کنترل هم‌زمانی عملیات در دیتابیس‌های تقسیم شده

پوهنیار زویا سحاب[۱۰]

تقریظ‌دهنده: پوهندوی امیر کرور شهیدزی

مجـلهی علمی-تحقیقی حـوزهی علوم طـبیعی پـوهنتـون کـابـل، ۲ (۳) ۱۳۹۹

چکیده

سـیسـتم‌های دیتابیس در جوامع مدرن نقش عمده داشـته که در ادارات مختلف مثل بانک‌ها، شـفاخانه‌ها، شـرکت‌های هوایی، و یا هم جسـت‌جو از انترنت که در اصـل به یک نوعی از طریق اشـخاص یا توسـط کمپیوتر و پروگرام‌های آن از دیتابیس‌ها استفاده صورت می‌گیرد. پیشرفت تکنالوژی مستلزم به وجود آمدن دیتابیس‌های بزرگ گردیده که می‌توانند انواع مختلف معلومات را ذخیره نمایند. برای اداره‌ی این معلومات، انواع جدید سـیسـتم‌های دیتابیس مانند دیتابیس‌های تقسیم شده معرفی گردیده است. مسأله‌ی که با این پیشرفت به میان می‌آید، کنترل هم‌زمانی عملیات در این دیتابیس‌ها می‌باشد. در این تحقیق ابتدا سیستم‌های تقسیم شده دیتابیس شرح گردیده و بعداً الگوریتم‌های که برای حل این مشکل اسـتفاده می‌گردد مورد بحث قرار گرفته اسـت. بعد از مقایسـه به این نتیجه می‌رسـیم که خواص ACID برای کنترل ترانزکشن‌ها مهم بوده و در این نوع دیتابیس‌ها باید برقرار باشد.

اصطلاحات کلیدی: دیتابیس‌های تقسیم شده؛ کنترل هم‌زمانی عملیات؛ قفل‌سازی؛ ترانزکشن؛ الگوریتم

Concurrency Control in Distributed Databases

Jr. Teaching Asstt. Zoia Sahab

Abstract

Database systems has an essential role in modern societies, which is used in different organizations such as banks, hospitals, airline companies, or in searching via internet that will involve somehow someone or some computers and it's programs accessing a database. Advances in technology have led to large databases that store different kinds of data. To manage this data, new exciting database systems and applications, like distributed database systems have been introduced. With the usage of these systems the concurrency control of transactions becomes a case of concern. In this research distributed database systems has been discussed at first and then the algorithms which is used to resolve case of concurrency has been described. After the comparison between algorithms, in conclusion it is cleared that ACID properties are important for controlling transactions in the database and has to be maintained.

Keywords: Distributed databases; Concurrency Control; Locking; Transaction; Algorithm

ارجاع

سحاب، زویا. (۱۳۹۹). کنترل هم‌زمانی عملیات در دیتابیس‌های تقسیم شده. مجـلهی علمی- تحـقیقی حـوزهی علـوم طـبیعی پوهنتـون کـابل، شماره ۲ (۳)، صص ۱۰۵ – ۱۱۶.

---

[۱] استاد پوهنځی کمپیوترساینس، پوهنتون کابل

مقدمه

دیتابیس عبارت از یک مجموعه دیتای باهم‌مرتبط بوده و دیتا در این زمینه عبارت از حقایق شناخته شده دارای معنی ضمنی است که می‌تواند در حافظه کمپیوتر ذخیره شود، مانند؛ آدرس، نمبر تیلفون، تاریخ تولد و غیره. دیتابیس‌های تقسیم شده (Distributed Databases (DDB) عبارت از مجموعه‌ی از دیتابیس‌های است که از نظر منطقی با هم مرتبط بوده و در قسمت‌های مختلف یک شبکه کمپیوتر پخش و ذخیره گردیده است.

این مجموعه توسط نرم‌افزازهای (Distributed Database Management System (DDBMS اداره می‌شود که تقسیم شدن دیتا را از نظر استفاده‌کننده‌گان پنهان نگه می‌دارد (۱). کنترل هم‌زمانی عملیات (Concurrency Control) عبارت از آن است که صحیح بودن دیتا را زمانی که چندین استفاده‌کننده به صورت همزمان از دیتابیس استفاده می‌کنند، تضمین نماید (۲).

