پایگاه داده پیشرفته

دکتر شجاعی مهر
علیرضا سلطانی نشان

۱۹ آبان ۱۴۰۲

فهرست مطالب

۱ مفاهیم

۱.۱ تراکنش

تراکنش واحد اجرای برنامه است. عملیاتی که در هر تراکنش می‌تواند شامل شود موارد زیر می‌باشد:

- Create
- Read
- Update
- Delete

۲.۱ قوانین ACID

۱.۲.۱ اتمیک یا Atomicity

هر تراکنش دیتابیس به صورت اتمیک می‌باشد. این قضیه بدان معناست که این تراکنش یا باید کاملا انجام شود یا کلا لغو و صرف نظر شود. در غیر این صورت اگر تراکنش به صورت ناتمام و ناقص انجام شود عواقب مختلفی روی دیتابیس خواهد گذاشت.

۲.۲.۱ جامعیت یا Consistency

هر تراکنش باید از قوانین جامعیت پیروی کند. نمی‌توان داده یا را وارد جدولی از دیتابیس کرد که به صورت معتبر نباشد. در برخی از مراجع این قانون را به اجرای صحیح و سازگار تراکنش می‌شناسند. مهم ترین مثال آن است که شما یک Validation روی یک مقداری از فیلد جدول تنظیم می‌کنید که هر داده‌ای بر روی آن فقط با شرایط تعریف شده بایستی وارد شود.

خالی از لطف نیست که در مورد مرجع پذیری داده‌ها در این قسمت نیز می‌توان صحبت کرد تا بتواند قوانین جامعیت را به طور صحیح کامل کرد. مرجع پذیری زمانی مطرح می‌شود که یک رکوردی از داده وقتی وارد جدولی از دیتابیس می‌شود ممکن است ارتباط مشخصی با جدولی دیگر داشته باشد. پس به همین خاطر کلیدهای اصلی و خارجی در خصوص جامعیت وجود دارند که داده‌ای معنادار را پس از پرس و جو از دیتابیس به برنامه نویس برگرداند. یادآوری، بخش جوین‌ها در دیتابیس و تعریف رفرنس در هنگام تعریف کلید جانبی.

۳.۲.۱ انزوا یا Isolation

هر سیستم جامع پایگاه داده‌ای باید بتواند روی همروند تراکنش‌ها مدیریت و کنترل کامل داشته باشد. انزوا تراکنش‌ها قابلیت کنترل و تنظیم بر اساس DBMS است.

به طور کل همروندی یا همزمانی به حالتی گفته می‌شود که چند تراکنش بخواهند در یک زمان به صورت موازی روی یک منبع عملیات خواندن و نوشتن را انجام دهند. اما این عملیات به طور کل هزینه خاص و مشخصی برای برنامه نویس و مدیر دیتابیس دارد.

۴.۲.۱ قابلیت اعتماد یا Duribility

قابلیت اعتماد یکی از مهم‌ترین ویژگی‌های هر سیستم دیتابیسی است. یعنی بتوان داده‌ها را در پایگاه‌داده به صورت پایدار و ثابت نگهداری و مراقبت کرد. در صورت بروز مشکل روی داده‌های یک دیتابیس می‌توان به عملیات انجام شده در این قسمت مراجعه کرد. بطور کلی این بخش قابلیت کنترل و مدیریت دارد و می‌توان مجموعه فرایندهای نگهداری و بک‌آپ را به صورت خودکار انجام داد.

۳.۱ تنظیم قابلیت انزوا

انزوا و مدیریت همروندی در دیتابیس به چهار طریق قابل انجام است:

۱. Read uncommitted

۲. Read commmitted

۳. Repeadable read

۴. Serializable

یادآوری: هر تراکنش دو حالت در پایان پیدا می‌کند:

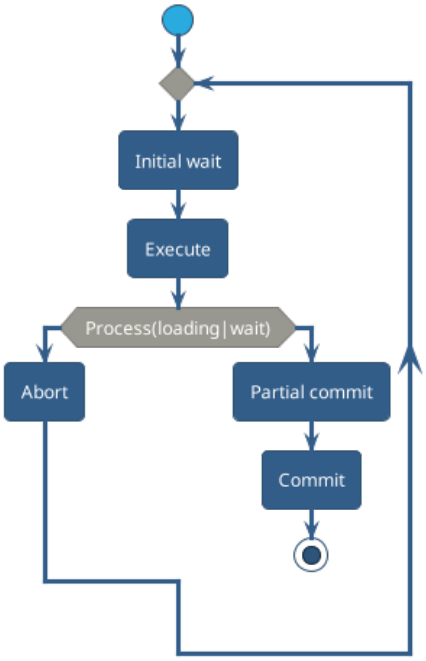
- Commit: تراکنش درنهایت تایید و انجام می‌شود
- Abort: تراکنش در نهایت سقط یا صرفه نظر می‌شود

۱.۳.۱ وضعیت تراکنش

نکته: Abort در دو شرط اتفاق می‌افتد:

۱. زمانی که اجرای تراکنش به خطای Run time دچار شود.

