زیرا فقط دو زوج‌مرتب برای مطابقت دارد. انتخاب ایندکس سن خیلی بهتر از یک اسکن ترتیبی ساده نخواهد بود. در سناریوی دوم، ایندکس سن اسکن‌های غیرضروری بیشتری را حذف می‌کند و انتخاب بهینه است.

انتخاب شاخص توسط DBMS شامل عوامل بسیاری است:

- چه ویژگی‌هایی در شاخص وجود دارند.
- چه ویژگی‌هایی پرس و جو را اشاره می‌کند.
- دامنه‌های مقداری ویژگی.
- ترکیب گزاره.
- اینکه آیا شاخص دارای کلیدهای یکتا یا غیر یکتا است.

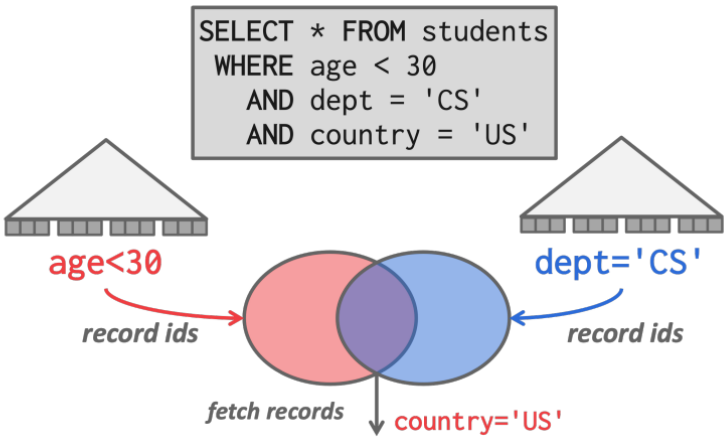
DBMS های پیشرفته‌تر از اسکن‌های چند شاخصی پشتیبانی می‌کنند. هنگام استفاده از چندین شاخص برای یک پرس و جو، DBMS مجموعه‌های آیدی های رکوردها را با استفاده از هر شاخص مطابقت داده شده محاسبه می‌کند، این مجموعه‌ها را بر اساس گزاره‌های پرس و جو ترکیب می‌کند، و رکوردها را بازیابی کرده و هر گزاره باقی‌مانده را اعمال می‌کند. DBMS می‌تواند از بیت‌مپ‌ها، جداول هش، یا فیلترهای بلوم برای محاسبه آیدی رکوردها از طریق تقاطع مجموعه‌ها استفاده کند.

۴ کوئری‌های تعدیلی

عملگرهایی که پایگاه داده را تغییر می‌دهند مسئول بررسی محدودیت‌ها و به‌روزرسانی شاخص‌ها هستند.

برای UPDATE/DELETE ، عملگرهای فرزند شناسه‌های رکورد را برای تاپل‌های هدف منتقل می‌کنند و باید تاپل‌های قبلاً دیده شده را ردیابی کنند. دو انتخاب برای نحوه مدیریت عملگرهای INSERT وجود دارد:

- انتخاب ۱: تاپل‌ها را داخل عملگر ماده‌سازی کنید.
- انتخاب ۲: عملگر هر تاپل انتقال یافته از عملگرهای فرزند را وارد می‌کند.



