درس5
پایگاه داده پیشرفته

کنترل همروندی(1)

کنترل همروندی در پایگاه داده

پروتکلهای عملی برای کنترل همروندی در دو دسته قرار می گیرند

1- پروتکل های مبتنی بر قفل

2- پروتکل های مبتنی بر برچسب(مهر) زمانی

پروتکل قفل گذاری

- یکی از پیش نیازهای اصلی برای ترتیب پذیری، تضمین انحصار متقابل است.
- به عبارت دیگر یک راه اطمینان از ترتیب پذیری این است که عناصر داده را به صورت انحصار متقابل مورد دسترسی قرار دهیم.
- یعنی وقتی تراکنشی به مورد داده ای دسترسی پیدا می کند تراکنش دیگری نمی تواند مورد داده ای را تغییر دهد.
- **قفل**. متغیری که به یک مورد داده ای نسبت داده شده و وضعیت آن را درباره اعمال قابل اجرا روی آن شرح می دهد.
- **هدف**: ایجاد برنامه ترتیب پذیر از برنامه ای که جابجا کردن دستورات آن به منظور ترتیبی شدن امکان پذیر نیست.

پروتکل قفل گذاری

- حجم اطلاعات قفل شده:
  - قفل فایل(جدول)
    - در این حالت فایلی که اطلاعات آن مورد استفاده تراکنش است قفل می شود.
      - کارایی پایین
      - در سیستم های قدیمی
  - قفل رکورد
    - در این حالت فقط رکوردی که مورد استفاده تراکنش است قفل می شود.
      - کارایی بالا

پروتکل قفل گذاری

داده ها به دو صورت می توانند قفل شوند:

- قفل انحصاری (X).
  - این قفل موقعی استفاده می شود که رکورد بازیابی شده ممکن است تغییر هم بکند. دستور Lock_X، عمل قفل گذاری انحصاری را انجام می دهد.
- قفل اشتراکی (S).
  - این قفل موقعی استفاده می شود که رکورد بازیابی شده اصلاح نمی شود به عبارت دیگر تراکنش فقط می خواهد رکورد را بخواند اما نمی تواند بنویسد.

پروتکل قفل گذاری (ادامه)

- ماتریس سازگاری _ قفل

	S	X
S	true	false
X	false	false

- یک تراکنش زمانی می تواند به یک داده قفل اعطا کند که این قفل با قفل هایی که در حال حاضر روی داده وجود دارد، سازگار باشد.
- بر روی یک داده، هر تعداد تراکنش می توانند قفل اشتراکی بگذارند.
  - ولی اگر تراکنشی به یک داده قفل انحصاری اعطا کند، هیچ تراکنش دیگری نمی تواند روی آن قفل بگذارد.
- اگر یک قفلی اجازه اعطا پیدا کند، درخواست تراکنش باید معلق بماند تا زمانی که قفل های ناسازگار اعطا شده روی داده توسط سایرتراکنش، آزاد شوند. سپس قفل اعطا می شود.

مثالی از یک تراکنش درحال قفل گذاری

T_2: lock-S(*A*);
read (*A*);
unlock(*A*);
lock-S(*B*);
read (*B*);
unlock(*B*);
display(*A*+*B*)

- استفاده از قفل گذاری به روش بالا، به خودی خود ترتیب پذیری اجرای تراکنش ها را تضمین نمی کند .
- اگر مقدار A و B در میانه خواندن A و B تغییر کنند، جمع آن ها عدد اشتباهی خواهد بود.
- یک پروتکل قفل گذاری، سلسله ای از قوانین هستند که در هنگام قفل گذاری یا آزادکردن آن ها توسط تمام تراکنش ها رعایت می شوند. پروتکل قفل گذاری برخی طرح های ممکن را محدود می کند.

تعجیل در قفل یا تاخیر در آزاد سازی

- اگر رکوردی زودتر از موعد مقرر قفل شود یا بلافاصله پس از اتمام عملیات روی آن آزاد نشود باعث اختلال در اجرای تراکنش هایی دیگری که نیاز به آن رکورد دارند می شود
- دراینصورت تراکنش های دیگر که نیاز به رکورد قفل شده دارند بایستی بی مورد منتظربمانند. این کار باعث کندی سیستم می شود.
- با توجه به شرایط لازم برای صحت عملیات روی پایگاه داده اقدام به قفل یا آزاد سازی می شود.