قفل‌سازی (Locking) یکی از میتودهای است که برای کنترل هم‌زمانی معلومات استفاده می‌شود. یک قفل عبارت از متحول است که همراه با خود دیتا داشته و مشخص می‌سازد که کدام عملیات بالای دیتا عملی است. برای هربخش دیتا در دیتابیس یک قفل فراهم بوده که برای هم‌زمان‌سازی دست‌رسی به دیتابیس توسط معاملات هم‌زمانی (concurrent transactions) استفاده می‌شود. بعضی میکانیزم‌های مانند phase locking، time stamping، multi-model time stamp و غیره وجود دارد که برای اداره‌ی هم‌زمانی عملیات استفاده می‌شود (۳). دراین مقاله کوشش به عمل آمده است تا دیتابیس‌های تقسیم شده، کنترل هم‌زمانی عملیات و الگوریتم‌های مورد استفاده‌ی کنترل هم‌زمانی عملیات مورد بحث قرار گرفته و تمام نکات مثبت و منفی که در هر الگوریتم وجود دارد، بیان گردد.

پیشینه‌ی تحقیق

وابسته‌گی به سیستم‌های معلومات با در نظر داشت امنیت، قابلیت اطمینان و دست‌رسی سریع به این معلومات در اداره و تجارت ضرورت به سیستم‌های تقسیم شده را بیشتر می‌سازد. دیتابیس‌های تقسیم شده یک نوع دیتابیس های مجازی است که بخش های آن می‌تواند به صورت فزیکی در دیتابیس‌های حقیقی در موقعیت‌های مختلف ذخیره گردد. سیستم اداره‌ی دیتابیس برای سیستم‌های تقسیم شده دست‌رسی به این دیتابیس‌ها را میسر ساخته و حالت تقسیم شده‌گی معلومات را از نظر استفاده‌کننده مخفی نگه‌می‌دارد و تصور می شود که از یک سیستم مرکزی استفاده گردیده است. دیتابیس‌های تقسیم شده نظر به دیتابیس‌های مرکزی قابلیت اطمینان بیشتر دارد (۴).

با بهتر ساختن ساختار دیتابیس‌های تقسیم شده و کار بالای خالی‌گاه‌های که در کنترل هم‌زمانی عملیات وجود دارد، می‌توان دیزاین این دیتابیس‌ها را پیشرفته‌تر ساخت، که این خود باعث بیشتر شدن سرعت دست‌رسی، مقیاس‌پذیری و انعطاف‌پذیری در زمان دست‌رسی به انواع مختلف دیتا گردد (۵).

روش تحقیق

این مقاله یک تحقیق توصیفی با دیدگاه مقایسه‌ای است. هدف تحقیق بررسی نکات مثبت و منفی الگوریتم‌های مختلف مورد استفاده سیستم دیتابیس‌های تقسیم شده در کنترل هم‌زمانی عملیات می‌باشد.

دیتابیس‌های تقسیم شده

دیتابیس‌های تقسیم شده عبارت از مجموعه‌ی از دیتاهای منطقی با هم مرتبط است که به صورت فزیکی در یک شبکه‌ی کمپیوتر تقسیم یا پخش گردیده است. دیتا می‌تواند با سرعت بسیار زیاد با استفاده ازین شبکه قابل دست‌رس باشد. هر سیستم با استفاده از وسایل انتقال معلومات مانند کیبل تیلفون و یا هم نتورک‌های بسیار سریع با هم می‌توانند مکاتبه نمایند. دیتابیس‌های تقسیم شده شامل بخش‌های است که توسط یک نتورک با هم وصل می‌شود و به صورت فزیکی کدام قسمت مشترک با هم ندارند. دیتا در هر بخش تحت کنترل DBMS (Database Management System) می‌باشد (٦).

انواع دیتابیس‌های تقسیم شده

- دیتابیس‌های تقسیم شده‌ی متجانس (Homogeneous DDBs): تمام بخش‌های تقسیم شده درین نوع دیتابیس از عین نوع DBMS استفاده می‌کنند. تمام بخش‌ها از موجودیت بخش‌های دیگر آگاه بوده و برای پروسس کردن خواست استفاده‌کننده‌گان باهم همکاری می‌کنند. این نوع DDB برای استفاده‌کننده شکل دیتابیس واحد را دارد.
- دیتابیس‌های تقسیم شده‌ی غیر متجانس (Heterogeneous DDBs): این نوع دیتابیس‌ها در هر بخش دارای مودل و DBMS مربوط به خود بوده و از هم‌دیگر متفاوت می‌باشد. هر بخش در این‌جا شاید از بودن بخش‌های دیگر بی‌خبر باشد. درین نوع DDB برای پروسس کردن خواست‌ها محدودیت‌های وجود دارد که مشکل عمده همانا استفاده‌ی مودل‌ها و نرم افزارهای مختلف در هر بخش می‌باشد.