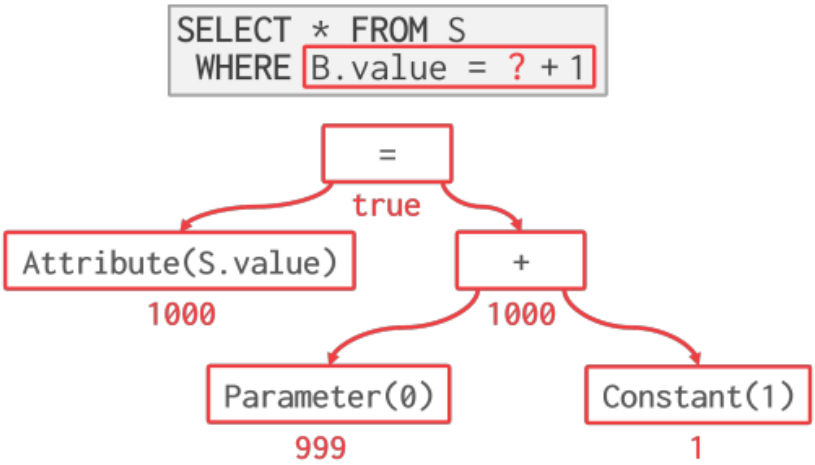
همان جدول در شکل قبل را در نظر بگیرید. با پشتیبانی از اسکن چند ایندکسی، ابتدا مجموعه‌های آیدی رکوردها که شرایط برای سن و دپارتمان را برآورده می‌کنند، به ترتیب با استفاده از ایندکس‌های مربوطه محاسبه می‌کنیم. سپس اشتراک دو مجموعه را محاسبه می‌کنیم، رکوردهای مربوطه را واکشی کرده و شرط باقی‌مانده `country='US'` را اعمال می‌کنیم.

۱.۴ مشکل هالووین

مشکل هالووین یک تناقض است که در آن یک عملیات به‌روزرسانی مکان فیزیکی یک تاپل را تغییر می‌دهد و باعث می‌شود یک عملگر اسکن چندین بار به تاپل مراجعه کند. این می‌تواند در جداول خوشه‌بندی شده یا اسکن‌های شاخص رخ دهد. این پدیده توسط محققان IBM در روز هالووین در سال ۱۹۷۶ هنگام ساخت System R کشف شد. راه حل این مشکل این است که شناسه‌های رکورد اصلاح شده را برای هر کوئری پیگیری شود.

۵ ارزیابی عبارات

DBMS ها یک شرط WHERE را به عنوان یک درخت عبارت نمایش می‌دهد. گره‌ها در این درخت نوع‌های مختلف عبارت را نمایندگی می‌کنند.



برخی از نمونه‌های نوع عبارات که می‌توانند در گره‌های درخت ذخیره شوند:

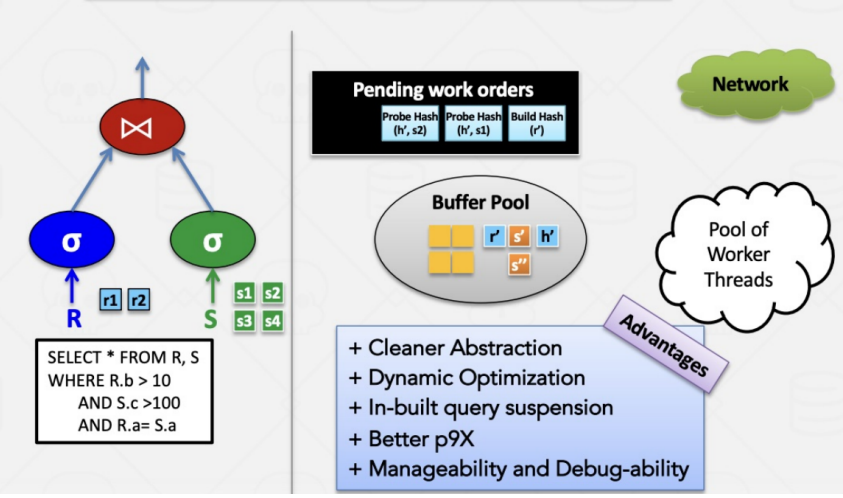
- مقایسه‌ها (=، <، >، !=)
- And ، Or
- عملگرهای حسابی (+، -، *، /، %)
- مقادیر ثابت و پارامتر
- مراجع صفت تاپل

برای ارزیابی یک درخت عبارت در زمان اجرا، سیستم مدیریت پایگاه داده (DBMS) یک نگهدارنده زمینه را که حاوی فرا داده برای اجرا است، مانند زوج‌مرتب فعلی، پارامترها، و طرحواره جدول، نگهداری می‌کند. سپس DBMS درخت را پیمایش می‌کند تا عملگرهای آن را ارزیابی کرده و نتیجه‌ای تولید کند.