۲. خرابی و نقص سیستم که روی اجرای تراکنش تاثیر می‌گذارد که کامل نشود



شکل ۱: نمودار شروع فرایند تراکنش‌ها

۴.۱ همروندی

۱.۴.۱ مزیت همروندی

۱. افزایش سرعت گذردهی یا throughput

۲. کاهش میانگین زمان پاسخدهی به تراکنش مورد نظر

۲.۴.۱ معایب همروندی

۱. Last update: تغییرات گمشده به دلیل همزمانی در خواندن و نوشتن قانون Write before Write

۲. Uncommitted: خواندن داده‌ای که معتبر نیست. معمولا به آن Dirty read هم گفته می‌شود. قانون Write before Read

۳. Inconsistent retrieval: بازیابی داده‌ای که ناهمگان است. Read before Write

۵.۱ زمان‌بندی

زمان‌بندی به اجرای همروند و همزمان چندین تراکنش با هم گفته می‌شود.

۶.۱ نظریه پی در پی پذیری زمان‌بندی‌ها

به دو روش می‌توان به پی در پی پذیری رسید:

۱. Conflict serializability

۲. View serializability

نمادهای مورد استفاده برای تعریف تراکنش‌ها:

- $R_i|Q|$
- $W_i|Q|$
- $C_i|Q|$
- $A_i|Q|$
- $B_i|Q|$
- $E_i|Q|$

۷.۱ سه شرط اصلی تصادم

اگر p_i و q_j دو تراکنش باشند:

۱. i != j

۲. هر دو به یک داده دسترسی داشته باشند

۳. حداقل یکی از دستورات عمل نوشتن یا write داشته باشد

جدول ۱: حالات تصادم

	$R_i(Q)$	$W_j(Q)$
$R_i(Q)$	ندارد	دارد
$W_j(Q)$	دارد	دارد

۸.۱ زمان‌بندی سریالی

در زمان‌بندی پی در پی، زمانی که یک تراکنش commit یا abort شود به دنبال تراکنش بعدی خواهد رفت که به آن تراکنش سریالی یا Serializable schedule می‌گویند.

$S_1 = R_1(A)W_1(A)a_1W_2(A)W_2(B)C_2$

زمان‌بندی سریالی بالا در حقیقت به دو فرایند تقسیم می‌شود. چرا که در انتهای تراکنش اول پیام سقوط کرده و برنامه به دنبال فرایند بعدی رفته است که روی منبع دیگری در حال انجام پردازش است.

فرایند نافرجام اول:

$S_1 = R_1(A)W_1(A)a_1$

فرایند commit شده دوم:

$S_1 = W_2(A)W_2(B)C_2$

جدول ۲: تراکنش‌های سریالی پی در پی

T_1	$R_1(A)$	$W_1(A)$	a_1			
T_2				$W_2(A)$	$W_2(B)$	C_2

۹.۱ زمان‌بندی‌های معادل در برخورد یا Conflict equivalent

زمانی که دستورات یک زمان‌بندی را وارد زمانبندی دیگر کنیم به گونه‌ای که باعث تصادم و برخورد نشود، این دستورات در این زمان‌بندی با هم معادل در برخورد هستند.

با توجه به تراکنش‌های t_1 و t_2 و t_3 و t_4 زیر، می‌توان دریافت که این دو تراکنش با یکدیگر معادل در برخورد هستند. به گونه‌ای که بعد از جا به جایی هیچ تصادمی رخ نداده است.

جدول ۳: تراکنش‌های معادل در برخورد اول

T_1	$R(Q)$	$W(Q)$		$R(P)$		$W(P)$	C			
T_2			$R(Q)$		$W(Q)$			$R(Q)$	$W(Q)$	C

جدول ۴: تراکنش‌های معادل در برخورد دوم

T_3	$R(Q)$	$W(Q)$		$R(P)$	$W(P)$		C			
T_4			$R(Q)$			$W(Q)$		$R(Q)$	$W(Q)$	C

اما در مثال بعد هر دو تراکنش t_1 و t_2 مستعد به برخورد در یکی از فرایندها در زمان هستند.