مشکل بن بست

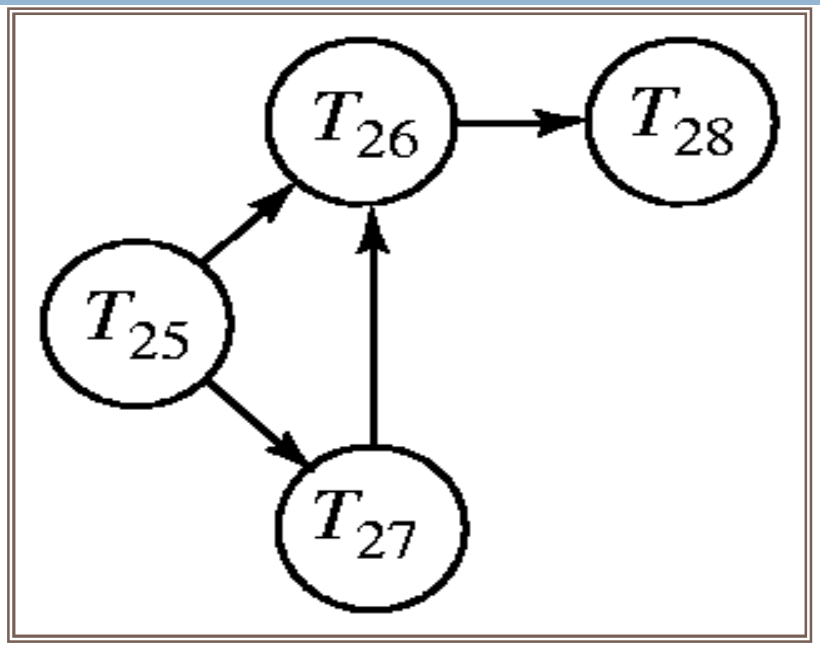
- یکی دیگر از مواردی که در اجرای همروند تراکنش ها اشکال ایجاد می کند ایجاد بن بست است.
- T3 عملیات روی B را شروع کرده ولی هنوز آن را آزاد نکرده است. از طرفی اجرای T4 شروع شده و A را قفل کرده و در صدد قفل رکورد B است ولی باید منتظر بماند تا B آزاد شود.
- از طرفی T3 درخواست قفل A را کرده و باید منتظر بماند.

T_3	T_4
lock-X(B)	
read(B)	
$B := B - 50$	
write(B)	
	lock-S(A)
	read(A)
	lock-S(B)
lock-X(A)	

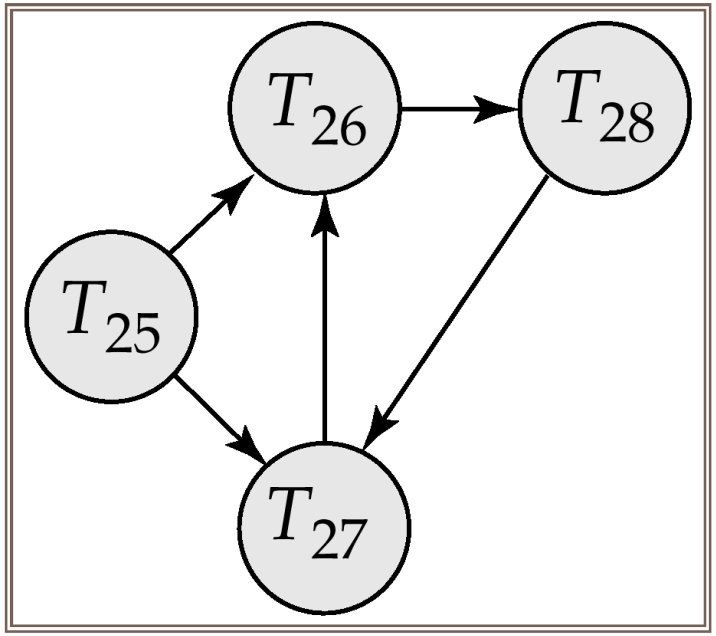
مشکل بن بست

- به حالتی که یک تراکنش در انتظار آزاد شدن رکورد مورد نیاز توسط تراکنش دیگری و بالعکس باشد بن بست گفته می شود.
- برای کشف بن بست از نمودار برداری wait-for استفاده می شود.
- در این نمودار گره ها بیان گر تراکنش های در حال اجرا بوده و بردار بین دو گره نشانگر نیاز گره مبدا به رکوردی است که توسط مقصد قفل شده است.
- اگر در نمودار W.F یک برنامه بردارها تشکیل دور بدهند در اینصورت در آن برنامه بن بست وجود دارد.

کشف بن بست (ادامه)



گراف Wait-for بدون حلقه



گراف Wait-for با حلقه

بن بست (ادامه)

- متد های رفع بن بست :
  - اجرای یکی از تراکنش ها توسط سیستم معلق شده تا قفل مورد استفاده آن نیز آزاد شود.
  - هر تراکنش در زمان شروع عملیات تمامی رکوردهای مورد نیاز خود را قفل کند.
- احتمال گرسنگی (Starvation) نیز در صورت طراحی بد سیستم مدیریت کنترل همروندی وجود دارد.
- این مشکل هنگامی بروز می کند که یک تراکنش برای مدت نامحدودی نتواند به حالت اجرا در آید در حالیکه تراکنش های دیگر مرتبا اولویت اجرا را به تراکنش های دیگر می دهد و تراکنش محروم نمی تواند قفل مورد درخواستش را از سیستم دریافت کند و این محرومیت مرتبا تکرار شده و ادامه می یابد.