ارزیابی شرایط با این روش کند است زیرا DBMS باید کل درخت را پیمایش کرده و عمل صحیح را برای هر عملگر تعیین کند. رویکرد بهتر این است که عبارت را مستقیماً ارزیابی کنیم. بر اساس مدل هزینه داخلی، DBMS تعیین می‌کند که آیا تولید کد برای تسریع یک پرس‌وجو اتخاذ خواهد شد یا خیر.

۶ برنامه‌ریز

مدل‌های پردازش پرس‌وجوی بالا توصیف روشنی از جریان داده ارائه می‌دهند، در حالی که جریان کنترل نسبتاً ضمنی است. زمان‌بند جداسازی روشنی بین جریان داده و جریان کنترل دارد. این روش در حالت دسته‌ای، به‌ویژه مدل بُرداری، به خوبی کار می‌کند. زمان‌بند دستورات کاری را ایجاد می‌کند و به جای فراخوانی درخت عملگر از بالا به پایین، درخت را پیمایش کرده و کارهای زمان‌بندی را در صف زمان‌بندی قرار می‌دهد. سپس رشته‌های کاری درخواست‌ها را از صف واکشی کرده و آنها را اجرا می‌کنند.



در شکل فوق مقایسه مدل پردازش پرس‌وجوی زمان‌بند و مدل پردازش پرس‌وجوی سنتی نشان داده شده است.

به طور کلی، این مدل دارای انتزاع تمیزتر، بهینه‌سازی پویا، تعلیق پرس‌وجوی داخلی، عملکرد بهتر و قابلیت مدیریت و اشکال‌زدایی بهتر است.

۷ پیش‌زمینه

در بحث‌های قبلی اجرای کوئری‌ها فرض بر این بود که کوئری‌ها با یک worker (یعنی thread اجرا می‌شوند. با این حال، در عمل، کوئری‌ها اغلب به صورت موازی با چندین worker اجرا می‌شوند. اجرای موازی مزایای کلیدی متعددی برای DBMS ها فراهم می‌کند:

- افزایش عملکرد در میزان گذردهی (تعداد بیشتر کوئری‌ها در ثانیه) و زمان پاسخ‌دهی (زمان کمتر برای هر کوئری).
- افزایش پاسخگویی و دسترس‌پذیری از دید مشتریان خارجی .DBMS
- کاهش کل هزینه مالکیت. این هزینه شامل هم خرید سخت‌افزار و لایسنس نرم‌افزار، و هم سربار نیروی انسانی برای پیاده‌سازی DBMS و انرژی مورد نیاز برای اجرای ماشین‌ها می‌شود.

دو نوع موازی‌سازی وجود دارد که هاDBMS پشتیبانی می‌کنند: inter-query parallelism و intra-query parallelism.

۸ پایگاه‌داده‌های Parallel در مقابل Distributed

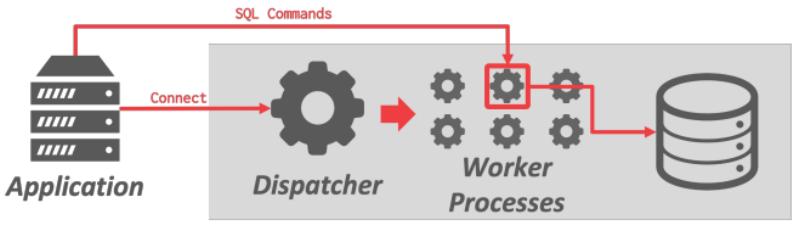
در هر دو سیستم موازی و توزیع‌شده، پایگاه‌داده در میان چندین "منبع" پراکنده شده است تا موازی‌سازی بهبود یابد. این منابع ممکن است محاسباتی (مثل هسته‌های CPU، سوکت‌های CPU، هاGPU ماشین‌های اضافی) یا ذخیره‌سازی (مثل دیسک‌ها، حافظه) باشند. تمایز بین سیستم‌های موازی و توزیع‌شده مهم است:

- **پایگاه‌داده موازی (Parallel-DBMS)**: در یک پایگاه‌داده موازی، منابع یا گره‌ها از لحاظ فیزیکی به هم نزدیک هستند. این گره‌ها از طریق یک اتصال سریع (high-speed-interconnect) با یکدیگر ارتباط برقرار می‌کنند. فرض بر این است که ارتباط بین منابع نه تنها سریع، بلکه ارزان و قابل اطمینان است.
- **پایگاه‌داده توزیع‌شده (Distributed-DBMS)**: در یک پایگاه‌داده توزیع‌شده، منابع ممکن است از هم دور باشند؛ این ممکن است به معنای پراکندگی پایگاه‌داده در قفسه‌ها یا مراکز داده در نقاط مختلف جهان باشد. در نتیجه، منابع از طریق یک اتصال کندتر (اغلب از طریق یک شبکه عمومی) ارتباط برقرار می‌کنند. هزینه‌های ارتباط بین گره‌ها بیشتر است و نمی‌توان از خرابی‌ها چشم‌پوشی کرد.

حتی اگر پایگاه‌داده به طور فیزیکی بر روی چندین منبع تقسیم شده باشد، هنوز به صورت یک نمونه پایگاه‌داده منطقی واحد برای برنامه ظاهر می‌شود. بنابراین، یک کوئری SQL که در برابر یک پایگاه‌داده تک‌گره‌ای اجرا می‌شود باید نتیجه یکسانی در یک پایگاه‌داده موازی یا توزیع‌شده تولید کند.

۹ مدل‌های فرآیند

یک مدل فرآیند DBMS تعیین می‌کند که سیستم چگونه از درخواست‌های همزمان از یک برنامه/محیط چندکاربره پشتیبانی می‌کند. DBMS شامل یک یا چند worker است که مسئول اجرای وظایف به نمایندگی از مشتری و بازگرداندن نتایج هستند. یک برنامه ممکن است یک درخواست بزرگ یا چندین درخواست را به طور همزمان ارسال کند که باید بین هایworker مختلف تقسیم شود.



دو مدل فرآیند اصلی وجود دارد که یک DBMS ممکن است اتخاذ کند: فرآیند به ازای worker و thread به ازای worker. یک الگوی استفاده رایج دیگر از پایگاه‌داده رویکرد تعبیه شده را اتخاذ می‌کند.

۱.۹ Process-per-Worker

پایه‌ای‌ترین رویکرد، فرآیند به ازای worker است. در اینجا، هر worker یک فرآیند جداگانه سیستم‌عامل است و بنابراین به برنامه‌ریز (scheduler) سیستم‌عامل متکی است. یک برنامه درخواست ارسال می‌کند و یک اتصال به سیستم پایگاه‌داده باز می‌کند. یک توزیع‌کننده (dispatcher) درخواست را دریافت می‌کند و یکی از فرآیندهای worker خود را برای مدیریت اتصال انتخاب می‌کند. برنامه سپس مستقیماً با یworker که مسئول اجرای درخواستی است که کوئری می‌خواهد، ارتباط برقرار می‌کند. این توالی از رویدادها در شکل ۱ نشان داده شده است.

اتکا به سیستم‌عامل برای برنامه‌ریزی به طور موثر کنترل DBMS بر اجرای کارها را کاهش می‌دهد. علاوه بر این، این مدل برای نگهداری ساختارهای داده جهانی به حافظه مشترک متکی است یا به پیام‌رسانی متکی است که سربار بیشتری دارد.

یکی از مزایای رویکرد فرآیند به ازای worker این است که خرابی یک فرآیند کل سیستم را مختل نمی‌کند زیرا هر worker در زمینه فرآیند سیستم‌عامل خود اجرا می‌شود.