کنترل هم‌زمانی عملیات (Concurrency Control)

کنترل هم‌زمانی عملیات در یک دیتابیس عبارت از اداره‌کردن عملیات مختلف در عین زمان می‌باشد، قسمی‌که صحیح بودن معلومات همیشه تضمین گردد. برای این کار از الگوریتم‌های مختلف استفاده می‌شود که الگوریتم خوب به الگوریتم گفته می‌شود که تمام ترانزکشن‌ها را به یک شکل حمایت کند. وظیفه‌ی کنترل هم‌زمانی عملیات اینست که تغییرات که یک استفاده‌کننده در دیتابیس ایجاد می‌کند، در تغییرات که استفاده‌کننده‌ی دیگر در سیستم ایجاد می‌کند، مداخله نکند. اگر این ترانزکشن‌ها در عین زمان در دیتابیس واقع شوند باعث ایجاد چندین مشکل در صحیح بودن دیتا می‌گردد (۵، ۷). برای درک بهتر موضوع دو ترانزکشن t1 و t2 را که در عین زمان اجرا می‌شوند در نظر می‌گیریم.

جدول ۱: اجرای ترانزکشن‌های t1 و t2 در عین زمان.

t1	Time	t2
	/* x= 100 */	
r(x)	1	
	2	r(x)
/*update x :=x + 50 */	3	
	4	/*update x :=x + 40 */
w(x)	5	
	/* x = 150 */	
	6	w(x)
	/* x = 140 */	

فرض می‌کنیم که X یک دیتای رقمی ا ست که دارای قیمت ۱۰۰ در زمان ۱ می‌با شد. ترانزکشن‌های t1 و t2 هر دو این قیمت را به متحولین محلی خود می‌خوانند و هر دو این قیمت را در متحولین محلی خود تعدیل می‌کنند. t1 عدد ۵۰ را اضافه می‌کند و t2 عدد ۴۰ را به قیمت اولیه ۱۰۰ اضافه می‌کند. بعد t1 و t2 قیمت به‌د ست آمده را به دیتای مشترک می‌نویسند، که در نتیجه دیتای مشترک باید قیمت ۱۹۰ را به خود می‌گرفت اما چون t2 آخرین ترانزکشن است که اجرا شده است پس قیمت

نهایی X در اینجا ۱۴۰ است که دیتای صحیح نیست. چون تغیرات آورده شده از جانب t1 از بین رفته است و مشکل ایجاد شده در این دیتابیس به‌خاطر نبود کنترل هم‌زمانی عملیات در دیتابیس است.

قفل‌سازی (Locking)

با استفاده از قفل سازی (locking) می‌توان دست‌رسی به دیتای مشترک را از حیث زمان با هم مطابق کرد. این قفل‌ها (lock) می‌تواند بالای دیتا ایجاد و یا هم حذف گردد. این به معنی اینست که اگر یک ترانزکشن قفل را بالای یک دیتا می‌گیرد، پس این دیتا برای ترانزکشن‌های هم‌زمان دیگر فراهم نمی‌باشد. زمانیکه یک ترانزکشن یک قفل درخواست می‌کند، دیده می‌شود که آیا قفل توسط کدام ترانزکشن دیگر گرفته شده است یا خیر. اگر قفل توسط کدام ترانزکشن دیگر گرفته شده بود در آن‌صورت ناسازگاری قفل به وجود میاید و درخواست ترانزکشن معلق می‌شود. ترانزکشن حالت انتظار قفل را گرفته و مسدود می شود. در غیر این صورت درخواست قفل قبول شده و گفته می شود که ترانزکشن قفل را بدست آورد. بلافاصله هر ترانزکشن قفل‌ها را آزاد می‌کند که درین وقت باز هم چک می‌شود که آیا کدام ترانزکشن در حالت انتظار وجود دارد تا بتواند قفل را به‌دست آورد یا خیر. اگر وجود داشت قفل برایش داده می‌شود تا بتواند به دیتای مورد نظر دست‌رسی پیدا کند (۷، ۸).