مثال از قحطی (محرومیت)

- برای مثال:یک تراکنش منتظر یک X-lock روی یک داده باشد، در حالی که از طرف دیگر، مجموعه ای از تراکنش ها در حال درخواست دادن و اعطای قفل S-lock روی همان داده باشند.
- فرض شود تراکنش T2 یک قفل اشتراکی روی D دارد و تراکنش T1 درخواست یک قفل انحصاری روی D بکند.
- T1 باید منتظر بماند تا T2 قفل روی D را بردارد. در حین انتظار T1، تراکنش دیگری مانند T3 در خواست قفل اشتراکی روی Dمی دهدو سیستم این قفل را به T3می دهد.
- حال اگر در این لحظه T2 قفل روی D را بردارد باز هم T1 نمی تواند قفل انحصاری روی D رادریافت کند. حتی ممکن است در این وضعیت تراکنش دیگری مانند T4 در خواست در خواست قفل اشتراکی روی D بدهدو...

روش های بیشتر برای پیش گیری از بن بست

- طرح های زیر از برچسب زمانی برای پیش گیری از بن بست استفاده می کنند:
- طرح Wait-Die — غیرانحصاری
  - تراکنش های قدیمی منتظر تراکنش های جوانتر برای آزادسازی داده می مانند. تراکنشهای‌جوان(برچسب زمانی آنها بزرگتراز بقیه است)هیچ گاه منتظر قدیمی‌ترها نمی شوند. به جایش برگشت می خورند.
  - یک تراکنش ممکن است بارها قبل از دسترسی به داده مورد نیاز خودش بمیرد.
- طرح Wound-Wait — انحصاری
  - تراکنش‌های‌قدیمی‌تراکنش‌های‌جدیدتررا (از طریق اجبار به برگشت خوردن) زخمی می کنند. تراکنش های جوان تر ممکن است در انتظار قدیمی ترها بمانند. ممکن است تعداد برگشت های کمتری نسبت به Wait-Die داشته باشد.

پیش گیری از بن بست (ادامه)

- در هر دو طرح Wait-Die و Wound-Wait تراکنش برگشت خورده از روی برچسب زمانی اولیه اش مجددا شروع می شود. بنابراین تراکنش های قدیمی تر ارجحیت بیشتری به نسبت جدیدترها دارند پس گرسنگی اتفاق نمی افتد.
- طرح مبتنی بر برچسب زمانی
  - یک تراکنش در یک مدت زمان خاص برای قفل منتظر می شود. پس از آن مدت زمان انتظار پایان یافته و تراکنش برگشت می خورد.
  - بنابراین برگشت اتفاق می افتد.
  - با وجود پیاده سازی آسان ولی احتمال گرسنگی وجود دارد. هم چنین تخمین میزان مدت زمانی مناسب دشوار است.

1- پروتکل قفل گذاری دو مرحله ای2PL

- این پروتکلی است که برنامه های ترتیب پذیری در برخورد (Conflict-Serializable) را تضمین می کند. این پروتکل بیان می کند هر تراکنش باید درخواست های قفل کردن و آزاد سازی را در دو مرحله بدهد.
- مرحله 1: رشد(اوج) (Grown phase)
  - تراکنش ها اجازه دارند قفل گذاری کنند.
  - تراکنش ها اجازه ندارند قفل ها را باز کنند.
- فاز 2: فاز تحلیل(افول) (Shrinking phase)
  - تراکنش ها اجازه دارند قفل ها را باز کنند.
  - تراکنش ها اجازه ندارند قفل گذاری کنند.

پروتکل قفل گذاری دو مرحله ای

- در ابتدا یک تراکنش در مرحله رشد قرار می گیرد.
  - تراکنش شروع به قفل رکوردهای مورد نظرش می کند.
  - به محض این که یک تراکنش یک قفل را آزاد می کند وارد مرحله تحلیل می شود.
  - نقطه ای که تراکنش آخرین قفل را انجام می دهد نقطه قفل Lock-point گویند.

پروتکل قفل گذاری دو مرحله ای

- قفل کردن دو مرحله ای رهایی از بن بست را تضمین نمی کند.
- مثال:

Lock-X(B)	
Read(B)	
B=B-50	
Write(B)	
	Lock-S(A)
	Read(A)
	Lock-S(B)
Lock-X(A)	

برگشت های پی درپی
cascading rollback

- در قفل گذاری دومرحله ای برگشت های پی درپی امکان پذیر است.

T5	T6	T7
Lock-X(A) Read(A) Lock-S(B) Read(B) Write(A) Unlock(A)		
	Lock-X(A) Read(A) Write(A) Unlock(A)	
Fail		Lock_S(A) Read(A)

قفل گذاری دو مرحله ای اکید S2PL

- برای جلوگیری از برگشت انتشاری، پروتکلی به نام (اکید) (Strict two-phase locking) قفل گذاری دومرحله ای قوی تعریف شده است.
- در این جا هر تراکنش باید تمام قفل های انحصاری را تا انجام کار نگه دارد.
- این نیاز تضمین می کند که هر داده نوشته شده توسط تراکنشی که به پایان نرسیده است تا زمان انجام شدن(commit) به صورت انحصاری قفل باشد که این کار باعث می شوداز خواندن داده توسط تراکنش های دیگرجلوگیری شود.

قفل گذاری دو مرحله ای سخت گیرانه
Rigorous two-phase locking (R2PL)(SS2PL)

- در این روش تمام قفل ها اعم از اشتراکی / انحصاری تا پایان کار نگه داشته می شوند. یعنی تا تراکنش انجام و تثبیت نشود هیچ یک از قفل های اشتراکی / انحصاری باز نخواهند شد.
- در این روش تراکنش ها می توانند به همان ترتیبی که انجام و تثبیت می شوند، ترتیب بگیرند.