این مدل فرآیند مساله کارگران متعدد در فرآیندهای جداگانه که نسخه‌های متعددی از همان صفحه را می‌سازند، ایجاد می‌کند. یک راه‌حل برای به حداکثر رساندن استفاده از حافظه، استفاده از حافظه مشترک برای ساختارهای داده جهانی است تا بتوانند توسط کارگرانی که در فرآیندهای مختلف اجرا می‌شوند، به اشتراک گذاشته شوند.

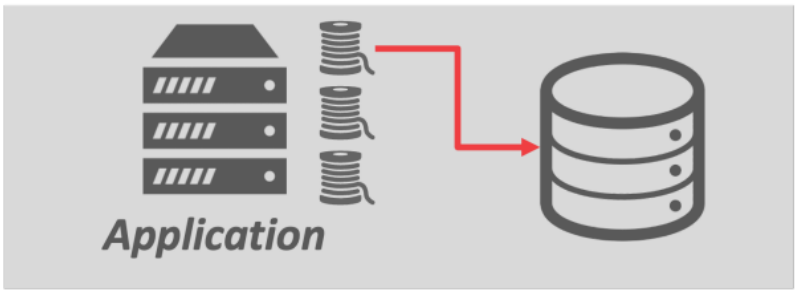
نمونه‌هایی از سیستم‌هایی که از مدل فرآیند به ازای worker استفاده می‌کنند شامل Postgres و Oracle هستند. زمانی که این هاDBMS توسعه یافتند، pthreads هنوز به استاندارد مدل threading تبدیل نشده بود. معنای threading از یک سیستم‌عامل به سیستم‌عامل دیگر متفاوت بود در حالی که `fork()` بهتر تعریف شده بود.

۲.۹ Thread-per-Worker

مدل رایج امروزی thread به ازای worker است. به جای داشتن فرآیندهای مختلف برای انجام وظایف مختلف، هر سیستم پایگاه‌داده فقط یک فرآیند با چندین worker thread دارد. در این محیط، DBMS کنترل کامل بر وظایف و thread ها دارد و می‌تواند برنامه‌ریزی خود را مدیریت کند. مدل چند thread ممکن است از یک thread توزیع‌کننده استفاده کند یا نکند.

استفاده از معماری چند thread مزایای خاصی فراهم می‌کند. اولاً، سربار کمتری برای هر تعویض زمینه (context switch) وجود دارد. علاوه بر این، نیازی به نگهداری مدل مشترک نیست. با این حال، ممکن است که خرابی یک thread کل فرآیند پایگاه‌داده را از کار بیاندازد. همچنین، مدل thread به ازای worker لزوماً به این معنا نیست که DBMS از موازی‌سازی درون کوئری (intra-query-parallelism) پشتیبانی می‌کند.

تقریباً هر DBMS ایجاد شده در ۲۰ سال گذشته از این رویکرد استفاده می‌کند، از جمله SQL Server و MySQL و Oracle مدل‌های خود را به‌روزرسانی کرده‌اند تا از این رویکرد پشتیبانی کنند. Postgres و پایگاه‌داده‌های مشتق شده از Postgres عمدتاً هنوز از رویکرد مبتنی بر فرآیند استفاده می‌کنند.



۳.۹ زمان‌بندی

در نتیجه، برای هر برنامه کوئری، DBMS باید تصمیم بگیرد کجا، چه زمانی و چگونه اجرا کند. سوالات مرتبط شامل موارد زیر هستند:

- از چند وظیفه باید استفاده کند؟
- از چند هسته CPU باید استفاده کند؟
- وظایف باید روی کدام هسته‌های CPU اجرا شوند؟
- خروجی وظیفه باید کجا ذخیره شود؟

هنگام تصمیم‌گیری در مورد برنامه‌های کوئری، DBMS همیشه بیشتر از سیستم‌عامل می‌داند و باید به همین ترتیب اولویت داده شود.

