درس پایگاه داده پیشرفته

مباحث درس :

۱- همروندی و کنترل همروندی

۲- ترمیم

۳- امنیت

۴- جامعیت

موضوعات تحقیقاتی :

1- no sql and big database
2- mobile database
3- fuzzy database
4- distributed database
5- RAID
6- anomaly detection in database
7- cloud database
8- سیستم های تصمیم یار
9- multi database

بارم بندی فعالیت ها:

۱- ارائه کنفرانس ۲ نمره

۲- ارائه مقاله ۴ نمره

۳- امتحان پایان ترم ۱۴ نمره

کنترل همروندی

تعریف پایگاه داده ها

مجموعه ای از داده های ذخیره شده و پایا تحت یک نام مشترک، بصورت صوری، مبتنی بر یک ساختارداده ای مشخص ،تحت مدیریت یک سیستم واحد ، مورد استفاده یک یا چند کاربر به صورت اشتراکی و همزمان.

صوری : سیستم باید به کاربران امکان دهد تا داده های خود را آن گونه که خود می بینند، به طور انتزاعی و به دور از جنبه های ذخیره سازی روی رسانه ذخیره کنند.

ساختارهای داده ای
- رابطه ای
- سلسله مراتبی
- شبکه ای
- object oriented

کاربران
- DA
- DBA
- PDB
- End user

سیستم مدیریت واحد:
نرم افزار DBMS
مجموعه ای از برنامه هاست که امکان کار با پایگاه را فراهم می کند.

سیستم پایگاه داده

مجموعه ای است از سخت افزارها و نرم افزارهایی که امکان دسترسی به داده ها و انجام عملیات روی آنها را فراهم میکند یعنی امکان انجام و اجرای تراکنش ها را میدهد.

تراکنش:

یک یا چند برنامه که با پایگاه داده ها کار میکنند.

خواص تراکنش :

❊اتمیک (Atomicity): به خاصیت همه یا هیچ معروف است. یعنی باید یا تمام دستورات یک تراکنش اجرا شوند و یا هیچکدام از آنها اجرا نشوند.

❊سازگاری (Consistency): هر تراکنش اگر به تنهایی اجرا شود پایگاه را از حالتی صحیح به حالت صحیح دیگری می برد.

ادامه خواص تراکنش

* انزوا (Isolation) :همروندی تراکنش ها باید کنترل شوند تا اثر مخرب روی همدیگر نداشته باشند.

* پایداری(Durability): تراکنش هایی که به مرحله تثبیت(Commit) برسند اثرشان در پایگاه داده ها ماندگار است.

اعمال روی پایگاه داده ها

Read(x)
Write(x)
Commit
Abort
Start
X***** یک ایتم داده ای است.

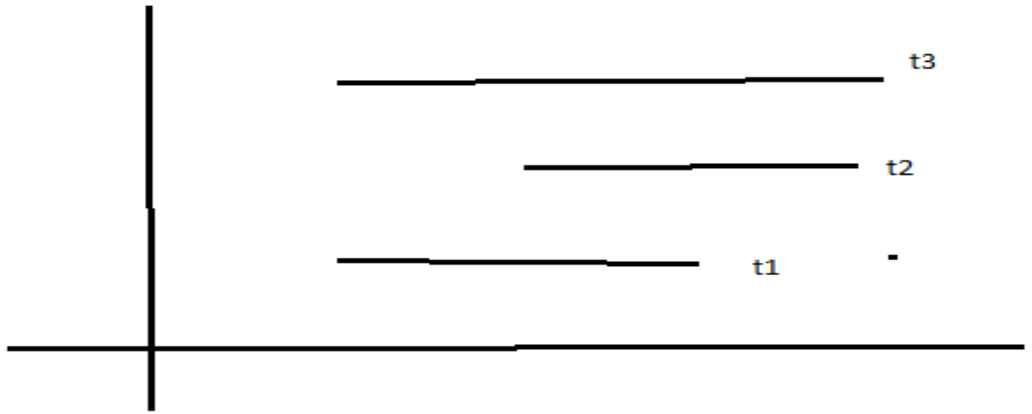
انواع تراکنش

- تراکنش فعال(Active): تراکنشی که شروع شده و درحال اجراست.
- تراکنش نهایی شده(Commited): تراکنش عمل Commit را اجرا کرده، سیستم هم آن را پذیرفته است و اثرات تراکنش در پایگاه داده ها ماندگار شده است.
- تراکنش سقط شده(Aborted): تراکنشی که دستور Abort برای آن صادر شده و اثری از آن در پایگاه نیست.
- تذکر: دستور Abort از منابع زیر صادر می شود:
- - خود تراکنش آن را اجرا میکند مثلا هنگامی که موجودی کافی نباشد.
- - سیستم مدیریت پایگاه داده ها(DBMS)

کنترل همروندی

تعریف

به مجموعه ای از تلاشها که جهت اجرای همروند تراکنش ها که احتمالا به داده مشترک دسترسی دارند بدون آنکه تداخل رخ دهد.(تخریب نشوند یعنی داده ها از حالت سازکار خارج نشوند)



تراکنش های t1 و t2 همروند هستند.

تراکنش های t1 و t3 همزمان و در نتیجه همروند هستند.

ترمیم(Recovery)

تضمین میکند که داده های ذخیره شده در پایگاه داده ها ماندگار باشد.

هدف اصلی از کنترل همروندی و ترمیم آن است که تراکنش ها بصورت اتمیک انجام شود. یعنی

- – بدون اینکه اثرات نامطلوب بر یکدیگر داشته باشند اجرا شوند.
- – یک تراکنش به صورت نرمال خاتمه یابد و تمام اثراتش ماندگار شود یا هیچ اثری نداشته باشد.

دو راه حل برای عدم تداخل

- - تراکنش ها به صورت پی درپی اجرا شوند.(اجرای پی در پی)

- - تراکنش ها به صورت تودرتو(قاطی) اجرا شوند و نتیجه درست باشد.(اجرای پی در پی پذیر)

*هدف از کنترل همروندی راه حل دوم است.

انواع تداخل

چنانچه کنترل همروندی نباشد ، تداخل باعث ناسازگاری می شود.

- بهنگام سازی گم شده(Last update): ناشی از تداخل دو عمل write.

مثال (R1(x)W1(x)R2(x)W2(x)C2C1

- بازیابی ناهنگام(تحلیل ناسازگار): ناشی از تداخل دو عمل Read و write.

مثال) R1(x)W2(x)R1(x)

ادامه انواع تداخل

- وابستگی تثبیت نشده(دستیابی به داده تثبیت نشده): ناشی از تداخل دو عمل writeوRead.

مثال)W1(x)R2(x) a1

کنترل همروندی طوری دستورات تراکنش های همروند را قاطی میکند (تقدم و تاخر دستورات مشخص میکند) که نتیجه درست حاصل شود گویی که تراکنش ها پی در پی اجرا شده اند.

اثرات تراکنش

- قلم داده ای را مینویسد(اثر write)←اثر گذاشتن روی سایر تراکنش ها
- مقادیری را از دیگر تراکنش ها می خواند(اثر read)←اثرپذیرفتن از سایر تراکنش ها

*وقتی تراکنشی Abort می شود هیچ اثری نباید داشته باشد یعنی :

- هر قلم داده ای که بازنویسی شده مقدارش به مقدار قبلی برگردد.
- هر تراکنشی که اثر پذیرفته باید برگردادنده شود.

راه حل مشکلات فوق

- راه حل مشکل اول : undo کردن. یک کپی از مقدار اولیه x نگه داریم (تصویر قبلی یا before image) ←منشا مشکل عمل write

راه حل مشکل دوم:بدانیم از کجا خوانده(خواندن از)←منشا مشکل عمل read

***پدیده انتشار سقط مطلوب نیست زیرا باعث اتلاف منابع می شود.

پدیده انتشار سقط(Abort Cascade)

تعریف: با سقط(Abort) یک تراکنش ممکن است یک یا چند تراکنش دیگر هم که به آن وابسته اند(مثلا داده ای را از آن خوانده باشند) نیزسقط می شوند این پدیده را انتشارسقط گویند.

- مثال:

- W1(x)R2(x)W2(x)R3(X)C2C3A1

با سقط تراکنش T1 تراکنش های وابسته به آن نیز سقط میشوند.

تعریف خواندن از

گوییم تراکنش Ti قلم داده x را از تراکنش Tj خوانده است اگر:

- ۱- قبل از خواندن Ti تراکنش Tj مقدارx را نوشته باشد.(شرط لازم)
- ۲- قبل از حواندن Tiتراکنش Tj سقط نکرده باشد.(شرط کافی)
- ۳- به ازای هر تراکنش مثل T_k ، اگر T_k پس از Tj روی x نوشت.قبل از خواندن Ti تراکنش T_k سقط شود.