الگوریتم‌های کنترل هم‌زمانی عملیات (Algorithms Distributed Concurrency Control)

قبل از آن‌که به شرح چند الگوریتم کنترل هم‌زمانی عملیات در دیتابیس پرداخته شود. بهتر خواهد بود تا بعضی از اصطلاحات مربوط به این الگوریتم‌ها مرور گردد:

- **Transaction:** یک ترانزکشن عبارت از مجموعه‌ی از عملیات است که بالای دیتای یک دیتابیس انجام می‌شود. یک ترانزکشن در DDB از خواص ACID (Isolation and Atomicity, Consistency, Durability) برخوردار می‌باشد (۸).
- **Atomicity:** یک ترانزکشن به شکل یک واحد عملیات درنظر گرفته می‌شود، طوری‌که یا تمام عملیات ترانزکشن به صورت مکمل در دیتابیس اجرا می‌شود یا هیچ‌کدام.
- **Consistency:** صحیح بودن یک ترانزکشن consistency گفته می‌شود. یک ترانزکشن آن‌وقت صحیح است که با هیچ یک از ترانزکشن‌های که در عین وقت اجرا می‌شود، صحیح بودن دیتابیس را زیر سؤال نبرد و از یک حالت صحیح به حالت صحیح دیگر برود.
- **Isolation:** درین خاصیت اگر چند ترانزکشن در عین زمان اجرا گردد، سیستم این‌را تضمین می‌کند که ترانزکشن اول ختم می‌شود قبل از این‌که ترانزکشن دوم شروع گردد و یا هم ترانزکشن دوم شروع

می‌شـود بعد از این‌که ترانزکشـن اول ختم گردد، در صـورتی‌که هر دو از یک‌دیگر و از هیچ ترانزکشن دیگر که در عین زمان در حال اجرا است، آگاه نیستند.

- **Durability:** بعد از این‌که ترانزکشـن موفقانه ختم می‌شـود، تغیراتی‌که در دیتابیس آورده شـده است، در سیستم به صورت دایمی باقی می‌ماند حتی اگر سیستم خراب شود.

خواص ACID در DDB با اسـتفاده از دو ویژه‌گی پروسـه‌های قابل بازیابی و پروتوکول اجرا شـدن عملی می‌شود:

- پروسـه‌های قابل بازیابی: درین ویژگی صـورت عملیات (log) نگه‌داری می‌شـود تا در صـورت ناکامی عملیات راه برگشت به حالات قبلی وجود داشته باشد.
- پروتوکول اجرا شدن: این پروتوکول برای چندین عملیه اجازه می‌دهد تا با هم به صورت هماهنگ برای یک ترانزکشن اجازه‌ی اجرا شدن و یا هم بی‌نتیجه ماندن را بدهد (۸).

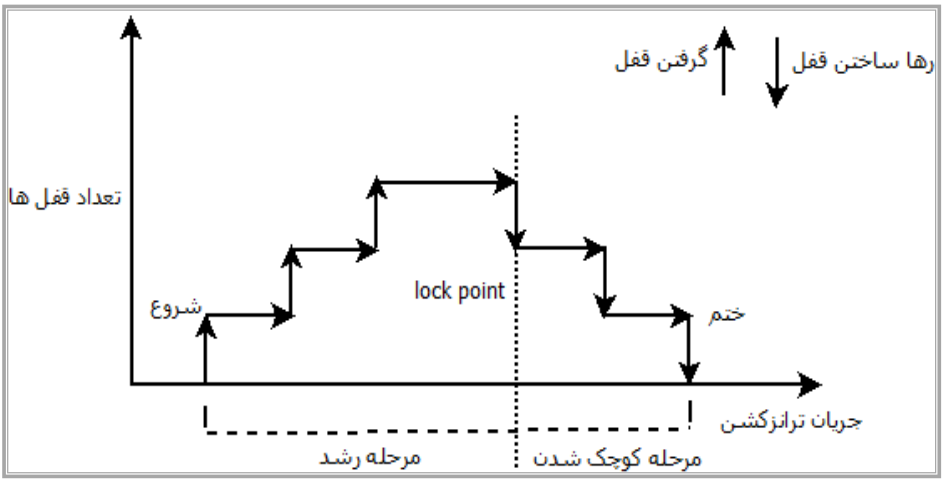
الگوریتم قفل‌سازی دو مرحله‌یی (2PL) (Two-Phase Locking Algorithm)