قفل گذاری دو مرحله ای محافظه کارانه
Conservative 2PL

- در این روش تراکنش ابتدا باید تمام قفل های مورد نیازش را اخذ کند و سپس اجرا شروع می شود. یعنی تا وقتی تمام قفل های مورد نیاز اخذ نشود اجرا شروع نمی شود.
- عیب :
  - مارا ملزم می کند قبل از اجرای تراکنش بدانیم به چند قفل و چه قفل هایی نیاز داریم.
  - درجه همروندی را پایین می‌آورد. زیرا باید صبرکند تا به تعداد قفل های مورد نیاز بگیرد و سپس شروع به اجرا کند که این باعث تاخیر می شود.
- مزیت :
  - بن بست رخ نخواهد داد. زیرا رقابتی در دریافت قفل ها وجود ندارد.

قفل گذاری دو مرحله ای همراه با تبدیل قفل

- در برخی مواقع با تغییر در قفل می توان همروندی را افزایش داد.
- مثال. فرض کنیم تراکنش های T1,T2 را بصورت زیر داریم.

اگر پروتکل 2PL استفاده شود آنگاه T1 باید a1 را در مود X قفل کند لذا اجرای همروند تراکنش ها به اجرای ترتیب پذیر شبیه می شود. به هر حال T1 به قفل انحصاری روی a1 تنها در پایان اجرا نیاز دارد.

READ(a_1)
READ(a_2)
.
.
READ(a_n)
Write(a_1)

READ(a_1)
READ(a_2)
Display(a_1+a_1)

- اگر T1 ابتدا a1 را در مود اشتراکی قفل کند و سپس قفل را
- به انحصاری تبدیل کند می توان همروندی بیشتری بدست آورد
- زیرا هردو همزمان می توانند به a1, a2 دسترسی پیدا کنند.
- این مشاهده مارا با پالایشی در 2PL راهنمایی می کند که در آن تغییر قفل مجاز است. لذا مکانیزیمی جهت تغییر قفل اشتراکی به انحصاری(**ارتقا**) و **تنزیل**(تبدیل قفل انحصاری به اشتراکی) تدارک دیده می شود.
- ارتقا در فاز رشد و تنزیل در فاز **تحلیل** رخ می دهند.

مثال از ارتقا و تنزیل

- مثال. فرض کنیم تراکنش های T1,T2 را بصورت زیر داریم.

Lock-S(a_1)	Lock-S(a_1)
Lock-S(a_2)	Lock-S(a_2)
Lock-S(a_n)	
	Unlock(a_1)
	Unlock(a_2)
Upgrade(a_1)	

تراکنشی که برای ارتقا قفلی روی عنصر داده Q تلاش می کند ممکن است وادار به انتظار شود.

این انتظار اجباری درصورتی رخ می دهد که Q توسط تراکنش دیگری به صورت اشتراکی قفل شده باشد.

قفل گذاری دو مرحله ای همراه با تبدیل قفل

- مرحله اول:
  - می تواند اجازه ایجاد یک قفل lock-S روی یک داده را پیدا کند.
  - می تواند اجازه ایجاد یک قفل lock-X روی یک داده را پیدا کند.
  - می تواند یک قفل lock-S را به یک قفل lock-X تبدیل کند. (ترفیع رتبه)
- مرحله دوم:
  - می تواند یک قفل lock-S را آزاد کند.
  - می تواند یک قفل lock-X را آزاد کند.
  - می تواند یک قفل lock-X را به یک قفل lock-S تبدیل کند. (تنزیل رتبه)
- این پروتکل ترتیب پذیری را تضمین می کند ولی هنوز برای تنظیم دستورات متنوع قفل گذاری متکی به برنامه نویس است.

استفاده خودکار از قفل ها

- تراکنش T_i دستورات خواندن/ نوشتن استاندارد را بدون فراخوانی قفل های انحصاری ایجاد می کند.
- The operation **read**(D) is processed as:

if T_i has a lock on D

then

read(D)

else begin

if necessary wait until no other transaction has a **lock-X** on D

grant T_i a **lock-S** on D;

read(D)

end

استفاده خودکار از قفل ها (ادامه)

write(*D*) is processed as:

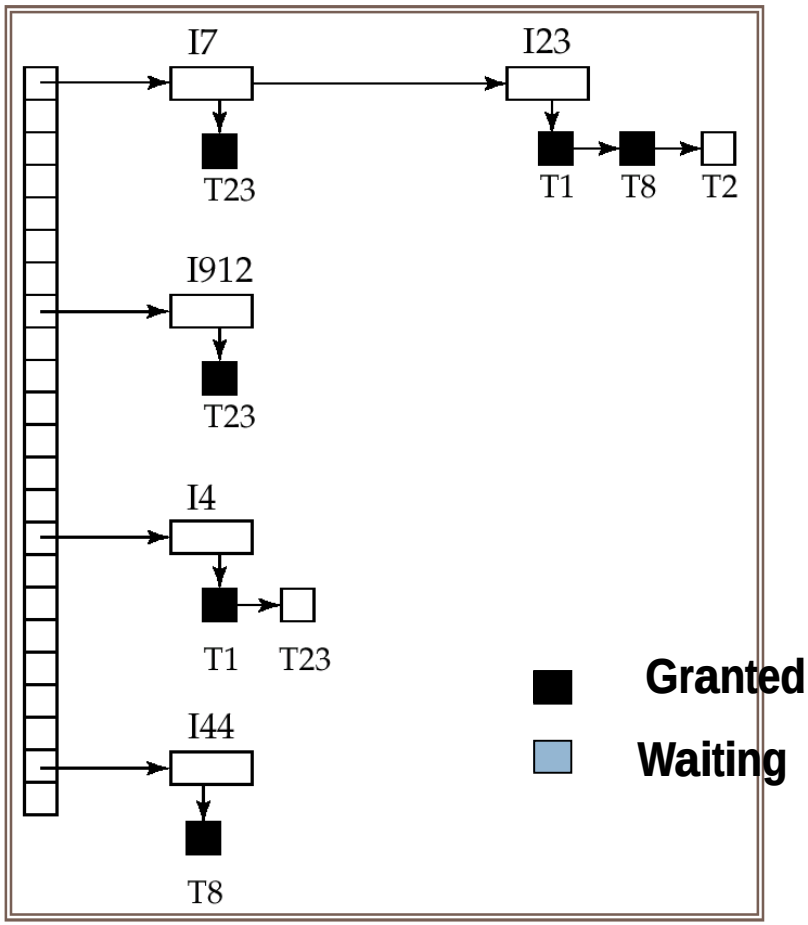
```
if Ti has a lock-X on D
  then
    write(D)
  else begin
      if necessary wait until no other trans. has any lock on D,
      if Ti has a lock-S on D
        then
          upgrade lock on D to lock-X
        else
          grant Ti a lock-X on D
        write(D)
  end;
```

- تمامی قفل ها پس از اجرا/توقف برنامه آزاد می شوند.

پیاده سازی قفل گذاری

- یک مدیر قفل می تواند به صورت یک پردازش مجزا پیاده سازی شود تا تراکنش های دیگر درخواست قفل گذاری یا قفل گشایی را برای او بفرستند.
- مدیر قفل به درخواست های قفل گذاری با فرستادن پیغام اعطای قفل (یا پیغام درخواست از تراکنش جهت برگشت در هنگام بن بست) پاسخ می دهد.
- تراکنش درخواست دهنده تا رسیدن به پاسخش منتظر می ماند.
- مدیر قفل یک ساختار داده ای به نام جدول قفل ایجاد می کند تا درخواست های اعطای قفل یا معلق مانده را ذخیره کند.
- یک جدول قفل معمولا به صورت یک جدول Hash روی نام داده ای که قفل شده است، پیاده سازی می شود.

جدول قفل

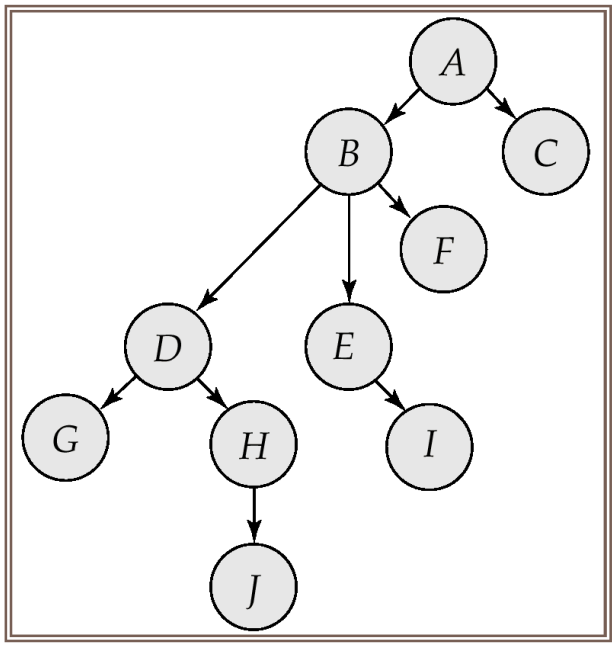


- مستطیل های تیره به قفل های اعطا شده و سفید به درخواست های منتظر اشاره می کند.
- جدول قفل هم چنین نوع قفل های اعطا شده یا درخواستی را ذخیره می کند.
- درخواست های جدید به انتهای صف درخواست های مربوط به یک داده اضافه می شود و اگر با قفل های اخیرا اعطا شده سازگاری داشت، اعطا می شود.
- درخواست های قفل گشایی در صورت حذف درخواست، اجرا می شوند و سپس سایر درخواست ها چک می کنند که آیا حالا می توانند عملی شوند یا خیر.
- اگر تراکنشی متوقف شود تمام درخواست های اعطای شده یا منتظر مربوط به آن، حذف می شوند.
  - مدیر قفل ممکن است فهرستی از قفل های هر تراکنش را جهت پیاده سازی بهینه این جدول نگه دارد.