انواع اجراها

- اجرای پی درپی (Ser) -
- اجرای ترمیم پذیر (RC) -
- اجرای پی در پی پذیر(SR) -
- اجرای اجتناب کننده از سقط (ACA) -
- اجرای محض(ST) -

اجرای پی در پی (Serial)

اجرایی که درآن تراکنش ها یکی پس از دیگری اجرا شوند(اتمیک)

$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow \ldots\ldots$

- این ساده ترین نوع اجراست .
- نتیجه این نوع اجرا همواره صحیح است.
- اجازه اجرای همروند تراکنش ها نمی دهد.
- توان عملیاتی سیستم در پایین حد خود است.

اجرای ترمیم پذیر(Recoverable)

تعریف

هر تراکنش زمانی commit کند که همه تراکنش هایی که از آنها خوانده است commit کرده باشند. به عبارت دیگر اگر تراکنش Ti از تراکنش Tj داده ای را خواند آنگاه ابتدا Cj و سپس Ci اتفاق افتد.

مثال:

W1(x)W2(y)R2(x)C1C2

اجرای اجتناب کننده از سقط
(Avoid Cascading Abort)

تعریف

هر تراکنش فقط از تراکنش های نهایی شده(Commited) بخواند.

*یعنی به تعویق انداختن Readها.

مثال:

W1(x)W2(y)C1R2(x)C2

اجرای پی درپی پذیر(SR)

تعریف

اجرای تراکنش ها به صورت همروند به طوری که نتیجه صحیح حاصل شود به عبارت دیگر معادل یک اجرای پی در پی باشد.

مثال:

R1(x)W1(x)W1(y)C1W2(z)R2(x)W2(y)C2

اجرای محض (Strict)

تعریف

هر تراکنش تنها قلم داده هایی را میخواند که اثر یک تراکنش نهایی شده باشد و تنها برروی قلم داده های می نویسد که اثر تراکنش های نهایی شده باشد.

*در این نوع اجرا پدیده انتشار سقط نداریم ولی همروندی به شدت کاهش می یابد.

مثال:

W1(x)W1(y) C1R2(x)W2(y)C2

تمرین: نوع اجرا ها را مشخص کنید

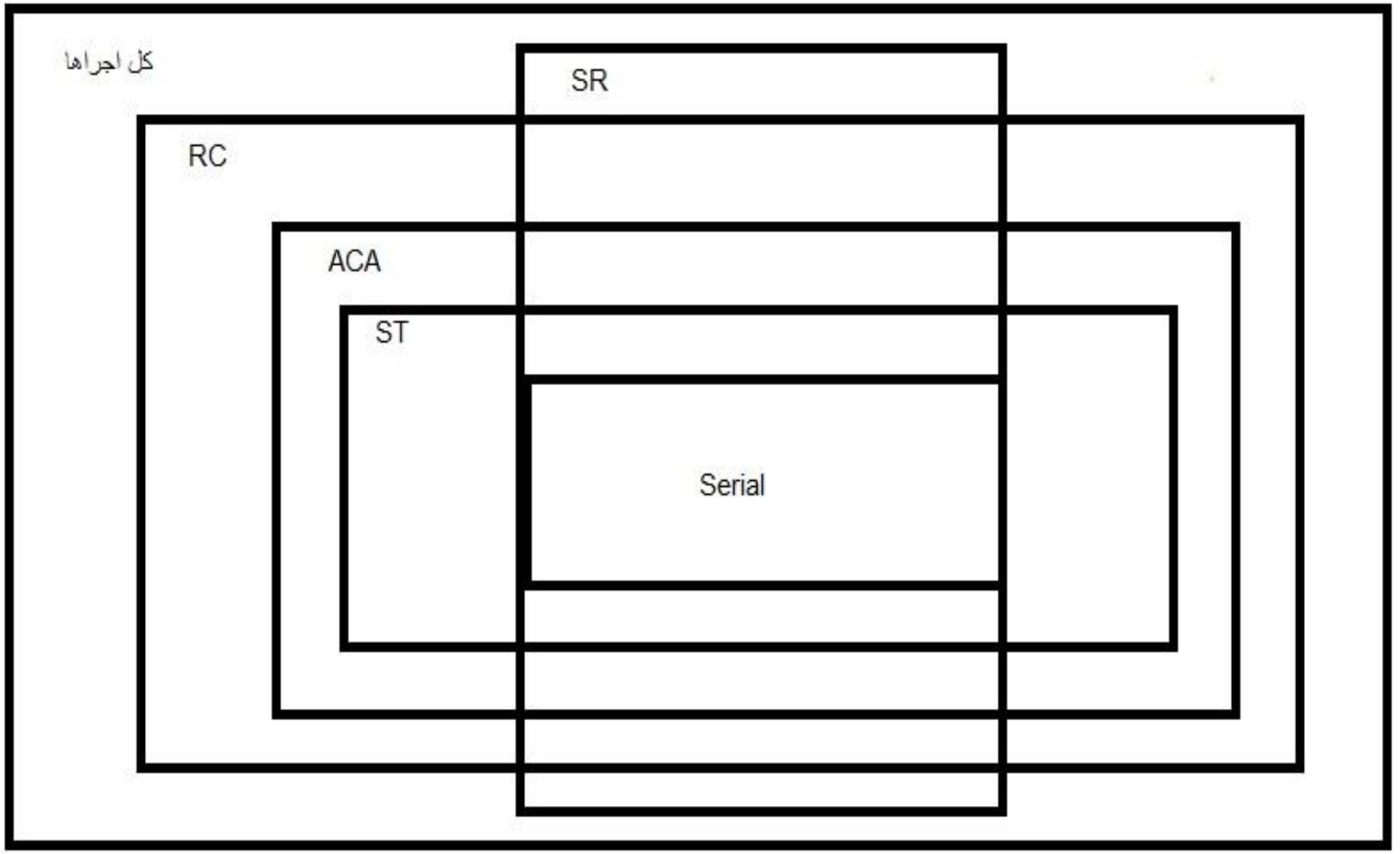
H1: W1(x)W1(y)R2(u)W2(x)R2(y)W2(y)C2W1(y)C1