الگوریتم 2PL یکی از پروتوکول‌های اسـاسـی کنترل هم‌زمانی دیتا می‌باشـد. نکته‌ی اصـلی درین الگوریتم طوری اسـت که "بخوان یکی را بنویس هر کدام را" (read any write all) به این معنی که قفل خواندن می‌تواند بالای دیتا قرار بگیرد و در صـورت ضرورت، قفل خواندن به قفل نوشـتن تبدیل می‌گردد. کافی اسـت که قفل خواندن بالای یکی از کاپی های دیتا قرار بگیرد. این دیتا به صـورت محلی قفل می‌گردد اما اگر ضرورت به تغیر در دیتا وجود داشـت قفل خواندن بالای تمام کاپی‌های دیتا در دیتابیس باید قرار بگیرد. و تا وقتی‌که ترانزکشـن موفقانه اجرا و یا بی‌نتیجه می‌ماند تمام کاپی‌های دیتا قفل گردیده است (۴). این الگوریتم دارای دو مرحله می‌باشد:

مرحله‌ی رشد (Growing Phase): درین مرحله یک ترانزکشن تنها قفل‌ها را به‌دست می‌آورد. هیچ قفل را درین مرحله رها نمی‌سازد. زمانی‌که ترانزکشن اولین قفل را بدست می‌آورد ترانزکشن وارد این مرحله گـردیده و به‌دست آوردن تمام قفـل‌های مورد ضـرورت خود می‌پردازد. در این مرحـله ترانزکشن نمی‌تواند هیچ یک از قفل‌ها را رها سازد حتـی اگر کار با آن دیـتا را ختـم کرده باشد. باالاخره زمانی می‌رسد که تمام قفل‌های که ترانزکشن ضرورت داشت را به‌دست آورده است که این نقطه به‌نام lock point یاد می‌شود.

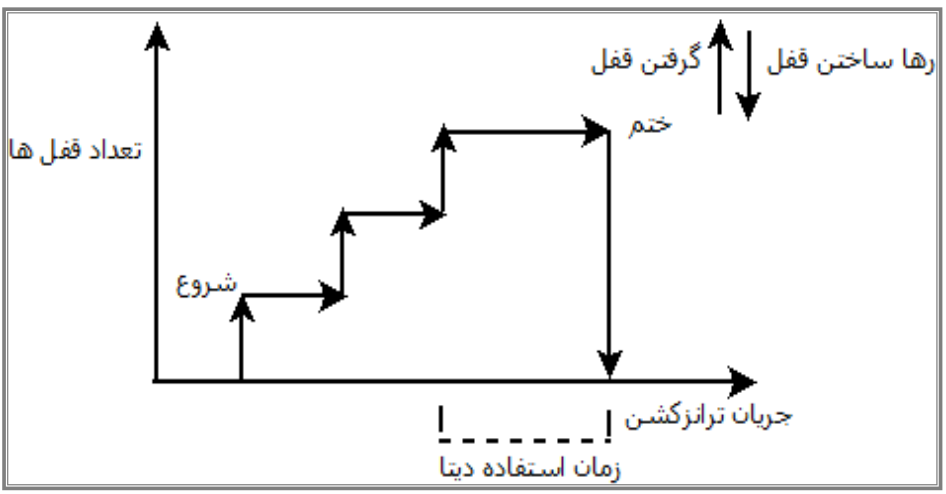
مرحله‌ی کوچک شدن (Shrinking Phase): زمانی‌که تمام قفل‌ها توسط ترانزکشن به‌دست آمد، پس وارد مرحله‌ی کوچک شـدن می‌شـود. درین مرحله ترانزکشـن تمام قفل‌های که به‌دسـت آورده

ا ست را باید رها سازد. در این مرحله دیگر نمی‌تواند که قفل به‌د ست آورد. بعد ازین‌که ترانزکشن به نقطه قفل می‌رسد، وارد مرحله‌ی کوچک شدن می‌شود که باید اولین قفل را رها سازد که بعداً به ترتیب تمام قفل‌ها را رها می‌سازد.