۴.۹ Embedded-DBMS

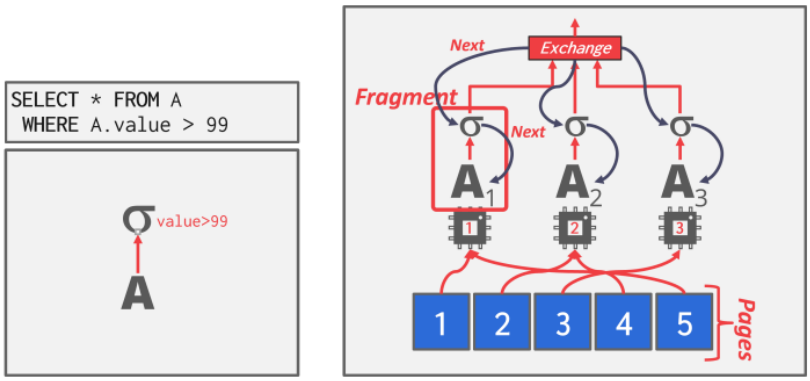
یک الگوی استفاده بسیار متفاوت برای پایگاه‌داده‌ها شامل اجرای سیستم در همان فضای آدرس برنامه است، برخلاف مدل مشتری–سرور که در آن پایگاه‌داده مستقل از برنامه است. در این سناریو، برنامه وظایف و هاthread را برای اجرا روی سیستم پایگاه‌داده تنظیم می‌کند. خود برنامه تا حد زیادی مسئول زمان‌بندی خواهد بود. نموداری از رفتارهای زمان‌بندی یک DBMS تعبیه‌شده در شکل ۳ نشان داده شده است.

SQLite DuckDB، و RocksDB مشهورترین هایDBMS تعبیه‌شده هستند.

۱۰ موازی‌سازی بین کوئری‌ها (Inter-Query-Parallelism)

در موازی‌سازی بین کوئری‌ها، DBMS کوئری‌های مختلف را به طور همزمان اجرا می‌کند. از آنجا که چندین worker به صورت همزمان درخواست‌ها را اجرا می‌کنند، عملکرد کلی بهبود می‌یابد. این امر گذردهی را افزایش داده و زمان تاخیر را کاهش می‌دهد.

اگر کوئری‌ها فقط خواندنی باشند، هماهنگی کمی بین کوئری‌ها مورد نیاز است. با این حال، اگر چندین کوئری به طور همزمان در حال به‌روزرسانی پایگاه‌داده باشند، تعارضات پیچیده‌تری ایجاد می‌شود.



برنامه کوئری برای این SELECT یک اسکن ترتیبی روی A است که به یک عملگر فیلتر تغذیه می‌شود. برای اجرای این به صورت موازی، برنامه کوئری به بخش‌های مجزا تقسیم می‌شود. یک بخش برنامه مشخص توسط یک worker متمایز اجرا می‌شود. عملگر تبادل (exchange-operator) به طور همزمان `Next` را روی همه بخش‌ها فراخوانی می‌کند که سپس داده‌ها را از صفحات مربوطه خود بازیابی می‌کنند.

۱۱ موازی‌سازی درون کوئری (Intra-Query-Parallelism)

در موازی‌سازی درون کوئری، DBMS عملیات یک کوئری واحد را به صورت موازی اجرا می‌کند. این امر زمان تاخیر کوئری‌های طولانی‌مدت را کاهش می‌دهد.

سازماندهی موازی‌سازی درون کوئری را می‌توان از دیدگاه پارادایم تولیدکننده/مصرف‌کننده در نظر گرفت. هر عملگر تولیدکننده داده‌ها و همچنین مصرف‌کننده داده‌ها از یک عملگر دیگر که در زیر آن اجرا می‌شود، است. الگوریتم‌های موازی برای هر عملگر رابطه‌ای وجود دارد. DBMS می‌تواند از چندین thread برای دسترسی به ساختارهای داده متمرکز استفاده کند یا از تقسیم‌بندی برای تقسیم کار استفاده کند.

درون موازی‌سازی درون کوئری، سه نوع موازی‌سازی وجود دارد: درون‌عملگر، بین‌عملگر، و بوشی (bushy). این روش‌ها متقابلاً انحصاری نیستند. مسئولیت DBMS این است که این تکنیک‌ها را به گونه‌ای ترکیب کند که عملکرد را بر روی بار کاری داده شده بهینه کند.