H2:W1(x)W1(y)R2(u)W2(x)R2(y)W2(y) W1(y)C1C2

H3:W1(x)W1(y)R2(u)W2(x))W1(z)C1R2(y)W2(y)C2

H4:W1(x)W1(y)R2(u)W2(z)C1W2(x)R2(y)W2(y)C2

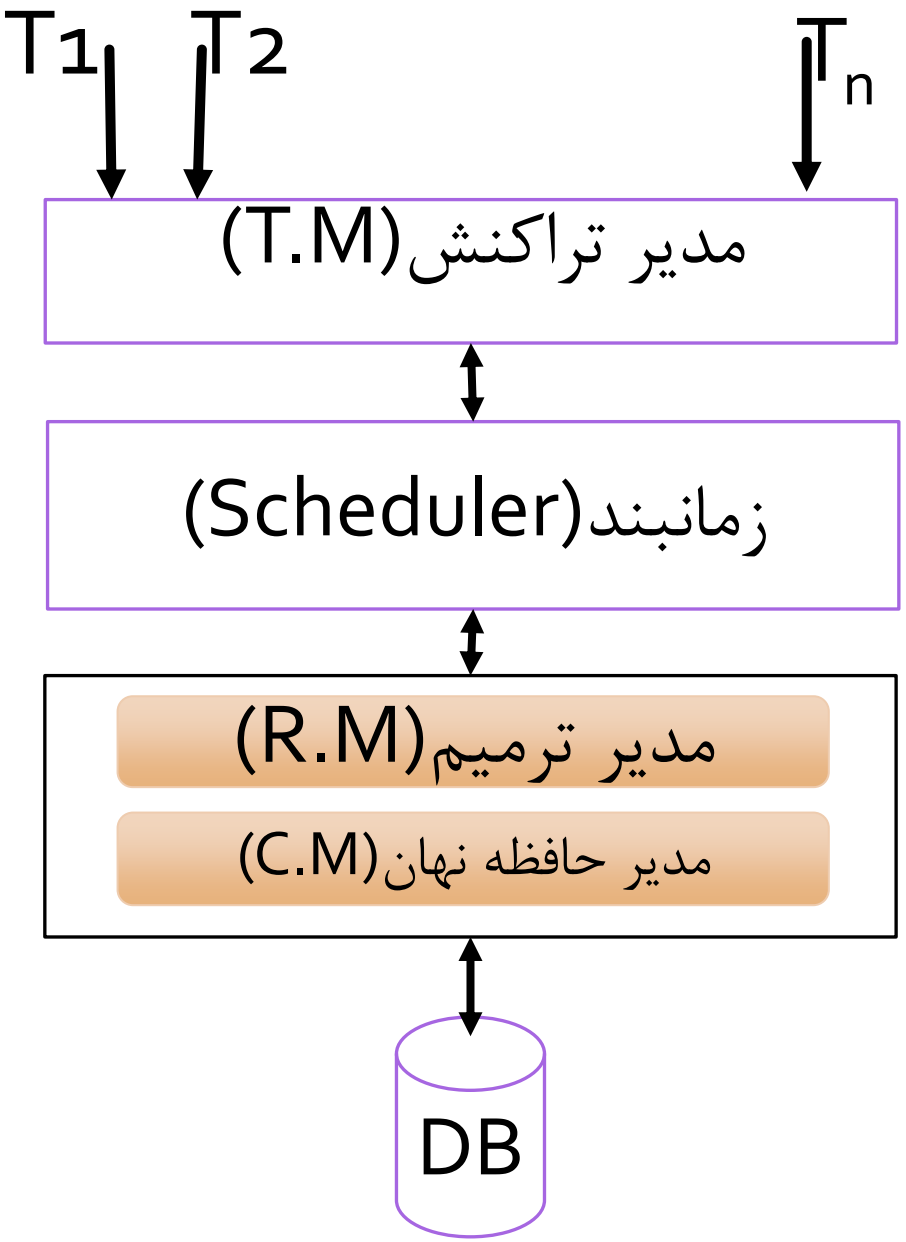
ارتباط بین اجراها



نتیجه

- – هر اجرای محض، ترمیم پذیر و ACA می باشد.
- – هر اجرای ACA ترمیم پذیر می باشد.
- – هر اجرای پی در پی ، اجرای محض و ترمیم پذیر وACA و پی در پی پذیر نیز می باشد.

معماری سیستم



زیر سیستم مدیریت تراکنش بخشی از DBMS است.

مدیر داده (D.M)

مدیر تراکنش

مدیر تراکنش تراکنش های برنامه های کاربردی را دریافت و هماهنگ میکند.
مدیر تراکنش با دیدن هر تراکنش، یک ID به آن تراکنش می دهد.
ممکن است لازم باشد مدیر تراکنش علاوه بر دستورات تراکنش بعضی از داده ها را نیز به زمانبند بفرستد.

زمانبند

هدف از این واحد تامین بیشترین همروندی تراکنش هاست.
زمانبند بر اساس یک استراتژی خاص خود تلاش میکند انواع اجراها را پیاده سازی کند وبدین منظور نحوه اجرای تودرتو (قاطی)تراکنش ها را فراهم میکند.
نکته: نمی توان ترتیب دستورات یک تراکنش را عوض کرد ولی می توان به منظور افزایش همروندی و داشتن اجراهای مختلف نحوه قاطی شدن دستورات چند تراکنش را با هم مدیریت کرد .

مدیر ترمیم

تضمین می کند که داده های پایگاه داده فاقد اثرات نهایی نشده است
و تمام نتایج موجود نهایی شده هستند .
مدیر ترمیم باید بتواند با انواع خطاها مقابله کند.

مدیر حافظه نهان(بافر)

حافظه نهان قسمتی از حافظه اصلی سیستم است که مسئول انتقال داده ها بین دیسک و حافظه اصلی است.

مدیرحافظه نهان دو دستور اجرا میکند :

- Fetch(x) : خواندن x از دیسک
- Flush(x): نوشتن x در دیسک

انواع خطاها

- خطاهای سیستمی: مانند قطع جریان برق. فقط تراکنش های در حال اجرا را مختل میکند که با استفاده از داده های موجود در پایگاه داده ها ترمیم می شوند.
- خطاهای رسانه ای : مانند خراب شدن هد یسک. با استفاده از Back up گیری با این خطا ها مقابله می شود. در صورت بروز این نوع خطا ها از دیسک پشتیبان استفاده میشود.

انواع زمانبندها

- بر اساس نحوه کارشان:

➢زمانبند مبتنی بر قفل : از روش قفل گذاری جهت کنترل همروندی استفاده می کنند.

➢زمانبند غیر قفلی: از روش های دیگری غیر از قفل گذاری برای کنترل همروندی استفاده میکنند.

- بر اساس ماهیت کارشان:

➢زمانبند مهاجم: از دو ابزار Run و Abort استفاده می کند.
وقتی دستوری از تراکنش میرسد فورا آن را اجرا یا کنسل میکند.

ادامه زمانبندها

زمانبند محافظه کار: این زمانبند با دقت بیشتری کار میکند . از ابزار Waitهم استفاده میکند تا تراکنش کمتری Abortکند.

این زمانبند با استفاده از ابزار Wait، اجرای دستورات را به تاخیر میاندازد تا بتواند آن توالی از دستورات که مدنظرش است را اجرا کند. بنابراین همروندی را افزایش داده و تراکنش های کمتری سقط می شوند.

ادامه زمانبندها

- نکته۱

زمانبند محافظه کار نیاز به داده های بیشتری برای زمانبندی دارد بنابراین ممکن است از مدیر تراکنش داده های بیشتری را درخواست کند.

- نکته۲

وقتی داده مشترک زیادی داریم محافظه کار بهتر عمل میکند.

- نکته ۳

هر زمانبند باید حداقل دو نوع اجرای پی در پی پذیر و ترمیم پذیر را پشتیبانی کند.

نظریه پی در پی پذیری

- گفته شد که اجرای پی در پی چند تراکنش درست است و جامعیت پایگاه را حفظ میکند.
- ممکن است اجرای همروند چند تراکنش درست نباشد و جامعیت پایگاه را خدشه دار کند.
- با استفاده از نظریه پی در پی پذیری (Serializable) می توانیم بفهمیم که یک اجرای همروند درست عمل میکند یا خیر.

روشهای پی در پی پذیری

- پی در پی پذیری برخوردی(CSR)
- پی در پی پذیری دیدی(VSR)

اعمال برخورددار

تعریف

دو عمل p_i و q_j برخوردارند اگر:

۱- بر روی داده مشترک کار کنند.

۲- حداقل یکی از آنها write باشد.

۳- این دو عملگر مربوط به دو تراکنش متمایز باشند.($i \neq j$)

جدول برخورد بین دستورات

	Ri(x)	**Wi(x)**
Rj(x)	سازگار	ناسازگار
Wj(x)	ناسازگار	ناسازگار

پی در پی پذیری در برخورد

تعریف:سابقه ای پی در پی پذیر برخوردی است که معادل با یک سابقه پی در پی باشد.

متناظر با پی در پی پذیری برخوردی ، هم ارزی برخوردی وجود دارد.

تعریف هم ارزی برخوردی دو سابقه H و H′ :

۱- هر دو بر روی مجموعه یکسانی از تراکنش ها تعریف شده باشند. (هر دستور در یکی هست در دیگری هم باشد.)

۲- در هر دو اعمال برخوردار به یک شکل مرتب شده باشند.

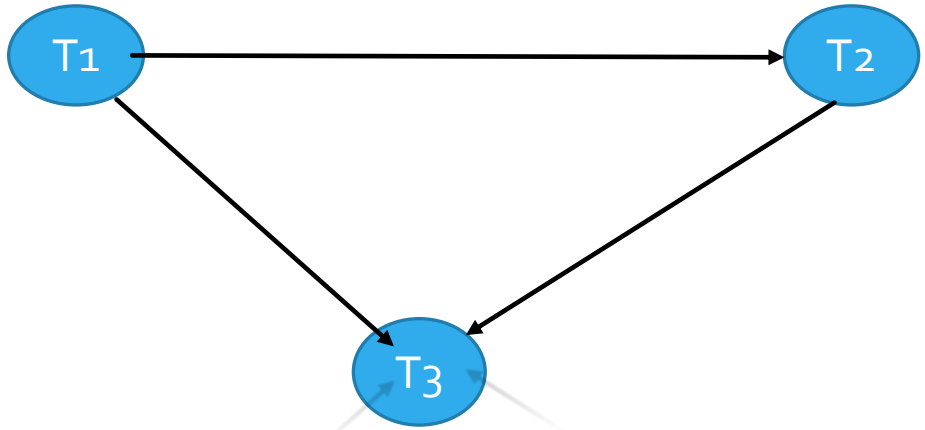
تشخیص پی در پی پذیری در برخورد

با استفاده از گراف پی در پی پذیری

- هر تراکنش یک راس گراف تشکیل میدهد.
- از راس Ti به راس Tj لبه ای ترسیم می شود هرگاه یک دستور برخوردار Tiقبل از Tjباشد.
- اگر در گراف حلقه نباشد، اجرای پی در پی پذیر است.