شکل ۱: الگوریتم قفل‌سازی دو مرحله‌ی (2PL)

Strict Two Phase Locking نیز یک نوع از 2PL است که دارای دو مرحله می‌باشد؛ مرحله‌ی اول مانند 2PL گرفتن قفل‌ها بوده اما مرحله‌ی دوم کمی متفاوت است. قفل‌ها وقتی رها می‌گردند که وقت اجرا شدن دستور برسد. بعد از اجرا شدن تمام lock ها به صورت یک‌جایی رها می‌گردند (۴، ۷، ۹).



شکل ۲: الگوریتم Strict Two Phase Locking

الگوریتم Wound - Wait و Wait - Die

برای درک درست الگوریتم Wound - Wait اگر دو ترانزکشن t1 و t2 را درنظر بگیریم، قسمی‌که t1 بخواهد بالای دیتای X یک قفل را بگیرد، در حالی‌که t2 قبل از t1 قفل بالای X را در دست دارد. در این حال اگر t1 برای ختم شدن کار t2 منتظر بماند، شاید منجر به ناکام شدن t1 گردد. پس چیزی که در نظر گرفته می‌شود به خاطر گرفتن قفل بالای X عمر هر ترانزکشن است. هر ترانزکشن که عمر بیشتر داشت حق گرفتن قفل بالای دیتا را دارد. در مثال t1 عمر بیشتر نسبت به t2 دارد پس وقتی‌که می‌خواهد قفل بالای X را به‌دست آورد می‌تواند که t2 را قطع کرده یا زخمی نماید و خودش قفل بالای X را به‌دست آورد. البته ترانزکشن t2 از بین برده نمی‌شود بلکه یک اندازه وقت معین برایش داده می‌شود تا ختم گردد اگر در وقت معینه ختم نگردید، ترانزکشن t2 از بین می‌رود. و اگر عمر t1 کوچک‌تر از t2 است پس t1 می‌تواند منتظر ختم شدن t2 بماند.

الگوریتم Wait - Die کمی متفاوت‌تر از Wound - Wait عمل می‌کند. اگر ترانزکشن t1 می‌خواهد قفل دیتای X را به‌دست آورد که قبلاً به‌دست t2 است و عمر t1 بیشتر از t2 است پس t1 می‌تواند منتظر t2 بماند. اما اگر t1 کوچک‌تر از t2 است در آن‌صورت t1 از بین می‌رود و بعداً دوباره با وقت به وجود آمدن قبلی به وجود می‌آید، نه با وقت جدید. البته درین الگوریتم حق منتظر ماندن تنها برای ترانزکشن‌های با عمر بیشتر است (۸).

الگوریتم Basic Timestamp Ordering (BTO)

نشان یا مهر زمان (Timestamp) عبارت از یک تعریف‌کننده‌ی بیمانند یا بی‌همتا است که توسط اداره‌کننده‌ی عملیات به وجود آمده و توسط آن هر ترانزکشن ti شناخته می‌شود. در اصل قیمت‌های مهرزمان به همان ترتیبی که ترانزکشن‌ها وارد سیستم می‌شود، تعیین می‌شود. پس هر مهرزمان نشان‌دهنده‌ی زمان داخل شدن هر ترانزکشن در سیستم می‌باشد. مهرزمان متشکل از ساعت محلی کمپیوتر یا یک شمارنده‌ی که به ترتیب یک-یک اضافه گردد و آدرس خود کمپیوتر می‌باشد، و چون هر پروسسر در واحد وقت تنها یک ترانزکشن را پروسس کرده می‌تواند پس هر مهرزمان دارای قیمت یگانه می‌باشد و هیچ دو ترانزکشن با عین مهرزمان نمی‌تواند به وجود بیاید (۸). بناءً با استفاده از مهرزمان می‌توان هم‌زمانی عملیات را مرتب و کنترل کرد. قانون مرکزی Timestamp Ordering (TO) طوری است که:

اگر Pi(X) و qj(X) دو عملیه از ترانزکشن‌های مختلف باشند که با هم در تناقض قرار دارند طوری‌که $i \neq j$ است پس pi(X) قبل از qj(X) اجرا می‌شود اگر $ts(ti) < ts(tj)$.