۱.۰.۱۱ موازی‌سازی درون‌عملگر (افقی)

در موازی‌سازی درون‌عملگر، عملگرهای برنامه کوئری به بخش‌های مستقل تجزیه می‌شوند که همان عملکرد را بر روی زیرمجموعه‌های متفاوت (مجزا) از داده‌ها انجام می‌دهند.

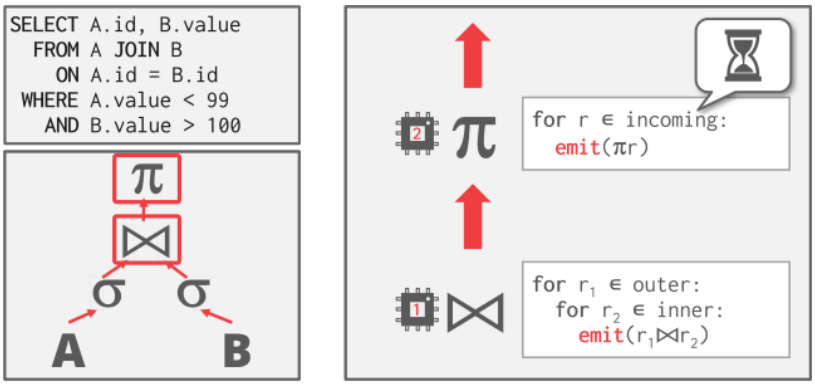
DBMS یک عملگر تبادل (exchange operator) را در برنامه کوئری درج می‌کند تا نتایج را از عملگرهای فرزند جمع کند. عملگر تبادل از اجرای عملگرهای بالاتر در برنامه تا زمانی که تمام داده‌ها را از فرزندان دریافت کند، جلوگیری می‌کند. نمونه‌ای از این در شکل ۴ نشان داده شده است.

به طور کلی، سه نوع عملگر تبادل وجود دارد:

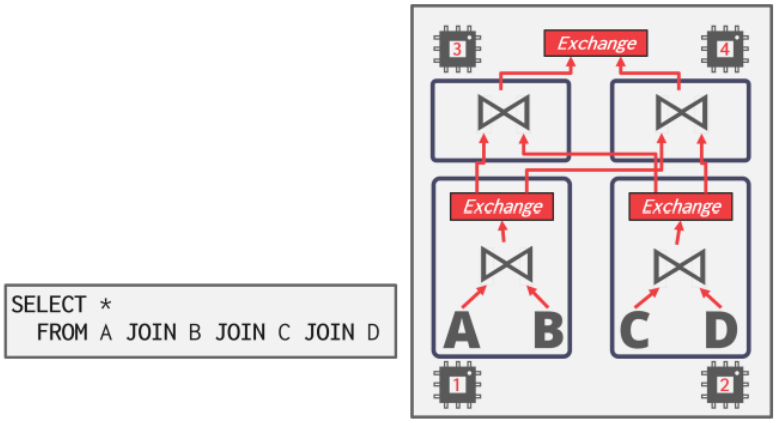
- **Gather**: ترکیب نتایج از چندین worker به یک جریان خروجی واحد. این نوع رایج‌ترین نوع مورد استفاده در هایDBMS موازی است.
- **Distribute**: تقسیم یک جریان ورودی به چندین جریان خروجی.
- **Repartition**: سازماندهی مجدد چندین جریان ورودی در چندین جریان خروجی. این امکان را به DBMS می‌دهد که ورودی‌هایی که به یک روش تقسیم شده‌اند را گرفته و سپس به روش دیگری توزیع کند.

۲.۰.۱۱ موازی‌سازی بین‌عملگر (عمودی)

در موازی‌سازی بین‌عملگر، DBMS عملگرها را به گونه‌ای همپوشانی می‌کند که داده‌ها را از یک مرحله به مرحله بعدی بدون ماده‌سازی انتقال دهد. این گاهی اوقات به عنوان موازی‌سازی لوله‌ای (pipelined-parallelism) نامیده می‌شود.