مثال

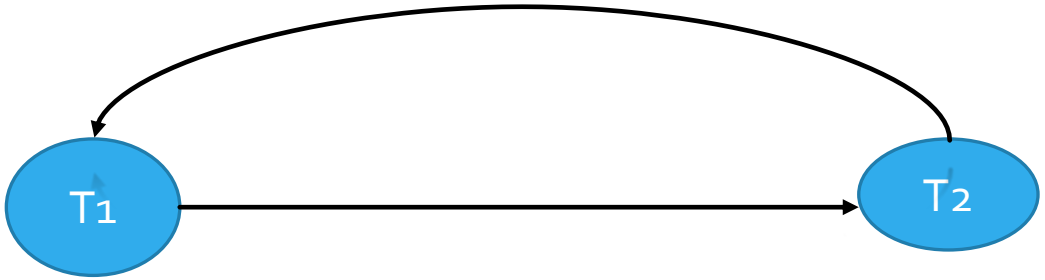
$$H= R_1(x)W_2(x)\ R_2(x)\ W_3(x)C_2C_1C_3$$



- در گراف حلقه نداریم پس سابقه معادل با سابقه $T_1<T_2<T_3$ است.
بنابراین پی در پی پذیر در برخورد است.

مثال

H:R2(x)W2(x)R1(x)R1(y)R2(y)W2(y)C1C2



در گراف حلقه داریم.

پی در پی پذیری در دید

متناظر با پی در پی پذیری دیدی ، هم ارزی دیدی وجود دارد.
تعریف هم ارزی دیدی دو سابقه H و H′
۱- هر دو بر روی مجموعه یکسانی از تراکنش ها تعریف شده باشند.
۲- اگر در سابقه H ، تراکنش Ti داده x را از تراکنش Tj خوانده باشد آنگاه در سابقه H′ هم تراکنشTi داده x را از Tj خوانده باشد.(هر دو از نوشتن های یکسانی خوانده باشند.)
۳- در هر دو سابقه Hو H′ ، به ازای هر قلم داده مانندx ، Wi(x) آخرین نوشتن باشد.

تشخیص پی در پی پذیری در دید

یادآوری:

گفتیم که دو سابقه H و H′ در صورتی هم ارز دیدی هستند که اولین خواندن ها و آخرین نوشتن های یکسان داشته باشند.

بنابراین :

- برای هر آیتم داده ای معین میکنیم که هر تراکنش برای اولین بار آن را از کجا خوانده است.
- برای هر ایتم داده ای معین میکنیم که آخرین نوشتن بر روی آن از جانب کدام تراکنش بوده است.

مثال

H:R1(a)R3(b)R2(a)W1(a)W1(c)C1W2(c)W2(d)C2W3(c)C3

- بررسی اولین خواندن ها:

T1 ← برای اولین بار a را از T0 خوانده است.(T0 تراکنشی است که در ازل اجرا شده)

T2 ← برای اولین بار a را از T0 خوانده است.

T3 ← برای اولین بار b را از T0 خوانده است.

- بررسی آخرین نوشتن ها:

T3 ← آخرین نوشتن را روی c انجام داده است.

T2 ← آخرین نوشتن را روی d انجام داده است.

T1 ← آخرین نوشتن را روی a انجام داده است.

ادامه مثال

حال اگر سابقه H′ را به صورت T1<T2<T3 در نظر بگیریم آنگاه داریم:

H′:R1(a)W1(a)W1(c)C1R2(a)W2(c)W2(d)C2R3(b)W3(c)C3

اولین خواندن ها :

T1 ← a را از T0 خوانده.

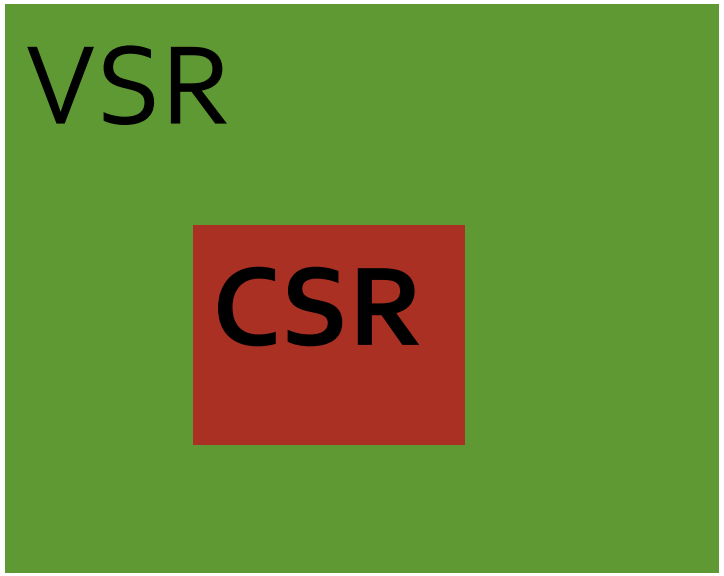
T2 ← a را از T1 خوانده.

بنابراین Hو H′ در خواندنها یکسان نیستند پس هم ارز در پی در پی دیدی نیستند.

ادامه مثال

حال اگر سابقه H’ را به صورت T2<T1<T3 در نظر بگیریم آنگاه دارای اولین خواندن ها و آخرین نوشتن های یکسانی هستند پس معادل با یک سابقه پی در پی می باشد بنابراین پی در پی پذیر دیدی است.

مقایسهCSR با VSR



VSR
CSR

پروتکل های کنترل همروندی

1- پروتکل های مبتنی بر قفل

۲- پروتکل های مبتنی بر مهر زمانی

پروتکل های مبتنی بر قفل(lock based)

این پروتکل ها کاربردی ترین روش کنترل همروندی می باشند.
هر گاه تراکنشی بخواهد برای خواندن یا نوشتن به داده ای دسترسی داشته باشد، ابتدا درخواست قفل مناسب با آن دستور را به واحدی به نام مدیر قفل(lock manager) می دهد.
مدیر قفل هر درخواست رسیده را با قفلهایی که احیانا توسط تراکنش های دیگر روی آن داده زده شده است مقایسه می کند چنانچه درخواست سازگار باشد اجابت می شود در غیر اینصورت تراکنش به انتظار می رود تا زمانی که قفل های زده شده آزاد گردند.

قفل دو حالتی (binary) :
در این حالت داده یا قفل است یا باز.
اشتراک داده ها وجود ندارد.
درخواست قفل گذاری در صورتی اجابت می شود که داده باز باشد.
قفل اشتراکی – انحصاری (shared- exclusive):
به منظور افزایش سطح همروندی دو نوع قفل اشتراکی (S) و انحصاری (X) تعریف می شود.
تراکنش جهت خواندن داده قفل S و جهت نوشتن داده قفل X درخواست میدهد.

جدول سازگاری قفل های S و X

$j \neq i$	S_j	X_j
S_i	سازگار	ناسازگار
X_i	ناسازگار	ناسازگار

چند نکته

❊هر تراکنش قبل از اجرای دستور read یا write باید درخواست قفل مربوطه را به مدیر قفل بدهد در صورتی که درخواست (با توجه به جدول سازگاری) اجابت شود، در این صورت می تواند دستور r یا w را اجرا کند.

❊چنانچه درخواست قفلی اجابت نشود، تراکنش به حالت انتظار(wait) میرود تا آن داده آزاد شود و حالت سازگازی پیش آید.

❊درخواست قفلs و x روی داده a برای تراکنش iبه ترتیب با $s_i(a)$ و $x_i(a)$ نمایش میدهیم و $u_i(a)$ هم جهت ازاد شدن قفل داده a را می دهد.

❊چند تراکنش می توانند بطور همزمان یک داده را قفل s کرده و همزمان بخوانند ولی چنانچه تراکنشی داده ای را از نوع x قفل کرده باشد ، دیگر هیچ تراکنشی نمی تواند هیچ نوع قفلی روی آن داده بگذارد.

مثال:

زمانبندی زیر را با قفل های s و x بنویسید.

H: $r_2(a)w_3(a)w_1(a)w_2(a)r_4(a)$

حل

$S_2(a)r_2(a)x_3(a)x_1(a)x_2(a)u_2(a)$ $w_3(a)u_3(a)c_3w_1(a)c_1u_1(a)$

$W2(a)u2(a)s_4(a)r_4(a)u_4(a)c_4$

تبدیل قفل

یک تراکنش باید در صورت لزوم قفل خود را تبدیل کند.
اگر تراکنشی روی داده ای قفل s گذاشته باشد و چند لحظه بعد بخواهد همان داده را بنویسد و یا بالعکس، میتواند از قانون تبدیل قفل استفاده کند.

افزایش درجه قفل(upgrade) : تبدیل قفل s به قفل x.
در صورتی که تراکنش های دیگر بر روی داده قفل s گذاشته باشند درخواست تبدیل قفل s به قفل x اجابت نمیشود.

کاهش درجه قفل(downgrade) : تبدیل قفل x به قفل s: همواره میتواند اجابت شود.
از مزایای کاهش درجه قفل افزایش درجه همروندی است.

**قفل انحصاری – اشتراکی چون به باز کردن قفل نمی پردازد واجازه میدهد که تراکنش بلافاصله پس از اتمام کارش قفل داده را باز کند بنابراین قادر به رفع مشکل بازیابی ناهمگام نمیشود.