پروتکل مبتنی گراف

- پروتکل مبتنی گراف جایگزینی برای پروتکل قفل گذاری دو مرحله ای است.
- اگر بخواهیم قانونی داشته باشیم که دو مرحله ای نباشد اما ترتیب پذیری را تضمین کند نیاز به اطلاعات اولیه درباره تراکنش ها داریم.
- مدلهای مختلفی وجود دارند. ساده ترین مدل، مدلی است که به اطلاعات اولیه در باره ترتیب دسترسی به داده های پایگاه داده نیاز دارد.
- برای این منظور بایستی یک ترتیب جزیی را برای داده ها به صورت مجموعه D، $D=\{d_1,d_2, \ldots,d_n\}$ تحمیل کرد.
  - اگر $d_i \rightarrow d_j$ آن گاه هر تراکنشی که به d_i و d_j دسترسی دارد، باید قبل از dj به di دسترسی پیدا کند.
  - در این روش مجموعه D به صورت یک گراف جهت دار و بدون حلقه در نظر گرفته می شود که به آن گراف پایگاه داده می گویند.
- پروتکل درخت نوع ساده ای از پروتکل مبتنی بر گراف است که فقط برروی قفل های انحصاری کار می کند.

پروتکل درخت



1. اولین قفل توسط T_i روی هر داده ای می تواند باشد
2. داده Q می تواند فقط توسط T_i قفل شود، اگر پدر Q در حال حاضر توسط T_i قفل شده باشد.
3. داده ها در هر زمانی می توانند آزاد شوند.
4. داده ای که توسط T_i قفل گذاری و آزاد سازی شده است، مجددا نمی تواند توسط T_i قفل شود.

توجه: تمام اجراهای معتبر تحت پروتکل درخت قابلیت ترتیب پذیری در برخورد را دارند.

در اینجا آزاد سازی قفل نسبت به دو مرحله ای سریعتر انجام می شود لذا زمان انتظار کاهش و همروندی افزایش می یابد.

پروتکل مبتنی بر گراف

- پروتکل درخت ترتیب پذیری و رهایی از بن بستdeadlock free را به خوبی پشتیبانی می کند.
- در پروتکل قفل گذاری مبتنی بر درخت قفل گشایی ممکن است زودتراز پروتکل قفل گذاری دو مرحله ای صورت گیرد.
  - زمان انتظار کوتاه تر و افزایش درجه همروندی
  - پروتکل فارغ از بن بست است و نیازی به برگشت خوردن تراکنش ها نیست.
- معایب
  - پروتکل ترمیم پذیری یا رهایی از برگشت های پی در پی را ضمانت نمی کند.
    - وابستگی های اجرا جهت تضمین ترمیم باید تعیین شود.
  - تراکنش ها باید داده هایی را که به آن دسترسی ندارند را قفل کنند. عوارض:
    - افزایش سربار ناشی از قفل و زمان انتظار اضافی
    - کاهش درجه همروندی
- برنامه هایی که تحت قفل گذاری دو مرحله ای عملی نیستند تحت پروتکل درخت امکان پذیرند و برعکس.

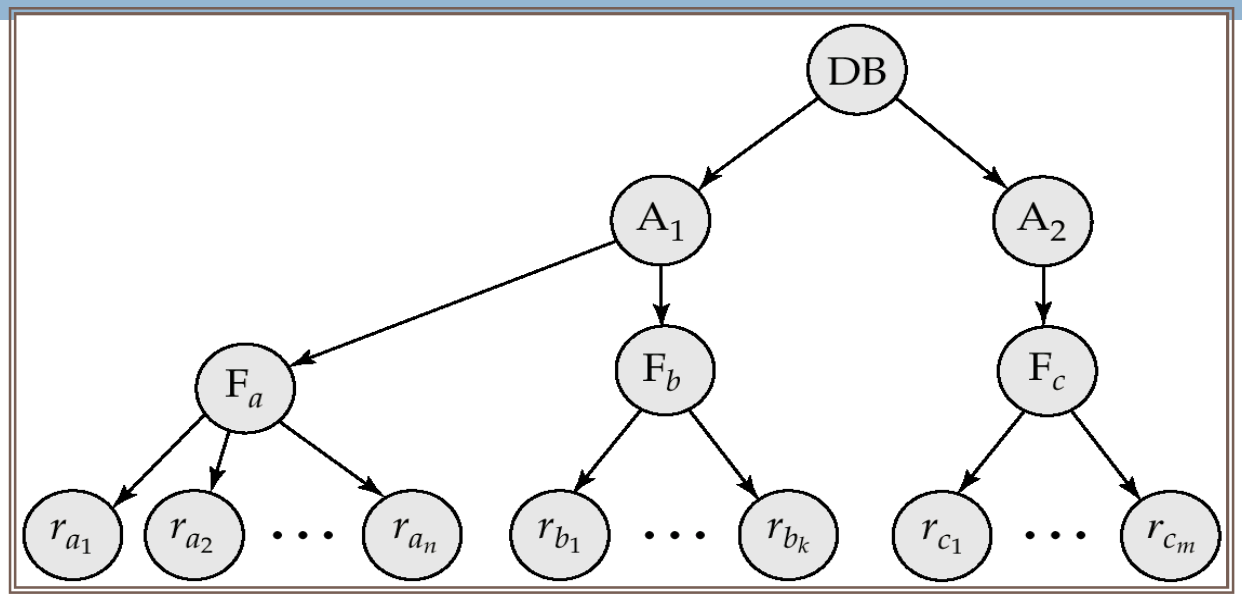
پروتکل دانه بندی چندگانه(چند واحد قفل شدنی)
Multiple Granularity