در عبارت JOIN سمت چپ، یک worker عملیات join را انجام می‌دهد و سپس نتیجه را به worker دیگری ارسال می‌کند که عملیات projection را انجام می‌دهد و سپس نتیجه را دوباره ارسال می‌کند.



برای انجام یک JOIN چهارطرفه روی سه جدول، برنامه کوئری به چهار بخش تقسیم می‌شود همان‌طور که نشان داده شده است. بخش‌های مختلف برنامه کوئری به طور همزمان اجرا می‌شوند، به روشی مشابه با موازی‌سازی بین‌عملگر. این رویکرد به طور گسترده در سیستم‌های پردازش جریان استفاده می‌شود، که سیستم‌هایی هستند که به طور مداوم یک کوئری را بر روی جریان ورودی هاtuple اجرا می‌کنند.

۳.۰.۱۱ موازی‌سازی بوشی (Bushy-Parallelism)

موازی‌سازی بوشی یک ترکیب از موازی‌سازی درون‌عملگر و بین‌عملگر است که در آن worker ها عملگرهای متعدد از بخش‌های مختلف برنامه کوئری را به طور همزمان اجرا می‌کنند. DBMS همچنان از عملگرهای تبادل برای ترکیب نتایج میانی از این بخش‌ها استفاده می‌کند.

۱۲ موازی‌سازی I/O

استفاده از thread اضافی برای اجرای کوئری‌ها به صورت موازی عملکرد را بهبود نمی‌بخشد اگر دیسک همیشه گلوگاه اصلی باشد. بنابراین، مهم است که بتوان پایگاه‌داده را در چندین دستگاه ذخیره‌سازی تقسیم کرد.

برای رفع این مشکل، DBMS ها از موازی‌سازی I/O برای تقسیم نصب بر روی چندین دستگاه استفاده می‌کنند. دو رویکرد برای موازی‌سازی I/O وجود دارد:

۱. موازی‌سازی چند دیسک

۲. پارتیشن‌بندی پایگاه‌داده

۱.۱۲ موازی‌سازی چند دیسک

در موازی‌سازی چند دیسک، سیستم‌عامل/سخت‌افزار برای ذخیره فایل‌های DBMS در چندین دستگاه ذخیره‌سازی پیکربندی می‌شود. این می‌تواند از طریق دستگاه‌های ذخیره‌سازی یا پیکربندی RAID انجام شود. تمام تنظیمات ذخیره‌سازی برای DBMS شفاف است، بنابراین هاworker نمی‌توانند بر روی دستگاه‌های مختلف عمل کنند زیرا DBMS از موازی‌سازی زیرساختی آگاه نیست.

۲.۱۲ پارتیشن‌بندی پایگاه‌داده

در پارتیشن‌بندی پایگاه‌داده، پایگاه‌داده به زیرمجموعه‌های مجزا تقسیم می‌شود که می‌توانند به دیسک‌های مجزا اختصاص داده شوند. برخی از DBMS ها اجازه مشخص کردن مکان دیسک هر پایگاه‌داده فردی را می‌دهند. این کار در سطح سیستم‌فایل آسان است اگر DBMS هر پایگاه‌داده را در یک دایرکتوری جداگانه ذخیره کند. فایل لاگ تغییرات معمولاً به اشتراک گذاشته می‌شود.

ایده پارتیشن‌بندی منطقی این است که یک جدول منطقی واحد به بخش‌های فیزیکی مجزا تقسیم شود که به صورت جداگانه ذخیره/مدیریت می‌شوند. چنین پارتیشن‌بندی به طور ایده‌آل برای برنامه شفاف است. یعنی برنامه باید بتواند به جداول منطقی دسترسی پیدا کند بدون اینکه نگران چگونگی ذخیره‌سازی باشد.

ما این رویکردها را در ادامه ترم و در زمان بحث درباره پایگاه‌داده‌های توزیع‌شده پوشش خواهیم داد.