برای رفع مشکل فوق در ادامه پروتکل های قفل گذاری معرفی میشوند.

پروتکل قفل دو مرحله ای (2PL)

این پروتکل علاوه بر قفل گذاری به قفل گشایی نیز می پردازد.
پروتکل 2pl تضمین میکند که زمانبندی پی در پی پذیر برخوردی باشد و هر سه مشکل همروندی را نیز حل کند.

مراحل پروتکل 2PL

۱- مرحله اول:مرحله رشد(growing)
در این مرحله تراکنش فقط میتواند قفل بگیرد و احتمالا با داده ها کار کند ولی نمی تواند قفل آزاد کند.
۲- مرحله دوم: مرحله عقب نشینی (shrinking)
در این مرحله تراکنش فقط میتواند قفل را آزاد کند و احتمالا با داده ها کار کند ولی نمی تواند قفل جدیدی روی هیچ داده ای بگیرد.

پروتکل B2PL

در این پروتکل تراکنش ها شروع به گرفتن قفل های مورد نیاز (در صورت توفیق در این امر)می کنند.
تراکنش هرگاه نیاز به انجام عمل read و write داشت همان لحظه درخواست قفل بدهد.
مشکل B2PL : وقتی تراکنش ها به صورت چرخشی به هم وابسته شوند بن بست ایجاد می شود.

پروتکل c2pl

- در این پروتکل هر تراکنش قبل از آنکه شروع به اجرای اولین دستور کند باید تمام قفلهای مورد نیازش را اخذ کند.و چنانچه موفق نشود حتی یکی از قفلهای مورد نیازش را اخذ کند باید تمام قفلهایی که ایجاد کرده است را آزاد کند.و دوباره در صف دریافت قفلهای مورد نظر بایستد.

- این پروتکل از بروز بن بست جلوگیری میکند.
همراه با کاهش همروندی است چون قوانین بیشتر و سخت گیرانه تر است.
هر چقدر قوانین کمتر باشد همروندی بیشتر است.

- مثال:

T1:w1(a)w1(b)
T2:r2(a)r2(b)

- H:x1(a)x1(b)w1(a)u1(a)s2(a)w1(b)u1(b)c1s2(b)r2(a)u2(a)r2(b)
u2(b)c2

پروتکل s2pl : (2pl محض- شدید)

- این پروتکل رایج ترین پروتکل قفل گذاری است–در این پروتکل باز کردن قفل نوشتن تا بعد از اتمام تراکنش (commit or Abort)به تعویق می افتد.اما قفل خواندن می تواند کمی زودتر باز شود.
- برای گرفتن قفل شرطی ندارد ولی قفل را زمانی آزاد می کند که نتیجه کار مشخص شده باشد یعنی بعد از اجرای commit or Abort

- مثال:
- T1=w1(a)w1(b)
- T2=r2(a)r2(b)

حل

- H:x1(a)w1(a)x1(b)w1(b)c1u1(a)u1(b)
  s2(a)r2(a)s2(b)r2(b)c2u2(a)u2(b)

مقایسه S2pl و c2pl

- S2pl مشکل سقط تسلسلی را حل می کند ولی بن بست دارد.
- C2pl بن بست ندارد اما مشکل سقط تسلسلی دارد.

پروتکل cs2pl

- برای قفل گذاری از پروتکل c2pl و برای قفل گشایی از پروتکل s2pl استفاده می کند.
- بدین ترتیب مشکل بن بست و سقط تسلسلی حل می شود.

- **مثال:**

- **H:x1(a)x1(b)w1(a)w1(b) c1/a1 u1(a)u1(b)s2(a)s2(b)r2(a)r2(b) c2/a2 u2(a)u2(b)**

پروتکل های مبتنی بر مهر زمانی

در این پروتکل هر تراکنش به محض ورود یک مهر زمانی تصاعدی تخصیص داده می شود.
مهر زمانی تراکنش Ti را با TS(Ti) نمایش میدهیم.
برای دو تراکنش Tiو Tj که Tj دیرتر وارد سیستم شده باشد TS(Ti) <TS(Tj).
پروتکل های مبتنی بر مهر زمانی تراکنش ها را به ترتیب مهر زمانی آنها به صورت پی در پی اجرا می کند .

به هر تراکنش یک مهر زمانی(TS(Ti)) وبه هر داده x دو مهر زمانی یکی مهر زمانی خواندن(R-TS(x)) و دیگری مهر زمانی نوشتن (W-TS(x))نسبت داده می شود.

قاعده مهر زمانی:
اگرPi(x) و Qj(x) دو عمل برخورددار باشند آنگاه زمانبند مهر زمانی، Pi(x) را پیش ازQj(x) اجرا میکند اگر و تنها اگر
TS(Ti)<TS(Tj).(به عبارت دیگر دستور تراکنش با مهر زمانی کوچکتر را زودتر از دستور تراکنش با مهر زمانی بزرگتر اجرا میکند)

مهر زمانی خواندن و مهر زمانی نوشتن

W-TS(X) مهر زمانی نوشتن داده X : برابر است با بزرگترین مهر زمانی تراکنشی که روی X به طور موفقیت آمیز نوشته است.

R-TS(X) مهر زمانی خواندن X: برابر است با بزرگترین مهر زمانی تراکنشی که به طور موفقیت آمیز X را خوانده است.

قواعد خواندن

فرض کنید تراکنش Ti شامل یک دستور read(x) است:
۱- اگر TS(Ti)< W-TS(X) باشد آنگاه تراکنش Ti داده ی را میخواند که انگار بعدا نوشته می شود . پس در این صورت با دستور خواندن تراکنش موافقت نمیشود و تراکنش رد می شود.

۲- اگر TS(Ti) = > W-TS(X) آنگاه دستور خواندن تراکنش Ti اجرا می شود و مهر زمانی خواندن داده x با ماکزیمم بین مهر زمانی تراکنش Ti و مهر زمانی خواندن x مقدار دهی می شود.

قواعد نوشتن

فرض کنید تراکنش Ti شامل یک دستور write(x) است:
۱- اگر TS(Ti)< R-TS(X) و یا TS(Ti)< W-TS(X) باشد آنگاه با دستور نوشتن تراکنش موافقت نمیشود و تراکنش رد می شود.

۲- در غیر اینصورت دستور نوشتن اجرا می شود و مهر زمانی نوشتن داده X نیز برابر با TS(Ti) مقدار دهی می شود.

با اعمال قواعد خواندن و نوشتن پروتکل های مبتنی بر مهر زمانی تضمین می کنند که دستورات W و R به ترتیب مهر زمانی اجرا شوند و زمانبندهای مربوطه پی در پی پذیر باشند.

دستورات دیر رسیده

اگر عمل برخورددداری از یک تراکنش با مهر زمانی کوچکتر بعد از یک عمل برخورددار از تراکنشی با مهر زمانی بیشتر رسید ، آن را یک دستور دیر رسیده گفته و بلافاصله عمل برخوردار با مهر زمانی کوچکتر رد می شود.

مثال : زمانبند زیر را با پروتکل های S2PL و مهر زمانی بنویسید.

H: r1(a)r2(b)r1(b)w1(b)r3(c)C3w2(a)

الف) با استفاده از پروتکل S2PL

S1(a)r1(a)s2(b)r2(b)s1(b)r1(b)x1(b)s3(c)r3(c)c3u3(c)x2(a) بن بست

انتظار انتظار

ب) با پروتکل مهر زمانی
در لحظه شروع داریم که:

H: r1(a)r2(b)r1(b)w1(b)r3(c)C3w2(a)

TS(T1)=1,TS(T2)=2,TS(T3)=3
W-TS(a)=W-TS(b)=W-TS(c)=0
R-TS(a)=R-TS(b)=R-TS(c)=0

دستور r1(a) پذیرفته می شود زیرا TS(T1)>= W-TS(a)
و پس از اجرای دستور R-TS(a)=1
بنابراین داریم:

r1(a)r2(b)r1(b)w1(b)r3(c)c3w2(a)

دستور دیر رسیده

مثال

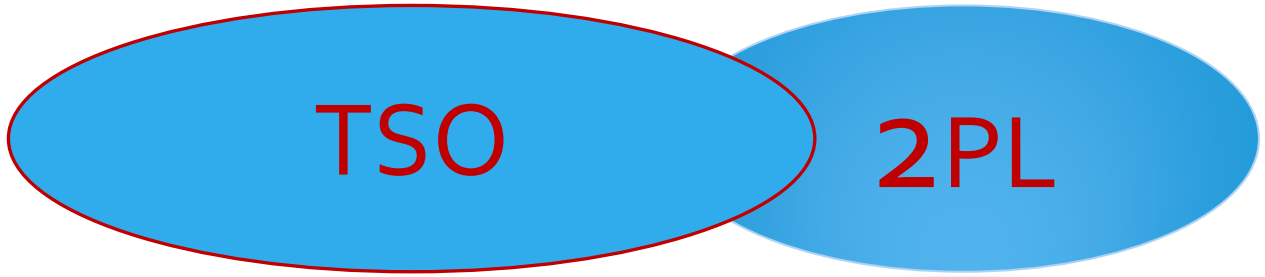
H: r1(a)r2(a)w2(a)r1(b)r2(b)w2(b)

زمانبند مهر زمانی پایه (BTSO)
زمانبند مهر زمانی محافظه کار(CTSO)

اگر رد شدن دستور دیر رسیده به معنای شروع مجدد تراکنش باشد آن را زمانبند مهر زمانی پایه گویند. در این صورت چون هیچ تراکنشی به انتظار نمیرود بنابراین بن بست رخ نمیدهد.