اگر عملیه‌ی pi برای اجرا شدن برای اداره‌کننده‌ی دیتا روان شود اما نظر به عملیه‌ی qj که همراه با این در تناقض قرار دارد طوری‌که $i \neq j$ و $ts(tj) > ts(ti)$ باشد. پس عملیه pi اجرا نشده و رد می‌گردد که در نتیجه ترانزکشن pi باید رد گردیده و بعداً با یک timestamp جدید با یک قیمت بزرگ‌تر دوباره شروع به کار می‌کند. برای تشخیص این‌که آیا عملیه‌ی که برای دیتامنیجر فرستاده شده است، دیر رسیده است یک تنظیم کننده‌ی BTO باید مهرهای ذیل را برای دیتای X ذخیره نماید:

- Max-r-scheduled (X): بزرگ‌ترین قیمت مهرزمان برای عملیه‌ی خواندن بالای دیتای X که قبلاً برای اداره‌کننده‌ی دیتا روان شده است.
- Max-w-scheduled (X): بزرگ‌ترین قیمت مهرزمان برای عملیه‌ی نوشتن بالای دیتای X که قبلاً برای اداره‌کننده‌ی دیتا روان شده است.

زمانی‌که عملیه‌ی pi(X) فرستاده می‌شود، ts(ti) با max-q-scheduled(X) برای هر عملیه q که همراه با p در تناقض است مقایسه می‌گردد. اگر ts(ti) < max-q-scheduled (X) باشد در این صورت pi(X) رد می‌شود چون دیر رسیده است. در غیر آن برای اداره‌کننده‌ی دیتا روان شده و ts(ti) > max-p-scheduled(X) تغیر داده می‌شود به ts(ti) اگر (۸) max-p-scheduled(X).

برای دیتابیس‌های تقسیم شده این الگوریتم به شکل "read any, write all" کار می‌کند قسمی که درخواست خواندن می‌تواند به یکی از کاپی‌های دیتا فرستاده شود اما درخواست نوشتن باید به تمام کاپی‌های دیتا فرستاده شود (۴).

مقایسه‌ی الگوریتم‌های کنترل هم‌زمانی عملیات در دیتابیس‌های تقسیم شده

جدول ذیل نشان دهنده‌ی نکات مثبت و منفی الگوریتم‌های مورد استفاده در کنترل هم‌زمانی عملیات در دیتابیس‌های تقسیم شده می‌باشد.

جدول ۲: نکات مثبت و منفی الگوریتم‌های مورد استفاده در کنترل هم‌زمانی عملیات.

الگوریتم	نکات مثبت	نکات منفی
2PL	عمل‌کرد این الگوریتم در سیستم‌های مرکزی خوب‌تر است	عمل‌کرد این الگوریتم در سیستم‌های تقسیم شده نظر به سیستم‌های مرکزی خوب نیست
S2PL	هیچ ترانزکشن دیگر نمی‌تواند دیتای که توسط یک ترانزکشن در حال نوشتن است را مورد استفاده قرار بدهد.	بعضاً ترانزکشن‌ها به‌خاطر انتظار طولانی بی‌نتیجه بمانند.
Wound – Wait	همیشه برد از آن ترانزکشن‌های با عمر بیشتر است.	گراف انتظار در سیستم‌های تقسیم شده بزرگ‌تر می‌گردد.
Wait – Die	ترانزکشن‌های با عمر بیشتر هیچ وقت از بین نمی‌روند.	ترانزکشن‌های با عمر بیشتر باید منتظر ختم شدن ترانزکشن‌های با عمر کم‌تر بمانند.
BTO	در سیستم‌های مرکزی و تقسیم شده مؤثر است	به حافظه اصلی بیشتر ضرورت دارد.