جدول ۵: تراکنش‌های معادل در برخورد اول

T_1	$R(Q)$	$W(Q)$		$R(P)$		$W(P)$	C			
T_2			$R(Q)$		$W(Q)$			$R(Q)$	$W(Q)$	C

جدول ۶: تراکنش‌های معادل در برخورد اول

T_1	$R(Q)$	$W(Q)$		$R(P)$		$W(P)$	C			
T_2		$R(Q)$	$R(Q)$		$W(Q)$			$R(Q)$	$W(Q)$	C

۱۰.۱ گراف پی در پی پذیر

کامپیوتر برای تشخیص وجود برخورد در تراکنش‌ها از تئوری گراف پی در پی پذیر استفاده می‌کند. در این روش به صورت بصری ارتباطات تراکنش‌ها را نسبت به یکدیگر را نمایش می‌دهیم. در صورتی که بین دو یا چند تراکنش دور یا حلقه ایجاد شود، می‌گوییم که این تراکنش‌ها با هم برخورد دارند.

سیستم DBM از گراف زمان اجرا خبر دارد و دائما در حال بروزرسانی آن است. اگر وجود دور یا حلقه را تشخیص دهد، برخورد را بررسی کرده و اعلام می‌کند که این تراکنش‌ها پی در پی پذیر در برخورد نیستند و از اجرای این تراکنش‌ها جلوگیری می‌کند.

۱.۱۰.۱ کشتن فرایند تراکنش‌ها

منظور از جلوگیری می‌تواند به دو روش باشد: یا کلا از اجرای تراکنش‌ها جلوگیری می‌کند یا بررسی می‌کند که کدام تراکنش یا تراکنش‌ها باعث ایجاد برخورد در تراکنش‌های دیگر می‌شود، آن را تشخیص داده و تراکنش آن را می‌کشد [۱].

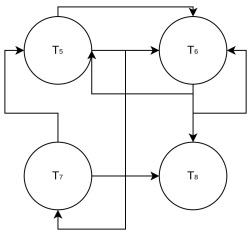
[۱] Kill transaction

برای مثال تراکنش‌های زیر را در نظر بگیرید:

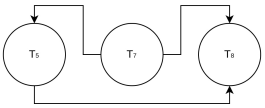
جدول ۷: تراکنش‌های بانکی

T_5			W(Q)		
T_6	R(Q)				W(Q)
T_7		W(Q)			
T_8				R(Q)	

گراف این تراکنش‌ها به شکل زیر است. توجه شود که هر تراکنش می‌تواند به صورت ترتیبی نسبت به تراکنشی بعدی خود ارتباط داشته باشد. در صورتی که حلقه ایجاد شود بایستی عامل ایجاد حلقه پیدا و سپس کشته شود.



شکل ۲: گراف تراکنش‌ها و ایجاد ارتباطات حلقه دار



شکل ۳: تراکنش حذف شده و ایجاد گرافی بدون حلقه

در این مثال برای حذف حلقه می‌تواند یکی یکی تراکنش‌های مورد نظر را بررسی کرد و در صورت حذف یکی از تراکنش‌ها حلقه حذف شد می‌توان آن را نتیجه گرفت و اعلام کرد این تراکنش‌ها باهم سازگارند و برخورد ایجاد نمی‌کنند. در نهایت سیستم DBM تصمیم به اجرای تراکنش‌ها خواهد کرد.

۱۱.۱ پی در پی پذیری در دید یا View equivalent

زمانی می‌گوییم پی در پی پذیری در دید برقرار است که نتایج یکسانی در سیستم DBM با یک زمان‌بندی پی در پی داشته باشیم.

سه قاعده اصلی پی در پی پذیری در دید:

۱. برای هر داده Q تراکنشی که در S مقدار اولیه داده‌ای Q را می‌خواند در 'S هم همان تراکنش اولیه مقدار Q را بخواند (خواندن‌های اولیه)

۲. برای هر داده Q اگر t_i در S داده Q را از t_j می‌خواند، در 'S هم t_i همان داده را از t_j بخواند. (خواندن‌های میانی)

۳. برای هر داده Q آخرین تراکنشی از S که روی Q می‌نویسد در 'S هم همان تراکنش نوشتن پایانی را روی Q انجام دهد. (نوشتن‌های پایانی)