ولی چنانچه هر در خواست خواندن یا نوشتن را برای مدتی به تاخیر اندازد به امید انکه درخواست های برخورددار از تراکنش های با مهر زمانی کوچکتر از راه برسند و به عنوان درخواست دیر رسیده باعث Abort تراکنش خود نشوند ، زمانبند را محافظه کار گویند.

عملکرد پروتکل های 2PL و TSO



TSO می تواند حجم بیشتری از زمانبندهای همروند را به شکل پی در پی پذیر برنامه ریزی کند

ترمیم
(Recovery)

ترمیم

یعنی بازگرداندن پایگاه داده به وضع سازگاری که درست قبل از بروز خرابی داشت.

تعریف سازگاری(consistency): وضعیتی که در آن تمام محدودیت های جامعیتی اعمال شده باشد.

خرابی ⇨ اشتباه ⇨ نقص

واحد مدیریت ترمیم (recovery management) مسئول ترمیم است.

نکته: فرایند ترمیم هزینه دار است.

روش های ترمیم

۱- تکنیک پیشرس(WAL): رایج ترین تکنیک ترمیم می باشد. اساس این تکنیک استفاده از فایل LOG است. در این تکنیک هر تغییری که تراکنش انجام میدهد، قبل از ثبت در دیسک ، در فایل LOG ثبت و ضبط میگردد.

۲- تکنیک ایجاد صفحه سایه (Shadowing): این تکنیک به دو صورت ، صفحه سایه پیش تصویر و صفحه سایه پسا تصویر وجود دارد. که تکنیک دومی رایج تر است.

تکنیک پیشرس (WAL)

مدیر ترمیم برای مقابله با خرابی دو استراتژی دارد:

۱- استراتژی Redoing : در این حالت مدیر ترمیم براساس اطلاعات فایل log، پایگاه داده ها را به وضع قبلی اش برمی گرداند و نتایج تراکنش های تثبیت شده را در پایگاه داده ها اعمال میکند.

۲- استراتژی Undoing: در این حالت مدیر ترمیم تغییراتی که باعث ناسازگارشدن پایگاه داده ها میشود را از پایگاه پاک میکند. در این استراتزی نیازی به نسخه پشتیبان نیست بلکه با استفاده از اطلاعاتی که در فایل LOG قرار دارد ، فرایند ترمیم را انجام میدهد.

انواع خرابی :

۱- خرابی رسانه ای(media failure): سبب ایجاد خرابی در تمام یا قسمتی از حافظه پایدار (دیسک) می شود و حداقل روی تراکنش هایی که در حال استفاده از داده ها هستند تاثیر می گذارد. مانند خرابی دیسک

۲- خرابی سیستمی(system failure) : سبب می شود حداقل یک یا تمام تراکنش های فعال (تراکنش های در حال اجرا) در سیستم آسیب ببینند ولی داده های ذخیره شده در حافظه پایدار (دیسک) دچار خرابی نشوند.مانند قطع جریان برق در حافظه اصلی

عوامل خرابی :

- اشتباه برنامه نویسی
- اشتباه اپراتور : تعویض دیسک
- اشتباه سیستمی(اشتباه سیستم عامل یا DBMD) : مانندرخداد بن بست
- اشتباه سخت افزاری : نقص در حافظه اصلی
- اشتباه عملیاتی : از بین رفتن نسخه پشتیبان

...

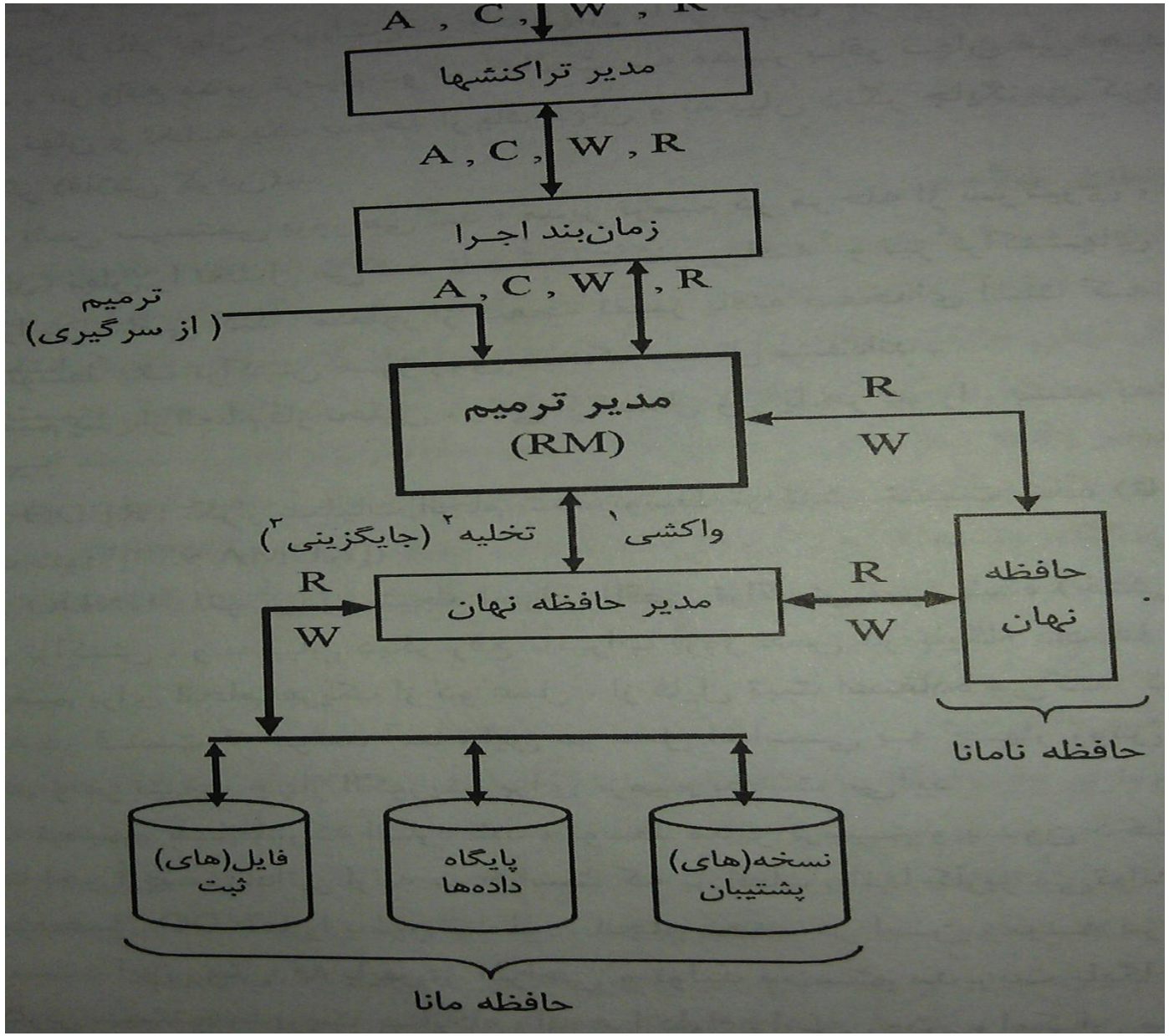
زیر سیتم مدیریت ترمیم :

بخشی از نرم افزار DBMS است که خود از اجزایی مانند: مدیر ترمیم ، مدیر حافظه نهان و... تشکیل شده است.
دشوارترین مرحله در طراحی DBMS ، طراحی واحد مدیریت ترمیم است.
بسامد انواع خرابی ها در یک سیستم یکسان نیست:

- بعضی از خرابی ها مانند خرابی تراکنش ممکن است در هر دقیقه چندین بار رخ دهد.
- خرابی سیستمی ممکن است چندین بار در ماه رخ دهد.
- خرابی رسانه ای ممکن است تنها چندین بار در سال رخ دهد.

به هر حال مدیر ترمیم باید انواع خطاها را به طور موثری مدیریت کند.