- الگوریتم 2PL در دیتابیس‌های که در یک سیستم مرکزی ذخیره می‌گردد خوب‌تر کار کرده می‌تواند نسبت به دیتابیس‌های تقسیم شده که در سیستم‌های متعدد ذخیره گردیده و توسط نتورک با هم وصل می‌باشند.
- الگوریتم Strict-2PL نیز مانند 2PL بوده، اما نکته‌ی مفید درین الگوریتم این است که هیچ ترانزکشن دیگر نمی‌تواند دیتای که توسط ترانزکشن دیگر در حال نوشتن است را حتی بخواند تا وقتی‌که ترانزکشن اولی بعد از نوشتن دیتا عملیه‌ی اجرا را انجام دهد. اما نکته‌ی ضعف این الگوریتم درین است که شاید بعضی ترانزکشن‌ها به‌خاطر انتظار طولانی بی‌نتیجه بمانند و یا هم پایان یابند.
- نقص الگوریتم wound wait این است که در دیتابیس‌های تقسیم شده نظر به دیتابیس‌های مرکزی جدول وقت انتظار بزرگ‌تر می‌گردد. این جدول شامل مجموعه‌ی از قفل‌ها از سیستم‌های مختلف می‌باشد که بعضاً منجر به عقب‌گرد شدن ترانزکشن‌ها می‌گردد.
- در الگوریتم Wait-Die ترانزکشن‌های که عمر بیشتر دارند باید منتظر ختم شدن ترانزکشن‌های بمانند که عمر جوان‌تر دارند که این با عث پایین آمدن سرعت اجرای ترانزکشن‌ها می‌شود.
- در Wait-Die نظر به درخواست ممکن ترانزکشن‌های جوان‌تر از بین بروند و یا هم دوباره از نو شروع به کار کنند. اگر با مهرزمان قبلی شروع به کار نمایند، شاید باز هم با ترانزکشن‌های قبلی با عین مهرزمان در تضاد واقع گردند.
- در Wait-Die ترانزکشن‌های با عمر زیاد هیچ وقت از بین نرفته و دوباره شروع به کار نمی‌کنند.
- الگوریتم BTO در هر دو نوع دیتابیس هم مرکزی و هم تقسیم شده خوب کار کرده می‌تواند.
- الگوریتم BTO برای نگه‌داری مهر زمان به حافظه اصلی بیشتر ضرورت است.
- بعضی ترانزکشن‌ها در الگوریتم BTO ممکن بی‌نتیجه بمانند و باز با مهرزمان جدید شروع به کار نمایند و این کار چندین بار به صورت دورانی اجرا گردد، بدون این‌که این ترانزکشن ختم گردد.

نتیجه‌گیری

در این تحقیق کنترل هم‌زمانی عملیات در دیتابیس‌های تقسیم شده شرح گردیده و الگوریتم‌های مورد استفاده این دیتابیس‌ها مانند 2PL, Wound - Wait, Wait - Die و Basic Timestamp Ordering نیز شرح و با هم مقایسه گردیده است. خواص ACID برای کنترل ترانزکشن‌ها خیلی مهم می‌باشد که درین نوع دیتابیس‌ها باید برقرار باشد. الگوریتم 2PL در دیتابیس‌های که در یک سیستم مرکزی ذخیره می‌گردد، خوب‌تر کار کرده می‌تواند در حالی‌که الگوریتم BTO در هر دو نوع هم مرکزی و هم تقسیم شده خوب کار کرده می‌تواند. بعضی قفل‌ها شاید در جریان ترانزکشن غیر ضروری باشد، اما راه خوب‌تر برای کنترل بهتر هم‌زمانی عملیات نگه‌داری قفل تا ختم ترانزکشن می‌باشد.

منابع

(1) S. B. N. Ramez Elmasri, Fundamentals of Database Systems, 7th ed., NJ United States of America: PEARSON, 2016.

(2) M. V. S. Andrew S.Tanenbaum, Distributed Systems Principles and Paradigms, 2nd ed., NJ United States of America: PEARSON, 2007.

(3) H. F. K. S. S. Abraham Silberschatz, Database System Concepts, 7th ed., New York: McGraw-Hill, 2019.

(4) C. E. B. Thomas M. Connolly, Database Systems A Practical Approach to Design, Implementation, and Management, 6th ed., Harlow, England: PEARSON, 2015.

(5) H. S. I. A. Qasim Abbas, "Concurrency Control in Distributed Database System," in International Conference on Computer Communication and Informatics, Coimbatore, INDIA, 2016.

(6) C. W. N. K. Paul Kahoro, "Concurrency Control In Distributed Databases," in United States International University Africa, Nairobi, Kenya, 2014.

(7) G. V. Gerhard Weikum, Transactional Information Systems Theory, Algorithms, and the Practice of Concurrency Control and Recovery, San Francisco, CANADA: Morgan Kaufmann Publishers, 2002.

(8) F. S. H. Saeed K. Rahimi, Distributed Database Management Systems A Practical Approach, 1st ed., NJ, United States of America: Wiley, IEEE Computer Society Press, 2011.

(9) H. H. K. Mandeep Kaur, "Concurrency Control in Distributed Database System," International Journal of Advanced Research in Computer Science and Software Engineering. 2013; vol 3, No. 7, pp 1444-1447.