- به داده ها اجازه می دهد که سایزهای متفاوتی داشته باشند و سلسله مراتبی از داده ها ایجاد می کند که دانه های کوچکتر درون (زیر مجموعه) دانه های بزرگ تر هستند.
- به صورت گرافیکی همانند یک درخت نمایش داده می شود. (با پروتکل قفل گذاری درختی اشتباه نشود.)
- وقتی تراکنشی گره ای را به صورت صریح در درخت قفل می کند به صورت ضمنی تمام زیر گره های آن (فرزندان)گره قفل می شوند.
- وجود چند واحدقفل شونده می تواند سبب افزایش کارایی شود.

پروتکل دانه بندی چندگانه(چند واحد قفل شدنی)

- دانه بندی (سطوح)قفل گذاری: (سطوحی در درخت که قفل گذاری در آن انجام گرفته است.)
  - دانه بندی ملایم: **fine granularity**
    - (سطوح پایین درخت): افزایش همروندی، افزایش سربار ناشی از قفل گذاری
  - دانه بندی خشن: **coarse granularity**
  - (سطوح بالای درخت): کاهش همروندی، کاهش سربار ناشی از قفل گذاری

مثالی از سلسله مراتب دانه بندی



سطوحی که از بالاترین لایه شروع می شوند، عبارتند از:

- پایگاه داده
- ناحیه
- فایل
- رکورد

مدل های قفل گذاری خیالی

- در مدل دانه بندی چندگانه به غیر از مدهای S و X، سه مد قفل گذاری دیگر وجود دارند:
  - ***خیالی _ اشتراکی (IS):*** یعنی روی لایه های پایینی درختی که قفلی خیالی داریم قفلی از نوع اشتراکی قرار داده شده است.
    - اگر تراکنشی بخواهد روی رابطه ای قفل IS قرار دهد یعنی قصد دارد روی تاپل های رابطه قفل اشتراکی قرار دهد و تاپل ها تغییر نمی کنند.
  - ***خیالی _ انحصاری (IX):*** روی لایه های پایینی درخت قفلی خیالی از نوع اشتراکی و انحصاری قرار داده می شود.
    - اگر تراکنشی بخواهد روی رابطه ای قفل IX قرار دهد یعنی قصد دارد روی تاپل های رابطه قفل X قرار دهد و تاپل ها تغییر خواهند کرد.

مدل های قفل گذاری خیالی

اشتراکی و خیالی _ انحصاری (SIX):

- زیردرخت مربوط به گره قفل گذاری از نوع SIX، به صورت صریح در مد اشتراکی قفل شده است و یک قفل گذاری صریح از نوع انحصاری در سطوح پایینی زیردرخت مربوط به آن صورت گرفته است.
- معنای این قفل این است که گره های زیر درخت به ریشه دارای قفل SIX با نوع S قفل گذاری می شوندو به علاوه گره هایی از سطح پایین تر گره دارای قفل SIX ممکن است با نوع X نیز قفل شوند.
- اگر تراکنشی بخواهد روی رابطه ای قفل SIX قرار دهد یعنی تراکنش های دیگر می توانند تاپلهای رابطه را بخوانند اما نمی توانند عمل نوشتن داشته باشند از طرفی خود تراکنش ممکن است تاپل هایی را تغییر دهد.
- قفل های خیالی اجازه می دهند تا گره های بالایی در **S** و **X** بدون نیاز به چک کردن گره های زیرین قفل شوند.

ماتریس سازگاری در مدل قفل گذاری خیالی

- ماتریس سازگاری برای تمام مدل های قفل گذاری عبارتند از:

	IS	IX	S	S IX	X
IS	✓	✓	✓	✓	×
IX	✓	✓	×	×	×
S	✓	×	✓	×	×
S IX	✓	×	×	×	×
X	×	×	×	×	×

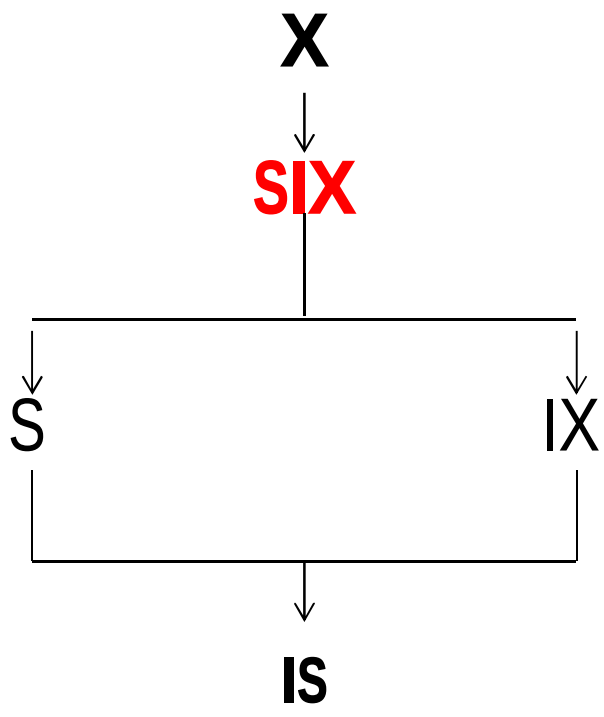
طرح قفل گذاری دانه بندی چندگانه

- تراکنش T_i گره Q را با استفاده از قوانین زیر قفل می کند:
  1. ریشه درماتریس سازگاری باید رعایت شود.
  2. ریشه درخت حتما باید در ابتدا قفل شود و نوع قفل مهم نمی باشد.
  3. گره Q می تواند توسط T_i در مدهای IS و S قفل شود، اگر پدر Q در حال حاضر توسط T_i در مدهای IX و IS قفل شده باشد.
  4. گره Q می تواند توسط T_i در مدهای IX و X و SIX قفل شود، اگر پدر Q در حال حاضر توسط T_i در مدهای IX و SIX قفل شده باشد.
  5. T_i در صورتی می تواند گره ای را قفل کند که قبل از آن هیچ گره ای را نگشوده باشد. (در این صورت T_i دو مرحله ای است.)
  6. T_i قفل گره Q را در صورتی می تواند باز کند که هیچ کدام از فرزندان Q در حال حاضر توسط T_i قفل نشده باشد.
- مشاهده می شود که قفل ها از ریشه به برگ اعطا می شوند و از برگ به ریشه آزاد می شوند.

طرح قفل گذاری دانه بندی چندگانه

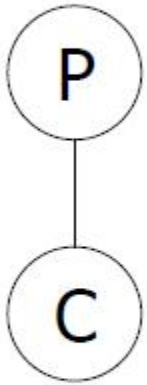
- قانون کلی :
  - قبل از اینکه هیچ تراکنشی بتواند روی گره ای قفل صریح (S,X) بگذارد باید روی تمام والدین آن قفل خیالی مناسب بگذارد.یعنی خبر می دهد من این پایین قفل قرارداده ام.
- روش تعدیل قفل: Lock Escalation
  - اگر تعداد قفل های یکسان روی گره های هم نیا از یک حد آستانه بیشتر شد آن قفل در گره پدر گرفته می شود.
  - عیب: ممکن است برخی گره ها بی جهت قفل گذاری شوند.

نمایش قدرت قفل ها در دانه بندی چندگانه



Locks With Multiple Granularity

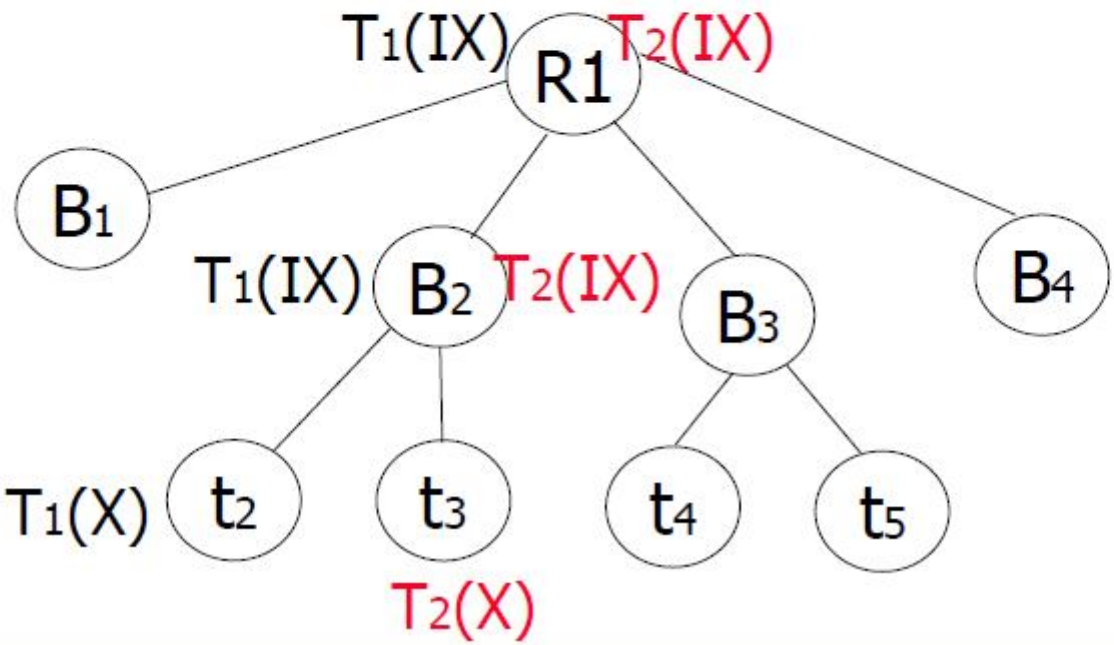
- An I lock for a superelement constrains the locks that the same transaction can obtain at a subelement.
- If Ti has locked the parent element P in IS, then Ti can lock child element C in IS, S.
- If Ti has locked the parent element P in IX, then Ti can lock child element C in IS, S, IX, X.



Locks With Multiple Granularity

- Example

  T2 wants to request an X lock on tuple t3



Locks With Multiple Granularity

- Example

  T2 wants to request an S lock on block B2