تصویر زیر سیستم ترمیم



تشریح عملکرد هر یک از بخش ای واحد مدیریت ترمیم :

- مدیر ترمیم : با واحدهای زمانبند و مدیر حافظه نهان در ارتباط است.
زمان بند دستورات نوشتن ، خواندن، تثبیت تراکنش و طرد تراکنش را به مدیر ترمیم ارسال میکند.
مدیر ترمیم نیز به نوبه خود دو درخواست به مدیر حافظه نهان ارسال میکند.
الف) عملیات fetch : واکشی یک صفحه از دیسک به حافظه نهان به منظور نوشتن یا خواندن داده ای از آن صفحه
ب) درخواست flush : درخواست انتقال(تخلیه) یک صفحه (یک بلاک) از حافظه نهان به حافظه دیسک .

- فایل ثبت تراکنش ها(log file) : فایلی است که حاوی اطلاعاتی در مورد چگونگی اجرای هر تراکنش و عملیاتی که تراکنش انجام میدهد. این فایل در حافظه پایدار نگهداری می شود در هنگام رخداد خرابی به فایل log آسیب نمیرسد . برای حصول اطمینان چند نسخه از فایل log ایجاد می کنند.

نکته ۱: قبل از انتقال یک صفحه از دیسک به حافظه نهان و یا بالعکس باید آن صفحه در فابل log ذخیره شود و همزمان با تغییر داده های یک صفحه در حافظه نهان این تغییرات در همان صفحه موجود در فایل log نیز اعمال می شود.

نکته ۲: در فواصل زمانی معین ، محتوی فایل log با دیسک (پایگاه داده ها) یکسان سازی می شود و اطلاعات یکسان سازی شده به بخش آرشیو فایل log منتقل میشود. به این عملیات نقطه بازرسی گویند.

عملیات مدیر ترمیم

عملیات مدیر ترمیم در دو فاز صورت می گیرد:

۱- فاز تحلیل : وقتی نقصی رخ میدهد مدیر ترمیم قبل از هر اقدامی ، وضع بافرهای حافظه نهان را بررسی و تحلیل میکند تا صفحات تغیر یافته(dirty page) و نیز تراکنش های فعال(تراکنش های در حال اجرا) در لحظه خرابی را بازشناسایی کند.(منظور از صفحات تغییر یافته : صفحاتی است که تغییرات اعمال شده در آن توسط تراکنش هنوز به دیسک منعکس نشده است.)

۲- فاز اقدام : بسته به استراتژی الگوریتم ترمیم یکی از اقدامات زیر انجام می گیرد.

الف) عمل redo:با شناسایی تراکنش های تثبیت شده، چنانچه نتایج آنها به پایگاه داده ها منتقل نشده ، نتایج تراکنش های تثبیت شده را از فایل LOG به دیسک منتقل میکنیم.

ب) عمل undo: با شناسایی تراکنش های نیمه کاره و تثبیت نشده ، چنانچه اثرات آنها به پایگاه داده ه منعکس شده است از پایگاه داده پاک شوند.

هر دو عملیات redo و undo به کمک فایل log صورت میگیرد.

روشهای تخلیه بافر حافظه نهان(flush):

چگونگی رفتار مدیر ترمیم بستگی به چگونگی تخلیه صفحات موجود در حافظه نهان دارد.
وقتی سیستم دستور نوشتن را اجرا می کند ، برای انتقال نتیجه عمل نوشتن به دیسک ۲ تکنیک وجود دارد:

۱- تکنیک بهنگام سازی با تاخیر(defered update) ۲- تکنیک بهنگام سازی فوری(immediate update)

بهنگام سازی با تاخیر

در این روش انتقال صفحات تغییر یافته توسط یک تراکنش فقط بعد از آنکه آن تراکنش تثبیت شد به دیسک صورت می گیرد.چون ممکن است ئر زمان انتقال صفحات مورد نظر سیستم دچار خرابی شود ، در حین عمل تثبیت تراکنش ، نسخه ای از داده تغییر یافته در فایل log نیز ذخیره میگردد.

در این حالت چنانچه تراکنش قبل از رسیدن به نقطه تثبیت دچار خرابی(abort) شود، چون هیچ تغییری در پایگاه داده ها انجام نشده است نیاز به عمل undo نیست ولی ممکن است نیاز به عمل redo باشد.تا به کمک فایل log اثرات تراکنش های تثبیت شده به دیسک منتقل نماید.

بهنگام سازی فوری:

در این روش نتیجه عملیات بهنگام سازی داده، بلافاصله پس از تغییر داده و قبل از تثبیت تراکنش در دیسک نوشته میشود.(البته همزمان در فایل log هم این تغییرات ذخیره میگردد).
در این حالت چنانچه تراکنش abort شد با کمک فایل log هر تغییری که این تراکنش روی داده ها انجام داده است بازگردانده و از پایگاه دا ه ها پاک می شود. بنابراین نیاز به عمل undo داریم.

ایجاد نقطه بازرسی(checkpoint)

عبارتست از نوشتن اطلاعات فایل log در حافظه پایدار در فواصل زمانی مشخص و منظم، به منظور کاهش تعداد رکوردهایی که مدیر ترمیم باید در فایل log بررسی کند.

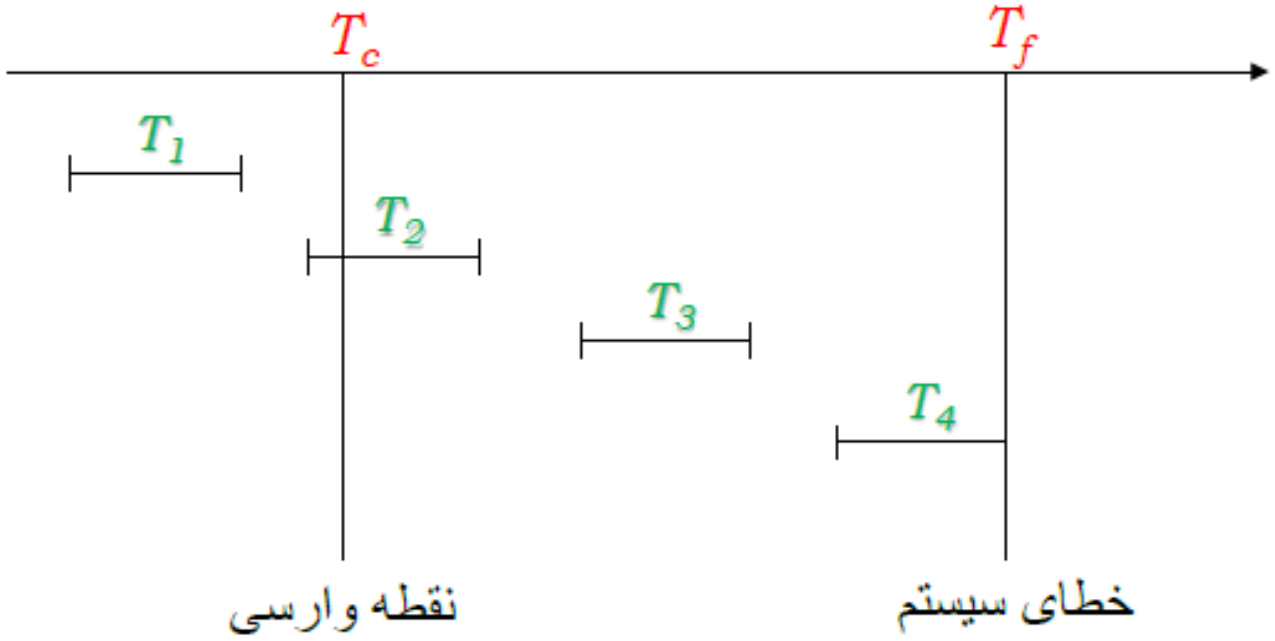
در این فرایند سیستم کارهای زیر را انجام میدهد:
۱- اجرای تمام تراکنش ها موقتا متوقف میشود.
۲- محتوای تمام بافرهایی که تغییر یافته اند(صفحات تغیر یافته) تخلیه اجباری میشوند.
۳- یک رکورد <checkpoint> در فایل log نوشته میشود.
۴- اجرای تراکنش ها از سر گرفته میشود.

وقتی ترمیم از سر گرفته میشود، سیستم با پویش وارون فایل log تا آخرین نقطه بازرسی پیش میرود . زیرا تا قبل از این نقطه نتیجه تمام تراکنش های تثبیت شده در پایگاه داده منعکس شده است.

مثال:

در نمودار زیر ، چگونگی اجرای همروند تراکنش ها و وضعیت هر یک در لحظه بروز خرابی دیده میشود.

رفتار مدیر ترمیم با تراکنش ها چگونه است؟



>>تراکنش T1را می توان نادیده گرفت .

>>تراکنش های T2و T3بایستی ، Redoشوند.

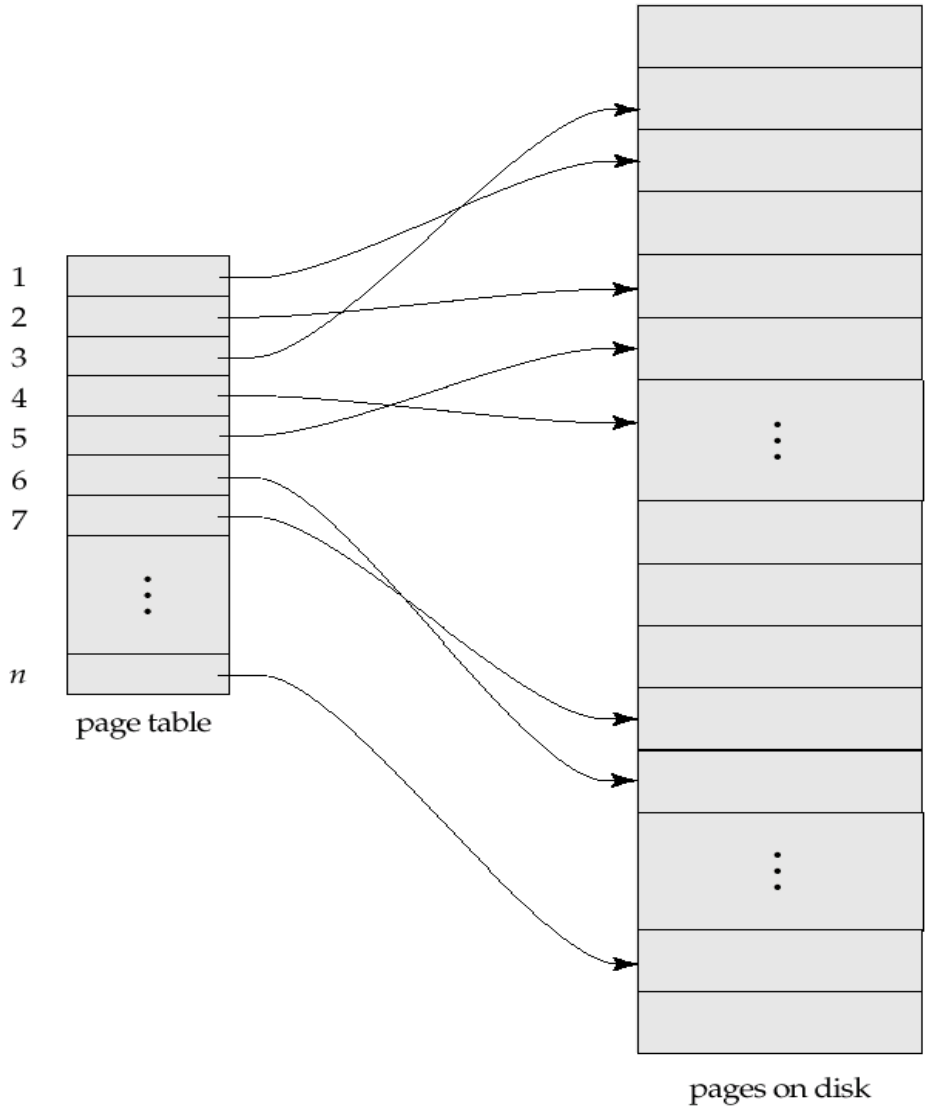
>> تراکنش T4بایستی ، Undoشود .

تکنیک ایجاد صفحه سایه

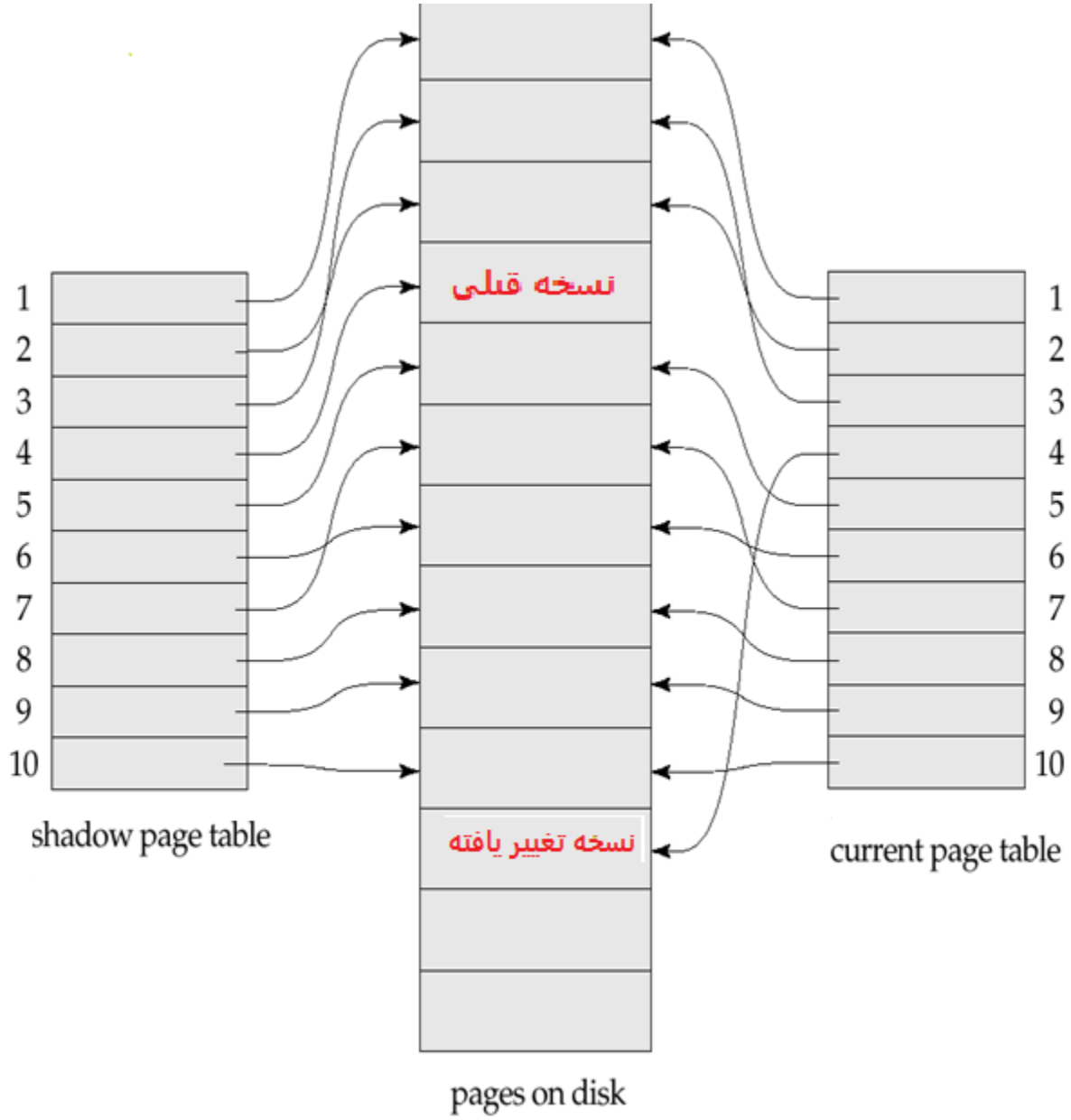
۱- صفحه سایه پیش تصویر
۲- صفحه سایه پسا تصویر

تکنیک صفحه سایه پسا تصویری

پایگاه داده از تعدادی صفحه با اندازه ثابت تشکیل شده است . سیستم یک جدول راهنما برای مدیریت این صفحات ایجاد میکند. مدخل iام این جدول راهنما به صفحه iام پایگاه داده بر روی دیسک اشاره می کند. وقتی تراکنش شروع به اجرا میکند ، جدول راهنما در یک جدول راهنمای دیگر به نام " جدول راهنمای سایه" کپی می شود.
راهنمای سایه در اثنایی که جدول راهنما مورد استفاده تراکنش قرار می گیرد روی دیسک نگهداری می شود و در اثنای اجرای تراکنش ، جدول راهنمای سایه تغییر نمی کند.
وقتی یک عمل نوشتن انجام می شود ، نسخه جدیدی از صفحه تغییر یافته ، ایجاد میگردد ولی در نسخه قبلی آن صفحه جایگزین نمی شود بلکه نسجه جدید ، در جای دیگری نوشته میشود(بهنگام سازی برون جا)



جدول راهنما



جداول سایه و صفحه فعلی پس از نوشتن صفحه ۴

تکنیک صفحه سایه پیش تصویری

بر خلاف تکنیک قبلی ، صفحه بهنگام درامده همیشه در جای قبلی اش نوشته می شود (بهنگام سازی درجا) و نسخه قبلی در جایی دیگر از پایگاه داده ها مثلا در انتهای دیسک نگهداری می شود. در این تکنیک قبل از انکه صفحه ای بهنگام شود و در جای قبلی اش باز نوشته شود ، پیش تصویر در مکانی نوشته میشود.